



Segmentation-based Image Similarity Search

Dissertation

zur Erlangung des Doktorgrades der Naturwissenschaften
(Dr. rer. nat.)

dem Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg
vorgelegt von

Master of Science (M.Sc.)
Nikolaus Korfhage
geboren in Frankfurt a. M.

Marburg, im Februar 2024

Vom Fachbereich Mathematik und Informatik der Philipps-Universität Marburg (Hochschulkennziffer 1180) als Dissertation am 6.5.2024 angenommen.

- 1. Gutachter:** Prof. Dr. Bernd Freisleben, Philipps-Universität Marburg
- 2. Gutachter:** Prof. Dr. Ralph Ewerth, Leibniz Universität Hannover

Tag der Einreichung: 26.2.2024

Tag der mündlichen Prüfung: 6.5.2024

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Dissertation selbstständig, ohne unerlaubte Hilfe Dritter angefertigt und andere als die in der Dissertation angegebenen Hilfsmittel nicht benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen sind, habe ich als solche kenntlich gemacht. Dritte waren an der inhaltlich-materiellen Erstellung der Dissertation nicht beteiligt; insbesondere habe ich hierfür nicht die Hilfe eines Promotionsberaters in Anspruch genommen. Kein Teil dieser Arbeit ist in einem anderen Promotions- oder Habilitationsverfahren der Antragstellerin oder des Antragsstellers verwendet worden. Mit dem Einsatz von Software zur Erkennung von Plagiaten bin ich einverstanden.

Datum

Unterschrift

Abstract

With the rapid and unprecedented growth of digital images, the need for effective image similarity search systems has become more important than ever. The application scenarios for image similarity search are numerous, ranging from e-commerce (where it enables customers to find products through image queries), over healthcare (where it supports diagnosis by comparing medical images), to digital archiving (where it helps to organize and access large volumes of visual data). Furthermore, there has been tremendous progress in the field of image segmentation in recent years, suggesting that image similarity search could possibly benefit from image segmentation.

This thesis provides contributions to two primary research areas: (1) detection and segmentation, and (2) image similarity search.

Two approaches are presented in the area of detection and segmentation: (a) a novel deep learning based workflow for automatic detection, alignment, and recognition of textual stamps on digitized index cards; (b) a novel cell segmentation approach for fluorescence microscopy images of morphologically complex eukaryotic cells.

In the area of image similarity search, we propose a novel approach to better understand the user's search intent. Moreover, we present a novel two-stage approach based on multi-index hashing to integrate deep hashing into Elasticsearch with query times comparable to state-of-the-art similarity search methods.

An important contribution of the thesis is a novel approach that combines insights from both domains into segmentation-based image similarity search, proposing the use of segmented images to enable querying image databases for specific regions within images. A novel versatile region-based similarity search approach for images couples two foundation models and enables users to utilize point, box, and text prompts to search for similar regions.

Finally, the thesis explores the practical implementation of image similarity search in different application domains. Real-world systems for analyzing large-scale image and video data benefit substantially from image similarity search, and image similarity search accelerates data acquisition and labeling when iteratively training specialized deep learning models.

Deutsche Kurzfassung

Mit dem rasanten und beispiellosen Anstieg der Anzahl digitaler Bilder ist der Bedarf an effektiven Bildähnlichkeitssuchsystemen wichtiger denn je geworden. Die Anwendungsszenarien für die Bildähnlichkeitssuche sind zahlreich und reichen vom elektronischen Handel (wo sie Kunden ermöglicht, Produkte durch Bildabfragen zu finden) über das Gesundheitswesen (wo sie Diagnosen durch den Vergleich medizinischer Bilder unterstützt) bis hin zur digitalen Archivierung (wo sie hilft, große Mengen visueller Daten zu organisieren und zu erschließen). Darüber hinaus wurden in den letzten Jahren enorme Fortschritte auf dem Gebiet der Bildsegmentierung erzielt, was darauf hindeutet, dass Bildähnlichkeitssuche von der Bildsegmentierung profitieren könnte.

Diese Arbeit liefert Beiträge in zwei primären Forschungsbereichen: (1) Detektion und Segmentierung von Objekten in Bildern und (2) Bildähnlichkeitssuche.

Im Bereich der Detektion und Segmentierung von Objekten in Bildern werden zwei Ansätze vorgestellt: (a) ein neuartiger, auf tiefen neuronalen Netzen basierender Workflow zur automatischen Erkennung, Ausrichtung und Erkennung von textuellen Stempeln auf digitalisierten Karteikarten; (b) ein neuartiger Zellsegmentierungsansatz für fluoreszenzmikroskopische Bilder von morphologisch komplexen eukaryotischen Zellen.

Im Bereich der Ähnlichkeitssuche von Bildern schlagen wir einen neuartigen Ansatz vor, der dazu dient, die Suchabsicht von Nutzer*innen besser zu verstehen. Darüber hinaus stellen wir einen neuartigen zweistufigen Ansatz vor, der auf Multi-Index Hashing basiert, um Deep Hashing in Elasticsearch zu integrieren, wobei die Abfragezeiten mit denen von aktuellen Ähnlichkeitssuchmethoden vergleichbar sind.

Ein wesentlicher Beitrag der Arbeit ist ein neuartiger Ansatz, der Erkenntnisse aus beiden Bereichen zu einer segmentierungsbasierten Bildähnlichkeitssuche kombiniert und die Verwendung von segmentierten Bildern zur Abfrage von Bilddatenbanken nach bestimmten Regionen innerhalb von Bildern beinhaltet. Dieser neuartige regionenbasierte Ansatz für die Ähnlichkeitssuche in Bildern verbindet zwei Basismodelle und ermöglicht es so Benutzer*innen, mit Hilfe von Punkt-, Box- und Text-Prompts nach ähnlichen Regionen zu suchen.

Zuletzt wird in dieser Arbeit der praktische Einsatz der Bildähnlichkeitssuche in verschiedenen Anwendungsdomänen untersucht. Reale Systeme zur Analyse großer Bild- und Videodaten profitieren erheblich von der Bildähnlichkeitssuche, und die Bildähnlichkeitssuche beschleunigt die Datenakquise und -annotation beim iterativen Training spezialisierter Modelle des tiefen Lernens.

Acknowledgments

First and foremost, my sincere appreciation goes to my supervisor, Prof. Dr. Bernd Freisleben, for his support, patience, and expertise. His mentorship was crucial in shaping this research and his encouragement and support sustained me through the challenges. I am also grateful for the many interesting and diverse projects in which he has enabled me to participate.

I would like to thank Prof. Dr. Ralph Ewerth from the Leibniz Universität Hannover for taking the time to review this work and for the opportunity to work with him and his research group in several projects.

I also want to thank all my past and present colleagues, collaborators, and advisors in our research group in Marburg who assisted me as co-authors of scientific papers, teaching, or other project-related tasks (in alphabetical order): Dr. Lars Baumgärtner, Hicham Bellafkir, Jakob Franz, Dr. Pablo Graubner, Dr. Jonas Höchst, Mechthild Kessler, Valeria Kizik, Mario Knapp, Dr. Patrick Lampe, Dr. Markus Mühling, Daniel Schneider, Michael Schwarz, Dr. Artur Sterz, Dr. Roland Schwarzkopf, Markus Sommer, Christian Strack, Thomas Trier, Christian Uhl, and Markus Vogelbacher.

Several projects provided financial support to me. I am very grateful for this financial support.

I gratefully acknowledge the financial support of the German Research Foundation (Deutsche Forschungsgemeinschaft; DFG) for the project “Bild- und Szenenrecherche in historischen Beständen des DDR-Fernsehens im Deutschen Rundfunkarchiv durch automatische inhaltsbasierte Videoanalyse” (project number: 230130757). I would like to thank all co-authors and the people who made our research possible (in alphabetical order): Prof. Dr. Ralph Ewerth, Prof. Dr. Bernd Freisleben, Angelika Hörth, Manja Meister, Dr. Markus Mühling, and Jörg Wehling.

I also participated in a project called “GoVideo – Automatische Verfahren zur kosteneffizienten Annotation von dokumentarischen Film- und Videoinhalten”, supported by the German Federal Ministry of Economic Affairs and Energy (BMWi) in the ZIM Programme. I would also like to thank all co-authors and contributors (in alphabetical order): Prof. Dr. Ralph Ewerth, Prof. Dr. Bernd Freisleben, Thomas Langelage, Dr. Markus Mühling, Dr. Eric Müller-Budack, Dr. Christian Otto, Matthias Springstein, and Dr. Uli Veith.

Furthermore, I received financial support during the project “FLORIDA: Flexibles, teilautomatisiertes Analysesystem zur Auswertung von Videomassendaten” (Förderkennzeichen 13N14250 bis 13N14256) supported by the German Federal Ministry of Education and Research (BMBF). In particular, I would like to thank (in alphabetical order): Prof. Dr. Bernd Freisleben, Dr. Markus Mühling, and all members of the project consortium.

As part of the DFG project “Denkfiguren|Wendepunkte. Kulturelle Praktiken und sozialer Wandel in der arabischen Welt” (Leibniz-Preis der DFG, 2012-2020), a database on contemporary Syrian literature for the Department of Arabic Literature and Culture was developed. I am grateful for the financial support and fruitful cooperation. I especially would like to thank (in alphabetical order): Dr. Felix Lang, Prof. Dr. Friederike Pannewick, and Anna Christina Scheiter.

My work on cell detection and segmentation was partly supported by the Hessian Ministry of Science and Arts (HMWK) (LOEWE SYNMIKRO Research Center and its Research Core Facility ‘Screening and Automation Technologies’). My thanks go to (in alphabetical order): Prof. Dr. Anke Becker, Prof. Dr. Bernd Freisleben, Dr. Markus Mühling, Stephan Ringshandl, and Prof. Dr. med. Bernd Schmeck.

I am also grateful for the financial support during my work on the DFG project “Szenen- und Personenerkennung für die automatische Erschließung von Videoarchiven” (project number: 388420599). I would like to thank my co-authors and other team members (in alphabetical order): Joanna Bars, Hicham Bellafkir, Sabrina Bernhöft, Prof. Dr. Ralph Ewerth, Prof. Dr. Bernd Freisleben, Angelika Hörth, Mario Knapp, Dr. Markus Mühling, Kader Pustu-Iren, Daniel Schneider, and Markus Vogelbacher.

I also would like to thank Prof. Dr. Elton Prifti and his team for the cooperation in the exciting stamp detection and recognition project as part of the Lessico Etimologico Italiano funded by the Akademie der Wissenschaften und der Literatur, Mainz.

Finally, I would like to thank my family, my friends, and particularly my partner Alexandra who gave me a lot of support, especially during the final phase of this work.

My Contributions

In the field of computer science, research papers are usually not written by a single person, but are the product of ideas, comments, and discussions among several people. New ideas are often developed and implemented collaboratively. Several works presented in this thesis reflect interdisciplinary collaborative projects. Parts of the content of this thesis are based on original publications, often verbatim. Therefore, I try to highlight my specific contributions in the following.

Chapter 3 is based on two publications from collaborative work in different fields. The first work [Kor+24] presents an approach for processing textual stamps and has been submitted but not published yet. The hope to accelerate the digitization process of the Lessico Etimologico Italiano (LEI) using deep learning methods was brought to me by Prof. Dr. Elton Prifti and Prof. Dr. Bernd Freisleben. The basic concept of the approach presented in Section 3.1 is my idea. It was implemented by myself with contributions from Hicham Bellafkir. The design of the dataset, the experiments, and evaluation were done by myself. Prof. Dr. Elton Prifti provided the raw data of scanned index cards and coordinated the annotation of the data, which was carried out by himself, Fabio Aprea, Elena Felicani, Anna Vaccaro, and Valentina Iosco. The experiments were performed by myself, Hicham Bellafkir, and Markus Vogelbacher. I wrote a first version of the manuscript with parts concerning the corpus of index cards written by Prof. Dr. Elton Prifti (Section 3.1.1). Prof. Dr. Elton Prifti, Dr. Markus Mühling, and Prof. Dr. Bernd Freisleben thoroughly reviewed the paper and suggested improvements to produce the final version of the paper.

The approach presented in Section 3.2 is a collaborative effort to improve segmentation of microscopic images of eucaryotic cells. Prof. Dr. Anke Becker and Stephan Ringshandl provided the initial motivation for this approach: until then, there was no adequate solution available for segmenting their particular fluorescence microscopy image data. The idea and implementation of the presented method is my work. The preparation and preprocessing of the data was done by myself, Dr. Markus Mühling and Markus Vogelbacher. Manual annotation of the dataset was done by Stephan Ringshandl. Markus Vogelbacher implemented an approach that was used for comparison in the experiments. Markus Vogelbacher participated in the postprocessing of the segmented cells. The initial version of the manuscript was written by myself and Prof. Dr. Bernd Freisleben, with parts concerning the microbiological background and microscopic images written by Stephan Ringshandl (Section 3.2.3). Prof. Dr. Bernd Freisleben, Dr. Markus Mühling and Prof. Dr. Anke Becker thoroughly reviewed and improved the manuscript.

Chapter 4 presents contributions in the area of image similarity search. The initial concept was jointly conceptualized by Prof. Dr. Bernd Freisleben, Dr. Markus Mühling, and myself. The approach presented in Section 4.1 was my idea and implemented by myself. The experiments were done by myself. The initial version of the manuscript was jointly written by myself and Dr. Markus Mühling. It was thoroughly revised and improved by Prof. Dr. Bernd Freisleben. The need for the approach presented in the second paper in Chapter 4 arose while working on a project with the DRA (German Broadcasting Archive). The approach for integrating image similarity search into Elasticsearch is my idea and the implementation and experimental

evaluation of the approach was done by myself. Prof. Dr. Bernd Freisleben and Dr. Markus Mühling made improvements to the initial version of the text that was written by myself.

The approach presented in Chapter 5 is based on my ideas on how to enable region-based similarity search for large datasets. It was implemented and evaluated by myself. I wrote a first version of the paper. Prof. Dr. Bernd Freisleben and Dr. Markus Mühling made suggestions to improve the text.

In Chapter 6, three systems in different application domains are presented. They are all collaborative work and involve the participation of many persons. The paper presented in Section 6.1 is the result of a project with the DRA (German Broadcasting Archive). A first version of the paper was mainly written by Dr. Markus Mühling and myself, with contributions to the introduction and evaluation of the results by Manja Meister. I contributed the approaches for visual concept classification and similarity search that I present in Section 6.1.2. The comparison between the bag-of-visual-words approach and multi-label convolutional neural networks in Section 6.1.3 was done by myself. Dr. Markus Mühling and myself implemented the approach for person recognition in Section 6.1.2. The experiments were carried out by Dr. Markus Mühling and myself. Improvements to the paper were suggested by Prof. Dr. Ralph Ewerth and Prof. Dr. Bernd Freisleben.

The system presented in Section 6.2 is the result of collaborative work. The initial version of the paper was mostly written by Dr. Markus Mühling and myself, with parts of Section 6.2.4 written by Dr. Eric Müller-Budack, Markus Otto, and Matthias Springstein. The requirements for the content-based video retrieval system were developed by Dr. Markus Mühling, Dr. Uli Veith, Prof. Dr. Ralph Ewerth, and Prof. Dr. Bernd Freisleben. I implemented and evaluated the approaches for visual concept detection, multi-task learning, and similarity search, which are presented in Section 6.2.3. I contributed significantly to the development of the system presented. Dr. Eric Müller-Budack, Markus Otto, and Matthias Springstein implemented and evaluated the approaches in Section 6.2.2. Prof. Dr. Ralph Ewerth and Prof. Dr. Bernd Freisleben reviewed and improved the work.

The paper presented in Section 6.3 contains results of a project on visual information retrieval in archives. It presents a system collaboratively developed by the DRA, Prof. Dr. Ralph Ewerth's research group and our research group in Marburg. Most parts of the initial manuscript were written by Dr. Markus Mühling, I wrote the parts concerning image similarity search. The approach in Section 6.3.3 was my idea, and I implemented and evaluated it in Section 6.3.5. Kader Pustu-Iren implemented the person recognition approach and evaluated it. The concept of the system is based on requirements defined by Dr. Markus Mühling, Joanna Bars, and myself. The development of the presented system was led by Dr. Markus Mühling and myself. The implementation was done by Mario Knapp, Hicham Bellafkir, Markus Vogelbacher, and myself. Prof. Dr. Ralph Ewerth and Prof. Dr. Bernd Freisleben reviewed and improved the text.

Contents

Abstract	iv
Deutsche Kurzfassung	v
Acknowledgments	vi
My Contributions	viii
Table of Contents	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Contributions of this Thesis	3
1.4 Publications	5
1.4.1 Publications in this Thesis	5
1.4.2 Further Publications	6
1.5 Open Source Software and Dataset Contributions	7
1.6 Organization of this Thesis	7
2 Fundamentals	9
2.1 Deep Learning	9
2.1.1 Feed-Forward Neural Networks	9
2.1.2 Activation Functions	11
2.1.3 Loss Functions	17
2.1.4 Optimization	19
2.1.5 Convolutional Neural Networks	25
2.1.6 Regularization	31
2.1.7 Transformers	34
2.2 Object Detection	38
2.2.1 Two Stage Detectors	39
2.2.2 Single Stage Detectors	42
2.3 Segmentation	45
2.3.1 U-Net	47
2.3.2 Mask R-CNN	48
2.4 Foundation Models	49
2.4.1 CLIP	49
2.4.2 SAM	50
2.4.3 FastSAM	50
2.5 Image Similarity Search	51
2.5.1 Similarity Measures	52
2.5.2 Approximate Nearest Neighbor Search	53
2.5.3 Locality Sensitive Hashing	53
2.5.4 Deep Hashing	54

2.5.5	Product Quantization	54
2.5.6	Multi-Index Hashing	55
3	Detection and Segmentation	57
3.1	Textual Stamp Recognition on Index Cards	58
3.1.1	Introduction	58
3.1.2	Related Work	62
3.1.3	Method	63
3.1.4	Results	69
3.1.5	Summary	74
3.2	Detection and Segmentation of Eukaryotic Cells via Feature Pyramid Fusion .	75
3.2.1	Introduction	75
3.2.2	Related Work	76
3.2.3	Design and Implementation	77
3.2.4	Results	82
3.2.5	Summary	89
4	Image Similarity Search	91
4.1	Intentional Image Similarity Search	92
4.1.1	Introduction	92
4.1.2	Related Work	94
4.1.3	German Broadcasting Archive	95
4.1.4	A Novel Approach to Intentional Image Similarity Search	96
4.1.5	Experimental Results	99
4.1.6	Summary	102
4.2	ElasticHash: Semantic Image Similarity Search in Elasticsearch	103
4.2.1	Introduction	103
4.2.2	Related Work	104
4.2.3	<i>ElasticHash</i>	105
4.2.4	Experimental Evaluation	110
4.2.5	Summary	113
5	Segmentation-based Image Similarity Search via Region Prompts	115
5.1	Introduction	115
5.2	Related Work	116
5.2.1	Foundation Models	116
5.2.2	Image Similarity	117
5.2.3	Prompts	118
5.2.4	Zero-shot Region Retrieval	118
5.2.5	Large-scale Similarity Search	118
5.3	Approach	119
5.3.1	Region Prompts	120
5.3.2	Deep Hashing	120
5.3.3	Training	121
5.3.4	Region Indexing	121

5.4	Experiments	122
5.4.1	Datasets	122
5.4.2	Region Features	123
5.4.3	Region Retrieval	127
5.4.4	Efficient Region Retrieval	128
5.4.5	Qualitative Results	129
5.5	Summary	136
6	Image Similarity Search in Applications	137
6.1	Video Retrieval in Historical Collections	139
6.1.1	Introduction	139
6.1.2	A Content-Based Video Retrieval System	140
6.1.3	Experimental Results	150
6.1.4	Summary	157
6.2	Content-based Video Retrieval in Film and Television Production	158
6.2.1	Introduction	158
6.2.2	Content-based Video Analysis for Media Production	159
6.2.3	Video Retrieval Tool	166
6.2.4	Experimental Results	168
6.2.5	Summary	175
6.3	Visual Information Retrieval in Video Archives	177
6.3.1	Introduction	177
6.3.2	Related Work	179
6.3.3	Image and Video Retrieval Approaches	179
6.3.4	VIVA	185
6.3.5	Experimental Results	189
6.3.6	Summary	194
7	Conclusion	195
7.1	Summary	195
7.2	Future Work	196
7.2.1	Detection and Segmentation	196
7.2.2	Image Similarity Search	197
7.2.3	Segmentation-based Image Similarity Search	197
	List of Figures	199
	List of Tables	203
	Listings	205
	Bibliography	207

1

Introduction

The need for fast and high-quality image similarity search emerges from the rapid growth of digital imagery and the consequent need to efficiently handle and navigate through vast visual databases. In today's digitally driven world, where billions of images are captured and shared daily, the ability to quickly and accurately find images that are visually similar to a given query image is becoming increasingly necessary.

Image similarity search has many applications in different fields. In e-commerce [Ak+18], for example, it allows customers to find products by uploading images, enhancing the shopping experience. In digital photo management, it helps to organize and categorize large image libraries. In healthcare, it can aid in diagnosis by allowing similar medical images to be retrieved for comparison, helping to detect and analyze diseases [Sil+22; KRS20; Qay+17].

While image similarity search has been an area of research for a long time, only recently have deep learning models significantly improved the retrieval quality, and data-driven indexing approaches, such as deep hashing and product quantization, have enabled image similarity search at scale.

1.1 Motivation

Although image similarity search is already commonly used in domains such as e-commerce and medical imaging, it is not yet widely used in several other domains. Thus, there are specific applications to be explored in other domains where image similarity search can make a significant contribution. For example, in the digital humanities, where large databases of digitized historical material are being created, image similarity search can be of great benefit. It can help to unlock cultural heritage by making large digitized image or video databases accessible.

Another more general area where there is great potential for image similarity search is in the dataset creation phase of the deep learning cycle. This is especially true for problems that require labeled training data and iterative training to improve machine learning models. Within the deep learning cycle, image similarity search can play a key role in the data acquisition phase. The application of image similarity search in this context primarily involves the enrichment and refinement of training datasets. Deep learning models require large, diverse, and high-quality datasets to learn effectively. Image similarity search can help to collect and enrich these datasets by retrieving images that are visually similar to existing images. This helps to cover a wider range of examples and variations, which is critical for training models to generalize well.

Where data labeling is required, similarity search can speed up the labeling process by grouping similar images together, making it easier and more efficient for human annotators to label them. This improved data acquisition process can substantially contribute to the development of more powerful deep learning models.

In addition to providing high quality retrieval results, an image similarity search system must be efficient. Ideally, this means an immediate response to a query, but at least a timely response in less than one second. It is often difficult to find a trade-off between search speed (usually determined by the length of the vector representing the images) and retrieval quality, and depends on, among other things, the size of the data set to be indexed, the available hardware resources, and the software environment. Therefore, it is necessary to develop solutions under specific conditions.

In addition to the semantic gap for images [Sme+00a], there exists an intentional gap in the context of image retrieval via image similarity search. This gap refers to the discrepancy between the query provided by the user and the user's actual search intention. Addressing this intentional gap is critical in improving the effectiveness of image similarity search and quality of search results.

Traditional image similarity search or instance retrieval methods that evaluate images as a whole and compress the whole image into a short, often binary, vector fail to capture the subtleties within different regions of the image, leading to less accurate or relevant results. They usually rely primarily on the use of global image features and are therefore unable to account for the intricacies and distinct patterns within local regions of an image. This can lead to sub-optimal search results, particularly in scenarios where specific image regions are of interest. In instance retrieval [Che+22], for example, in addition to global features, local descriptors are extracted, which, however, are still used on the whole image level. Furthermore, instance retrieval is limited to instances of a particular object. In contrast, segmenting images into smaller parts and indexing these regions can enable a more detailed search, focusing on specific areas of interest within images. Segmentation-based image similarity search can be particularly useful when only a specific region of an image is of interest, rather than the entire scene.

1.2 Problem Statement

Image similarity search works by comparing features extracted from images. After selecting a single query image, a ranked list of images *similar* to this query image is returned. What similar means usually depends on the dataset and the model trained on that data, respectively. However, this does not necessarily coincide with a user's search intention, and there is often a discrepancy between the returned retrieval list and a user's expectations of what aspects of the present query image should be considered for similarity search. A challenge in image similarity search is therefore to bridge this gap by exploring how to better understand and interpret the user's underlying search intentions, which may not be explicitly conveyed by the query image alone.

This dissertation investigates how these search intentions can be more precisely specified. Special emphasis is placed on search scenarios where specific regions, rather than the entire

image, are of interest. This leads to the main research question of this thesis, which is how image similarity search can benefit from image segmentation. Since there can be potentially many representations extracted for a segmented image, this raises further questions concerning the efficiency of the search system that are important for segmentation-based image similarity search. The aim of this thesis is to develop a segmentation-based image search system that fulfills these requirements, i.e., delivers high-quality retrieval results and does this in an efficient manner.

1.3 Contributions of this Thesis

The main contribution of this thesis is a novel approach to image similarity search based on segmented images. Segmentation-based image similarity search covers the areas of image similarity search on the one hand, and image segmentation on the other. Additionally, this thesis provides contributions in both areas, segmentation and similarity search, but also in related areas, such as content-based video retrieval and analysis, stamp detection and recognition, concept classification, and person recognition.

The contributions of this thesis can be roughly categorized into three areas. First, contributions in the area of object detection and segmentation in images are addressed. Following that, advancements in the area of image similarity search are presented. Additionally, there are contributions in several application domains.

The contributions of this thesis in different areas are as follows:

Detection and Segmentation

- An innovative approach to accelerate the production of the Lessico Etimologico Italiano (LEI), a comprehensive historical and etymological dictionary of the Italian language and dialects, which traditionally relied on manual lexicographic methods. The proposed deep learning workflow for automatic detection, alignment, and recognition of textual stamps on digitized index cards significantly streamlines the philological work involved in processing large numbers of scanned cards.
- A novel approach for the detection and segmentation of macrophage cells in fluorescence microscopy images, addressing challenges like crowded cell environments and morphological complexity. It involves the integration of previously learned nucleus features into the cell detection and segmentation architecture, resulting in improved performance. The proposed feature pyramid fusion architecture outperforms other state-of-the-art approaches on this challenging dataset.

Image Similarity Search

- Implementation and evaluation of a similarity search method to identify new textual stamps within the corpus of philological index cards of the Lessico Etimologico Italiano (LEI).

- Implementation of multi-task learning for visual concept detection and similarity search within a single common architecture, allowing network weight sharing to reduce computation time significantly.
- Introduction of intentional image similarity search as a novel approach to image similarity search that addresses the limitations of query-by-example in content-based image and video retrieval by allowing users to more effectively specify their search intent for a query image. It includes a plugin mechanism to support fine-grained neural network models tailored to specific search tasks, a hybrid feature method that combines features extracted from convolutional neural networks with hand-crafted features, and a deep similarity network analysis technique to find relevant image regions and improve the relevance of retrieved results.
- Introduction of *ElasticHash*, a novel approach for efficient and large-scale semantic image similarity search in Elasticsearch. It utilizes a deep hashing model to generate hash codes for fine-grained image similarity search in natural images. A two-stage search method that combines multi-index hashing and terms lookup enables fast and accurate similarity search. The first stage involves a coarse search based on short hash codes, while the second stage re-ranks results based on Hamming distance computed on long hash codes.
- *Search Anything*, a novel approach for performing similarity search in images, enabling users to use point, box, and text prompts to search for similar regions within a set of images. By using self-supervised learning and foundation models to learn binary hash code representations for image regions, it enables fine-grained region-level indexing and searching. It automatically segments the region selected by a prompt and extracts a binary feature vector, which is then used to query an image region index to retrieve images containing the corresponding regions.

Applications

- Introduction of a system designed for automatic video content analysis and retrieval within the historical collections of GDR television recordings. As a distributed, service-oriented architecture, it incorporates various video analysis algorithms, including shot boundary detection, concept classification, person recognition, text recognition, and similarity search.
- Introduction of a system that includes deep learning approaches to assist professional media production by automating content-based labeling video footage. This includes the development of specific methods for visual concept detection, similarity search, face detection, face recognition, and face clustering, which are integrated into a multimedia tool for efficient video inspection and retrieval with innovative visualization components.
- Introduction of VIVA, a software tool designed for building content-based video retrieval methods using deep learning models. VIVA enables non-expert users to perform visual information retrieval for concepts and persons within video archives and adapt the models to changing search requirements. It provides a novel semi-automatic data acquisition workflow including a web crawler, image similarity search, and review and user feedback components to reduce the time-consuming manual effort for collecting training samples.

1.4 Publications

1.4.1 Publications in this Thesis

1. **Nikolaus Korfhage**, Markus Mühling, and Bernd Freisleben. “Search Anything: Segmentation-based Similarity Search via Masked Region Prompts.” in: *Submitted; Under Review*. 2024 [KMF24]
2. **Nikolaus Korfhage**, Hicham Bellafkir, Markus Mühling, Markus Vogelbacher, Elton Prifti, and Bernd Freisleben. “Deep Learning for Textual Stamp Recognition on Index Cards of the Lessico Etimologico Italiano.” in: *Submitted; Under Review*. 2024 [Kor+24]
3. Markus Mühling, **Nikolaus Korfhage**, Kader Pustu-Iren, Joanna Bars, Mario Knapp, Hicham Bellafkir, Markus Vogelbacher, Daniel Schneider, Angelika Hörth, Ralph Ewerth, and Bernd Freisleben. “VIVA: Visual Information Retrieval in Video Archives.” in: *International Journal on Digital Libraries* 23.4 (2022), pp. 319–333 [Müh+22]
4. **Nikolaus Korfhage**, Markus Mühling, and Bernd Freisleben. “ElasticHash: Semantic Image Similarity Search by Deep Hashing With Elasticsearch.” in: *Computer Analysis of Images and Patterns: 19th International Conference, CAIP 2021, Virtual Event, September 28–30, 2021, Proceedings, Part II* 19. Springer. 2021, pp. 14–23 [KMF21]
5. **Nikolaus Korfhage**, Markus Mühling, Stephan Ringshandl, Anke Becker, Bernd Schmeck, and Bernd Freisleben. “Detection and Segmentation of Morphologically Complex Eukaryotic Cells in Fluorescence Microscopy Images via Feature Pyramid Fusion.” in: *PLOS Computational Biology* 16.9 (9 Sept. 2020), e1008179. ISSN: 15537358. DOI: 10.1371/JOURNAL.PCBI.1008179 [Kor+20]
6. **Nikolaus Korfhage**, Markus Mühling, and Bernd Freisleben. “Intentional Image Similarity Search.” in: *Artificial Neural Networks in Pattern Recognition: 9th IAPR TC3 Workshop, ANNPR 2020, Winterthur, Switzerland, September 2–4, 2020, Proceedings* 9. vol. 12294 LNAI. Springer. Springer Science and Business Media Deutschland GmbH, 2020, pp. 23–35. DOI: 10.1007/978-3-030-58309-5_2 [KMF20]
7. Markus Mühling, Manja Meister, **Nikolaus Korfhage**, Jörg Wehling, Angelika Hörth, Ralph Ewerth, and Bernd Freisleben. “Content-based Video Retrieval in Historical Collections of the German Broadcasting Archive.” in: *International Journal on Digital Libraries* 20 (2019), pp. 167–183 [Müh+19]
8. Markus Mühling, **Nikolaus Korfhage**, Eric Müller, Christian Otto, Matthias Springstein, Thomas Langelage, Uli Veith, Ralph Ewerth, and Bernd Freisleben. “Deep Learning for Content-based Video Retrieval in Film and Television Production.” in: *Multimedia Tools and Applications* 76 (21 Nov. 2017), pp. 22169–22194. ISSN: 15737721. DOI: 10.1007/s11042-017-4962-9 [Müh+17]

1.4.2 Further Publications

During the work on this thesis, the following further papers have been published, but are not presented in this thesis:

1. Markus Vogelbacher, Finja Strehmann, Hicham Bellafkir, Markus Mühling, **Nikolaus Korfhage**, Daniel Schneider, Sascha Rösner, Dana G Schabo, Nina Farwig, and Bernd Freisleben. “Identifying and Counting Avian Blood Cells in Whole Slide Images via Deep Learning.” in: *Birds* 5.1 (2024), pp. 48–66 [Vog+24]
2. Finja Strehmann, Markus Vogelbacher, Clara Guckenbiehl, Yvonne Ramona Schumm, Juan Masello, Petra Quillfeldt, **Nikolaus Korfhage**, Hicham Bellafkir, Markus Mühling, Bernd Freisleben, Nina Farwig, Dana Schabo, and Sascha Rösner. “Intrinsic Factors Influence Physiological Stress in a Forest Bird Community: Adults and Females Have Higher H/L Ratios Than Juveniles and Males.” in: *Submitted; Under Review*. 2024 [Str+24]
3. Hicham Bellafkir, Markus Vogelbacher, Daniel Schneider, Markus Mühling, **Nikolaus Korfhage**, and Bernd Freisleben. “Edge-based Bird Species Recognition via Active Learning.” in: *International Conference on Networked Systems*. Springer. 2023, pp. 17–34 [Bel+23]
4. Hicham Bellafkir, Markus Vogelbacher, Jannis Gottwald, Markus Mühling, **Nikolaus Korfhage**, Patrick Lampe, Nicolas Frieß, Thomas Nauss, and Bernd Freisleben. “Bat echolocation call detection and species recognition by transformers with self-attention.” in: *Intelligent Systems and Pattern Recognition: Second International Conference, ISPR 2022, Hammamet, Tunisia, March 24–26, 2022, Revised Selected Papers*. Springer. Cham: Springer International Publishing, 2022, pp. 189–203 [Bel+22]
5. Daniel Schneider, **Nikolaus Korfhage**, Markus Mühling, Peter Lüttig, and Bernd Freisleben. “Automatic Transcription of Organ Tablature Music Notation With Deep Neural Networks.” in: *Transactions of the International Society for Music Information Retrieval* 4 (1 Feb. 2021), pp. 14–28. ISSN: 25143298. DOI: 10.5334/tismir.77 [Sch+21]
6. Markus Mühling, Jakob Franz, **Nikolaus Korfhage**, and Bernd Freisleben. “Bird Species Recognition via Neural Architecture Search.” in: ed. by L. Cappellato, C. Eickhoff, N. Ferro, and A. Névél. CEUR-WS.org, 2020. URL: https://ceur-ws.org/Vol-2696/paper_188.pdf [Müh+20]
7. Kader Pustu-Iren, Markus Mühling, **Nikolaus Korfhage**, Joanna Bars, Sabrina Bernhöft, Angelika Hörth, Bernd Freisleben, and Ralph Ewerth. “Investigating Correlations of Inter-coder Agreement and Machine Annotation Performance for Historical Video Data.” in: *Digital Libraries for Open Knowledge: 23rd International Conference on Theory and Practice of Digital Libraries, TPDL 2019, Oslo, Norway, September 9-12, 2019, Proceedings* 23. vol. 11799 LNCS. Springer. Springer Verlag, 2019, pp. 107–114. DOI: 10.1007/978-3-030-30760-8_9 [Pus+19]
8. Johannes Drönner, **Nikolaus Korfhage**, Sebastian Egli, Markus Mühling, Boris Thies, Jörg Bendix, Bernd Freisleben, and Bernhard Seeger. “Fast Cloud Segmentation Using Convolutional Neural Networks.” in: *Remote Sensing* 10.11 (11 Nov. 2018), p. 1782. ISSN: 20724292. DOI: 10.3390/rs10111782 [Drö+18]

1.5 Open Source Software and Dataset Contributions

During the work on this thesis, the following software has been co-developed and released under permissive open source licenses:

1. ElasticHash implementation that enables large-scale image similarity search in Elasticsearch including Tensorflow model serving and web front-end [KMF21]. Available at <https://github.com/umr-ds/ElasticHash>.
2. ElasticHash Python package which builds on the Elasticsearch Python package. Available at <https://github.com/nik-ko/elasticash>
3. VIVA, a web-based software tool for building content-based retrieval methods for video archives based on deep learning models [Müh+22]. Available at <https://github.com/umr-ds/VIVA>
4. Mask R-CNN implementation modified to enable feature pyramid fusion for detection and segmentation in fluorescence microscopy images [Kor+20]. Available at https://github.com/umr-ds/feature_pyramid_fusion

The following publicly available datasets have been released:

- Text stamps on index cards dataset consisting of a detection dataset of 6,991 scanned index cards, containing bounding box annotations of 6,759 stamps, and a recognition dataset, containing 170,494 images of 4,304 different stamp classes [Kor+24]. Available at <https://github.com/umr-ds/stamp-detection-recognition>
- Cell segmentation dataset of 82 2-channel fluorescence microscopy images including more than 2,500 macrophage cells and nuclei [Kor+20]. Available at <https://data.uni-marburg.de/handle/dataumr/231>
- Hashcode dataset for about 6.9 million images of the OpenImages dataset [KMF21]. Available at <https://data.uni-marburg.de/handle/dataumr/233>

1.6 Organization of this Thesis

This thesis is organized as follows:

Chapter 2 introduces topics fundamental to the research in this thesis. The focus is on deep learning. In particular, convolutional neural networks (CNNs) and neural network architectures for detection and segmentation, including foundation models, are discussed. This chapter also provides an introduction to image similarity search and methods for efficient nearest neighbor search.

Chapter 3 covers research in the fields of detection and segmentation. First, we present a deep learning workflow for efficiently processing index cards, focusing on automatic detection, alignment, and recognition of text stamps, along with an image similarity search method for identifying new stamps. Then, we present an approach for detecting and segmenting morphological complex cells in fluorescence microscopy images.

Chapter 4 presents research aimed at improving image similarity search in two ways: query time on the one hand, and retrieval quality at the other hand. Our work on intentional image similarity search aims to improve retrieval quality according to the user's expectations by enabling users to specify their search intentions more precisely. In the second work, we present an efficient and robust two-stage approach for integrating deep hashing into Elasticsearch.

Chapter 5 builds on the areas covered in Chapters 3 and 4 by presenting a novel approach that combines segmentation and image similarity search. By not considering whole images as data points, but by segmenting them first, this, together with region prompts, allows a finer grained image similarity search at the level of image regions.

Chapter 6 presents three content-based image retrieval systems. In each of these systems, the use of image similarity search is an important component. The first system is dedicated to making the heritage of GDR television accessible. The second system is a multimedia tool for rapid video inspection and retrieval to support media production. The third system allows non-expert users to train and apply deep learning models on a corpus of videos, including the creation of new concepts to be searched for. Here, image similarity search plays an essential role in data acquisition.

Chapter 7 concludes the thesis and discusses possible areas of future work.

2

Fundamentals

This chapter discusses the basic concepts and technologies used throughout this thesis. Section 2.1 introduces deep learning with a focus on convolutional neural networks (CNNs). These form the basis for the object detection models discussed in Section 2.2 and the architectures used for segmentation, which are discussed in Section 2.3. Section 2.4 introduces several foundation models used in the work presented in this thesis. Finally, concepts related to image similarity search are introduced in Section 2.5.

2.1 Deep Learning

Deep learning as a subarea of machine learning involves algorithms based on artificial neural networks. These networks were originally inspired by the structure and function of the brain.

2.1.1 Feed-Forward Neural Networks

A typical artificial neural network (ANN) consists of two or more layers: an output layer with one or more hidden layers, with the inputs connected to the first hidden layer. The elements in the hidden layers and the output layer are called neurons or units. Figure 2.1 shows a small feed-forward neural network that receives four input values and consists of four hidden and three output neurons, i.e., it returns a vector with three components.

A feed-forward network is a directed acyclic graph, which means that it must not contain any cycles or connections from higher to lower layers. Units between two adjacent layers are fully pairwise connected, while units within a single layer share no connections. We will refer to the type of hidden layer in the network shown in Figure 2.1 as a fully connected layer, although it has different names depending on the context. It is also called dense or linear layer in deep learning frameworks. In the context of transformers, multiple fully connected layers are referred to as multi-layer perceptron (MLP).

The size of a neural network is usually measured in terms of learnable parameters, i.e., weights and biases. Recent neural networks can contain hundreds of millions or billions of parameters, organized in hundreds of layers; this is why they are called *deep* neural networks.

When the network is used for inference, an input vector n , fed into the input layer, is passed through the network, resulting in an output vector, the prediction. In the forward pass, the inputs to the network are propagated through the network. Starting from the input layer, the

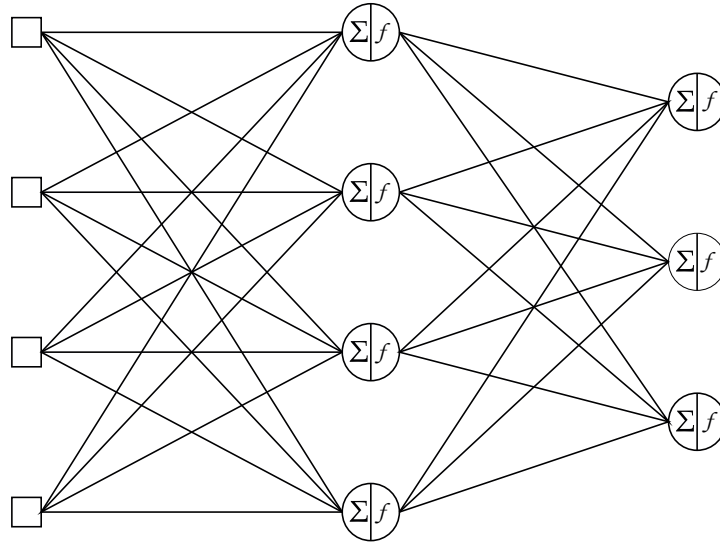


Figure 2.1: A feed forward neural network with four inputs, three outputs and one hidden layer with four neurons.

outputs (activations) of the input units serve as inputs to the units of the next layer, i.e., they are passed forward to the units of the next hidden layer, which in turn may be followed by several hidden layers. Finally, this results in activations in one or more units of the output layer. This output vector is the prediction for a given input.

In the training phase of the network, the forward pass is followed by a backward pass, which adjusts the units according to a loss function. The output activations are compared to the true outputs by an error function, also called the loss. If the deviation from the true output is large, the loss function returns a large loss, while if the network output is close to the ground truth, the loss is small. To correct the parameters of the network, a backward pass is performed. In the backward pass, the loss of the network is propagated backward: all parameters (i.e., weights and biases) of the network are changed by a small amount towards the true output, depending on how much they contributed to the current output. This adaptation of the parameters can be achieved by, for example, stochastic gradient descent, and the calculation of the weight adaptations is usually done by the backpropagation algorithm.

When dealing with a large number of units, it is useful to specify the network in terms of layers rather than individual units. The weights between layers can then be viewed as matrices \mathbf{W} where the columns are the outputs of the previous layer and the rows are the inputs of the current layer. More precisely, the weight matrix of the l^{th} layer is defined as \mathbf{W}^l . Its elements are then identified as w_{ji}^l , where j corresponds to the j^{th} unit of the actual l layer and i corresponds to the i^{th} unit of the $(l-1)^{\text{th}}$ layer. Similarly, the biases of a layer l are defined as a vector \mathbf{b}^l with b_j^l for the j^{th} unit of the layer l . It follows that the output of a single unit j in the hidden layer l is given by

$$a_j^l = f\left(\sum_i w_{ji}^l a_i^{l-1} + b_j^l\right) \quad (2.1)$$

and thus the output of the layer, i.e., the output of all neurons in this layer, by

$$\mathbf{a}^l = f\left(\mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l\right), \quad (2.2)$$

where f is the activation function and $\mathbf{a}^0 = \mathbf{x}$ is the input to the neural network. However, f need not be the same for all layers. The matrix notation reflects an important feature of neural network layers: all computations can be performed in parallel. For example, the weighting of the incoming connections \mathbf{a}^{l-1} in Figure 2.2 is simply a matrix multiplication.

In the following, all components of the neural network will be explained in more detail. First, some common activation functions f are introduced. Then, the training procedure is described in more detail. This includes the choice of a suitable loss function and the application of the backpropagation algorithm. Furthermore, some kind of regularization is needed to make the training less prone to overfitting.

2.1.2 Activation Functions

In case of fully connected layers, the summed inputs constitute the input of the activation function. In the following, the input of an activation function is defined as

$$z = \sum_i w_{ji} a_j + b_j, \quad (2.3)$$

or, in vectorized form as

$$\mathbf{z} = \mathbf{W}\mathbf{a} + \mathbf{b}, \quad (2.4)$$

whereas the layer superscripts and unit subscripts are sometimes omitted for the sake of simplicity.

An important requirement for the units in a neural network is the use of a nonlinear activation function. The nonlinearity of the activation is crucial. It allows the neural network to learn complex patterns. In contrast, with a layered composition of linear functions, the entire network itself is a linear function and would reduce the neural network to a linear model.

For the output units it is important to choose an activation function that fits to the characteristics of the target values (e.g., sigmoid for binary classification, softmax for multi-class classification, or linear for regression). In following, several common activation functions and their characteristics are introduced.

Sigmoid

A common activation function is the logistic (or sigmoid) activation. It maps input values to values between 0 and 1, thereby transforming the inputs into a probability, as shown in Figure 2.2. While it was the standard activation function in earlier feed-forward networks, today it is mainly used in the output layer for binary classification, representing the probability of an instance belonging to one of two classes. The logistic function is defined as

$$f_{\sigma}(z) = \frac{1}{1 + e^{-z}}. \quad (2.5)$$

Backpropagation of the forward pass error in neural networks requires the computation of derivatives of the activation functions. In more complex networks, with a larger number of units, the computational cost of these derivatives becomes significant. One attractive property of the logistic function for neural networks is its simple derivative, which is given by

$$f'_\sigma(z) = f_\sigma(z) (1 - f_\sigma(z)). \quad (2.6)$$

However, this type of activation function has drawbacks. The logistic function can saturate, resulting in a very small gradient in deeper layers during backpropagation. Since the error is backpropagated to the inputs by multiplication, the adaptation of the incoming weights will also be very small - the network will hardly learn due to a vanishing gradient. In addition, to ensure fast convergence of neural networks, the use of non-zero mean inputs for neurons should be avoided [LeC+12]. This is easily achieved in the first layer by normalizing the inputs. However, since the sigmoid activation function is non-symmetric, the output of a neuron is restricted to the interval $[0, 1]$. According to Glorot et al. [GB10], logistic sigmoid activations are inappropriate as hidden layer activation functions in deeper networks (more than four layers deep) because they can saturate the top hidden layer very early and lead to poor learning dynamics. Based on these observations, the *tanh* activation function is a better choice in many situations.

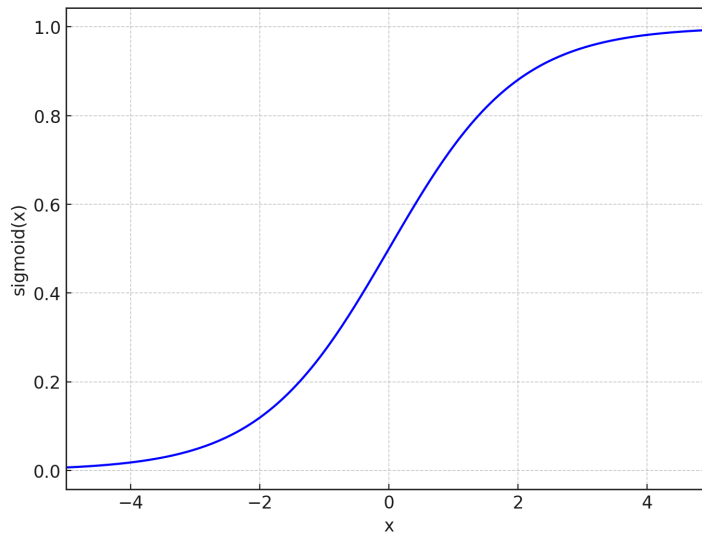


Figure 2.2: Sigmoid activation given by $1/(1 + e^x)$.

Tanh

The hyperbolic tangent activation function, usually called *tanh*, provides the same advantages as the sigmoid. Additionally, it is an antisymmetric activation function and thus the neuron outputs both positive and negative values in the interval $[-1, 1]$. *Tanh* units are more likely to produce outputs, that are on average close to zero [LeC+12]. The *tanh* function is given by

$$f_{\tanh}(z) = \frac{\sinh z}{\cosh z} = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z} - 1}{e^{2z} + 1}. \quad (2.7)$$

Tanh can also be written as a linear combination of the sigmoid activation function

$$f_{\tanh}(z) = 2 \cdot f_{\sigma}(2z) - 1. \quad (2.8)$$

Its derivation is simple as well:

$$f_{\tanh}(z)' = 1 - f_{\tanh}^2(z) = (1 + f_{\tanh}(z))(1 - f_{\tanh}(z)). \quad (2.9)$$

Tanh should be initialized with weights that are not too small because of their very flat error surface near the origin.

While more efficient activation functions such as ReLU are usually preferred in current networks, *tanh* is often used in deep hashing approaches where outputs can be mapped to approximations of binary outputs (either 1 or -1) and are preferred over sigmoid outputs.

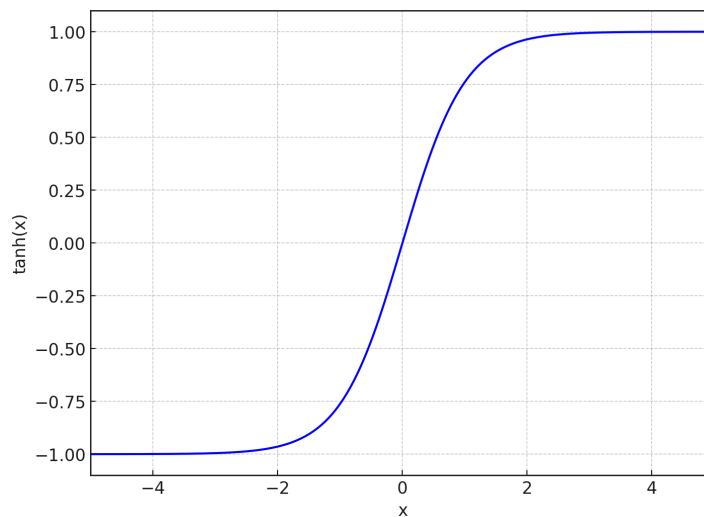


Figure 2.3: *Tanh* activation.

ReLU

A common choice for activation functions in current neural networks is Rectified Linear Units (ReLUs) [NH10]. ReLUs are units that use a very simple activation function, called a rectifier, defined as

$$f_{\text{rec}}(z) = \max(0, z). \quad (2.10)$$

Figure 2.4 shows the activation function for a ReLU.

Until the successful application of ReLUs in deep convolutional neural networks by Krizhevsky et al. [KSH12], which led to 6 times faster convergence, *tanh* activation functions were the common choice for hidden unit activation. Today, ReLUs are widely used in deep neural networks. They have some advantages over sigmoid and *tanh* activations, although they are neither symmetric nor fully differentiable. Non-differentiability at 0 is actually not a real

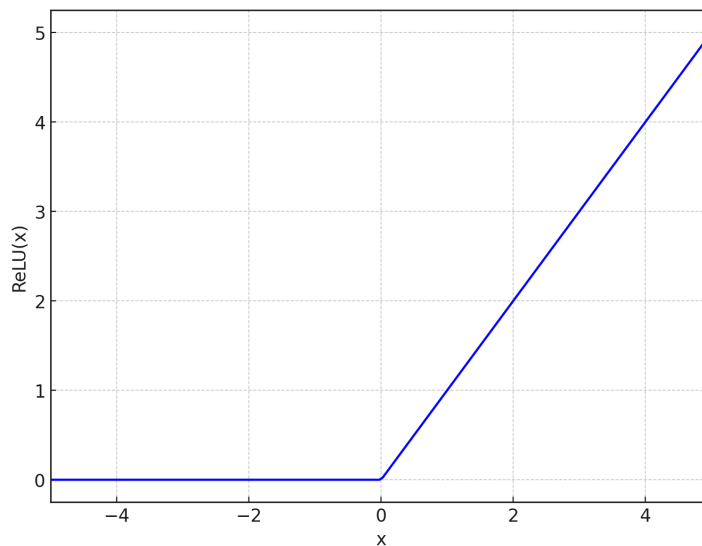


Figure 2.4: ReLU activation.

problem, since it is differentiable at any point arbitrarily close to 0 and a subgradient can be used. A subgradient generalizes the notion of a gradient to functions that are not differentiable at certain points. In the case of ReLU, the subgradient at zero can be chosen as either 0 or 1, or any value in between, allowing backpropagation to proceed even when the exact derivative is undefined.

Obviously, the output of the function, as well as its derivation, is much more efficient to compute (thresholding a matrix of activations at zero). It is precisely this property that makes ReLUs so well suited for deep networks, which often contain hundreds of thousands or millions of neurons, by allowing much faster and more effective training. Furthermore, ReLUs do not suffer from the vanishing gradient problem: unlike sigmoid neurons, they do not saturate, whereas the derivatives of sigmoid functions become very flat far away from 0. Using unbounded ReLUs makes sense in hidden layers, but usually not in the output layer. While sigmoid neurons tend to return values close to 0 or 1, and less often anything in between, ReLUs have a high probability of returning zero. If the learning rate is too high, a large gradient can lead to a weight update that sets all weights to zero, meaning that the neuron will never be active again (referred to as the problem of "dead ReLUs"). A variant of the ReLU that solves this problem is the Leaky ReLU, which has a small negative slope instead of zero for $z < 0$:

$$f_{\text{rec}} = \mathbb{1}_{(s < 0)} \alpha x + \mathbb{1}_{(z \geq 0)} z, \quad (2.11)$$

where α is a small constant and $\mathbb{1}$ is the indicator function, which is one if the condition is true and zero otherwise. The generalization of the ReLU is the maxout neuron [Goo+13b] which can learn the activation function itself during training:

$$f_{\text{maxout}} = (w_1^T z + b_1, w_2^T z + b_2). \quad (2.12)$$

The ReLU is then a special case with $w_1, b_1 = 0$. While maxout neurons do not suffer from irreversible deactivation like ReLUs, they have twice the number of parameters.

The importance of rectifiers for deep neural network models has also been confirmed by He et al. [He+15b], where a modification of ReLU, called Parametric Rectified Linear Unit (PReLU), in combination with an appropriate initialization method outperforms humans on the ImageNet 2012 classification dataset [Rus+15].

Sigmoid Linear Unit

The SiLU (Sigmoid Linear Unit) activation function [EUD18], also known as the Swish function, is defined as follows:

$$\text{SiLU}(x) = x \cdot f_{\sigma} = x \cdot \frac{1}{1 + e^{-x}}. \quad (2.13)$$

This results in a smooth, non-monotonic function as show in Figure 2.5. Unlike ReLU, which blocks negative inputs completely but similar to other improved variants of ReLU, SiLU provides a non-zero gradient for negative inputs. This characteristic counteracts the dead ReLU problem. It has replaced ReLU in many state-of-the art CNN architectures such as EfficientNet [TL19] or newer versions of YOLO [JCQ23].

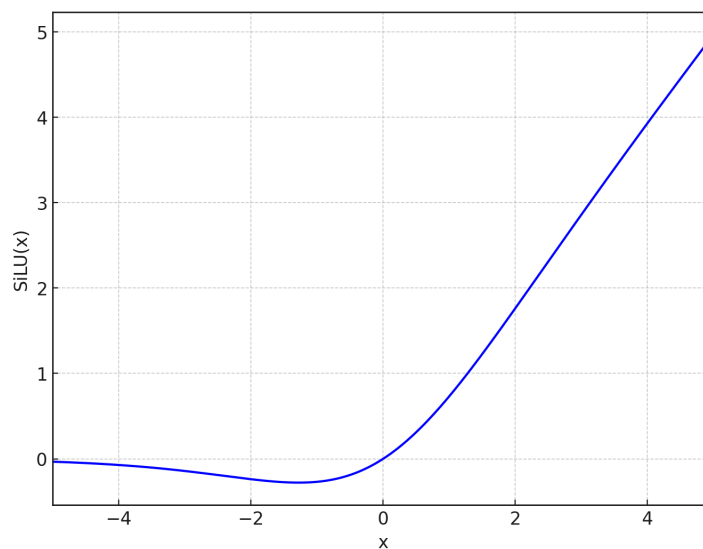


Figure 2.5: SiLU activation.

Softmax

In multi-class problems, where the input should be assigned to one of C classes, a categorical probability distribution of the outputs is often preferred to a single independent activation for each output. For this purpose, the softmax activation function is used in the last layer. It is the generalization of the logistic function for binary problems to multi-class problems. A softmax

layer normalizes the outputs of the units so that they sum to 1. For the j^{th} unit, the softmax is defined as

$$s_j = \frac{e^{z_j}}{\sum_{c=1}^C e^{z_c}}, \quad (2.14)$$

where $s = f_{\text{softmax}}(\mathbf{z})$. The exponentiation ensures positive values for the outputs. In this way, the softmax function interprets each element in \mathbf{z} as holding the unnormalized probabilities (often referred to as logits) of the C classes. Applying the softmax function results in a C dimensional vector of probabilities for a network input \mathbf{x} . This requires that the number of output units in the final layer be equal to the number of classes.

Unlike the sigmoid activation function, the output of a unit in the softmax layer must also consider its neighbors. For the previous activation functions, it was appropriate to look at a single neuron and its derivative, but in the softmax layer, the errors from each of the C outputs should be propagated to each of the C inputs, so the gradient for the j^{th} unit with respect to \mathbf{z} depends on the index of the unit:

$$\frac{\partial s_j}{\partial z_c} = \begin{cases} s_j(1 - s_c) & \text{if } j = c \\ -s_j s_c & \text{else} \end{cases} \quad (2.15)$$

Using the Kronecker delta¹ this simplifies to

$$\frac{\partial s_j}{\partial z_c} = s_j(\delta_{jc} - s_c) \quad (2.16)$$

Typically, the softmax function is used with logarithmic loss (cross-entropy loss) because it works much better than squared error [GB10]. Also, other loss functions than logarithmic loss can lead to a vanishing gradient when an output unit saturates [BGC17]. Softmax is invariant under additive changes of its input vector, so the bias can be omitted, i.e., $\text{softmax}(\mathbf{z}) = \text{softmax}(\mathbf{z} - \mathbf{b})$ (see equation 2.17). If the softmax is calculated directly, numerical stability problems are likely to occur. As a result of the exponentiation, the intermediate terms e^{z_j} and $\sum_{c=1}^C e^{z_c}$ can become very large, making the division of the two numerically unstable. To solve this problem, the following normalization trick can be used:

$$\frac{e^{z_j}}{\sum_{c=1}^C e^{z_c}} = \frac{K e^{z_j}}{K \sum_{c=1}^C e^{z_c}} = \frac{e^{z_j + \log K}}{\sum_{c=1}^C e^{z_c + \log K}}, \quad (2.17)$$

where $\log K$ is usually set to $-\max_j z_j$, which corresponds to shifting the values within the vector so that the highest value is zero.

The softmax function is also used in recurrent neural networks such as LSTMs [HS97] or in attention mechanisms [Vas+17] used in transformer architectures.

¹The Kronecker delta is 1 if the variables are equal and 0 otherwise: $\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$

2.1.3 Loss Functions

In the training phase, the output of a network is compared to the desired output (the target). To measure the discrepancy between these values, a loss function (also called an error function, cost function, or objective function) maps the network outputs to a real value. The higher the value that the loss function returns, the worse it rates the current model. In this section, some common loss functions used in CNNs and other neural networks are introduced and considered for regression, binary, and multi-class problems.

During training, the gradient of the loss function is computed during backpropagation. In practice, it is not necessary for the function to be fully differentiable, and the problem is solved by using the subgradient. For each instance in the training set, the loss is computed separately and then averaged over the N training samples:

$$L = \frac{1}{N} \sum_{i=1}^N L_i \quad (2.18)$$

Usually, a loss function is used together with a certain output layer activation function, which allows to express the gradient in the output layer as the difference between the actual output and the target output. An example of such a pairing are softmax together with logarithmic loss or mean squared error for regression tasks.

Mean Squared Error

A loss function commonly used for regression problems, called the mean squared error loss, is based on the squared error, defined as

$$L_i = \frac{1}{2}(t_i - y_i)^2 \quad (2.19)$$

for the output for a training instance i , where t_i is the target vector and y_i is the output of the network. The multiplication of $\frac{1}{2}$ is for convenience in computing the gradient, which is simply $y_i - t_i$. For N training instances, the mean squared error (MSE) loss is then defined as

$$L = \frac{1}{N} \sum_{i=1}^N (\mathbf{t}_i - \mathbf{y}_i)^2, \quad (2.20)$$

Squaring each term amplifies outliers, i.e., large false outputs are weighted much more heavily.

Cross Entropy Loss

Cross entropy is a measure of the difference between two probability distributions. In information theory, it is used to quantify the information lost when a given distribution Q (the estimated distribution) is used to approximate another distribution P (the true distribution), and is defined as

$$H(P, Q) = - \sum_x p(x) \log q(x). \quad (2.21)$$

Thus, a lower cross entropy value indicates that the estimated distribution Q is closer to the true distribution P .

In neural networks, the cross entropy loss (often called the log loss) is used when the outputs can be thought of as representing independent probabilities (e.g., each output represents a different class). The probabilities $p(x)$ correspond to the target value and the probabilities $q(x)$ correspond to the actual output. If the problem is binary and thus the network has a single output o_i , the loss L_i for a training sample with the true label $y_i \in \{0, 1\}$ is computed as follows:

$$L_i = -y_i \log o_i + (1 - y_i) \log(1 - o_i). \quad (2.22)$$

When there are multiple output units, i.e., the output is a vector, it is common to use the sigmoid if each output represents an independent probability (e.g., in multi-label classification), or the softmax activation if the ground truth is a one-hot encoded vector. For a multi-class problem with K classes, the cross-entropy loss is computed as

$$L_i = - \sum_{k=1}^K L_{ik} = - \sum_{k=1}^K y_{ik} \log(o_{ik}). \quad (2.23)$$

For example in softmax, with output vector $\mathbf{o}_i = f_{\text{softmax}}(\mathbf{z}_i)$, the cross-entropy loss for the output corresponding to the position in the target vector with the positive value ($y_{ik} = 1$) results in

$$L_{ik} = -y_{ik} \log o_{ik} = -\log o_{ik} = -\log \left(\frac{e^{z_{ik}}}{\sum_{c=1}^C e^{z_{ic}}} \right), \quad (2.24)$$

where z is the summed input of the activation function as defined in equation 2.3. For non-target classes k $y_{ik} = 0$ and the term $y_{ik} \log o_{ik}$ becomes zero, so these classes do not contribute directly to the sum in the loss calculation.

If the probability of the true class is very small (close to 0), the loss will go to infinity, and if the probability is very high ($\log 1 = 0$), the loss will go to 0.

With softmax activation, the derivation of the cross-entropy loss at \mathbf{z} results in $\mathbf{y} - \mathbf{t}$: the gradient of the loss with respect to the input of the softmax function \mathbf{z} (the logits) is the difference between the predicted probabilities and the true labels.

Contrastive Loss

Contrastive loss functions are used when labels about similarity are available or in self-supervised learning. While the previous loss functions are typically used on labeled datasets, there are scenarios where distances or orders are available instead of class labels, or can be obtained from the data to train the model in a self-supervised manner. Contrastive losses are used to learn embeddings or representations of the data by considering how similar or dissimilar pairs of samples are.

An example of a contrastive loss is the pairwise contrastive loss [HCL06]. The idea is to ensure that pairs of similar data points (positive pairs) are closer together in the learned feature space, while pairs of dissimilar data points (negative pairs) are further apart. The pairwise contrastive loss for N pairs in a batch is defined as

$$L = \sum_{i=1}^N \left[(1 - y_i) \frac{1}{2} d(a_i, b_i)^2 + y_i \frac{1}{2} \max(0, \text{margin} - d(a_i, b_i))^2 \right], \quad (2.25)$$

where margin is a hyperparameter that measures the minimum distance between two points a_i and b_i that are dissimilar. The binary label y_i for the i^{th} pair is 0 if the pair is similar, 1 if the pair is dissimilar. The distance function d is often the Euclidean distance between the embeddings of the two samples.

Another example of a contrastive loss is the triplet loss, which is often used in tasks such as face recognition and similarity learning. It is computed over a batch of N samples and is defined as

$$L = \sum_{i=1}^N \max(d(a_i, p_i) - d(a_i, n_i) + \text{margin}, 0), \quad (2.26)$$

where a_i is the anchor sample in the i^{th} triplet, p_i is the positive example, and n_i is the negative example. The distance between the samples is measured by a distance function $d(x, y)$, which can be the Euclidean distance. The margin is a hyperparameter that defines how far apart the distances between the anchor-positive and anchor-negative pairs should be. It prevents the model from trivial solutions and encourages it to learn useful embeddings. The triplet loss function aims to ensure that the distance between the anchor and the positive example is less than the distance between the anchor and the negative example by at least the margin. If this condition is satisfied, the loss is zero.

2.1.4 Optimization

During training, the network learns by iteratively updating its weights to achieve a lower loss. This is done by error backpropagation combined with gradient descent.

Error Backpropagation

The backpropagation algorithm [RHW88] performs parameter updates for all parameters of the network. The amount of parameter adjustment depends on their individual contribution to the network error.

After the forward pass, in which one or more training examples are passed through the network, the discrepancy between the output of the network and the desired output is measured by a loss function. Then, all parameters in the network are adjusted to make the network perform better in the next iteration, i.e., to achieve a lower loss next time. Essentially, the parameters of

the network are changed towards a (usually local) minimum of the loss function. Since it is computationally infeasible to compute a global minimum for a large model on the entire data set, backpropagation is usually combined with the gradient descent method.

Gradient descent finds a minimum in parameter space by iteratively taking steps proportional to the negative of the gradient of the function until it no longer yields a relevant improvement. The combination of parameters that minimizes the loss function is considered the solution to the learning problem. Figure 2.6 illustrates gradient descent for a function with two parameters.

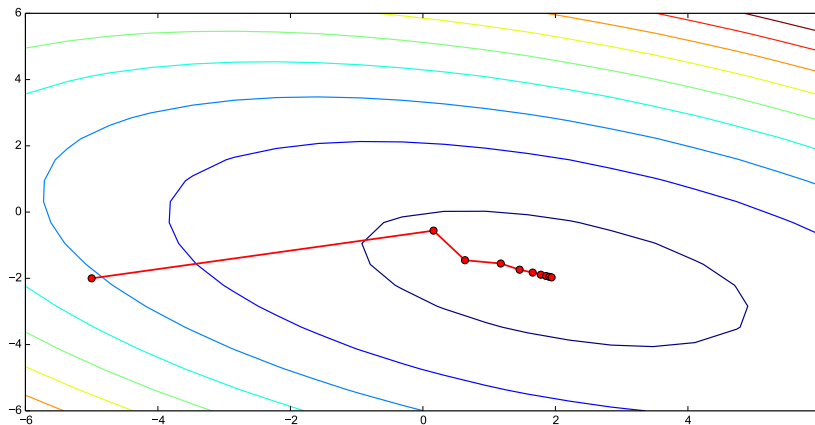


Figure 2.6: Gradient descent on a 3-dimensional error surface.

The training process consists of several iterations. In each iteration, there are two subsequent steps. First, the gradient for the whole network is computed efficiently by error backpropagation, and then the parameters are adjusted. In larger networks, the second step of weight adjustment is usually realized by optimization schemes such as Stochastic Gradient Descent (SGD) or Adam [KB15], which offers improvements like adaptive learning rates.

To obtain the overall gradient of a network, it is necessary to compute the gradient of the loss function with respect to each parameter of the network. A multi-layer network represents a potentially very deeply nested function, i.e., each layer can be thought of as a function in its own right whose output depends on the output of the previous layer. What the backpropagation algorithm basically does is to apply the chain rule repeatedly, layer by layer, starting from the output layer. When computing the local gradient (for example, at a given layer), two functions g and f are considered. Their composition is given by $F(x) = f(g(x))$, i.e., the local function $g(x)$ is followed by $f(x)$, e.g., the next layer towards the output of the network. The chain rule then expresses the derivative of the composition $F'(x) = f'(g(x))g'(x)$, which can be equivalently written as

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx'} \quad (2.27)$$

where $z = f(y)$ and $y = g(x)$, so z is a function of x . From the multivariable chain rule it follows for $f : \mathbb{R}^m \rightarrow \mathbb{R}$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}$ with $y = f(u)$ and $u = g(x) = (g_1(x), \dots, g_m(x))$ that

$$\frac{\partial y}{\partial x} = \sum_{\ell=1}^m \frac{\partial y}{\partial u_\ell} \frac{\partial u_\ell}{\partial x}. \quad (2.28)$$

In the case of neural networks, one is interested in the gradient of the loss function L with respect to the parameters in the network. There are two cases to consider: Either a unit is in the output layer, or the unit is in the hidden layer. In the first case, the gradient of the loss function can be computed directly. However, if a unit is in a hidden layer, the gradients of its input weights also depend on the gradients of the upper layers. Thus, backpropagation starts by computing the gradient of the loss with respect to the weights connecting the output units to units in the last hidden layer. A unit in the output layer is identified by the index j , units in

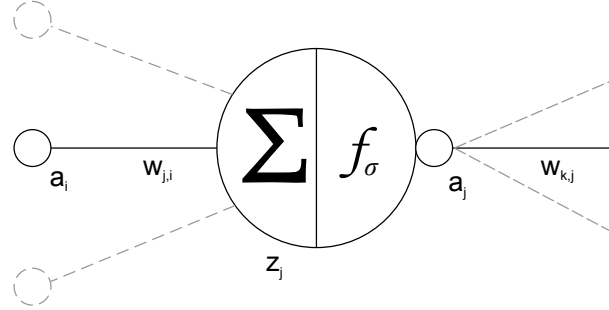


Figure 2.7: Unit j sums the weighted outputs of units i of the previous layer to z_j . After applying the nonlinearity, a_j serves as the input for the units k in the following layer.

the previous layer by the index i . Applying the chain rule then results in

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial a_j} \underbrace{\frac{\partial a_j}{\partial z_j}}_{\delta_j} \frac{\partial z_j}{\partial w_{ji}}. \quad (2.29)$$

For the output layer, the derivative with respect to w_{ij} , the weight between unit i and unit j , can be computed easily. For example, in the case of mean squared error loss, with $L = \frac{1}{2} \sum_j (y_j - t_j)^2$, it follows that

$$\frac{\partial L}{\partial a_j} = \frac{\partial L}{\partial y_j} = (y_j - t_j), \quad (2.30)$$

where $y_j \in \mathbf{y}$ are the outputs of the network and $t_j \in \mathbf{t}$ are the values of the destination vector. The second factor in Equation 2.29 simply expresses the derivative of the activation function, which in the case of sigmoid activation is given by f_σ :

$$\frac{\partial a_j}{\partial z_j} = \frac{\partial f_\sigma(z_j)}{\partial z_j} = f'_\sigma(z_j) = a f_\sigma(z_j)(1 - f_\sigma(z_j)) = a_j(1 - a_j). \quad (2.31)$$

To complete the computation of the gradient for the output units, the sum of the weighted inputs z_j is differentiated with respect to a specific input weight, w_{ji} , and thus only the input associated with that weight will have a non-zero derivative:

$$\frac{\partial z_j}{\partial w_{ji}} = a_i. \quad (2.32)$$

Plugging the equations 2.38 and 2.32 into 2.29, we get the following for the output layer

$$\frac{\partial L}{\partial w_{ji}} = \delta_j a_i. \quad (2.33)$$

The calculation of the gradients for the hidden layers is different, since the gradients depend on higher layers. In the following, we will examine the gradient with respect to w_{ji} . As shown in figure 2.7, unit i is located in the previous layer of unit j . Units with index k are in the layer above j .

Similar to Equation 2.29, the gradient of the loss with respect to the weights in the hidden layers is also given by

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial a_j} \underbrace{\frac{\partial a_j}{\partial z_j}}_{\delta_j} \frac{\partial z_j}{\partial w_{ji}}. \quad (2.34)$$

While the last two factors are computed analogously to those in the output layer (see Equations 2.31 and 2.32), the first factor is now a recursive expression. The loss function for unit j now depends on the units k in the subsequent layer:

$$\frac{\partial L(a_j)}{\partial a_j} = \frac{\partial L(z_u, z_v, \dots, z_w)}{\partial a_j}, \quad (2.35)$$

where $K = u, v, \dots, w$ are units in the subsequent layer which are receiving input from unit j . The outgoing connections from unit j to units in K lead to a sum of gradients due to the multi-variable chain rule (Equation 2.28):

$$\begin{aligned} \frac{\partial L(a_j)}{\partial a_j} &= \sum_{k \in K} \frac{\partial L}{\partial a_k} \frac{\partial a_j}{\partial z_k} \frac{\partial z_k}{\partial a_j} \\ &= \sum_{k \in K} \underbrace{\frac{\partial L}{\partial a_k} \frac{\partial a_j}{\partial z_k}}_{\delta_k} w_{kj}, \end{aligned} \quad (2.36)$$

where the δ_k correspond to the errors already calculated for the units $k \in K$ in the subsequent layer. Eventually, the above equations result in

$$\frac{\partial L}{\partial w_{ji}} = \underbrace{\sum_{k \in K} [\delta_k w_{kj}] f'(z_j)}_{\delta_j} a_i, \quad (2.37)$$

where δ_j is the error of unit j .

Gradient Descent

When using gradient descent in the next stage, weight updates are performed with the learning rate η by adding

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \delta_j, \quad (2.38)$$

where

$$\delta_j = \frac{\partial l}{\partial a_j} \frac{\partial a_j}{\partial z_j} = \begin{cases} (a_j - t_j) f(z_j) & \text{if } j \text{ is an output unit,} \\ (\sum_{k \in K} \delta_k w_{kj}) f(z_j) & \text{if } j \text{ is a hidden unit.} \end{cases} \quad (2.39)$$

Only gradients with respect to the weights have been considered here, but gradients for the biases are obtained in a similar way. Since all computations are local, they can be processed efficiently in parallel. For this purpose it is convenient to write the forward and backpropagation equations in matrix notation. For an output layer L , the errors are defined as

$$\mathbf{d}_L = \mathbf{t} - \mathbf{y}_L. \quad (2.40)$$

The errors for the layers $l = L - 1, L - 2, \dots, 1$ are given by

$$\mathbf{d}_l = (\mathbf{W}_{l+1}^T \mathbf{d}_{l+1}) \circ f_l(\mathbf{z}_l), \quad (2.41)$$

where \circ is the Hadamard product. The weight and bias updates are then computed as

$$\Delta \mathbf{W}_l = \mathbf{d}_l \mathbf{a}_{l-1}^T \quad \text{and} \quad \Delta \mathbf{b}_l = \mathbf{d}_l. \quad (2.42)$$

This example assumes fully connected layers. For sparsely connected layers (such as convolutional layers), the equations are different. Although there are other ways to perform optimization, gradient descent remains the most widely used method in neural networks.

Stochastic Gradient Descent

Summing and then averaging the loss over the entire training set and then performing the weight updates is called full-batch learning. However, this approach becomes very expensive for large datasets. If the dataset contains millions of examples, learning would be slowed down significantly by considering all training samples in each iteration. In practice, the gradient is usually computed for a mini-batch. A mini-batch consists of m examples that are randomly selected from the training set for each gradient descent learning step. The number of images in a mini-batch is typically in the range of about 10-1000 examples, depending on the problem, and is a hyperparameter, although it is usually not cross-validated. For current neural network architectures, the batch size may also depend on the memory limitations of a GPU (although gradients can be accumulated over several batches before performing a weight update).

Originally, stochastic gradient descent (SGD), sometimes called online gradient descent, is a special case of minibatch gradient descent. The batch then contains only a single example. However, this approach is not widely used because it is much more efficient to evaluate the gradient for m examples than to evaluate the gradient m times. The term stochastic gradient descent is now used synonymously with mini-batch gradient descent.

Momentum-Based Gradient Descent

The simplest form of weight updating is simply to add the negative gradient to the weights at a fixed learning rate (see Equation 2.38). While gradient descent is the most commonly used method for updating weights, there are other approaches and variations of it. Improvements in the rate of convergence can be achieved by using momentum-based methods. The motivation for momentum comes from a physical analogy. The loss can be interpreted as the height of a hilly landscape and the parameter vector as a ball rolling on that landscape. While in standard gradient descent the position depends only on the fixed learning rate, the parameter vector now depends on a velocity and a friction term. Integrating a velocity leads to the following equations for gradient descent updates:

$$v_j^{\text{new}} = \mu v_j^{\text{old}} - \eta \delta_j \quad (2.43)$$

$$w_j^{\text{new}} = w_j^{\text{old}} + v_j^{\text{new}}, \quad (2.44)$$

where the hyperparameter $\mu \in [0, 1]$ controls the amount of friction, misleadingly called the momentum coefficient. A low value of 0 for μ corresponds to a suppression of the velocity term, while a value of 1 means a strong effect of velocity. In cross-validation, μ should be set to values such as 0.5, 0.9, 0.95, and 0.9.

A variation of the momentum method is called Nesterov Accelerated Momentum (NAG) [Nes83] and can further improve gradient descent [Sut+13]. Instead of looking at the current position x of the parameter vector, the idea is to look ahead to the position that μv would push the parameter vector to, i.e., compute the gradient at that future position $x + \mu v$.

Other less common approaches include second-order methods based on Newton's method, which use the inverse of the Hessian matrix to update the weight according to local curvature. However, for large neural networks, computing the Hessian matrix requires a lot of memory.

Adaptive Learning Rate Methods

Another group of optimization approaches is based on the idea of assigning an individual learning rate to each parameter. For example, the Adaptive Gradient Algorithm (AdaGrad) [DHS11] adapts the learning rate so that weights that receive large gradients have their learning rate decreased, while weights that receive small updates have their learning rate increased. RMSprop (Root Mean Square Propagation) is another optimization algorithm. It adjusts the learning rate for each parameter by dividing the gradient by a running average of its recent size, solving the vanishing or exploding gradient problem common to deep neural networks.

A widely used optimization algorithm is ADAM (Adaptive Moment Estimation) [KB15]. It combines the advantages of AdaGrad and Root Mean Square Propagation (RMSProp).

ADAM maintains an exponential running average of the gradients similar to SGD with momentum. It also maintains an exponentially decaying average of the squared gradients. The decay rates β_1 and β_2 are set close to 1 (e.g. $\beta_1 = 0.9$ and $\beta_2 = 0.999$). The initial learning rate α is another hyperparameter.

Initially, at $t = 0$, the first moment vector $m_0 = 0$ and the second moment vector $v_0 = 0$.

For each iteration t , the first and second moments are updated as follows

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \quad (2.45)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \quad (2.46)$$

where the gradient g_t with respect to the loss function is obtained by backpropagation. Since the momentum vectors are initialized with zeros, they are biased towards zero in the early time steps and at high decay rates. Therefore, a bias correction is applied to the momentums:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (2.47)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (2.48)$$

Finally, the parameters are updated:

$$w_{t+1} = w_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}, \quad (2.49)$$

where ϵ is a small constant to ensure numerical stability and w is the parameter vector.

2.1.5 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a specialized type of neural network, particularly suited to image data. They were introduced by Fukushima [Fuk80] and later successfully applied to the MNIST dataset of handwritten digits [LeC+98]. They are inspired by the biological image processing of the primate brain, which is hierarchically organized in layers of increasing processing complexity [Ben+09]. Basically, CNNs use convolutions instead of general matrix multiplication (as in standard feed-forward neural networks). Convolutions allow a localized view on the input data and are very efficient since they share locally connected and thus sparse parameters. This section first introduces the mathematical operation of convolution. Then, the architecture of convolutional networks is explained. CNNs usually consist of several successive groups of stacked layers. There are three types of layers that are part of a typical convolutional neural network. First, there is a convolutional layer, in most cases followed by a pooling layer and finally one or more fully connected layers (the same as the hidden layers explained earlier when introducing regular feed forward neural networks).

Convolution

An illustrative application of a 2D convolution can be found in image processing: When applying a particular filter to an image, the filter function (kernel) is applied to the image, centered at each position (i.e., pixel). For example, in the case of Gaussian blur, the kernel is a discretized Gaussian function. This fixed-size kernel is then shifted across the entire image, pixel by pixel. For each pixel, the following steps are performed: Each pixel of the original image within the range of the filter is multiplied by the value of the filter for that filter position. These values are then added together to form the new value for that particular pixel (which is in the center of the current filter position). Figure 2.8 shows a Gaussian blur filter mask of size 3×3 and its effect.

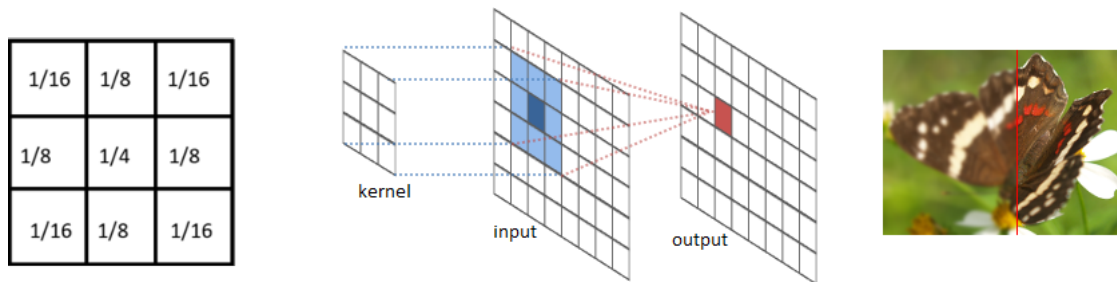


Figure 2.8: Example of a 2-D convolution: 3×3 Gaussian blur filter mask (left), convolution applied to one pixel (center), sample output with Gaussian blur applied to half of the image (right).

In general, convolution is an operation on two functions f and g . It is defined as

$$(f * g)(t) = \int f(a)g(t - a)da \quad a, t \in \mathbb{R}, \quad (2.50)$$

where f is the input, g is the kernel, and the output is called the feature map in the context of CNN. In practice, time is discretized and t takes on integer values. For the discrete case, the convolution is defined as follows

$$(f * g)[t] = \sum_{a=-\infty}^{\infty} f[a]g[t - a], \quad a, t \in \mathbb{Z} \quad (2.51)$$

In the case of convolutional neural networks, the input is an array of data and the kernel is an array of parameters learned by backpropagation. However, these parameters are not learned separately for each location, but rather a single set of parameters is learned for all locations. This is called parameter sharing and will be explained in detail in the next section.

The first layer in a deep CNN is an input layer, which usually accepts fixed-size images with three RGB channels. This layer is typically followed by convolution and pooling layers. A convolution layer is followed by an activation layer, for example a ReLU. There may be one or more convolution layers with corresponding activation layers. In most cases, a pooling layer is placed between these groups. Finally, there is usually at least one fully connected layer. Recent CNN architectures often include batch normalization layers after convolution layers or before activation functions to improve training stability and training speed.

Convolutional Layer

In the context of neural networks, the convolution operation differs slightly from the usual definition of the standard discrete convolution operation. The term actually refers to an operation that consists of many convolutions in parallel. A convolution in a CNN extracts many types of features at many locations. In general, the parameters of a convolutional layer consist of a set of filters. Figure 2.9 shows the organization of an input layer, a filter, and a convolutional layer. In the forward pass, the filter is convolved over the input volume. More precisely, convolving here means computing the dot product between the entries of the filter and the input. This computation is exactly what characterizes a unit in a regular feed forward neural network, but here it only considers a small region of the input. In fact, the filter entries correspond to the weights of the input. Since the filter slides over the entire input volume

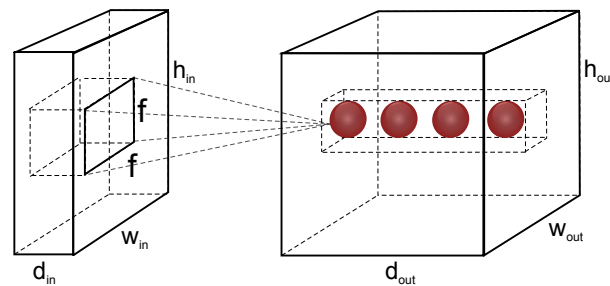


Figure 2.9: A filter of size $f \times f$ convolves over a 3-dimensional input layer of size $w_{in} \times h_{in} \times d_{in}$ and produces a slice of size $w_{out} \times h_{out}$. The filters are the input weights of the output layer units, and their number corresponds to the depth d_{out} .

along its width and height, there will be an output for each location. Together, these outputs of a given filter form a 2-dimensional *activation map*. It is important to note that the filter weights are the same for each output, i.e., the parameters are shared by units within the same activation map. If the input is large compared to the filter, this leads to an enormously reduced number of parameters. In practice, there is usually more than one filter: in a convolutional layer, the resulting activation maps (also called feature channels) are stacked, and the depth of the output volume corresponds to the number of filters or activation maps. Figure 2.10 shows a typical set of learned first-layer filters.

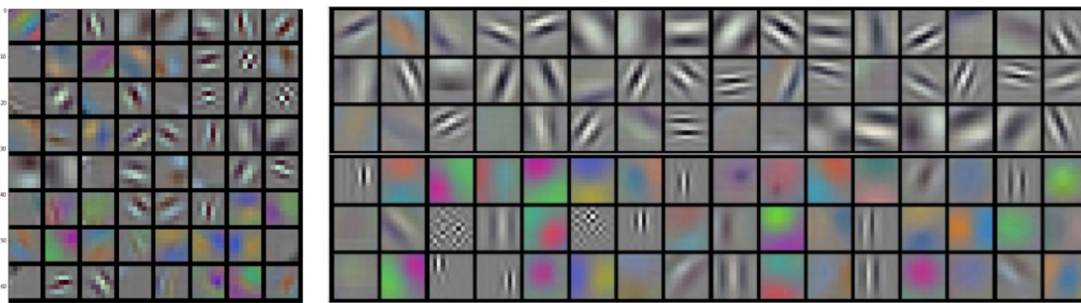


Figure 2.10: Visualization of filters (input weights to the first convolutional layer) of GoogLeNet [Sze+15] of size 7×7 (left) and AlexNet [KSH12] of size 11×11 (right).

The spatial extent $f \times f$ of a filter is called the receptive field. While f can be adjusted, its depth depends on the previous layer, since it is required that the depth of the filter be equal to the depth d_{in} of its input. Thus, the connections are local along the width and height, but full along the depth of the input volume. Typically, a receptive field is no larger than 7×7 .

While f determines the number of inputs to consider for a single output, three other hyperparameters specify the extent of the output volume. First, the depth d_{out} , i.e., the number of filters k , can be chosen freely, since it only affects the following layer. The remaining two hyperparameters, stride s and zero-padding p , together with the width w_{in} and height h_{in} of the input volume, control the width w_{out} and height h_{out} of the output volume. When interpreting the filter as sliding over the input volume, the stride is the distance between any specific position of a filter at two consecutive steps. In other words, the stride specifies how many units in the input volume are skipped before a unit is multiplied by the same filter entry and assigned to the next unit in the activation map. For example, if the stride is 1, the next receptive field will overlap a lot and produce a large activation map. Conversely, a higher stride will reduce both the overlap of the receptive field and the size of the output activation map. So, if $s \geq f$, the receptive fields will not overlap. Assuming $f > 1$ (a receptive field larger than 1×1) in combination with a stride of 1, the spatial dimension $w_{in} \times h_{in}$ of the input. More precisely, the width of the output volume will be $w_{out} = w_{in} - \lfloor \frac{f}{2} \rfloor$. This motivates zero padding: by spatially padding the input at the border, it is possible to generate output volumes that have the same spatial size as their input volumes. Setting the padding to $p = \frac{f-1}{2}$ when $s = 1$ ensures that the input and output volumes are of the same spatial size. In general, zero padding allows to directly control the spatial size of the output volumes to match s and f .

In summary, the dimension of the convolutional layer input is defined as $w_{in} \times h_{in} \times d_{in}$. It produces an output volume of dimension $w_{out} \times h_{out} \times d_{out}$. The relationship between the hyperparameters f , s , and p is reflected in the following equation, which calculates the number of units along the width of the output volume:

$$w_{out} = \left\lfloor \frac{w_{in} - f + 2p}{s} \right\rfloor + 1, \quad (2.52)$$

where it must be ensured that $w_{out} \in \mathbb{Z}$. Similarly, the number of units h_{out} resulting from convolving over height can be calculated. Thus, the number of units in an activation map is $w_{out} \cdot h_{out}$ and the total number of units in the convolutional layer is $(w_{out} \cdot h_{out}) \cdot k$, with the hyperparameter $k = d_{out}$. Each filter corresponds to $f \cdot f \cdot d_{in}$ weights. So, in total there are $(f \cdot f \cdot d_{in}) \cdot k$ weights and k biases that belong to a convolutional layer.

When implemented, the convolution is performed by a large matrix multiplication, for example on a GPU. The filters are expanded to $f \cdot f \cdot d_{in}$ columns and the locations of the receptive fields to $w_{out} \cdot h_{out}$ rows. Thus, the resulting matrix of stretched inputs is of dimension $(f \cdot f \cdot d_{in}) \times (w_{out} \cdot h_{out})$. The weights of the convolutional layer are also stretched, resulting in a weight matrix of dimension $k \times (f \cdot f \cdot d_{in})$. The resulting multiplied matrix is $k \times (w_{out} \cdot h_{out})$ is then reshaped to the dimensions $w_{out} \times h_{out} \times k$.

Formally, the summed input of a unit in the convolutional layer can be written as

$$z_{ijk}^\ell = \sum_{a=1}^f \sum_{b=1}^f \sum_{c=1}^{d_{in}} w_{abck} a_{(i+a-1)(j+b-1)c}^{\ell-1} \quad (2.53)$$

where $i \in \{1, \dots, w_{\text{out}}\}$, $j \in \{1, \dots, h_{\text{out}}\}$, $k \in \{1, \dots, d_{\text{out}}\}$ and ℓ is the convolutional layer. Including the stride, it follows that

$$z_{ijk}^{\ell} = \sum_{a=1}^f \sum_{b=1}^f \sum_{c=1}^{d_{\text{in}}} w_{abck} a_{(s \cdot i + a - 1)(s \cdot j + b - 1)c}^{\ell-1} \quad (2.54)$$

with stride s . As usual, a nonlinear activation function is applied to the output of the units. In case of CNN, the convolution is often followed by a rectifier activation (implemented as a ReLU layer). The output of a single unit is then defined as

$$a_{ijk}^{\ell} = f_{\text{rect}}(z_{ijk}^{\ell}). \quad (2.55)$$

In the derivation of the backpropagation equations, stride and padding are omitted for simplicity. In the backward pass, the derivative of the loss with respect to the input of the following layer $\frac{\partial L}{\partial a_{ijk}^{\ell}}$ has already been computed in the previous step. Next, the derivative with respect to the weights is needed. Since the weights are shared, a weight w_{abck} is connected to all z_{ijk}^{ℓ} . Thus, due to the multivariate chain rule (Equation 2.28), we get

$$\frac{\partial L}{\partial w_{abck}^{\ell}} = \sum_{i=1}^{(w_{\text{in}}-f+1)} \sum_{j=1}^{(h_{\text{in}}-f+1)} \sum_{k=1}^{d_{\text{out}}} \frac{\partial L}{\partial z_{ijk}^{\ell}} \frac{\partial z_{ijk}^{\ell}}{\partial w_{abck}^{\ell}}. \quad (2.56)$$

Since $\frac{\partial z_{ijk}^{\ell}}{\partial w_{abck}^{\ell}} = a_{(i+a-1)(j+b-1)c}^{\ell-1}$, Equation 2.56 can be rewritten as

$$\frac{\partial L}{\partial w_{abck}^{\ell}} = \sum_{i=1}^{(w_{\text{in}}-f+1)} \sum_{j=1}^{(w_{\text{in}}-f+1)} \sum_{k=1}^{d_{\text{out}}} \underbrace{\frac{\partial L}{\partial z_{ijk}^{\ell}}}_{\delta^{\ell}} a_{(i+a-1)(j+b-1)c}^{\ell-1}. \quad (2.57)$$

As with normal backpropagation, the computation of δ^{ℓ} is straightforward:

$$\delta^{\ell} = \frac{\partial L}{\partial z_{ijk}^{\ell}} = \frac{\partial L}{\partial a_{ijk}^{\ell}} \frac{\partial a_{ijk}^{\ell}}{\partial z_{ijk}^{\ell}} = \frac{\partial L}{\partial a_{ijk}^{\ell}} \frac{\partial}{\partial z_{ijk}^{\ell}} \left(f_{\text{rect}}(z_{ijk}^{\ell}) \right) = \frac{\partial a_{ijk}^{\ell}}{\partial z_{ijk}^{\ell}} f'_{\text{rect}}(z_{ijk}^{\ell}). \quad (2.58)$$

Now the gradient with respect to the weights can be used for weight updates in the convolutional layer. In addition, the error must be propagated back to the previous layer $\ell - 1$. The following equation assumes that the upper and left edges of the convolutional layer ℓ are padded with f zeros. Again, since each $a_{ijk}^{\ell-1}$ is connected to a patch of size $f \times f$ in each activation map, i.e. a volume of size $f \times f \times d_{\text{out}}$, it follows from the multivariate chain rule that

$$\begin{aligned} \frac{\partial L}{\partial a_{ijk}^{\ell-1}} &= \sum_{a=1}^f \sum_{b=1}^f \sum_{c=1}^{d_{\text{out}}} \frac{\partial L}{\partial z_{(i-a+1)(j-b+1)c}^{\ell}} \frac{\partial z_{(i-a+1)(j-b+1)c}^{\ell}}{\partial a_{ijk}^{\ell-1}} \\ &= \sum_{a=1}^f \sum_{b=1}^f \sum_{c=1}^{d_{\text{out}}} \frac{\partial L}{\partial z_{(i-a+1)(j-b+1)c}^{\ell}} w_{abck}. \end{aligned} \quad (2.59)$$

This is also a convolution using the same filter (i.e. weights w_{abck}), but with both spatial axes flipped.

A popular variant of convolutional layers are dilated convolutions [YK15], also called atrous convolutions. They are designed to capture broader contextual information without increasing the number of parameters. While in a regular convolution the filter is applied to the input in contiguous manner (each element of the filter is multiplied by adjacent elements of the input), in dilated convolutions, the filter is applied over an area larger than its actual size by skipping input values at a certain rate (dilation factor r . With $r > 1$, spaces are introduced between the kernel elements. Dilated convolutions are especially useful when information of larger areas is important, e.g., in semantic image segmentation.

In many CNNs, a pooling layer is inserted periodically in between successive convolutional layers.

Pooling Layer

A convolutional layer can optionally be followed by a pooling layer. The pooling layer takes small rectangular blocks from the convolutional layer and subsamples them to produce a single output from each block. Unlike the convolutional layer, the pooling is performed for each depth slice of the input layer (i.e., activation map) individually. This preserves the depth of the output volume.

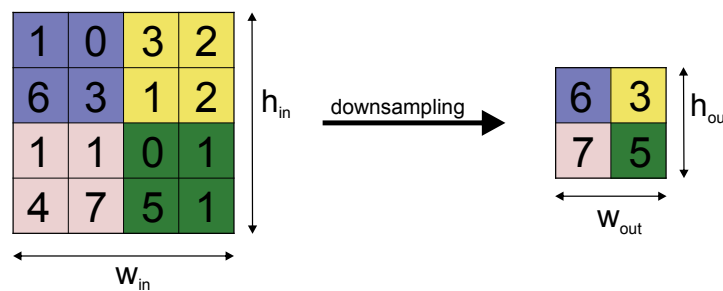


Figure 2.11: Max-pooling with step 2 and step 2 downsamples the input.

There are two hyperparameters to adjust: stride s and spatial extent f . From a convolutional layer of size $w_{in} \times h_{in} \times d$, pooling results in a volume with width $w_{out} = \frac{w_{in}-f}{s} + 1$ and height $h_{out} = \frac{h_{in}-f}{s}$, while the depth is still d (see Figure 2.11).

Although there are other options for the pooling function, such as average pooling or L2 norm pooling, max pooling is most commonly used in CNN. That is, the highest activation in the spatial extent of the filter is selected. The output of a max pooling layer indicates whether a feature was present in a region of the convolutional layer, but not exactly where.

In practice, two max pooling settings are usually used. The more commonly used form applies a 2×2 filter with a step of 2 to the input volume, discarding 75% of the activations. The second type of max pooling uses a 3×3 filter with stride 2, called overlapping pooling. It turned out that larger receptive field sizes were too destructive.

In backpropagation, the gradient is propagated only to the unit that had the highest activation value in the forward pass, i.e., the derivative of the loss with respect to the input of the previous layer $\ell - 1$ is defined as

$$\frac{\partial L}{\partial a_{(i+a)(j+b)k}^{\ell-1}} = \begin{cases} 0 & \text{if } a_{ijk}^{\ell-1} \neq a_{(i+a)(j+b)k}^{\ell-1} \\ \frac{\partial L}{\partial a_{ijk}^{\ell-1}} & \text{else} \end{cases}. \quad (2.60)$$

This can be implemented efficiently by keeping the index of the maximum in the forward pass.

After several convolution and max-pooling layers, most CNNs have at least one fully connected layer before the output layer. One of the advantages of a CNN architecture is that there are many more connections than weights involved, and thus a kind of regularization is already introduced by the architecture. Furthermore, a certain degree of translation invariance is automatically provided by the shared weights. Since images contain hierarchical structures (e.g., a car consists of wheels, a wheel consists of edges, etc.), filters in convolutional layers, which can be interpreted as more and more abstract detectors the deeper the layers are, seem to be a suitable approach to handle image or similar data. Finally, the local connectivity induced by the receptive field reflects that neighboring pixels are correlated in images.

Learning Rate

If the learning rate is too high, the loss will not converge. Conversely, if it is too low, learning will be slower than necessary. For deep networks, it is common to choose an initial learning rate that decreases over time during the training process. There are several approaches to learning rate annealing. In step decay, the learning rate is reduced by a factor after some epochs. How to choose the interval of epochs (the step size) and the factor depends on the nature of the problem and the model. A common heuristic is to reduce the learning rate by a constant when the validation error stops improving. The learning rate with exponential decay is defined as $\eta = \eta_0 e^{-kt}$ with the hyperparameters η_0 and k and the number of iterations t . For $1/t$ decay, the learning rate is $\eta = \frac{\eta_0}{1+kt}$. Another frequent choice for a learning rate schedule is cosine annealing [LH16], where the learning rate decreases following a cosine curve.

In addition, adaptive learning rate methods such as Adam are often used. They start with an initial learning rate and adjust the learning rate dynamically during training.

2.1.6 Regularization

When a model with many parameters is trained on a comparatively small training data set, the model usually overfits. That is, the model achieves a small error on the training data, but when applied to new examples, the error is large. In the worst case, the model has learned the training examples but does not generalize. To prevent a neural network from overfitting, some kind of regularization must be applied. There are several approaches, of which L2 regularization and dropout are often used in the field of deep convolutional neural networks.

L1 and L2 Regularization

L2 regularization penalizes the squared magnitude of all parameters. This is done by modifying the loss function to produce a higher loss for large weights. For each weight in the network, the following regularization term is added

$$\frac{1}{2}\lambda w^2, \tag{2.61}$$

where λ controls the influence of the regularization and the factor $\frac{1}{2}$ is just for simplifying the derivation. The effect of L2 regularization is to prevent the network from learning large parameters. Since the inputs are multiplied by the weights, the network generally achieves lower loss by using all of its inputs a little, rather than some of its inputs a lot. Thus, over time, each weight decays linearly toward 0. Usually, L2 regularization is not applied to penalize the biases. That is, it usually has little effect on the results, and large biases make it easier for units to saturate, which can sometimes be desirable. In L1 regularization, the following term is added to the loss:

$$\lambda|w|. \tag{2.62}$$

In contrast to the L2 regularization, this leads to sparse weight vectors, i.e. many weights are close to zero and only a few are large. The combination of both L1 and L2 loss is called elastic net regularization [ZH05]. Another type of regularization are max-norm constraints, which introduce an absolute upper bound c for weight vectors in a network, i.e. each weight vector must satisfy $\|\mathbf{w}\|_2 \leq c$.

Dropout

The above regularization methods can be used together with a simple and powerful regularization technique called dropout [Sri+14]. The idea of dropout is to keep a unit active with some probability p and otherwise deactivate it during training (with probability $1 - p$). When a unit is deactivated, it neither receives input from neighboring units nor does it receive output from previously connected units. Thus, in each iteration, only a subset of units is active, as shown in Figure 2.12. This can be interpreted as training a different network in each iteration, each overfitting in a different way. Then averaging over all these networks regularizes the full network.

Applying dropout during training means that the expected output of a unit will be $pa + (1 - p)0$. Therefore, when using the network for inference, the output a must be adjusted by multiplying it by p to keep the same expected output. However, since time is more important during inference, the outputs are scaled at training time. Another explanation for the regularization effect of dropout is that it counteracts complex co-adaptations of units by preventing units from relying on the presence of particular other units, and thus forcing them to learn more robust features that are useful together with many different random subsets of the other units [KSH12]. Dropout is most effective when taking relatively large steps in the parameter space [Goo+13a]. A related method is called DropConnect [Wan+13b], where a random set of weights is set to 0 during a forward pass.

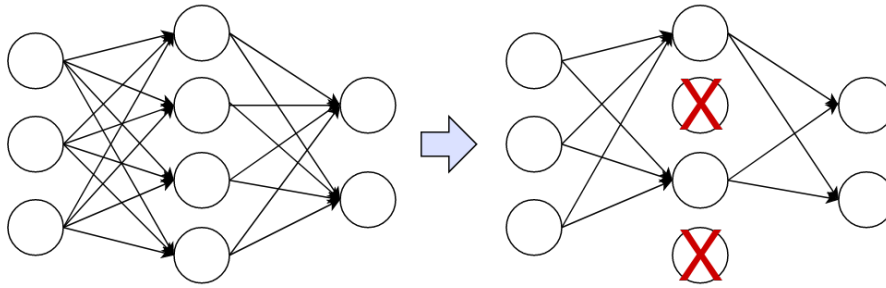


Figure 2.12: Dropout randomly deactivates neurons in training.

Batch Normalization

Batch normalization [IS15] addresses the problem of internal covariate shift, where the distribution of each layer's inputs changes during training as the parameters of previous layers change. It therefore normalizes the inputs of each layer for each minibatch by adjusting the inputs across the minibatch to have a mean of 0 and a standard deviation of 1.

The input x in a batch is normalized as follows

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (2.63)$$

This normalized input is used together with two learnable parameters; a scaling factor γ and a shift parameter β , allowing the network to undo the normalization if it is more beneficial during training. The output y of a batch normalization layer is then given by

$$y = \gamma \hat{x} + \beta \quad (2.64)$$

During training, the mean and variance are computed for each mini-batch. During inference, an exponential moving average computed during training is used instead. In addition to regularization, batch normalization improves gradient flow, thereby speeding up training and making the network less sensitive to initial parameters.

Finally, unsupervised pre-training can also have a regularizing effect.

Data Preprocessing and Weight Initialization

One of the most important preprocessing steps when using images as input for neural networks is mean subtraction. Typically, the individual mean is subtracted from each input feature, which centers the data around the origin in each dimension. In the case of images, it is often sufficient to subtract a single value from all pixels or a single value for each color channel. In contrast, normalization is not always necessary, since for images the relative scales of the pixels are

already approximately equal and in the range from 0 to 255. Another way to normalize the input is to apply batch normalization to the input layer.

Before starting training, it is important to initialize the weights. The weight initialization has a strong influence on how and if the network learns during training. On the one hand, it should be avoided that activations excessively saturate since then the gradients would not propagate well. On the other hand, they should not be too linear, as they would not compute anything interesting [GB10]. First of all, the worst thing to do would be to initialize all weights with the same value, for example setting them to 0. If every unit in the network outputs the same value, the gradients would all be the same and each unit would perform exactly the same weight updates. To break symmetry, weights are usually initialized to small random values around 0. A common procedure is to sample from a Gaussian distribution with zero mean and one standard deviation. However, smaller numbers do not always work better. For example, in a deep network, it is problematic if the gradients are very small and decrease too much during backpropagation. If the initial weights are sampled from a random distribution, another problem arises. As the number of inputs increases, so does the variance. To solve this, the variance of a unit's output can be normalized to 1 by scaling its weight vector by the square root of the number of inputs, i.e., multiplying the random weight by $\frac{1}{\sqrt{n}}$ for n inputs. This factor follows from the fact that each input should have a variance $\frac{1}{n}$ to normalize the output variance [Bot12]. The mean of the output of a unit is assumed to be zero-centered. This is however not the case for e.g. ReLUs. Glorot et al. [GB10] recommend a variance of $2/(n + m)$ is recommended for weight initialization, where n is the number of units in the previous layer and m is the number of units in the next layer. In newer deep neural networks, the variance for ReLU initialization is set to $\sqrt{2.0/(n + m)}$ or $\sqrt{2.0/n}$ [He+15b] which is referred to as Xavier initialization. Biases can be initialized to 0 without concern. Another option is sparse initialization, where all weights are set to 0, but each unit is randomly connected to a fixed number of units in the layer below.

2.1.7 Transformers

Transformers were originally introduced in the field of natural language processing (NLP) [Vas+17], but have recently been used in many other fields. In contrast to earlier sequence-to-sequence models such as RNNs or LSTMs [HS97], they do not process inputs sequentially, but are able to process sequence inputs in parallel. This, along with self-attention, enhances their efficiency and ability to capture long-range dependencies within the data. This section briefly introduces the main concepts behind transformer models before discussing their adaptations for processing image data.

Encoder-Decoder Architecture

A transformer consists of two main components, an encoder and a decoder. From a high level perspective, the encoder processes embedded sequence inputs, which are combined with positional encodings to preserve the order of the sequence, and processes them through several similarly structured encoder blocks. Within these blocks a crucial part is the attention mechanism which allows to consider the whole input sequence. The decoder part, on the other

hand, takes the output of the last encoder block as input. The decoder part itself consists of several decoder blocks, each of which includes attention blocks that work slightly different from the encoder attention mechanism. Inputs at a position after the actual input within the input sequence are masked and attention is linked to the encoder output. Figure 2.13 depicts the transformer architecture as proposed by Vaswani et al. [Vas+17].

At inference time, all encoder inputs are processed in parallel. However, the decoder outputs are generated sequentially. This is because each token generated by the decoder depends on previously generated tokens. The decoder part predicts the next output token given the encoded embeddings and the output embeddings up to the current position. More precisely, the final linear layer outputs a probability distribution over the vocabulary for the current output embedding position. After the output token for a particular position has been generated, the output embedding for that token is added to the sequence of output embeddings and the next position is processed. For training, cross entropy loss is used.

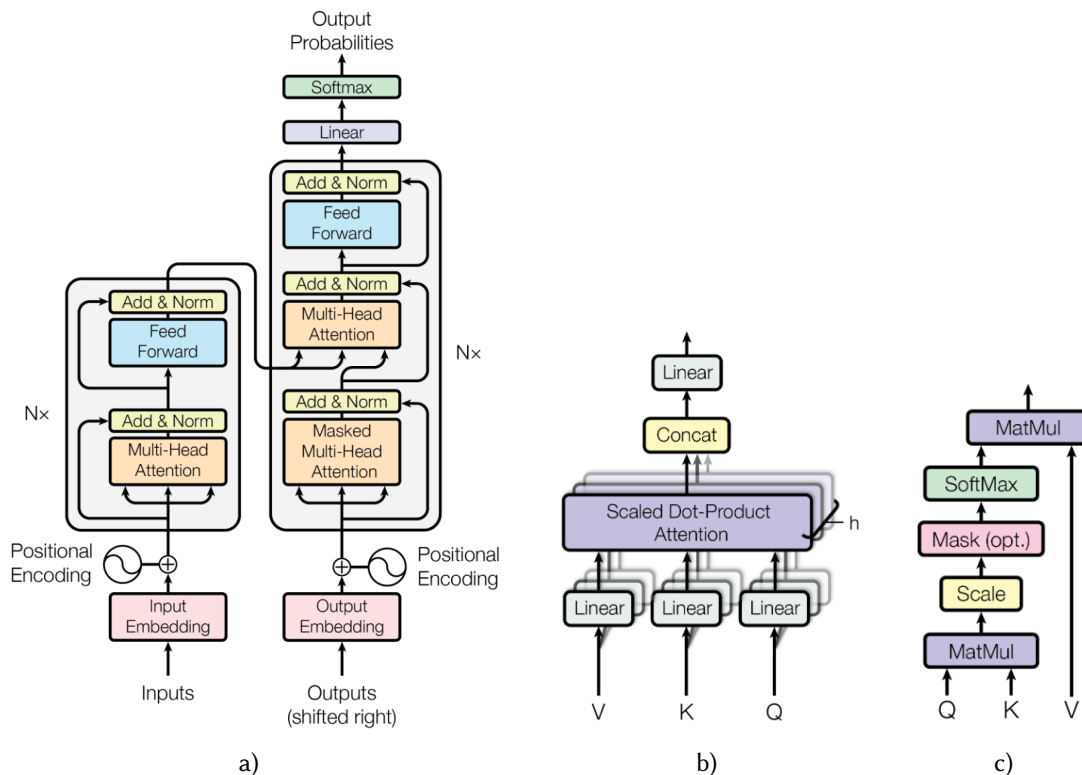


Figure 2.13: The Transformer model (a) includes different kinds of Multi-head Attention blocks (b). Each of them performs scaled dot-product attention (c). Images taken from Vaswani et al. [Vas+17].

Attention

Attention allows the encoder and decoder to consider the entire input sequence when processing a given embedding. This mechanism is called Scaled Dot-Product Attention. It maps a set of

query vectors (Q), key vectors (K), and value vectors (V) to an output, as illustrated in Figure 2.13c. The attention function is computed as follows

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.65)$$

where Q , K , and V are matrices obtained by projecting the input embedding matrix X of dimension d_{model} through learned weight matrices $W^Q \in \mathbb{R}^{d_{\text{model}} \times d_q}$, $W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, and $W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, respectively. For example, the key matrix is computed as $K = XW^K$. The dimensions d_q , d_k , and d_v are hyperparameters that define the size of the query, key, and value vectors (or matrices in batch processing). The resulting matrix of the dot product between each query vector in Q and each key vector in K is referred to as the scores, which represent how much each element in the sequence (represented by the queries) should care about every other element (represented by the keys). The scaling factor ($\sqrt{d_k}$) scales the dot products to stabilize the gradients by preventing large values. The softmax function normalizes the scaled dot products to produce a set of weights for each query, with these weights reflecting the relevance of each key to the query. It is applied across the rows of the score matrix for each query over all keys.

These normalized scores are then used to perform a matrix multiplication with the value vectors (V), effectively computing a weighted sum of the value vectors for each query. This operation aggregates the information across the sequence, weighted by the computed attention scores, resulting in a set of vectors that form the output of the attention mechanism.

In the encoder blocks, this attention is usually called self-attention. In the decoder blocks, attention is applied twice, each time in a slightly different way. First, the attention is applied to the input embeddings of the decoder block in the same way as in the encoder, with a small difference: before applying the softmax to obtain the weight vector for V , those inputs of the softmax at positions corresponding to positions after the position of the current input embedding are masked. This is done by setting the corresponding elements in the scaled vector to $-\infty$. The purpose of the masking is to preserve that the generation of each output embedding can only depend on previously generated embedding, not future ones. The second attention is also called cross-attention: It computes query vectors Q from the previous embedding within the decoder as for self-attention, but V and K are computed on the outputs of the final encoder block. This provides a connection to the output sequence of embeddings of the encoder for each decoder block.

The described computation of the attention function is limited to computing the attention of a single embedding to all other embeddings within the input sequence. Multi-Head Attention allows the model to generate multiple attention weight distributions, enabling it to attend to different aspects of the information contained in the sequence (see Figure 2.13b). More precisely, it allows to compute h weight matrices W^V , W^K and W^Q and to compute the attention function h times, thus allowing the model to attend to different positions. The resulting vectors are concatenated and multiplied by a weight matrix $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$, resulting in output vectors of dimension d_{model} . It is defined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_h)W^O, \quad (2.66)$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

Attention is followed by layer normalization [BKH16], which is defined as

$$\text{LN}(x) = \gamma \left(\frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \right) + \beta, \quad (2.67)$$

where γ and β are learnable parameters.

This normalized output is added to the pre-attention input vectors, allowing direct information flow through a bypass connection, thus serving a similar purpose as residual connections in ResNet [He+15a] or U-Net [RFB15].

Within the encoder and decoder blocks, attention is followed by a feed-forward neural network with ReLU activation shared across input positions:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2. \quad (2.68)$$

Positional Encoding

Since transformers do not inherently have access to information about the position of an embedding within the input sequence, a positional encoding is used to bring this information into the model.

This is achieved by adding position encoding vectors to the input embeddings. Position encodings are defined as follows

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad \text{and} \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (2.69)$$

where pos is the position within the input sequence, $i \in \{0..d_{model} - 1\}$ is the dimension index. The use of sine and cosine allows relative positions to be learnt and inputs to be scaled to unseen lengths of sequences.

Vision Transformers

Vision Transformer (ViT) [Dos+20] adapt the Transformer architecture to address vision tasks such as image classification. It treats an image as a sequence of fixed-size patches (e.g. 16×16). Each patch is flattened and linearly projected to a certain dimension. This corresponds to the input word embeddings in original Transformers. The input image is thus converted into a sequence of 1D patch embeddings. In contrast to the original Transformer architecture, ViT does only use an encoder part and adds linear layers on top (see Figure 2.14). The encoder blocks work slightly different. ViT introduces a learnable class token as the model is trained for image classification.

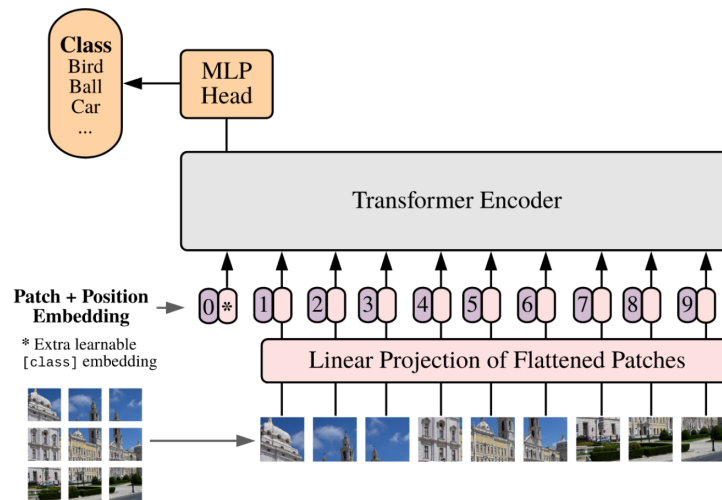


Figure 2.14: Vision Transformer uses the encoder part and introduces a class token. Image taken from Dosovitskiy et al. [Dos+20].

2.2 Object Detection

In object detection, the goal is to identify and locate objects in an image or video. It involves two primary tasks: identifying what objects are present (recognition) and determining where they are in the image (localization). Object detection has seen significant advances with the advent of CNNs. This section provides a brief overview from the beginnings of object recognition with CNNs to the current state of research.

Object detectors can be divided into two groups: Two-Stage Detectors and Single-Stage Detectors. Two-stage detectors are characterized by a two-stage object detection process. The first step in these models is to generate region proposals. These are essentially candidate object bounding boxes that the model predicts might contain objects. In the second step, each of the proposed regions is classified into different object categories. This allows for a more refined detection process, as the model first narrows down the areas of interest before classifying them, resulting in potentially higher accuracy. More recent architectures belong to the group of single-shot detectors. These models eliminate the region proposal step entirely. Instead, they directly predict object categories and their bounding box coordinates in a single pass through the network. This approach is usually faster than the two-stage detectors, as it reduces the computational overhead. It is particularly beneficial for applications requiring real-time processing. Usually there is a trade-off between speed and accuracy [Hua+17], as skipping the region proposal step might lead to less accurate localization of objects compared to the two-stage approach.

We will start chronologically with R-CNN [Gir+14] and then pointing out the improvements made by its successors, Fast R-CNN [Gir15] and Faster R-CNN [Ren+15]. As an representative example for architectures that belong to the group of Two-Stage Detectors, Faster R-CNN will be explained in more detail. We will then introduce the YOLO architecture [Red+16; RF18; JCQ23; WBL23] representing single stage detectors.

2.2.1 Two Stage Detectors

Shortly after the breakthrough of CNNs in image classification, they were used for object detection. A pioneering object detection approach based on CNNs is R-CNN [Gir+14].

R-CNN

R-CNN first generates region proposals using a method such as selective search [Uij+13]. These proposals are candidate regions in the image that may contain objects. Each proposed region is then resized and fed into a pretrained CNN to extract features. Then R-CNN uses a set of class-specific SVMs [CV95] for classifying these features. In addition to the SVM, for each class a linear regressor is trained as well. Depending on the predicted class the corresponding regressor is used for refining the bounding box of the proposal. Finally, a Non-Maximum Suppression (NMS) step removes multiple bounding boxes for each object: the proposals are ordered according to their confidence scores from the SVMs. Those boxes with the highest scores are kept, while boxes with lower scores and a predefined IoU overlap are removed.

Fast R-CNN

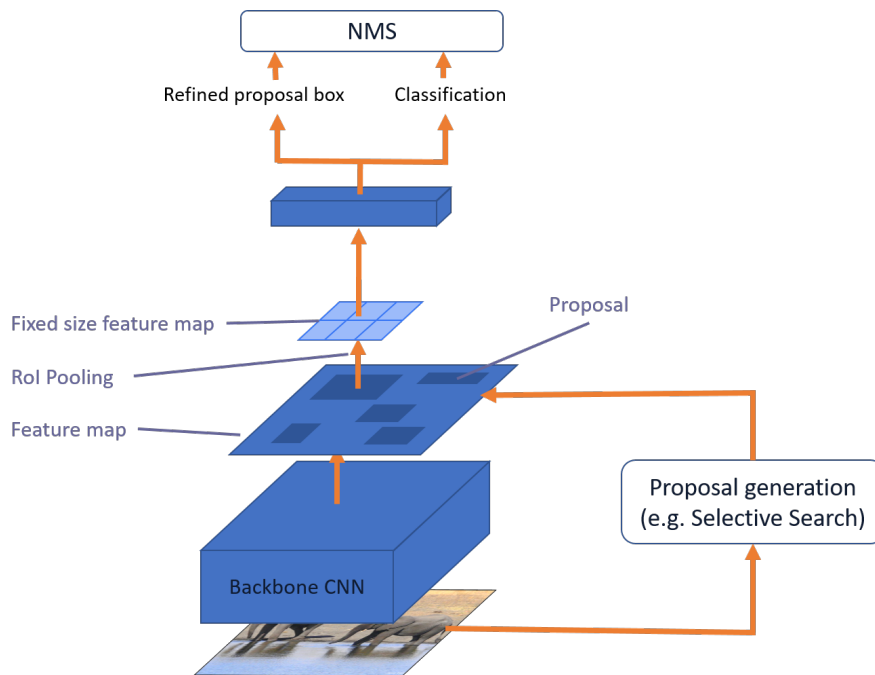


Figure 2.15: Fast R-CNN: Proposals are extracted from the feature map of the backbone CNN.

The main drawback of R-CNN, the expensive CNN application for each proposal, is addressed in its successor Fast R-CNN [Gir15]. In Fast R-CNN, region proposals are generated from the input image and then mapped to a shared convolutional feature map created by applying the backbone CNN to the entire image. From this global feature map fixed-size feature maps are

extracted via max (or average) pooling. This is referred to as RoI (Region of Interest) pooling, eliminating the need to apply the whole CNN for each proposal. Furthermore, the separate SVMs for classification and the linear regressors for bounding box regression used in R-CNN are replaced with a single head consisting of fully connected layers. This head performs both object classification, using a softmax layer, and bounding box regression, directly predicting refinements for the proposal boxes.

Figure 2.15 depicts the object detection process with Fast R-CNN: Proposal generation and feature extraction from the backbone CNN are performed once per image, while RoI pooling, bounding box prediction and classification are performed for each proposal. Finally, NMS is applied to filter out redundant bounding boxes.

For training, a multi-task loss $L = L_{cls} + \lambda L_{loc}$ is used. For classification L_{cls} is the log loss over $K + 1$ classes (including background). The loss for bounding box regression is defined as $L_{loc}(t, t^*) = \sum_{i \in \{x, y, w, h\}} \text{smooth}_{L1}(t_i - t_i^*)$ is used with

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1, \\ |x| - 0.5 & \text{otherwise.} \end{cases}$$

where t are the predicted and $t^* = (t_x^*, t_y^*, t_w^*, t_h^*)$ are the ground truth transformation parameters, calculated from the ground truth bounding box relative to the proposal box with

$$t_x^* = \frac{(x_{gt} - x_a)}{w_a}, \quad t_y^* = \frac{(y_{gt} - y_a)}{h_a}, \quad t_w^* = \log\left(\frac{w_{gt}}{w_a}\right), \quad t_h^* = \log\left(\frac{h_{gt}}{h_a}\right).$$

Faster R-CNN

Fast R-CNN still relies on an external region proposal algorithm such as selective search. For this reason, Faster R-CNN introduces a Region Proposal Network (RPN), a fully convolutional network that simultaneously predicts object bounding boxes and objectness scores at each position in the image. This RPN is integrated into the architecture, allowing for end-to-end training. Figure 2.16 shows the Faster R-CNN architecture, where the region proposal algorithm has been replaced by an RPN. It operates directly on the backbone CNN. For each spatial position of this feature map, a set of anchors is defined. These anchors are predefined bounding boxes of different scales and aspect ratios. They serve as reference points for potential objects. For each anchor box, the RPN predicts bounding box refinements and an objectness score. First, the objectness scores and refined bounding boxes are used to filter out likely background boxes. Then, redundant region proposals are removed using NMS.

By using an RPN for region proposals, Faster R-CNN can be trained end-to-end. The losses used to train the RPN are similar to the final outputs. The multi-task loss includes a binary cross-entropy loss for the objectness score (foreground and background) and a regression loss (smooth L1 loss) for the bounding box refinements.

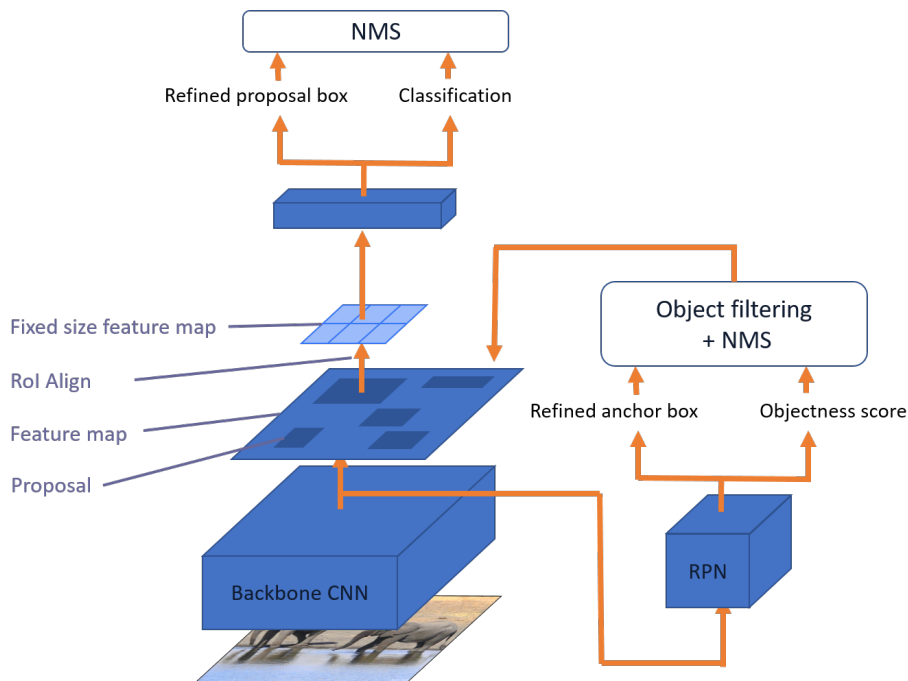


Figure 2.16: Faster R-CNN: Proposals are generated by a Region Proposal Network (RPN).

Feature Pyramid Networks

A weakness of the original Faster R-CNN architecture is that it struggles with small objects. This is due to the loss of fine-grained spatial information as the input passes through multiple convolutional and pooling layers, which leads to reduced resolution in deeper layers of the network.

Feature Pyramid Networks (FPN) [Lin+17a] apply the idea of feature pyramids to CNNs for object detection. It extends the backbone CNN by adding a top-down pathway with lateral connections to the existing bottom-up pathway of the CNN. The top-down pathway of the FPN starts from the highest-level feature map (with the lowest spatial resolution) in the backbone CNN and progressively upsamples it, as shown in Figure 2.17. The corresponding CNN feature maps are connected to those of the feature pyramid by adding them after applying a 1×1 convolutional layer. These typically five levels of the pyramid all contribute to the prediction of bounding boxes and classes, as anchor boxes are extracted from all levels. However, each level corresponds to a range of bounding box extents in the input image and thus predicts only for that range. Higher levels with smaller feature maps are used to detect larger objects, while lower levels with larger feature maps are used to detect smaller objects. When Faster R-CNN is extended with FPN, the feature pyramid is used in the RPN as well as in the subsequent classification and regression head.

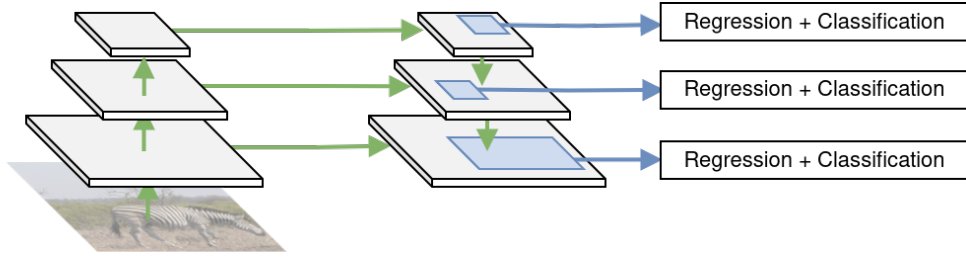


Figure 2.17: Feature Pyramid Network (FPN) for predicting bounding boxes and scores from different spatial resolutions.

2.2.2 Single Stage Detectors

In contrast to two-stage detectors such as Faster R-CNN, which first generate region proposals and then refine and classify them, Single Shot Detectors accomplish both tasks in a single pass through the network. This makes them faster and better suited for real-time applications. Among the first architectures for single shot detection were Single Shot Multi-Box Detector (SSD) [Liu+16b] and You Only Look Once (YOLO) [Red+16]. In contrast to the first version of YOLO, SSD relies on anchor boxes and uses multi-scale feature maps according to different object sizes, similar to Faster R-CNN with FPN, improving detection particularly for smaller objects. However, subsequent versions of YOLO followed these principles and employed anchor boxes and multi-scale feature maps. As one of the most influential approaches for single shot object detection we will give a short overview on the evolution of YOLO and focus on YOLO v8 in this section.

YOLO

YOLO v1 [Red+16] is fundamentally a CNN with a loss tailored to object detection. It is characterized by its ability to process the entire image in a single forward pass, enabling real-time object detection. The main idea is to divide the input image into a grid and predict two bounding boxes for each grid cell (e.g. 7×7 cells results in 49×2 box predictions per image), making it significantly faster than two-stage approaches since there is only a single NMS step on an order of magnitude fewer proposals. The loss used for training YOLO is a weighted sum of classification, objectness and localization loss. For objectness and classification, mean squared error is used. For bounding box regression, the localization is different from other detection architectures:

$$L_{loc} = \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right],$$

where S^2 represents the total number of grid cells, B is the number of bounding boxes per cell, $\mathbb{1}_{ij}^{obj}$ is 1 if an object is present in the j -th bounding box of the i -th grid cell, and 0 otherwise.

In YOLO v2 [RF17], anchor boxes for grid cells, batch normalization, connections to higher-resolution feature maps and multi-scale training were used. The architecture is fully convolutional. Among other improvements, anchor boxes were optimized via k-means clustering.

YOLO v3 [RF18] further improved the architecture by using a deeper backbone, Spatial Pyramid Pooling [He+15c] and changing the loss function to logistic regression for bounding boxes and cross-entropy loss for classification. It has three outputs corresponding to three ranges of bounding box sizes. In summary, architectural features previously used in two-stage detectors such as anchor boxes or FPN were also incorporated into YOLO. Additionally, the loss function became more similar to those used in two stage detectors, e.g. by adding an objectness loss term.

A detailed description of the further evolution of the YOLO series is beyond the scope of this thesis, so we will focus on a more detailed description of a recent version of YOLO that achieves state-of-the-art performance in real-time object recognition. Next, we will briefly outline the main improvements and architectural components of YOLO v8.

YOLO v8

YOLO v8 shifts from anchor box-based box prediction back to an anchor-free approach [Tia+19; Ge+21] like YOLO v1 and predicts the center of an object directly instead of the offset from a predefined anchor box. This is intended to counteract differences between the distribution of anchor boxes in the dataset used for training and other data at inference time, which may have different distributions of anchor boxes [TC23]. It also reduces the number of box predictions, thus speeding up the NMS step.

The architecture is divided into three parts: backbone, neck, and head. The backbone is a CNN consisting of blocks of C2f modules and convolutional layer and Spatial Pyramid Pooling (SPPF) module on top. In addition to the SPPF outputs, output feature maps of the backbone are taken from several intermediate layers corresponding to different spatial resolutions. The levels are referred to as P3, P4, and P5 (from low level high resolution feature maps to high level low resolution feature maps). They are connected to the neck which performs upsampling on P4 and P5 and provides skip-connections. The idea of up- and downsampling with skip-connections is similar as for U-Net (see Section 2.3.1), although more complex as it involves C2f modules. The head part consists of three decoupled heads for the three scale levels P5, P4 and P3, where each head separately predicts bounding boxes and class probabilities. The convolutional layers in YOLO v8 perform batch normalization (see Section 2.1.6) and use SiLU activations (see Section 2.1.2).

A central architectural block used in YOLO v8 is the C2f module. By its design it combines high-level features with contextual information to improve detection accuracy. The C2f module is an evolution of the Cross-Stage Partial (CSP) layer introduced in CSP Nets [Wan+20]. CSPNets have been shown to improve information flow and learning efficiency. Their main idea is to split the feature maps along the channel dimension, process one part through multiple convolutional layers, and then merge both parts. This avoids potentially redundant computations in two parallel branches. Specifically, within the YOLO v8 architecture, a part is processed through a

series of bottleneck blocks. These blocks consist of two convolutional layers with an optional shortcut connection.

Spatial Pyramid Pooling Fusion (SPPF) process features at various scales. SPPF extends the idea of Spatial Pyramid Pooling (SPP) [He+15c]. Figure 2.18 shows how SPP works on a convolutional layer: pyramid pooling spatially divides the input feature maps into a fixed number of bins (1×1 , 2×2 , and 4×4). The resulting outputs are concatenated. The size of the output is the same regardless of the size of the pooled feature map. This allows CNNs to process arbitrarily sized inputs. SPPF stacks pooling layers and adds shortcut connections rather than applying pyramid pooling for different scales in parallel, thus reducing computational cost.

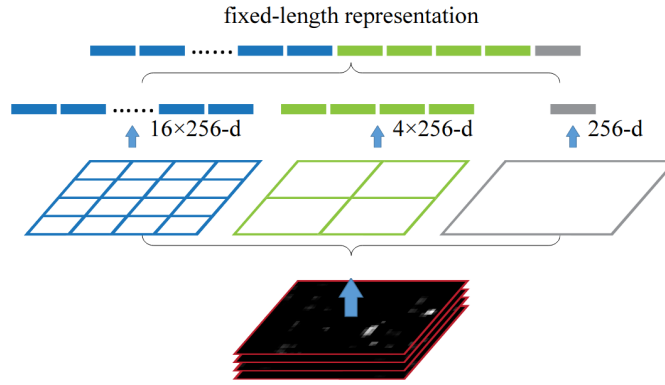


Figure 2.18: Spatial Pyramid Pooling (SPP). Image taken from He et al. [He+15c].

The loss function for YOLO v8 is binary cross entropy for classification and a combination of Complete Intersection over Union (CloU) [Zhe+20] and Distribution Focal Loss (DFL) [Li+20] for bounding box regression.

Unlike IoU, which considers the overlap between bounding boxes, CloU loss also considers the distance between their centers and the aspect ratio. It is defined as

$$\text{CloU} = \text{IoU} - \frac{\rho^2(b, b^{gt})}{c^2} - \alpha v,$$

$$\text{with } \alpha = \frac{v}{(1 - \text{IoU}) + v} \quad \text{and} \quad v = \frac{4}{\pi^2} \left(\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h} \right)^2, \quad (2.70)$$

where IoU is the intersection over union between the predicted bounding box b and the ground truth bounding box b^{gt} , $\rho(b, b^{gt})$ is the Euclidean distance between the center points of the boxes, c is the diagonal length of the smallest enclosing box that covers both boxes, v is the aspect ratio consistency term that is balanced by α . For the predicted box, w and h are the width and height, and w^{gt} and h^{gt} are the width and height of the ground truth bounding box, respectively.

DFL extends focal loss to continuous values and can therefore be used in bounding box regression when predicting attributes such as center coordinates, width, and height.

The original focal loss (FL) [Lin+17b] addresses the scenario where there is a strong imbalance between classes during training. It is defined as

$$\text{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t), \quad (2.71)$$

where p_t is the estimated probability for the class with the true label, α_t is a weighting factor for the class that adjusts the importance of positive or negative examples, and γ is the focusing parameter that adjusts the rate at which easy examples are down-weighted, helping the model focus more on hard examples.

DFL is applied to each bounding box regressor. It is applied to a predicted discretized probability distribution (realized as a softmax) for a given number of bins (e.g., 14 bins). The minimum and maximum values of the interval reflect the minimum and maximum possible offsets for the corresponding bounding box attribute. Each bin is associated with an interval of offsets. The idea then is to focus on the probability of the predicted values in the vector at positions i and $i + 1$ around the ground truth value y . The values y_i and y_{i+1} represent the start and end points of an interval around the ground truth value y with $y_i \leq y \leq y_{i+1}$. DFL is defined as

$$\text{DFL}(S_i, S_{i+1}) = -(y_{i+1} - y) \log(S_i) + (y - y_i) \log(S_{i+1}), \quad (2.72)$$

where S_i and S_{i+1} are the softmax probabilities predicted by the model for the merged interval (spanning two bins) from y_i to y_{i+1} . As with the focal loss, hard examples, such as boxes with ambiguities in the object boundaries, have a higher impact on the total loss because their respective softmax probabilities are lower.

Unlike its predecessors, which focused primarily on object detection, YOLO v8 also supports other vision tasks such as segmentation and pose estimation.

Transformers for Detection

In this section, the focus was on CNN-based architectures. However, there exist also models based on Transformer architectures. DETR (Detection Transformer) [Car+20] performs similarly well as Two-Stage CNN based detectors like Faster R-CNN. While object detection is performed in an end-to-end manner (single-stage object detection), this has several drawbacks. It generally requires more training data and training time. Inference speed can be slower due to its more complex architecture. Additionally, DETR struggles in detection of small objects as it lacks multi-scale features, which are critical for small object detection [Liu+23].

2.3 Segmentation

In image segmentation, the goal is to divide an image into several parts or regions in order to simplify its representation. While there are different approaches to image segmentation that are more or less suitable depending on the specific problem, in this section we will focus on deep learning based image segmentation. These approaches can be used in cases where classical

image processing methods such as thresholding (e.g. OTSU [Ots79]) or edge detection [Can86], region-based (e.g. region growing [PP93]), clustering-based or watershed segmentation [BM18] fall short. This is the case, for example, in natural image segmentation, which is particularly challenging due to the inherent complexity and variability of real-world images.

In deep learning-based image segmentation, we can categorize segmentation tasks into three groups: Semantic Segmentation, Instance Segmentation, and Panoptic Segmentation. Figure 2.19 shows the different segmentation tasks for an input image.

- Semantic segmentation involves labeling each pixel in an image with a class label that corresponds to a category or type of object. The goal is to classify each pixel into a category, such as trees, sky, etc. As shown in Figure 2.19b, this does not distinguish between instances of a particular object type or class (i.e., all pixels for elephants are assigned the same label).
- Instance segmentation focuses on segmenting individual objects. Rather than assigning each pixel of the input image to a specific class, it requires that each instance's pixels are assigned correctly. This usually involves object detection to first detect instances, and a subsequent segmentation step. In Figure 2.19c, the pixels inside the detected bounding boxes are segmented.
- Panoptic segmentation [Kir+19] requires labeling every pixel in an image, like semantic segmentation, but also distinguishes between instances of the same object class, similar to instance segmentation. In panoptic segmentation, classes are typically grouped into "stuff" (amorphous regions such as grass, sky, road) and "things" (countable objects such as people, animals, cars). For example, the stuff classes sky and grass in Figure 2.19d are the same as in Figure 2.19b, but each pixel belonging to an elephant is assigned to a specific instance class.

In this section, we will focus on semantic segmentation and instance segmentation, and outline approaches that represent methods in each area. We will take a closer look at U-Net [RFB15] as an architecture for semantic segmentation. Then, we will introduce Mask R-CNN [He+17] as a typical architecture for instance segmentation. Finally, we will highlight the extensions needed to adapt YOLO v8 for instance segmentation.

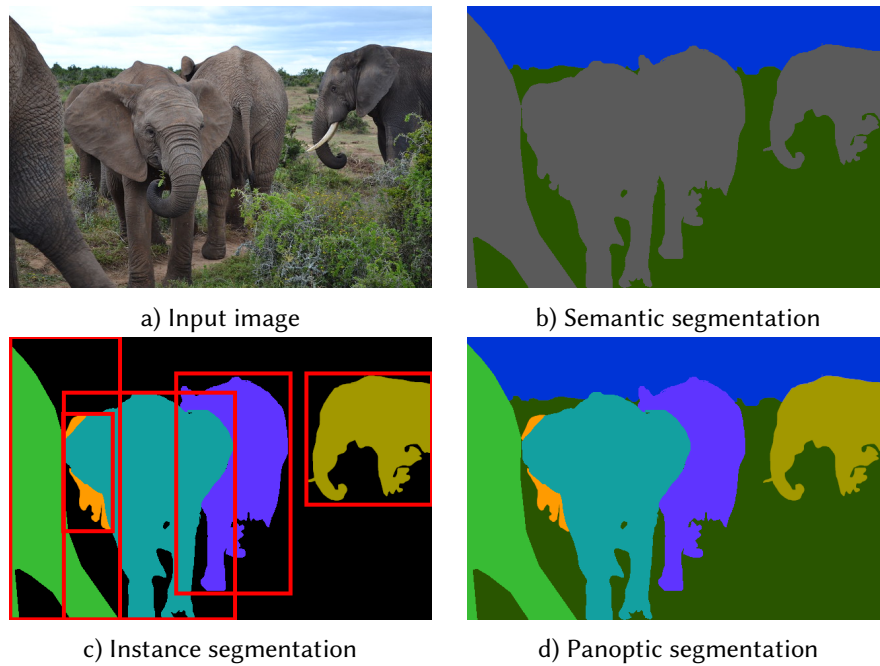


Figure 2.19: Different segmentation tasks for an input image a)

2.3.1 U-Net

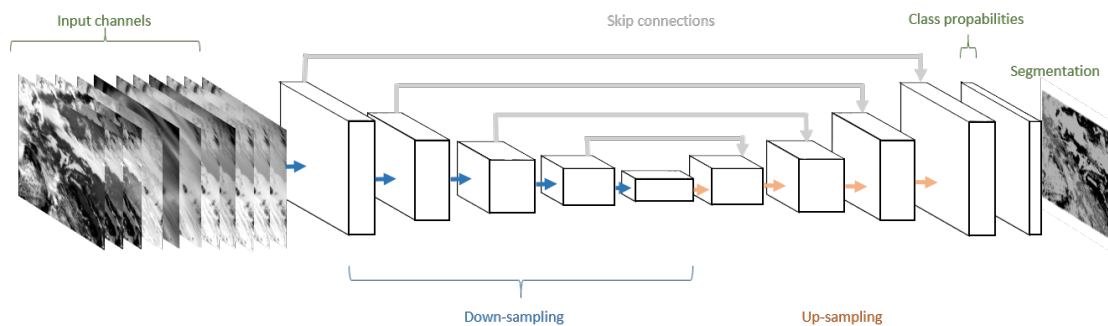


Figure 2.20: U-Net for cloud segmentation.

One of the most widely used architectures for semantic segmentation is the U-Net. It was originally developed for and applied to biomedical image segmentation. Its name refers to its typical shape. It consists of two parts: a downsampling and an upsampling path. The downsampling path is similar to a typical CNN, i.e., alternating convolution and pooling layers. The upsampling path uses transposed convolutions [Zei+10], which aim to reverse the transformation of a convolution, effectively increasing the spatial dimensions of the input. An important feature is the use of skip connections between downsampling and corresponding upsampling layers. This allows a flow of contextual information from high-resolution layers, which helps with precise localization in upsampling layers.

Figure 2.20 shows a U-net architecture that we adapted for cloud segmentation in satellite imagery [Drö+18]. It works with 9 input channels, has fewer filters for a lighter network, and uses strided convolutional layers instead of pooling layers for downsampling. For training a U-Net for predicting one of C classes per pixel a multi-class cross entropy is used:

$$L_{CE} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^H \sum_{k=1}^W \sum_{c=1}^C y_{ijk} \log(\hat{y}_{ijkc}), \quad (2.73)$$

where \hat{y}_{ijkc} is the softmax output for class c at position (j, k) and y_{ijkc} is the 1 if the groundtruth at pixel position (j, k) is c and 0 otherwise.

2.3.2 Mask R-CNN

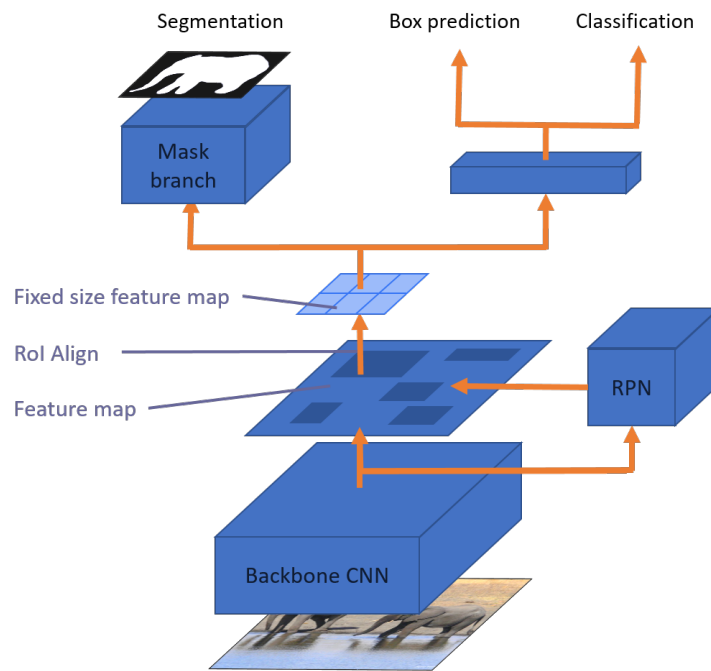


Figure 2.21: Mask R-CNN.

Mask R-CNN [He+17] is based on Faster R-CNN (see Section 2.2.1) and extends it for instance segmentation. It includes an additional branch for segmentation: In addition to box prediction and classification, the extracted region of interest is processed by a mask branch that outputs a segmentation map (see Figure 2.21). To train the mask branch, the following binary cross-entropy loss is used:

$$L_{mask} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^H \sum_{k=1}^W [y_{ijk} \log(\hat{y}_{ijk}) + (1 - y_{ijk}) \log(1 - \hat{y}_{ijk})], \quad (2.74)$$

where \hat{y}_{ijk} is the predicted and y_{ijk} is the groundtruth value of for each pixel at position (j, k) . The total loss used for training is therefore $L = L_{cls} + L_{box} + L_{mask}$. Other improvements include replacing RoI pooling with RoIAlign, which addresses the quantization errors introduced by RoI pooling. Instead of dividing the RoI into a grid, it computes exact floating-point values of cell boundaries and computes values in the output feature map via bilinear interpolation.

2.4 Foundation Models

Foundation models are large-scale models trained on large datasets of text, images or audio. They are often trained in a self-supervised manner. Because of the general representations learned, they can be fine-tuned or adapted for a variety of downstream tasks.

In this section, we present three foundation models. First, we introduce the Contrastive Language-Image Pre-training (CLIP) [Rad+21], an image-to-text model trained on unified image and text representations. Next, the Segment Anything Model (SAM) [Kir+23] is described, a general purpose image segmentation model. Trained on the same data as SAM, FastSAM [Zha+23] provides comparable segmentation results to SAM, but by using a CNN architecture instead of Transformer, it significantly improves inference time.

2.4.1 CLIP

CLIP (Contrastive Language-Image Pre-training) is a model trained to understand both text and images. The training data are images and their descriptions collected from the Internet. It consists of two main components: the text encoder and the image encoder. The image encoder can be based on various architectures, such as a Convolutional Neural Network (CNN) or a Vision Transformer (ViT). The text encoder processes textual input (the image descriptions) and typically uses a transformer-based architecture. Both encoders are trained to project their input into a common embedding space. This is achieved by learning to minimize the distance between the text and image vectors of an image-text pair, as shown in Figure 2.22. CLIP uses the following contrast loss function for training:

$$\mathcal{L} = -\frac{1}{2N} \sum_{n=1}^N \left[\log \frac{\exp(\text{sim}(i_n, t_n)/\tau)}{\sum_{k=1}^N \exp(\text{sim}(i_n, t_k)/\tau)} + \log \frac{\exp(\text{sim}(t_n, i_n)/\tau)}{\sum_{k=1}^N \exp(\text{sim}(t_n, i_k)/\tau)} \right] \quad (2.75)$$

where $\text{sim}(i, t)$ denotes the similarity score between an image embedding i and a text embedding t , i_n and t_n represent the image and text embeddings of the n^{th} pair, respectively. The temperature τ is a scaling parameter that controls the degree of concentration of the distribution, affecting the separation between positive and negative pairs in the embedding space. The loss is averaged over all N pairs in the batch, with the sum running over both image-to-text and text-to-image comparisons to ensure symmetry in learning.

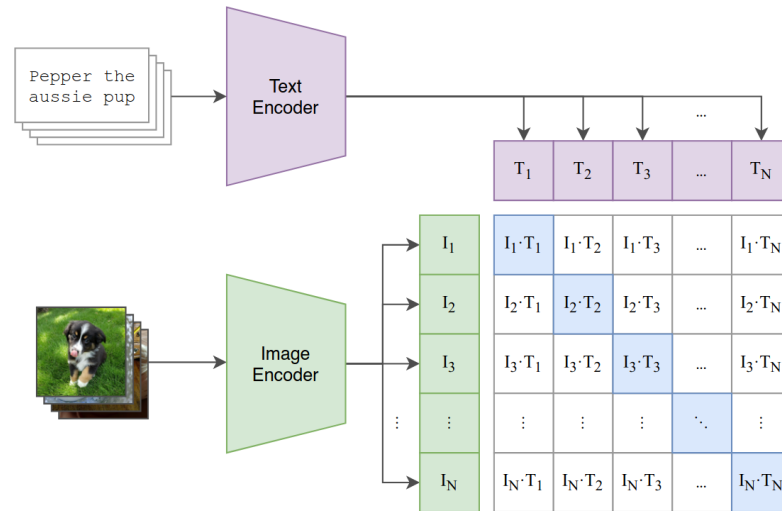


Figure 2.22: CLIP contrastively learns unified image- and text representations. Image taken from Radford et al. [Rad+21].

2.4.2 SAM

The Segment Anything Model (SAM) is a basic image segmentation model capable of segmenting a wide range of images without task-specific training. It is trained on the SA-1B dataset, which consists of over 1 billion masks on 11 million images. The images were labeled iteratively and semi-automatically. It is based on a Vision Transformer backbone. In addition to the Vision Transformer (stacked encoder blocks) that encodes the images, an additional prompt encoder is trained that learns a unified embedding for point, box, and mask prompts, and, based on CLIP text encodings, also text prompts. These two encoders are the input to a mask decoder that outputs the segmentation maps. For training focal loss together with dice loss, which is defined as:

$$\text{Dice Loss} = 1 - \frac{2 \times |Y \cap \hat{Y}|}{|Y| + |\hat{Y}|}, \quad (2.76)$$

where Y denotes the ground truth mask and \hat{Y} denotes the predicted mask.

2.4.3 FastSAM

Unlike SAM, FastSAM uses a CNN-based architecture, namely the YOLO v8 detector with segmentation. The use of a CNN dramatically reduces computation time while maintaining comparable segmentation performance. It is trained on a fraction of the SA-1B dataset introduced for SAM. While FastSAM cannot directly process prompts within its architecture, the resulting masks are post-processed and matched to box, dot, and text prompts.

2.5 Image Similarity Search

Image similarity is a subjective concept, depending on the context in which it is being applied. How similar two images are depends on the specific scenario. For instance, two images might be considered similar based on their color distribution but dissimilar when considering the arrangement of elements within them. Furthermore, when considering the similarity of colors, shapes, textures, and patterns this is often not reflected in semantic concepts of objects present in an image. Instance retrieval [CAS20; Che+22] is closely related to image similarity search. It can be considered as a subfield of image similarity search, since it focuses on identifying and retrieving images that contain the same specific instance depicted in the query image.

In practice, image similarity is usually measured between feature vectors that represent those images. Similarity search then corresponds to nearest neighbor search in feature space, whether binary or real-valued. While earlier image descriptors such as SIFT [Low99] were used, nowadays the feature vectors are usually obtained from deep learning models.

Figure 2.23a shows a result list of a similarity search system based on CNN features that were pre-trained for image classification on OpenImages [Kuz+20]. In this example, two images are similar, if they are from the same class, i.e., *bicycle*. A more fine-granular search could consider color or type of the bicycle. The determination how images are similar to each other is thus determined by the model and dataset, respectively.

Image similarity search is used to find images that are visually similar to a given query image. Figure 2.23b shows the process from a high level point of view: A user's search intention is expressed via a query image which is then used to query a database to find similar images. The results are then presented to the user. The search process involves two main steps: feature extraction and nearest neighbor search.

First, features need to be extracted from an image. These features implicitly define the concept of similarity used for image similarity search. An image is represented by a vector in this feature space. Similar images are those which are close in feature space (e.g., in terms of Euclidean distance). When indexing an image dataset, the speed of feature extraction is not the most important issue, but it is very important at query time, since this step needs to be performed before using the feature vector for nearest neighbor search.

The second step, nearest neighbor search, is even more important for search performance. As the dataset to be searched grows, the requirement for fast and efficient nearest neighbor search becomes more important. The performance of the image similarity search depends on various parameters, such as the indexing structure and the corresponding search algorithm, the type of the feature vectors (e.g., binary or float values), and the length of the feature vector.

2 Fundamentals

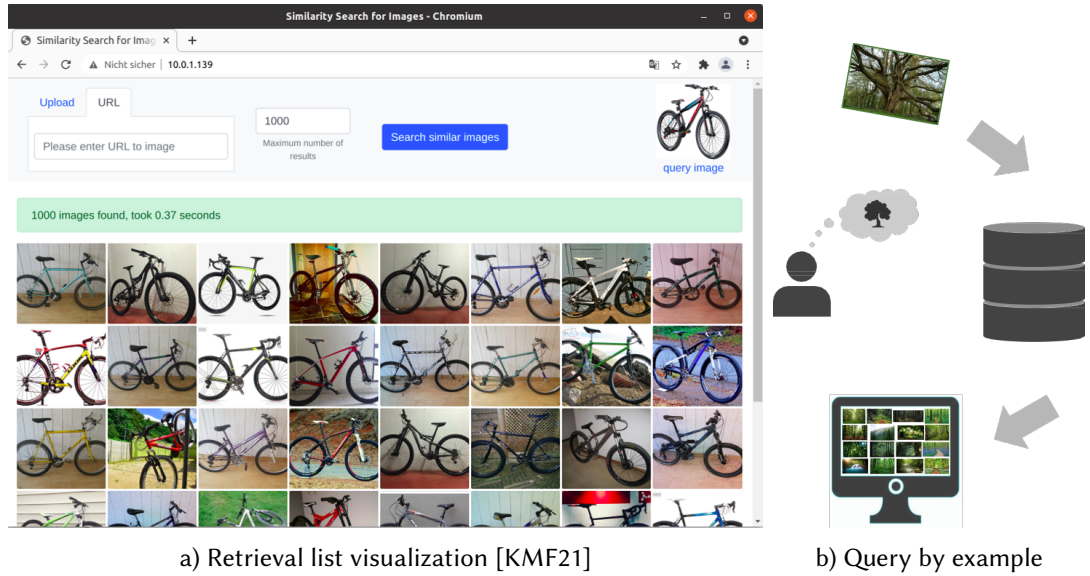


Figure 2.23: Image similarity search: A query image is used to find similar images (b). The result is visualized as retrieval list (a).

2.5.1 Similarity Measures

The following measures are used to quantify how similar two feature vectors are to each other. The query vector is \mathbf{q} and a vector from the database is \mathbf{x} .

Euclidean Distance

The simplest way to measure distance between feature vectors is the Euclidean distance. It computes a straight line distance between points in feature space:

$$d_e(\mathbf{q}, \mathbf{x}) = \sqrt{\sum_{i=1}^n (q_i - x_i)^2} \quad (2.77)$$

The smaller the value, the closer the feature vectors are.

Cosine Similarity

The cosine similarity calculates the cosine of the angle θ between two vectors. One advantage is that it can be used to compute how similar two feature vectors are, regardless of their magnitude. Contrary to the Euclidean distance, a cosine similarity close to 1 indicates high similarity, a cosine similarity close to -1 indicates high dissimilarity (opposite directions).

$$s_{\cos}(\mathbf{q}, \mathbf{x}) = \cos(\theta) = \frac{\mathbf{q} \cdot \mathbf{x}}{\|\mathbf{q}\| \|\mathbf{x}\|} = \frac{\sum_{i=1}^n q_i x_i}{\sqrt{\sum_{i=1}^n q_i^2} \cdot \sqrt{\sum_{i=1}^n x_i^2}} \quad (2.78)$$

Cosine similarity can be used to measure distance instead of similarity by computing $1 - s_{cos}$.

If the vectors are L_2 normalized (dividing the vectors by their magnitude to get a vector of unit length), then computing the Euclidean distance results in the same order as when using the cosine distance (also the actual distance values will differ).

Hamming Distance

If the vectors are \mathbf{x} and \mathbf{q} binary vectors, i.e., in Hamming space, the Hamming distance can be used:

$$d_H(\mathbf{q}, \mathbf{x}) = \sum_{i=1}^n \mathbb{1}_{q_i \neq x_i} \quad (2.79)$$

Although there is a quantization error, an advantage of using binary vectors is that the Hamming distance computation is very fast, especially in hardware implementations or using bitwise operations in software, which is significantly faster than computing Euclidean distance in high-dimensional spaces.

2.5.2 Approximate Nearest Neighbor Search

Nearest Neighbor Search (NNS) includes algorithms that are used to find the data points in a given dataset that are closest to a query point. Its simplest form is the brute-force search, which calculates the distance from the query point to every other point in the dataset and then selects the points with the smallest distance. This is also referred to as exhaustive search or linear scan. While straightforward and exact, this approach is expensive and obviously does not scale for larger datasets. For this reason, Approximate Nearest Neighbor (ANN) methods partition the search space or use some kind of quantization to enable a more efficient search. ANN methods can be grouped into hashing-based methods, product quantization based methods, and graph-based methods [Wan+15]. Hashing methods map input feature vectors to a lower dimensional binary space. In this way, the learned representations (referred to as compact binary codes or hash codes) enable memory-efficient storage and fast retrieval of similar items from large databases. In the context of ANN, hash functions are supposed to achieve the opposite of hash functions know from other fields: instead of avoiding collisions, they are explicitly designed the way that similar items have the same or similar hash values (in terms of the Hamming distance).

2.5.3 Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) [GIM+99] is one of the first methods in the field of hashing-based ANN methods. The idea of LSH is to hash data points such that similar points are more likely to be mapped to the same bucket. LSH uses a set of hash functions which map a data point to a bucket. The number of hash functions n is the length of the binary string, i.e., there is one hash function for each position in the binary string. The bucket a data point is mapped to

then corresponds to the particular value of the resulting binary string. Although the definition of LSH is more general, when used for ANN, where the input is a d -dimensional embedding vector and the output is an n -dimensional binary vector, Signed Random Projection (SRP) is used as hash function, that implicitly assumes cosine similarity of the embedding vectors and is defined as:

$$h_{\mathbf{a}}(\mathbf{v}) = \text{sign}(\mathbf{a} \cdot \mathbf{v}) \quad (2.80)$$

where \mathbf{a} is a projection vector with each element is randomly chosen from a Gaussian distribution $N(0, 1)$. In this way, the hash function defines a hyperplane that acts as a decision boundary for the hash value to be either 0 or 1.

While LSH aims to map data points that are close to each other into the same buckets, the decision boundaries for the buckets are data-independent because they are chosen randomly. For this reason, when LSH is used for ANN of image feature vectors, the performance is usually inferior to that of deep hashing methods.

2.5.4 Deep Hashing

Similar to LSH, the goal of deep hashing is to represent large-scale, high-dimensional data such as images in a lower-dimensional binary space while preserving the similarity relations of the original space between the data points. The fundamental idea to use deep learning methods to learn binary hashcodes was introduced by Salakhutdinov et al. as Semantic Hashing [SH09] where an auto-encoder learns a binary hashcode representation of the input data. In contrast to data-independent hashing methods, the hashing function is directly learned from the input data. The field of deep hashing can be roughly divided into supervised and unsupervised methods. Supervised deep hashing methods [Cao+17b; Su+18; Cao+18b] usually utilize class labels of large image datasets to train an output layer that produces outputs that approximate binary vectors of a certain length. The performance depends on the network architecture and the loss function. Typical loss functions are pairwise or triplet losses (see Section 2.1.3). Often the quantization error is incorporated into the loss function. In unsupervised deep hashing [Su+18; Yan+19], the similarity information is usually derived from the relationship in the original space.

2.5.5 Product Quantization

The idea of product quantization [JDS10; Mat+18; JDJ19] is to divide the high-dimensional space into smaller sub-spaces and then quantize each of these sub-spaces separately. The input vector of length D is split into M subvectors \mathbf{x}_m of equal length $\frac{D}{M}$, corresponding to M subspaces. For each of the subspaces a set of K representative values $c_{m,k}$ is derived (called codes or centroids). The learning of the codes typically involves a clustering algorithm, such as k-means, applied to each of the M sub-spaces. Each centroid is assigned to a unique id. By accepting a quantization error between the floating point sub vector and the representative value assigned to its code, the subvector can be stored as an id referencing to the value, thereby significantly reducing storage costs. Query costs are reduced by using an inverted index for the codes that point to the corresponding subvectors in the dataset.

The whole code is quantized as follows:

$$Q(\mathbf{x}) = [Q_1(\mathbf{x}_1), Q_2(\mathbf{x}_2), \dots, Q_M(\mathbf{x}_M)] \quad \text{with} \quad Q_m(\mathbf{x}_m) = \underset{\mathbf{c}_{m,k} \in \mathcal{C}_m}{\text{argmin}} \|\mathbf{x}_m - \mathbf{c}_{m,k}\|^2, \quad (2.81)$$

The approximate reconstruction $\hat{\mathbf{x}}$ of \mathbf{x} from its centroids can be used to efficiently perform distance computations and is defined as:

$$\hat{\mathbf{x}} = [\mathbf{c}_{1,Q_1(\mathbf{x}_1)}^T, \mathbf{c}_{2,Q_2(\mathbf{x}_2)}^T, \dots, \mathbf{c}_{M,Q_M(\mathbf{x}_M)}^T]^T \quad (2.82)$$

The Euclidean distance between a query vector \mathbf{q} and a database item \mathbf{x} , after quantization, can be approximated by summing the distances between their corresponding sub-vector centroids:

$$\|\mathbf{q} - \mathbf{x}\|^2 \approx \sum_{m=1}^M \|\mathbf{c}_{m,Q_m(\mathbf{q}_m)} - \mathbf{c}_{m,Q_m(\mathbf{x}_m)}\|^2 \quad (2.83)$$

2.5.6 Multi-Index Hashing

Usually, it is not necessary to compute the distance to all neighbors, since it is often sufficient to search neighbors within certain radius of the query point. Multi-Index Hashing (MIH) [NPF12] divides a hashcode h into m partitions, i.e. $h = (h^1, \dots, h^m)$, where h^k and g^k are the k^{th} subcodes. The Hamming distance between the query code and the database code is the same as computing the distance between all subcodes and adding them up. However, when searching only within a specific radius, not all subcodes need to be considered. The idea of MIH is based on the observation that for two binary codes $h = (h^1, \dots, h^m)$ and $g = (g^1, \dots, g^m)$ the following proposition holds:

$$\|h - g\|_H \leq r \Rightarrow \exists k \in \{1, \dots, m\} \quad \|h^k - g^k\|_H \leq \left\lfloor \frac{r}{m} \right\rfloor, \quad (2.84)$$

where H is the Hamming norm.

For example, a 64-bit binary code could be divided into $m = 4$ partitions. Then, for every neighboring code within radius $r = 3$, there must exist a subcode with $\lfloor \frac{3}{4} \rfloor = 0$. In this case, instead of comparing the whole binary string, it is sufficient to check if any of the subcodes are equal. Multi-Index Hashing can be combined with the previously introduced methods.

3

Detection and Segmentation

This chapter presents work in two related areas: object detection and image segmentation. Both works present solutions to specific problems that could not be solved by available state-of-the-art methods alone. The first work presents an approach to solve the problem of textual stamp detection and recognition. The second work presents a solution for improved segmentation of morphologically complex eukaryotic cells.

Detection and Recognition

Detecting textual stamps on index cards that contain various other text itself poses various challenges. Our approach integrates several deep learning approaches into a pipeline for textual stamp detection and recognition. We show that the learned textual stamp representations generalize well and can be used to identify and group unknown textual stamps.

Detection and Segmentation

Segmentation of morphologically complex eukaryotic cells in fluorescence microscopy images is a challenging task due to the inherent variability in cell shapes, sizes, overlapping cells, and the complex intracellular structures that may be present. When working with fluorescence microscopic images of cells, it is common to stain cells. It is also possible to stain cell nuclei separately. Our approach extends the Mask R-CNN architecture to learn features for cell nuclei separately and achieves better results than using both channels directly.

3.1 Deep Learning for Textual Stamp Recognition on Index Cards of the *Lessico Etimologico Italiano*

3.1.1 Introduction

The objective of the *Lessico Etimologico Italiano* (LEI)¹ [PS79] is the documentation and historical analysis of the entire Italian vocabulary from its beginnings until today. The LEI is published since 1979 by the Academy of Sciences and Literature (Akademie der Wissenschaften und der Literatur) in Mainz, Germany, under the direction of Max Pfister, Wolfgang Schweickard, and Elton Prifti. In addition to the standard Italian language, all italo-romance dialects are taken into account. With this conception, the LEI makes an important contribution to the knowledge of Italian as well as European language history and to the preservation of the linguistic and cultural historical traditions reflected in it.

The traditional workflow of etymological dictionary research is based on the compilation of large collections of source material in the form of annotated index cards, as shown in Figure 3.1. In the LEI project, these annotated index cards were started to be digitized in recent years for two reasons: first, to store and save the precious collected material on digital storage devices, and second (and above all), to be able to process the scanned index cards automatically to speed up the redaction process of the LEI. A typical index card is shown in Figure 3.2. It contains at least three different areas of text: (a) an etymon (green box) that corresponds to an entry in the published dictionary, (b) an area containing the content, i.e., the source text (orange box), and (c) a single textual stamp that itself contains information about the origin of content (red box).

The index cards of the LEI were produced manually by the LEI team members over the years, since 1968. The total number of index cards collected for the LEI is about 8,000,000.

Typically, each index card is processed manually by a team of philologists to produce an entry for a particular word of the Italian vocabulary, taking the etymon as a starting point and considering semasiological and onomasiological aspects. The dimensions of an article of the LEI can oscillate from a few lines up to nearly 200 pages of text. Each article of the LEI is composed of four parts: a) the head, containing the etymon, morphological information as well as the meaning, b) the body, containing a summary as well as the history of the word, c) the commentary, d) the bibliography. The body of the article is composed of lexicographic information chains, systematically ordered by meaning, as well as geolinguistic and chronological classification. The completion of the LEI is planned for the year 2033.

Since 2009, several software tools are used to support the creation of the LEI [Pri22]. In particular, a collaborative work environment is provided, based on a shared database system containing entries for each word of the Italian vocabulary currently being processed, with all the required information. The final database entries are used to produce the printed edition and the online version of the LEI. The information in the database also includes the currently relevant scanned index cards.

¹<https://lei-digitale.it>



Figure 3.1: Example of boxes of collected index cards for the LEI.

The textual stamp in an index card refers to a specific literature source (e.g., "Menegus Tamburin" in Figure 3.2) and contains usually also the related geolinguistic information (e.g., "3222 lad.cador.(oltrechius.)" in Figure 3.2) as well as the chronological coordinates. The stamp is important to determine the relevant entry in the database the index card belongs to. Thus, automatically recognizing textual stamps in scanned index cards and automatically inserting the corresponding index card into the database can save significant amounts of time.

However, the recognition of textual stamps on index cards poses several challenges. For example, the stamp itself and all other parts of an index card contain text. Correctly identifying the text of interest (i.e., the stamp's text) can be difficult and cannot be solved, e.g., by standard text spotting approaches [Zha+16b; JVZ14] or scene text recognition methods [WBB11; SBY16]. Other challenges are variations in background, layout, size, and position of a stamp. Stamps can occur in various orientations, might not be complete or have very low contrast. They may be occluded, modified by hand-written corrections or partly stamped on other text, resulting in background clutter. This is particularly hard to distinguish, since both background and the object to be detected in the foreground are text. Even hand-written "stamps" occur on numerous index cards.

Another challenge is that two different stamp classes may differ only in one or few letters, which makes them visually very similar. In Figure 3.3, several particularly challenging stamps are shown. Overall, this leads to the fact that the use of available methods for stamp recognition [You+17] or text recognition [WBB11; SBY16] do not provide satisfactory results.

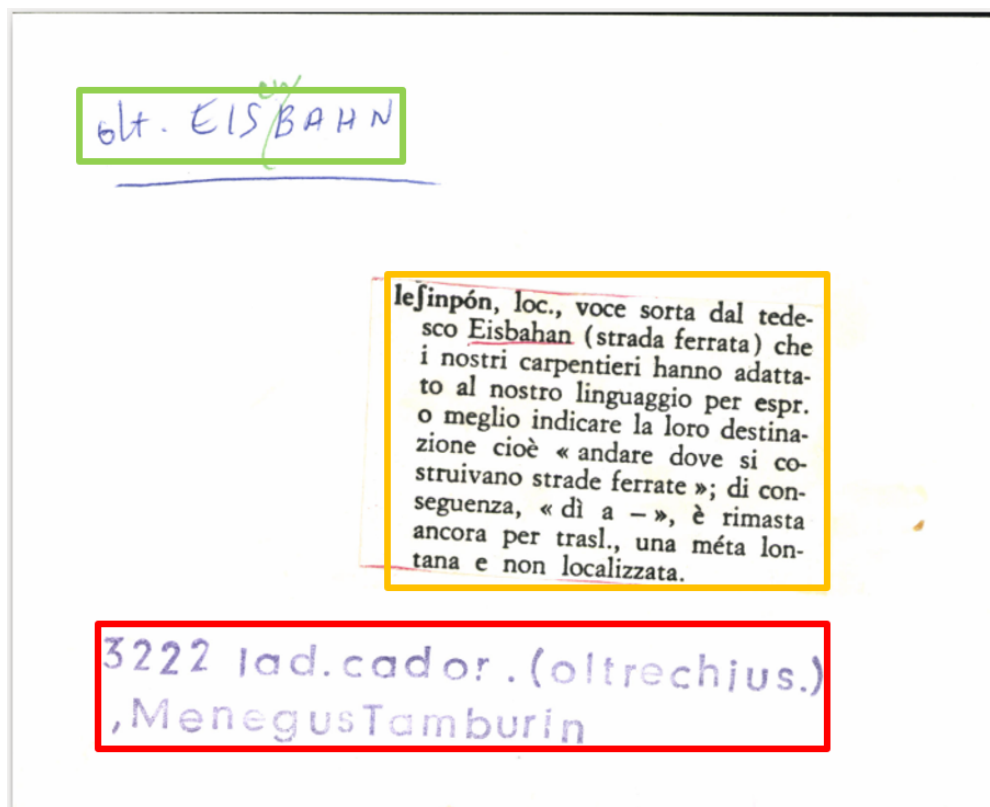


Figure 3.2: A typical index card with the stamp to be recognized (red), content (yellow), and etymon (green).

To address these challenges, we present a novel approach to automatically process a collection of digitized index cards by detecting, aligning, and recognizing textual stamps on scanned index cards. The proposed approach is based on deep learning models for the three tasks of stamp detection, alignment, and recognition, in order to speed up the lexicographic workflow of the LEI and facilitate editorial work.

The contributions of this section can be summarized as follows:

- We propose a novel deep learning approach for processing a large corpus of scanned index cards in the fields of lexicography, philology, linguistics, as well as further disciplines working with large numbers of index cards. In particular, we present deep neural network models for textual stamp detection, alignment, and recognition. To the best of our knowledge, our work is the first work proposing a solution to the problem of textual stamp recognition on index cards.
- Since the total number of stamp classes in the LEI corpus is unknown, we trained a deep learning model to map stamp images to the embedding space. These embedding vectors are compact representations of the stamps and can be used to compare two stamps and determine whether they belong to the same class. Although trained iteratively only on a small portion of the data, we demonstrate that the learned textual stamp embeddings generalize well and are highly discriminative to identify unknown stamp classes. Our

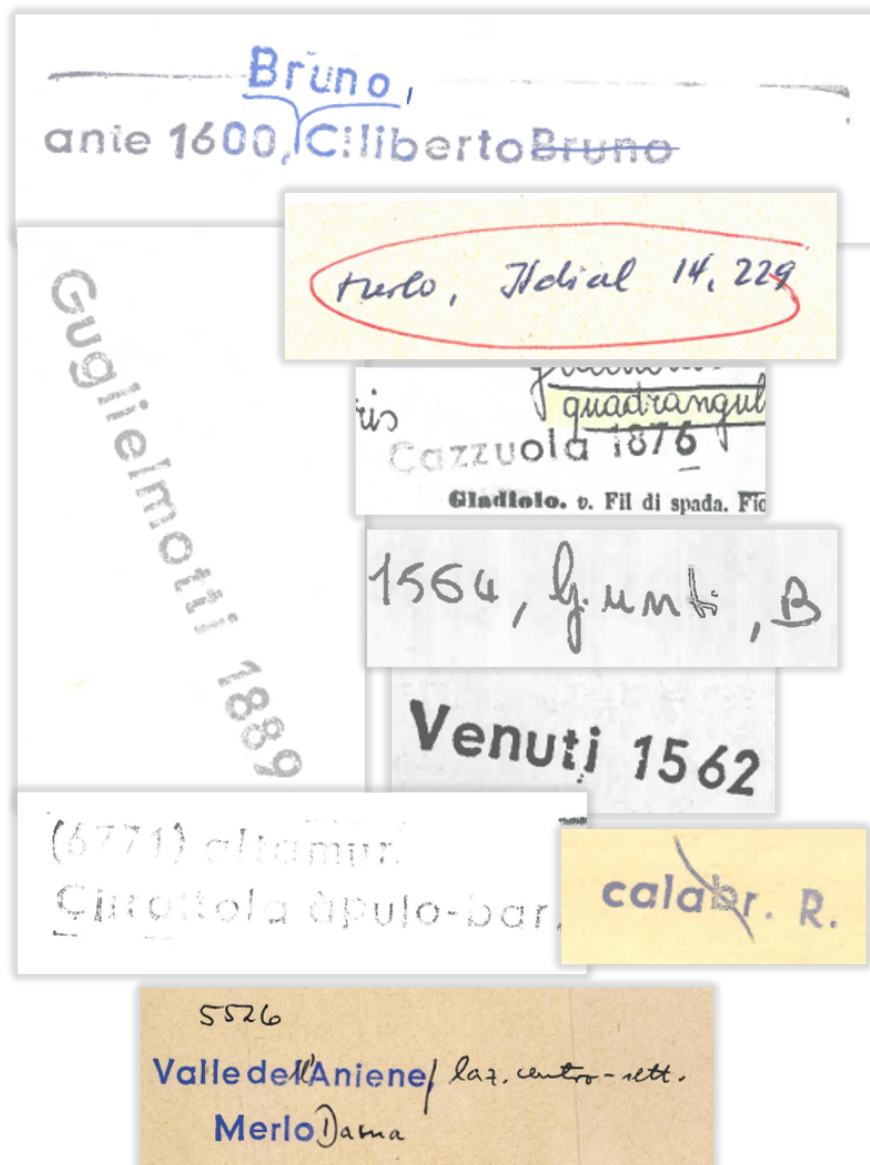


Figure 3.3: Visually challenging textual stamps.

experimental evaluations show excellent results for the stamp detection task as well as for the stamp recognition task, with a mean average precision of 98.80% and an accuracy of 97.02%, respectively.

- We present a semi-automatic stamp recognition approach based on similarity search to identify new textual stamps not present during training in the entire corpus of philological index cards and thus accelerate the traditional time-consuming workflow of etymological dictionary research.
- We make annotated datasets for textual stamp recognition on index cards available to the scientific community (i.e., 6,819 scanned index cards with bounding boxes for

stamps and 170,494 cropped and aligned stamps for recognition)². Although we focus on textual stamps in the context of the LEI, other philological projects that also digitize their analogue collections and have similar workflows based on recognizing textual stamps on index cards may profit from our work. Examples of such projects are: *Diccionario del español medieval* (DEM) [Mül87]; *Dictionnaire étymologique de l'ancien français* (DÉAF) [Bal+74]; *Dictionnaire de l'occitan médiéval* (DOM) [Ste+96]; *Dictionnaire onomasiologique de l'ancien gascon* (DAG) [BPW75].

Parts of this section are based on: Nikolaus Korfhage, Hicham Bellafkir, Markus Mühling, Markus Vogelbacher, Elton Prifti, and Bernd Freisleben. “Deep Learning for Textual Stamp Recognition on Index Cards of the *Lessico Etimologico Italiano*.” in: *Submitted; Under Review*. 2024.

3.1.2 Related Work

In recent years, deep learning methods, in particular deep convolutional neural networks (CNNs), have led to breakthroughs in many computer vision tasks, such as image classification, object detection, image segmentation, person recognition, and text spotting. To the best of our knowledge, the particular problem of textual stamp recognition in general, and the problem of textual stamp recognition on index cards for creating an etymological dictionary in particular have not been considered in the literature - neither using deep learning approaches, nor using classical computer vision methods. Nevertheless, there are related fields such as text spotting, text recognition, optical character recognition (OCR), and generic stamp recognition.

Existing methods and deep learning models in the field of text spotting [Zha+16b; JVZ14] are usually trained and applied end-to-end, but cannot be successfully applied to our corpus of scanned index cards. Text spotting is aimed at recognizing text in natural images, where the intra-class variation is usually very high. In contrast, textual stamps, and stamps in general, have a low intra-class variation, since they mostly differ in color, position, and orientation, but not in shape and size. In the case of our scanned index cards, we need to distinguish textual stamps from other text.

Similarly, OCR cannot be applied directly. Although OCR works well on textual stamps cropped and aligned accurately, it still requires a method to distinguish the unimportant text from the important text (i.e., textual stamps). To tackle this problem, we divide the task into an initial stamp detection step, a subsequent stamp alignment step, and a final stamp recognition step. Dividing the textual stamp recognition task into subtasks, rather than training an end-to-end model, eliminates the need for a sufficiently large dataset with full bounding box annotations for thousands of stamp classes. For stamp detection, we rely on the recent CNN object detection architecture Mask R-CNN [Wu+19] with feature pyramids [Lin+17a]. The cropped detection results are aligned by using a recent word-level text detection framework [Bae+19]. Finally, for stamp recognition, we use the deep learning model EfficientNet-B3 [TL19], which achieves state-of-the-art results in image recognition tasks.

Several other methods relying on classical image processing methods for special use cases (e.g., passport stamp recognition [Zaa+20], seal detection [RPL09], logo detection [ARP16;

²http://link_to_datasets

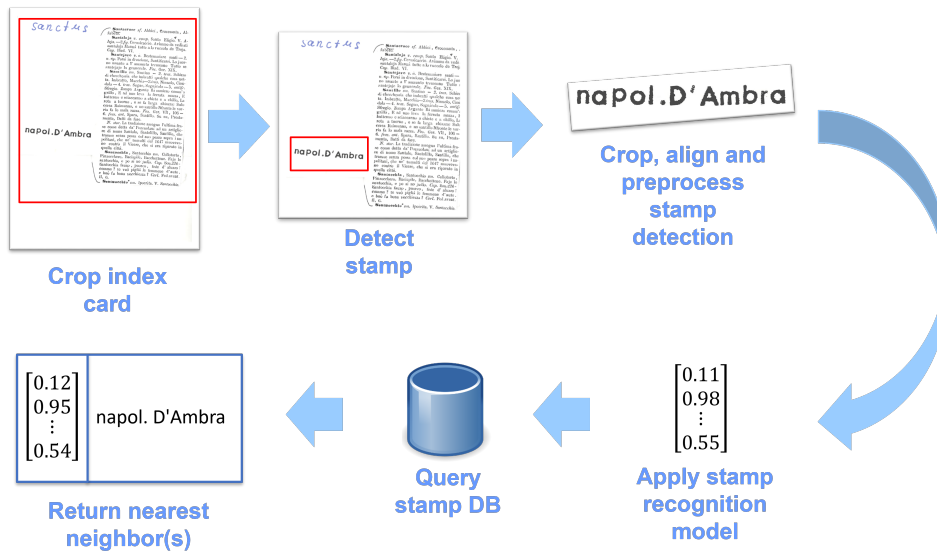


Figure 3.4: Textual stamp detection, alignment, and recognition workflow.

Cha+19] or photo timestamp recognition, [CZ03]) exist. These methods use features such as shape [FM13; DMS15], color [MB11; MBS12; Ued95], or local keypoint descriptors [Ahm+13; DMS15; FM13]. They assume that a fixed set of stamps is represented by the corresponding features. Thus, these methods are not applicable to the more challenging problem of (arbitrary) textual stamp recognition. A method for the segmentation of textual, graphical, official, and fun purpose stamps based on fully convolutional neural networks has been presented by Younas et al. [You+17]. It predicts whether a pixel belongs to a stamp or to background. In our case, pixel-level segmentation is not necessary. This approach could be used to derive the bounding boxes from the segmented regions, but the training of such a model requires ground truth segmentation masks for the index cards. However, providing this information during the manual data acquisition phase is very expensive and time-consuming.

Finally, another requirement that is not fulfilled by any of the available stamp detection or stamp recognition approaches, is the ability to re-identify stamp instances that have not occurred in the training data. With increasing sizes of the underlying database of index cards and stamps, respectively, the need for an efficient automatic indexing strategy arises.

3.1.3 Method

In this section, we present our proposed approach for textual stamp recognition on index cards. Figure 3.4 shows the entire workflow from a scanned document to the final assignment of a stamp class. It consists of the following steps:

1. The scanned document is automatically cropped to the detected borders of an index card.

3 Detection and Segmentation

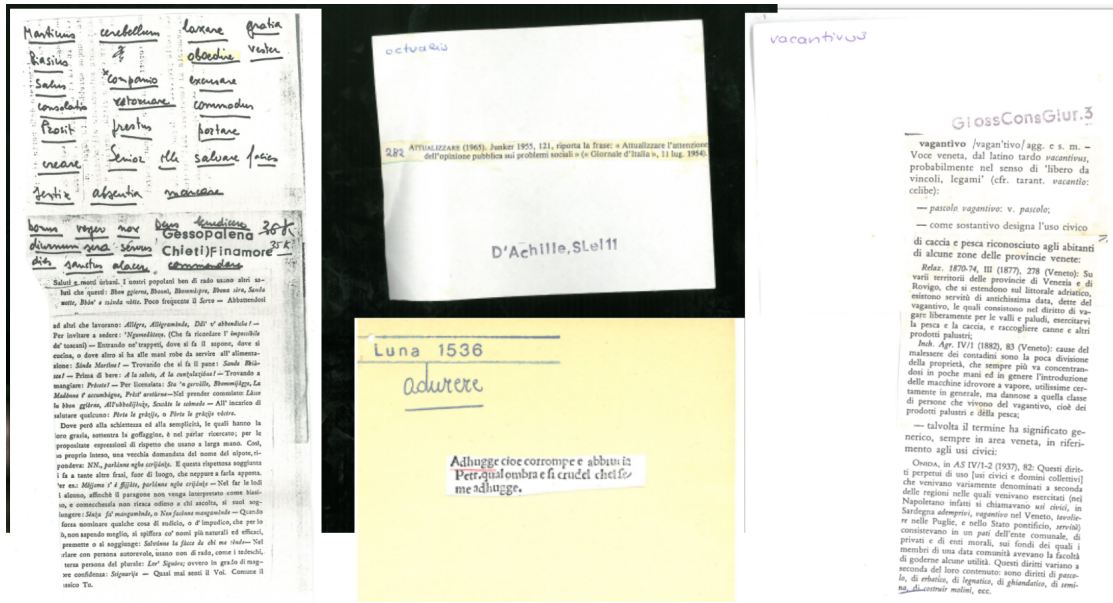


Figure 3.5: Variations of index cards.

2. The deep neural detection model is applied to identify stamp region, content, and etymon. While the other object classes (content and etymon) are of interest and further processed in other contexts in LEI, we focus only on the detected stamps.
3. Since the detected textual stamps may have various orientations, the stamp region is aligned using a character-level text detection approach.
4. The crop of the aligned stamp is fed to the deep neural stamp recognition model that assigns the corresponding stamp class.

The individual steps are discussed in more detail below.

Index Card Cropping

A large number of index cards had to be scanned manually due to their variable extents and attachments. Thus, the documents were often not captured accurately. In addition to the relevant region of an index card, the scanned documents contain various backgrounds, such as hands, black or white regions, and other noise such as light artifacts (as shown in Figure 3.5). Therefore, we cropped the scanned documents automatically to the region of the index card. Due to different forms of background clutter in the corpus of the scanned index cards, we developed a robust method³ for automatically cropping the relevant regions of the index cards. The method uses image processing methods based on the OpenCV library [Bra00] for detecting the scanned index card boundaries on various backgrounds. The cropped index cards are then passed to the neural network for stamp detection.

³https://github.com/nik-ko/crop_scans

Textual Stamp Detection

We trained a deep neural stamp detection model that not only detects stamps, but also other regions of index cards. These regions are further processed within the LEI project, but are not considered in the subsequent steps of our textual stamp recognition workflow. Altogether, we detect three object classes: stamp, content, and etymon (see Figure 3.2). We use the Faster R-CNN [Ren+15] deep neural network architecture with feature pyramids [Lin+17a] and a ResNext backbone deep neural network [Xie+17]. Since the training of a deep neural network for object detection requires millions of training images and many object regions, we follow a fine-tuning strategy. We initialized the parameters of the base model using pre-trained weights learned on the COCO object detection dataset [Lin+14], which itself was used to initialize the ResNext backbone on pre-trained weights of ImageNet [Rus+15]. We trained the object detection model using the Detectron2 framework [Wu+19].

Furthermore, we applied data augmentation to extend the dataset by flipping the images horizontally and vertically. Additionally, we rotated the images by 90°, 180° and 270°, respectively. We trained the object detection network for 29 epochs on the stamp detection dataset.

After detecting a stamp on an index card, the stamp identity needs to be recognized. However, the stamps are often not aligned horizontally and therefore require an alignment step.

Textual Stamp Alignment

Textual stamps may occur in very different orientations, which may result in quite large bounding boxes with a large proportion of background. This background, especially if it is text, may hinder the subsequent stamp recognition. For example, the extreme case occurs when a stamp is rotated by 45 degrees. Another option would be to train with rotated bounding boxes instead. However, this would require the manual annotation of rotated bounding boxes, which would be significantly more time-consuming. For this reason, we decided not to use rotated bounding boxes.

To obtain well aligned stamps, we utilized a character-level text detection approach [Bae+19]. The predicted rotated word-level bounding boxes are used to infer the rotation angle of the stamp. Since there may be multiple detections per stamp, the calculation of the rotation angle is based on the largest word-level bounding box.

Using the inferred rotation angle, we rotated the cropped stamps accordingly. Next, we applied the stamp detection model again and cropped the detected stamp region once more. This approach leads to high-quality aligned crops of textual stamps that are the input for the subsequent stamp recognition task.

Textual Stamp Recognition

The exact number of stamp classes in the entire corpus of index cards will not be known until the entire corpus of the LEI has been processed. Therefore, it is unknown at the current stage of the LEI project how many different textual stamps occur in the entire dataset of index cards. Hence, we need to create a database of known stamp classes in an iterative manner. For this

purpose, we do not classify all stamps directly, but learn a mapping from stamp images to feature embeddings, where distances in the embedding space correspond to a measure of similarity between stamps. This embedding can then be used to cluster, verify, or classify stamp images by calculating distances between feature embeddings. For classification, the feature embedding of an unknown stamp is compared to all embeddings in the database of known stamps, and the stamp class with the smallest Euclidean distance is assigned.

The workflow of textual stamp recognition in the context of LEI using a database of known stamps is presented in detail in Section 3.1.3. In this section, we describe our approach for iteratively obtaining a deep neural stamp recognition model that can be used to extract stamp embeddings.

The index cards in the LEI are grouped by the first letter of their etymons. As a data basis for building the recognition model, we selected the stamps of the index cards with the first letter *P* and *S* in their etymons. This is motivated by the assumption that this subset contains a large number of different stamp classes, since many words of the Italian language begin with the letters *P* or *S*.

Due to the large manual effort for the acquisition of training data, we generated the ground truth data semi-automatically and performed the training of the neural stamp recognition model in several steps, as described below.

Initial Textual Stamp Recognition Model

To obtain training data for an initial textual stamp recognition model, we applied the stamp detection model to all index cards of etymons beginning with letter *P*. In the first step, we applied the Tesseract OCR software [Smi07] to the aligned and cropped stamp images. We clustered the resulting text representations hierarchically using average linkage clustering [Sib73]. The distance function between two clusters *A* and *B* is given by

$$\frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d_L(a, b).$$

where $|A|$ is the number of text representations in cluster *r*, $|B|$ is the number of text representations in cluster *s*, and d_L is the Levenshtein distance between two text representations (stamps). The stamps are agglomeratively clustered until the distance exceeds a predefined threshold. The threshold is manually chosen so that the purity of the clusters with respect to the stamp classes is as high as possible.

After clustering, we revised the resulting groups manually, e.g., we deleted incorrectly assigned stamps and merged groups representing the same stamp class. In total, we obtained a dataset of 13,882 stamp images with 557 different stamp classes. However, the distribution of the stamp classes is very imbalanced. Many classes have only one or a few samples. Therefore, we applied data augmentation to increase the number of examples per class as well as to add new synthetic stamp classes to the dataset. For the synthetic stamp classes, we used a generated list of “pseudo stamps” similar in structure to real stamps used in philological research. The generation of synthetic stamp images works as follows: a background image is randomly selected from a set of 270 index cards. Since the layout of the index card is known

due to the preprocessing by the object detection model, the synthetic stamp can be drawn to a random region, non-overlapping with the original stamp region. The style of the stamp is randomly chosen from 7 free stamp-like fonts. Finally, the corresponding stamp region is cropped. Examples of synthetically generated stamp images are shown in Figure 3.6. Altogether, we obtained an augmented training dataset of 1,800 stamp classes with 1,000 samples per class. Based on this dataset, we trained an initial stamp recognition model using an EfficientNet-B3 architecture [TL19] and cross-entropy loss pre-trained on ImageNet[Rus+15].

Final Textual Stamp Recognition Model

The initial deep neural network model is applied to the stamp images of index cards for etymons beginning with *S* in the second step. Again, the examples are hierarchically clustered, but this time based on the deep neural network features extracted from the last layer of the initial model. Again, we corrected the resulting groups manually. We removed noise samples, such as false detections, and we merged stamp classes that were wrongly partitioned into multiple clusters. This additional effort results in a high-quality stamp recognition dataset that we used for training and validating the final neural network model on 170,494 images of 4,304 different stamp classes.



Figure 3.6: Examples of synthetic textual stamps.

Semi-automatic Stamp Recognition via Similarity Search

In this section, we present a semi-automatic workflow for indexing the entire corpus of scanned index cards. We can not simply apply our trained classification model to the whole corpus, since it is currently unknown how many and which stamps are present in the LEI corpus. However, when indexing the corpus, many of the index cards can already be assigned to one of the stamps present in the stamp database, which in the beginning contains the stamps used to train the model.

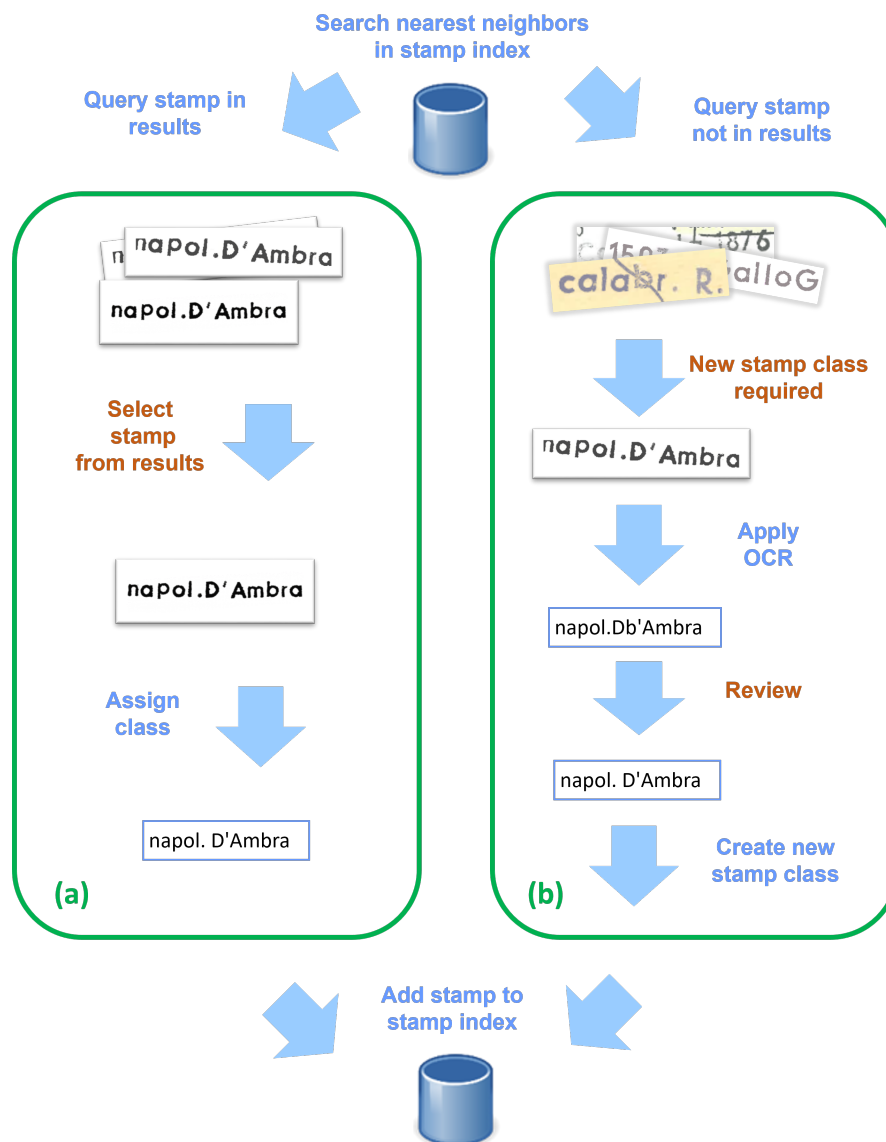


Figure 3.7: Semi-automatic stamp recognition workflow. A stamp is either assigned to an existing stamp class and added to the stamp index (a) or a new stamp class is created and the stamp is added to the index (b).

As the digitization of LEI's index cards continues, many index cards with previously unseen stamps will appear. Instead of continuously learning new deep neural network models, which requires manual labeling in each iteration, we want to use the learned neural network model to eventually index the entire dataset of scanned index cards.

We now outline our approach to identify new, unseen stamps and to semi-automatically add them to the stamp database. The entire semi-automatic workflow is shown in Figure 3.7. Steps that require user interaction are colored in orange.

We assume that the new stamps will come from a similar distribution as the stamps used for training. Therefore, the learned feature embeddings of the trained stamp classification model can be used to identify new stamp classes without retraining the model. Thus, after training the stamp classification model as described in Section 3.1.4, we applied the neural stamp recognition model to a dataset of about 1.6 Million cropped stamps that we used in this section to (a) extract feature vectors, and (b) find examples that are not assigned to any class with certainty by the neural network model. The stamps with low confidence for any trained class are frequently new stamps that were not present during training. Such new stamp representations can be easily added to a database with a new stamp label for classifying future occurrences of this stamp in a semi-automatic workflow for indexing the entire corpus. For all stamps assigned to a class we extracted feature vectors from the penultimate layer of the deep neural network. We normalized these 1,536-dimensional embeddings and used them to build a Faiss index [JDJ19] for finding similar stamps.

First, we took the annotated stamps from the training and validation sets to set up an initial database of stamp representations, as shown in Figure 3.4. We used feature vectors obtained from the classification model to map the stamps to a high dimensional stamp embedding space. During processing the corpus of stamped index cards, this database can be queried. If a query stamp matches a stamp from the database of already processed stamps (i.e., a nearest neighbor with a stamp label assigned shows the same stamp as the query stamp), the corresponding stamp class can be assigned to this index card (Figure 3.7 a). Since an accuracy of 100% is necessary to satisfy the quality requirements of LEI, this assignment needs to be confirmed by a human expert: The stamp classes of the nearest neighbors found in the stamp index are presented to the user from which the user selects the class representing the query stamp. Otherwise, if none of the nearest neighbors found does show the same stamp as the query stamp, it is assumed that the stamp is not present in the database. In this case, the stamp needs to be added to the database (Figure 3.7 b). To store the new stamp, a class name is required, which in case of textual stamps will be the stamp's text. The new stamp is first processed via OCR. Then, the result is presented to the human expert who may correct the stamp text if necessary and then confirm the stamp text. Finally, a new stamp class is added and the stamp is assigned to that class and inserted into the stamp index.

3.1.4 Results

We evaluated the detection and recognition performance of our approach separately. Additionally, we present qualitative results of the nearest neighbor search in the stamp embedding space for unseen stamps.

Datasets

The proposed deep neural network models for detection and recognition are trained on different datasets. In the following, we will explain in more detail how the datasets for detection and recognition were compiled.

Our detection dataset consists of 6,991 scanned index cards containing bounding box annotations of which 6,759 are stamps, 13,961 etymons, and 7,012 content. All index cards were

Class	Training	Validation	Total
Stamp	6115	644	6759
Etymon	12731	1230	13961
Content	6349	663	7012

Table 3.1: Scanned index card detection dataset.

manually annotated by human experts. For annotating the stamp detection dataset, the human experts used the labeling tool Labelbox⁴. We split the data into a training and a validation dataset where we used 10% of the index cards for validation. The number of bounding box annotations per class and dataset are shown in Table 3.1.

Our recognition dataset contains 170,494 images of 4,304 different stamp classes, which were iteratively collected, as described in 3.1.4. Each sample is a detected stamp cropped from a scanned index card. The distribution of the samples is highly imbalanced, since only 756 stamp classes have more than 50 samples and 1,131 classes have less than 5 samples. As shown in Figure 3.8, many classes have indeed very few training samples. We do not consider this to be a particularly serious problem, since for stamps, variation in their appearance is usually very limited. The unequal distribution of the stamp classes can be counteracted during the training by sampling from a uniform distribution.

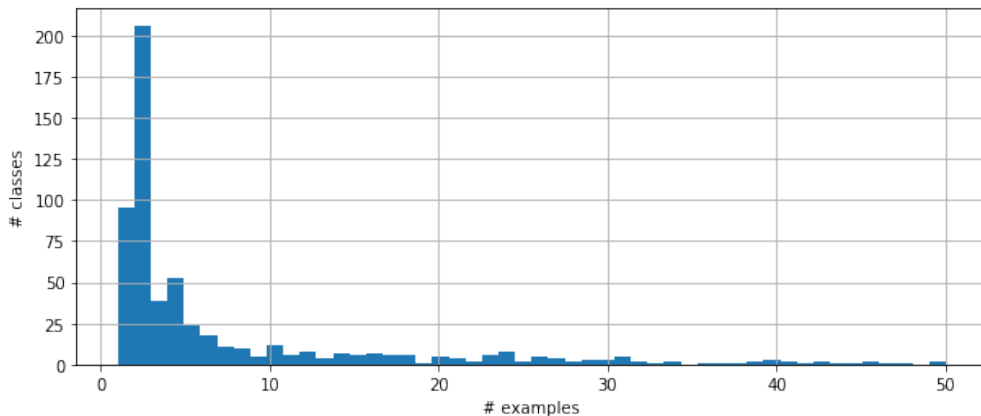


Figure 3.8: Number of classes with less than 50 training samples.

From the imbalanced stamp dataset, we used 4,304 stamp classes for training. However, for validation, we only considered stamps with at least 40 examples, which holds for 899 stamp classes. The validation dataset has 10 images per stamp class, resulting in 8,990 stamp images for validation. The remaining 161,504 images were used for training.

⁴<https://labelbox.com>

Class	$AP_{IoU=0.50}$	$AP_{IoU=0.75}$
Stamp	0.988	0.953
Etymon	0.980	0.805
Content	0.988	0.954

Table 3.2: Stamp detection performance.

Textual Stamp Detection

We evaluated our deep neural stamp detection model in terms of average precision (AP). AP summarizes the shape of the precision-recall curve. It is defined as the mean precision at a set of equally spaced recall levels. AP is computed for different Intersection over Union (IoU) thresholds. IoU between sets of pixels A and B is computed as

$$IoU(A, B) = \frac{A \cap B}{A \cup B} \quad (3.1)$$

on the 100 top-scoring detections per image for bounding boxes and segmentation masks, respectively. For example, for threshold $t = 0.5$, detections with an IoU overlap of at least 50% with a ground truth object are counted as true positives (TP). Unmatched predicted objects are false positives (FP) and unmatched ground truth objects are false negatives (FN). Precision is computed as $p = \frac{TP}{TP+FP}$ and recall as $r = \frac{TP}{TP+FN}$. We use the 101-point interpolated AP, similar to Lin et al. [Lin+14]. AP for a given IoU threshold t is computed over all images in the test set as

$$AP(t) = \frac{1}{101} \sum_{r \in \{0, 0.01, \dots, 1\}} p_{interp}(r), \quad (3.2)$$

where $p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$ and $p(\tilde{r})$ is the measured precision at recall \tilde{r} .

On the validation data, we achieved $AP_{IoU=0.50} = 0.988$ for stamps. In addition to stamps, the object classes *content* and *etymon* were also detected. The full results are shown in Table 3.2.

Textual Stamp Recognition

To evaluate the performance of the deep neural stamp recognition model, we computed its classification accuracy on the validation set as

$$Accuracy = \frac{\text{number of correctly classified stamps}}{\text{total number of stamps}}. \quad (3.3)$$

Table 3.3 shows the results of different methods for text recognition applied to the detected textual stamps. As a baseline we apply the Tesseract OCR software [Smi07] without alignment on the cropped detections. The vocabulary are the 3,817 textual stamp strings and the OCR results are matched to the closest string via the Levenshtein distance. This approach can only achieve good results for clearly visible, not occluded, and well-aligned stamps. Since many stamps in the dataset meet these conditions, we can already achieve 81.904% accuracy with

Method	Accuracy
OCR + no alignment	81.904
OCR + alignment	85.997
Recognition + no alignment	90.466
Recognition + alignment	96.754
Recognition + rotation augmentation	89.849
Recognition + alignment + synthetic stamps	97.016

Table 3.3: Stamp recognition performance.

k	Weight	Accuracy
1	uniform	97.197
3	uniform	97.286
3	distance	97.297
5	uniform	97.319
5	distance	97.386
10	uniform	97.297
10	distance	97.342

Table 3.4: k -Nearest neighbor performance on the validation data.

this simple approach. By aligning the cropped stamps, the OCR approach can achieve 85.997% accuracy.

For our trained neural stamp recognition model, we evaluated different settings. Without any data augmentation or alignment, 90.466% accuracy is achieved. While using rotation augmentation does not lead to an improvement, aligning the stamps greatly improves the accuracy to 96.754%. We applied stamp alignment on training and validation images equally. Furthermore, using synthetic stamps improved the results slightly to 97.016% accuracy. We assume that the slight improvement by data augmentation with synthetic stamps is due to the fact that stamps have relatively little intra-class variation.

Quality of Stamp Embeddings

In our semi-automatic stamp recognition workflow, we use the embeddings obtained by the classification model to compare stamps. We first evaluated the proposed nearest neighbor approach in terms of accuracy for the validation images. We used the training data to assign a stamp class via the k nearest neighbor approach to stamps from the validation set. The k nearest neighbors are weighted either uniformly by 1 or by the inverse distance in the embedding space, i.e., a closer element has a higher weight. Table 3.4 shows the accuracy for different settings of the k nearest neighbor approach. The k nearest neighbor accuracy is similar to the classification accuracy in Table 3.3 for all settings. The best accuracy is achieved with $k = 5$ and weighting by distance. This confirms that the features of the penultimate layer can be used to assign stamp classes to query stamps.

To be useful for creating new stamp classes in the semi-automatic stamp recognition workflow, it is necessary that the stamp embeddings generalize well. We therefore computed AP and accuracy for some query examples showing stamp classes that were not present in the training data.

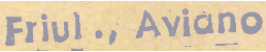
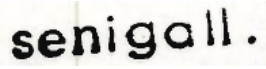


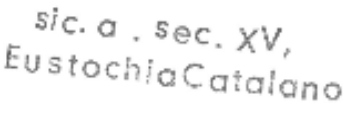



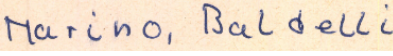

Query stamp	AP@10	AP@5	AP@3	ACC@5
	100.00	100.00	100.00	100.00
	100.00	100.00	100.00	100.00
	100.00	100.00	100.00	100.00
	100.00	100.00	100.00	100.00
	94.30	100.00	100.00	100.00
	75.75	80.34	91.67	60.00
	77.95	84.34	91.67	60.00
	83.44	96.67	100.00	100.00
	89.61	96.67	100.00	100.00
	86.46	96.67	100.00	100.00

Table 3.5: Evaluation of the retrieval performance for some unknown stamp classes

To find stamp classes the model was not trained for, we selected some stamps from the set of stamps that the model assigned to a stamp class of the training data set with low confidence. These stamps were not assigned to any class of the set of stamp classes used for training with high confidence and can thus be crops of new stamps that did not occur in the training data. For several of these stamps, we checked that they indeed show new stamp classes and used 10 of them to query the stamp index.

For these 10 query images, we computed the AP for the top k retrieved stamps for $k \in \{3, 5, 10\}$ and the accuracy for the top 5 retrieved stamps, as shown in Table 3.5. For example, an accuracy of 60% means that 3 of the top 5 stamps belong to the correct class. For some stamps, very high AP values are obtained. For some other stamps at the bottom of table, the AP@10 values are lower, but nevertheless our approach is still useful in practice, since even for AP@10 values of around 75%, most of the retrieved stamps belong to the same stamp class. In our scenario, it makes more sense to consider an even shorter retrieval list, i.e., the first 3 or 5 items, since within the semi-automatic stamp recognition workflow, only a small number of stamps are presented to the user, e.g., the top 5. For $k = 5$, the AP and accuracy values are still very high. If we only consider the top 3 stamps of the retrieval list, all stamps often belong to the correct

class, resulting in both accuracy and AP values of 100%. These results indicate that the neural stamp recognition model can be successfully used to create new stamp classes that have not been present in training data.

3.1.5 Summary

We presented a novel approach for textual stamp detection, alignment, and recognition on index cards of the LEI. Our approach keeps the manual annotation effort as low as possible, while achieving high quality results. Using a trained stamp embedding for similarity search allowed us to continuously grow the stamp database used to assign new index cards to stamp classes as digitization of index cards proceeds. Since both the neural stamp detection and recognition components of our semi-automatic stamp recognition workflow achieve very good results in terms of mean average precision (detection) and accuracy (recognition), they are of great benefit for processing index cards in the context of creating the LEI etymological dictionary. We showed that the learned stamp embeddings can generalize well to unknown stamps and thus can be used to identify stamps the neural network models were not trained for. The presented approach could also be helpful for other projects in the field of philological research in digitizing their scanned index cards and speed up the digitization process.

3.2 Detection and Segmentation of Morphologically Complex Eukaryotic Cells in Fluorescence Microscopy Images via Feature Pyramid Fusion

3.2.1 Introduction

High-throughput cell biology incorporates methods such as microscopic image analysis, gene expression microarrays, or genome-wide screening to address biological questions that are otherwise unattainable using more conventional methods [RSS15].

Nevertheless, fluorescence microscopy is often avoided in high-throughput experiments, since the generated data is tedious to analyze by humans or too complex to analyze using available image processing tools. However, fluorescence microscopy offers information about subcellular localization, supports morphological analysis, and permits investigations on the single cell level [Usa+16].

A limiting factor of automated fluorescence microscopy image analysis is the separation of signals in close proximity, regardless of whether signals originate from neighboring cells or single spots. High cell densities or cluster formation increase the probability of such situations on the cellular level [Che+15], while high background or low spatial resolution complicate the problem on the signal level. Another limitation is the detection of morphologically complex cells, such as macrophages or neurons. Their indefinite morphology causes identification issues when looking for slight variations of fixed shapes.

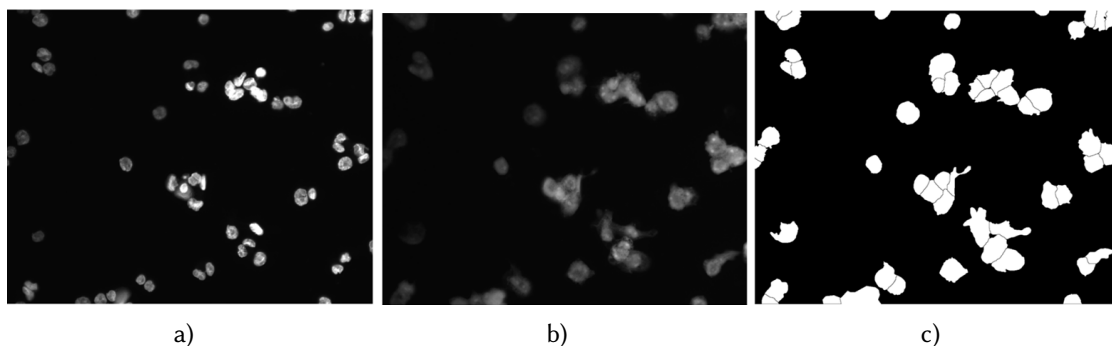


Figure 3.9: Image with ground truth segmentation. Nucleus signal (a), cytoplasm signal (b), and ground truth segmentation (c).

The fluorescence microscopy images considered in this section originate from high-throughput screening, with the aim of analyzing phenotypic changes and differences in bacterial infection rates of macrophages upon treatment. To analyze cell infection and changes in cell morphology, proper cell detection and segmentation are required. Upon adhesion to a surface, these cells tend to form clusters showing faint signal changes in areas with cell to cell contact. These restrictions prevent appropriate analysis with conventional microscopy analysis software.

Compared to merely segmenting cytoplasm, instance-based segmentation is a much harder task, since the assignment of a cell instance identity to every pixel of an image is required. Fig 3.9 shows that the accurate separation and segmentation of clustered cells without any

further information is extremely difficult, if not impossible. Even for human experts, the correct separation of individual cells is often only possible using nucleus information. Fig 3.9 indicates that taking the nucleus signal into account alleviates the identification of individual cells significantly.

In this section, a deep learning approach to cell detection and segmentation based on a convolutional neural network (CNN) architecture is applied to fluorescence microscopy images containing channels for nuclei and cells. The contributions in this section are as follows:

- We provide a novel dataset of macrophage cells for public use, including ground truth bounding boxes and segmentation masks for cell and nucleus instances.
- We utilize nucleus information in a deep learning approach for improved cell detection and segmentation. The nucleus channel is used to improve the quality of cell detection and segmentation. To the best of our knowledge, this is the first deep learning approach that uses additional nucleus information to improve cell detection and segmentation.
- We present a CNN architecture based on Mask R-CNN and a novel feature pyramid fusion scheme. This CNN architecture shows superior performance in terms of mean average precision compared to early fusion of nucleus and cell signals. It clearly outperforms a state-of-the-art Mask R-CNN [He+17] applied to cell detection and segmentation with relative mean average precision improvements of up to 23.88% and 23.17%, respectively.

Parts of this section have been published in: Nikolaus Korfhage, Markus Mühling, Stephan Ringshandl, Anke Becker, Bernd Schmeck, and Bernd Freisleben. “Detection and Segmentation of Morphologically Complex Eukaryotic Cells in Fluorescence Microscopy Images via Feature Pyramid Fusion.” in: *PLOS Computational Biology* 16.9 (9 Sept. 2020), e1008179. ISSN: 15537358. doi: 10.1371/JOURNAL.PCBI.1008179.

3.2.2 Related Work

Our approach is related to several instance segmentation approaches for fluorescence microscopy images. Methods that do not rely on deep learning, such as graph cut algorithms [Al+10; Abr+17; Dim+14; Cai+19a; Cai+19b], usually struggle with morphologically complex objects. Similarly, other methods either consider nuclei [CR+09; Gud+08; Al+10] or segment similarly sized and mostly round objects [Sch+18] or different shapes [Sol+17; Ama+14]. Most notably, all of these methods consider only a single signal, either cell or nucleus. A method utilizing nucleus information together with the cell signal is described by Held et al. [Hel+11] and Wenzel et al. [Wen+11]. Their segmentation algorithm uses a fast marching level set [Set99], i.e., a classic computer vision method that does not rely on machine learning methods. A more recent approach proposed by Al-Kofahi et al. [Al+18] refines deep learning based nucleus segmentation by a seeded watershed algorithm to segment cells.

Recent CNN-based segmentation algorithms can be roughly divided into two groups. Algorithms in the first group perform full image segmentation and require additional post-processing to be applicable to instance segmentation [LSD15; NHH15; BKC17]. A well-known example in bio-medical image segmentation is the U-Net [RFB15] architecture. However, such methods

fail in case of clustered cells. Due to few misclassified pixels, neighboring cells are fused into a single cell, resulting in poor detection performance.

For this reason, we consider methods of the second group of CNN-based segmentation algorithms to be more suitable for our data. These methods perform detection followed by segmentation, i.e., detected bounding boxes are segmented rather than the whole image. For example, van Valen et al. [Van+16] use object detection and perform bounding box segmentation in a subsequent step. Akram et al. [Akr+16] describe a CNN architecture using region of interest (RoI) pooling for cell detection and instance-based segmentation. Recently, Mask R-CNN [He+17] based on Faster R-CNN [Ren+15] with feature pyramids [Lin+17a] achieved state-of-the-art results in natural image object detection and segmentation. Hence, our nucleus and cell detection and segmentation approach is based on Mask R-CNN. However, in contrast to recent work applying Mask R-CNN to a number of segmentation problems in biomedical imaging including nucleus segmentation [Som+19; Joh19; VAK19], we extend the architecture to meet the requirements of data sets containing both cell and nucleus signals.

3.2.3 Design and Implementation

The fluorescence microscopy images used in our work are collected as part of high-throughput screening to detect treatments that have an influence on the bacterium *Legionella pneumophila* and modify the infection of human macrophages. Microscopic images do not only allow assessment of bacterial replication, but also enable investigations of morphological changes.

Cell preparation and image acquisition

Monocytic THP-1 cells were obtained from ATCC, inoculated from a -80°C culture and passaged in RPMI-1640 medium with 10% fetal calf serum (Biochrom) at 37°C and 5% CO_2 . The used cell passage numbers were in the range of 5 to 14. Cells were seeded in 100 μl on 96 well Sensoplate Plus plates (Greiner Bio-One) at a concentration of 1.45×10^4 cells per well. Differentiation to macrophages was induced 24 h after seeding by adding 20 nM phorbol 12-myristate 13-acetate (PMA; Sigma-Aldrich) for 24 h. After medium renewal, cells were infected with GFP-expressing *Legionella pneumophila* strain Corby at a multiplicity of infection of 20 for 16 h. For infection, bacteria were plated on BCYE agar, incubated for 3 days at 37°C and 5% CO_2 , resuspended and added to macrophages [Jun+17]. After infection, the cells were washed, fixed with 4% paraformaldehyde for 15 min and permeabilized with 0.1% Triton X-100 for 10 min. Staining of cells was achieved with HCS CellMask Red (2 $\mu\text{g}/\text{ml}$; Thermo Fisher Scientific) and Hoechst 33342 (2 $\mu\text{g}/\text{ml}$; Invitrogen) for 30 min.

The images were acquired using an automated Nikon Eclipse Ti-E fluorescence microscope with a 20x lens (Nikon CFI Plan Apo VC 20X) and a Nikon Digital Sight DS-Qi1Mc camera. Finally, the images were converted to TIFF format via the Fiji/ImageJ Bio-Formats plugin.

In total, several hundred thousands of 3-channel images with a size of 1280×1024 pixels were collected.

Dataset of macrophage cells

For our benchmark dataset, 82 representative images were selected and augmented with ground truth information by a biomedical researcher. In our work, only the nucleus and cell channels are of interest.

These 82 2-channel images were randomly split into a training set, containing 64 images, and a test set, containing 18 images. While the training set contains 2044 cells and 2081 nuclei, the test set contains 508 cells and 514 nuclei. The discrepancy between the number of cell and nuclei instances is due to border cells that are not completely visible in the images. Furthermore, some of the cells are in their mitotic phase, i.e., multiple nuclei may occur in a single cell.

Manually generating ground truth labels for segmentation tasks is very time-consuming. Therefore, we generated the ground truth segmentation masks for cell and nuclei instances in a two-step process. In the first step, classical computer vision algorithms were applied to produce an initial segmentation using a threshold-based approach and contour processing functions from the OpenCV library [Bra00] to find cell and nuclei instances. Furthermore, the structures of these instances were analyzed, and cells with more than one nucleus were separated similar to a Voronoi segmentation where each pixel of a cell is assigned to the nearest nucleus. In the second step, these segmentations were manually refined using an image processing tool designed for scientific multidimensional images called Fiji/ImageJ [Sch+12]. Most of our manual corrections were necessary at cell to cell borders.

To segment macrophage cells, we follow an instance-based segmentation approach by introducing a new neural network architecture based on Mask R-CNN. It combines nucleus and cell features in a pyramid fusion scheme. The Mask R-CNN [He+17] architecture is an extension of a Faster R-CNN [Ren+15] that predicts bounding boxes with class probabilities and additional instance-based segmentation masks.

Like Faster R-CNN, Mask R-CNN is a two-stage method. In the first stage, a Region Proposal Network (RPN) is used to find object proposals (regions of interest, RoI). Therefore, candidate bounding boxes together with objectness scores are predicted. In the second stage, features are extracted for these candidate bounding boxes using a RoI Align layer that computes fixed-size feature maps through bilinear interpolation. Based on these feature maps, the candidate bounding boxes are refined, classified and segmented using regression, classification, and segmentation heads, respectively. The features of the two stages are shared in a backbone architecture for runtime improvements. The CNN backbone architecture is typically pre-trained on an image classification task. The overall neural network is fine-tuned and trained end-to-end using the multi-task loss $L = L_{box} + L_{cls} + L_{mask}$, where L_{box} is the bounding box regression loss, L_{cls} is the classification loss, and L_{mask} is the mask loss (i.e., per pixel sigmoid with binary loss), respectively.

To improve detection and segmentation performance within small object regions, Mask R-CNN relies on a Feature Pyramid Network (FPN) [Lin+17a], a top-down architecture with lateral connections to the backbone layers for building high-level semantic feature maps at all scales. Feature maps for regression, classification, and segmentation of candidate bounding boxes are extracted from the pyramid level according to bounding box sizes. Larger bounding boxes are

assigned to higher levels of the pyramid and smaller boxes to lower levels, respectively. In Fig 3.10, the underlying Mask R-CNN architecture is highlighted in green.

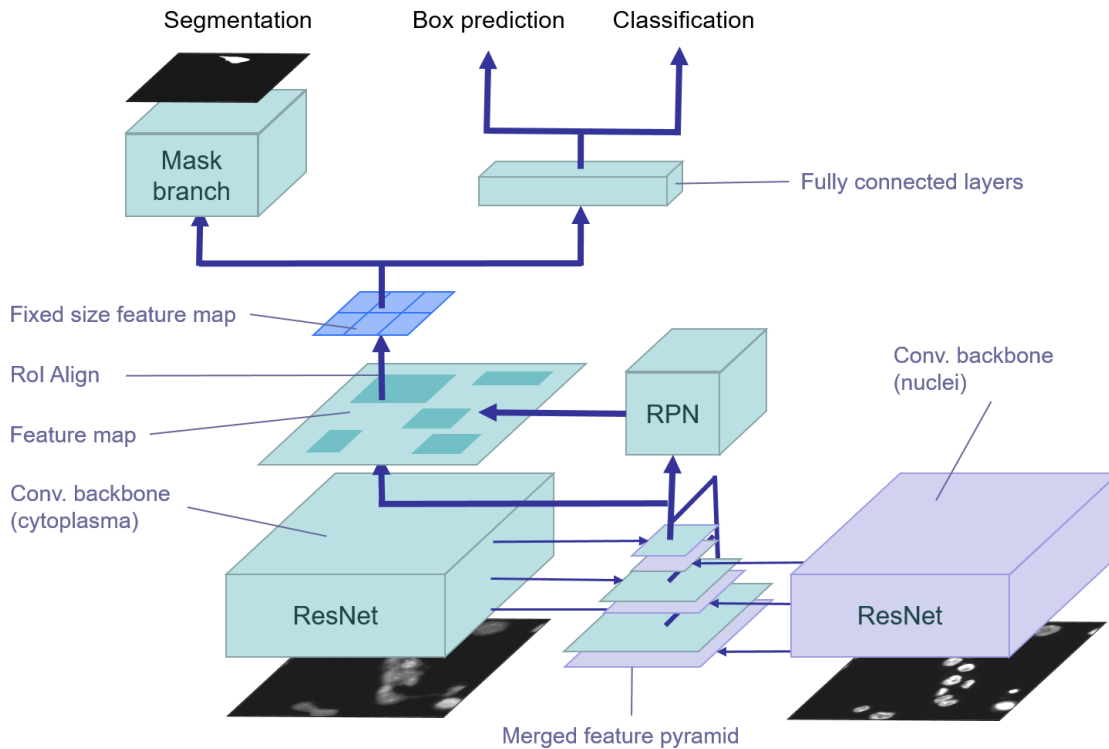


Figure 3.10: Feature pyramid fusion of nucleus features. Pre-trained nucleus features (violet) are fused with features of the feature pyramid in the cell detection and segmentation model (green) by either concatenation or addition.

The remainder of this section is organized as follows. We first describe the used backbone CNN. Then, the new network architecture using a pyramid fusion scheme to integrate nucleus features for cell detection and segmentation is presented.

Furthermore, a weighted segmentation loss is introduced to focus the training process on difficult to segment pixels at the cell borders. Finally, we describe the post-processing steps to further improve the detection and segmentation results.

Reduced ResNet-50 backbone

Residual Neural Networks (ResNet) [He+16] achieve state-of-the-art performance in natural image classification and object detection tasks. Due to the limited number of object classes (i.e., cells or nuclei), the reduced background noise of fluorescence microscopy images compared to natural images, and the runtime requirements caused by the high-throughput experiments, a reduced ResNet-50 is used as the backbone architecture for our segmentation network. To speed up training, the number of filters as well as the number of building blocks in the ResNet architecture was halved, which results in about ten times less parameters compared to the original ResNet-50. Table 3.6 shows the reduced ResNet-50 architecture in detail. This

architecture was trained for image classification. A pre-trained network for RGB images is not suitable here, since the inputs are 1-channel images for nuclei and cells, respectively. Thus, we converted the ImageNet [Den+09] dataset to grayscale. Based on the 1-channel ImageNet dataset, the backbone model was trained for 120 epochs using batch normalization [IS15].

layer name	output size	blocks
conv1	112×112	$7 \times 7, 64, \text{stride } 2$
conv2_x	56×56	$3 \times 3 \text{ max pool, stride } 2$
		$\begin{bmatrix} 1 \times 1, 32 \\ 3 \times 3, 32 \\ 1 \times 1, 128 \end{bmatrix} \times 2$
conv3_x	28×28	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 2$
conv4_x	14×14	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 3$
conv5_x	7×7	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 2$
	1×1	avg. pool, fc, softmax

Table 3.6: Reduced ResNet-50 architecture

Feature pyramid fusion

Fig 3.11 shows architectures for instance-based cell segmentation using different ways of integrating nucleus information. The architecture in Fig 3.11a does not include any nucleus information at all. It basically is a Mask R-CNN trained on cell masks for single-class object detection. Similar to an early fusion scheme, the nucleus image is added as an additional channel to the cell image and fed directly into the Mask R-CNN (Fig 3.11bb). However, this architecture cannot utilize available ground truth information for the nucleus channel during the training process.

Nuclei usually have similar shapes and sizes, and in most cases they are clearly separable from each other compared to the appearance of macrophage cells. Therefore, we designed our cell segmentation model in two stages. In the first stage, we train a model for nucleus segmentation to obtain useful nucleus features for the cell segmentation task (Fig 3.10, violet). This Mask R-CNN architecture using feature pyramids based on the reduced ResNet-50 backbone is trained for nucleus segmentation.

In the second stage, the learned FPN features of the nucleus segmentation task are incorporated into the cell segmentation architecture using feature pyramid fusion (FPF). From each feature

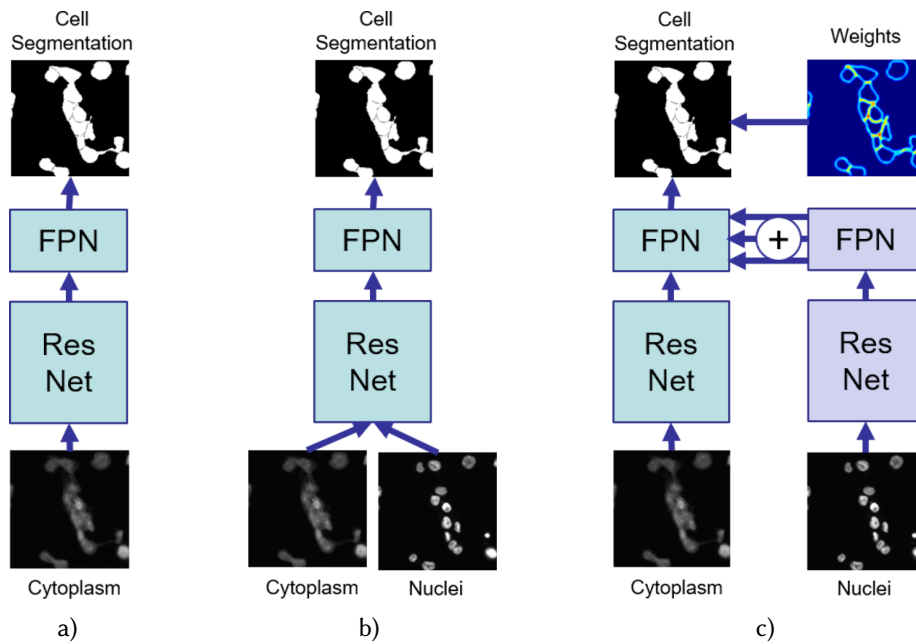


Figure 3.11: Including nucleus information for cell segmentation. (a) without nucleus information, (b) with additional input for the nucleus channel, and (c) with fused nucleus features.

pyramid level, we obtain a stack of activations that is merged into the cell segmentation architecture at the corresponding scale. This means that at each stage of the feature pyramid the pre-trained features of the cell nuclei are available and can be used during training in addition to the feature maps for cell segmentation (Fig 3.10, green). Except for the fused feature pyramids, the backbone architecture for cell segmentation is the same ResNet-50 as for nucleus segmentation.

We also evaluated two merging operations for residual and lateral connections: concatenation and addition. The merged nucleus parameters are fixed during training on cell segmentation, which allows us to combine both models at inference time for concurrently predicting nucleus and cell segmentation masks. The final cell segmentation model has two inputs: the nucleus image and the cytoplasm image. The model architecture was implemented in Tensorflow [Aba+16] and is based on a Mask R-CNN implementation [Abd17].

Weighted segmentation loss

Since it is more difficult to segment pixels at the cell borders especially within cell clusters, we applied a weighting scheme for the mask loss to focus the model on the edges of cells. The weighting is similar to the weighting used by Ronneberger et al. [RFB15]. By putting more weight to the edges, the model is supposed to learn predicting cell contours more accurately, which requires exact and consistent segmentation masks. We computed a Gaussian blurred weight matrix based on the contours of the cells in the full-image segmentation mask. The Gaussian blur function for pixels x, y is defined as $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$ with the horizontal

distance x and vertical distance y from the origin. We used a 25×25 kernel and standard deviation $\sigma = 5$. Using the weight matrix, pixels near border pixels are weighted up to two times higher than regular pixels, i.e., pixels inside cells. Fig 3.12d shows a visualization of the crop from the weight matrix based on contours computed on Fig 3.12b. The crop corresponds to the instance mask in Fig 3.12c. It is used for weighting the mask loss for a box proposal processed in the mask branch.



Figure 3.12: Weights for instances. For each box proposal, crops are resized to 28×28 pixels. Crop from the input image (a), full-image segmentation mask (b), cell mask (c) and weight matrix (d).

Post-processing

To further improve the performance of cell detection and segmentation, the following post-processing steps are performed. First, contour processing methods from the OpenCV library are used to detect nucleus regions that are overlapped by more than one cell. In this case, the corresponding cell segmentation masks are merged. Second, a threshold-based segmentation method is applied to find cell regions not covered by the instance-based segmentation masks due to rarely occurring false positives or misaligned bounding boxes. Therefore, the predicted segmentation masks are subtracted from the threshold-based segmentation, morphological operations and contour processing are applied to detect regions, and regions that are smaller than a predefined threshold are discarded. The remaining regions are handled as follows:

- The region is added to a cell if it can be connected to the corresponding cell instance.
- The region is discarded if it is connected to multiple cell instances.
- Otherwise, a new cell instance is generated.

Third, nucleus regions that are not completely covered by cell regions are used to either enlarge the overlapping cell region or to create a new cell with the same shape as the nucleus.

However, in order to ensure a fair comparison, no post-processing was carried out in the experiments when compared to other methods.

3.2.4 Results

In this section, the performance of the three network architectures shown in Fig 3.11 is investigated. We additionally compare the performance to the performance of U-Net in order to evaluate how well segmentation without detection (U-Net) performs compared to detection

and segmentation with Mask R-CNN architectures. For U-Net, we trained two settings, one with only the nucleus channel and one with both cells and nuclei as input channels. To train the U-Net architecture, we tuned the weighting of the gaps between neighboring cells (and nuclei, respectively) to achieve better results on the corresponding datasets than with the settings used in the original paper [RFB15]. Other training parameters such as learning rate and number of epochs were also optimized to achieve the best results. Since the U-Net model does not return bounding boxes, the contours of cells were used to obtain bounding boxes. In favor of U-Net, contours within contours and very small bounding boxes were ignored in our evaluation. For cell segmentation, the inputs Additionally, we compare the performance to Stardist [Sch+18], which achieves state-of-the-art results for nucleus segmentation.

First, the Mask R-CNN architecture in Fig 3.11a is investigated. Without any information about nuclei, its performance is expected to be lower than for the other architectures.

Second, to verify that incorporating nucleus information is indeed beneficial, Mask R-CNN is extended to accept an additional input channel (Fig 3.11b). This is a straightforward way of incorporating nucleus information. However, it does not utilize available ground truth segmentation masks for nuclei.

Third, the proposed feature pyramid fusion architecture shown in Fig 3.11c is evaluated, where pyramid features of a pre-trained nucleus model are combined with cell features using the pyramid fusion scheme. Within this fusion scheme, two merging operations are evaluated: concatenation and addition. Recent work on learning CNN architectures suggests that addition operations are more beneficial in many cases than concatenation operations [Liu+18]. Furthermore, the feature pyramid fusion architectures are evaluated in combination with the weighted segmentation loss described above.

The layers of the backbone architecture were initialized with pre-trained weights from an image classification task using the ImageNet dataset. In all experiments, the same pre-trained weights were used for FPF. For the architecture in Fig 3.11b with two input channels, the pre-trained weights of the first convolutional layer were duplicated and halved to not disturb dependencies to weights of subsequent layers.

In all experiments with FPF, the same training schedule was applied, with a learning rate of 0.001, a momentum of 0.9, and a weight decay factor of 0.0001. The training schedule is as follows: on top of the pre-trained backbone segmentation, regression and classification heads are trained for 100,000 iterations. Next, all layers deeper than conv4 in the backbone are trained for another 250,000 iterations. Finally, the whole network is trained for 500,000 iterations with a reduced learning rate of 10^{-4} . Both regression losses and the mask loss are weighted by a factor of 2.

All segmentation models, except the configuration with two input channels, were trained on 512×512 1-channel input images. Within the training process, data augmentation was applied to the training images, which increases the number of training images to 497,355 of size 512×512 for each channel, containing 3,557,425 nuclei in 4,288,104 cells. The variants of the FPF architecture use the same model for nucleus features. It was trained with the same previously described training schedule for cell segmentation models.

All models were evaluated on the test dataset described above. Additionally, to verify that FPF performs better particularly for clustered cells, we created a subset of the test dataset. This subset contains all cell clusters occurring in the test images, but no isolated cells. The resulting 78 images have 256×256 pixels. Each image contains at least one cluster of two or more cells. In total, the dataset contains 255 cells with 258 nuclei.

Evaluation metrics

All experiments were evaluated in terms of average precision (AP) for detection and segmentation. AP summarizes the shape of the precision/recall curve. It is defined as the mean precision at a set of equally spaced recall levels. AP is computed for different Intersection over Union (IoU) thresholds. IoU between sets of pixels A and B is computed as

$$IoU(A, B) = \frac{A \cap B}{A \cup B} \quad (3.4)$$

on the 100 top-scoring detections per image for bounding boxes and segmentation masks, respectively. For example, for threshold $t = 0.5$, detections with an IoU overlap of at least 50% with a ground truth object are counted as true positives (TP). Unmatched predicted objects are false positives (FP) and unmatched ground truth objects are false negatives (FN). Precision is computed as $p = \frac{TP}{TP+FP}$ and recall as $r = \frac{TP}{TP+FN}$. We use the 101-point interpolated AP as Lin et al. [Lin+14]. AP for a given IoU threshold t is computed over all images in the test set as

$$AP(t) = \frac{1}{101} \sum_{r \in \{0, 0.01, \dots, 1\}} p_{interp}(r), \quad (3.5)$$

where $p_{interp}(r) = \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r})$ and $p(\tilde{r})$ is the measured precision at recall \tilde{r} . For a detailed description of interpolated AP, we refer to Everingham et al. [Eve+10]. The final mean average precision score (mean AP) is the mean over all threshold values between 0.5 and 0.90 with steps of 0.05. However, it should be noted that higher IoU thresholds are less meaningful in our dataset, since in many cases there are no sharp cell borders and thus the corresponding ground truth masks may be somewhat uncertain (see Fig 3.13).

Detection and segmentation results

First, we evaluated the performance of the nucleus model used in the FPF architectures (as described in Table 3.6), Stardist, and U-Net. Table 3.7 shows results for both detection and segmentation performance in terms of AP on the nuclei test images. Detection and segmentation results are similar, since the majority of the nuclei are round and isolated. However, in some cases nuclei appear adjacent. These instances are hard to separate for U-Net that cannot be trained to detect instances before segmentation. Stardist performs slightly better for segmentation, while Mask-RCNN performs slightly better for detection.

Next, the performance of the architecture without any nucleus information, the architecture with two input channels, and the FPF architectures are evaluated for cell detection and segmentation. In the following, \odot is used for concatenation and \oplus for addition. The experiments

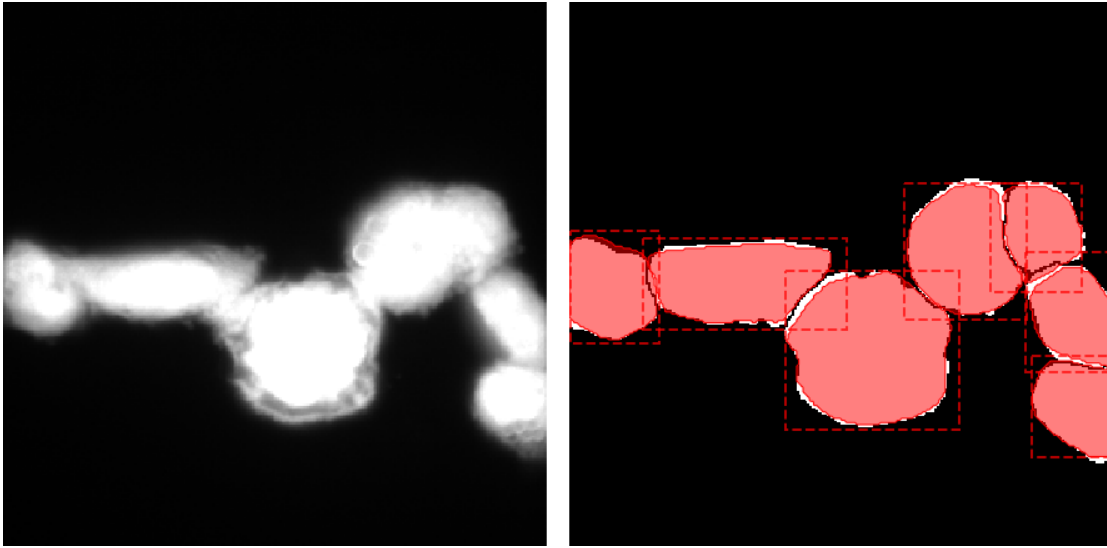


Figure 3.13: Visualization of cell segmentation errors on a 256×256 patch of clustered cells in test data: predicted masks (red) differ only slightly from ground truth masks (white).

	IoU threshold									mAP
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	
<i>Detection</i>										
U-Net	67.4	65.6	61.0	60.4	57.8	52.9	41.8	24.1	-	46.7
Stardist	93.8	93.7	92.5	90.9	89.5	87.4	85.2	76.8	52.4	84.7
Mask R-CNN	95.3	94.1	92.5	91.5	91.5	89.6	86.7	82.2	56.9	86.7
<i>Segmentation</i>										
U-Net	69.4	67.1	63.4	61.0	59.1	52.8	40.8	18.7	-	47.9
Stardist	94.2	92.4	92.2	90.5	89.4	89.2	88.0	81.2	60.0	86.3
Mask R-CNN	95.2	93.6	92.5	91.2	91.2	89.0	86.2	81.0	53.6	85.9

Table 3.7: Detection and segmentation results for the nuclei test dataset in terms of AP.

are conducted on the whole cell test dataset as well as on a subset of the cell test dataset that contains exclusively clustered cells, to have a closer look at those instances that are difficult to detect and segment.

AP scores for detection and segmentation on the cell test dataset are shown in Table 3.8 and Table 3.9, respectively. As expected, the performance of cell segmentation is much worse when knowledge about nuclei is not incorporated at all. By simply adding one more input channel for the nucleus signal to Mask R-CNN (see Fig 3.11b), significantly better results are achieved. For the U-Net model, the same method is used to integrate the nucleus channel, i.e.,

the model has two instead of one input channel. The drawbacks of the U-Net architecture on the cell test dataset are two-fold. First, it cannot use available ground truth of nuclei directly and second, there are even more hard-to-separate signals in this dataset than in the nucleus dataset. Since Stardist is trained for cell segmentation only, it cannot use nucleus information. However, Stardist still performs worse than the Mask R-CNN model without nucleus information. Compared to the performance of Stardist on the nuclei test set, this is most probably caused by the irregular shapes of cells and clustered cells.

FPF performs better than merely using an additional input for Mask R-CNN. However, it does not depend on the merge operation: merging features by concatenation or addition results in similar performance. AP scores for detection (Table 3.8) and segmentation (Table 3.9) are similar for all pyramid fusion configurations. Although mean AP performance is slightly better for both architectures trained under a weighted loss, a higher weighting of edges at gaps between cells and near-border pixels in the loss function does not result in a significant performance gain, neither in detection, nor in segmentation. Relative to the mean AP of the model without any nucleus information, the best performing FPF architecture (FPF \oplus with weighted loss) achieves a performance gain of 10.25%. Compared to the model with the nucleus input channel, the relative performance gain is 3.02%. Similarly, the performance gain for detection is 10.48% (no nucleus) and 3.5% (nucleus input channel). A detailed evaluation including the number of true positives (TP), false positives (FP), and true negatives (FN) of the segmentation results for an IoU threshold of 0.75 is shown in Table 3.10.

	IoU threshold									mAP
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	
U-Net (cell + nucleus input)	51.5	47.3	42.1	36.0	29.1	25.0	18.1	9.9	-	48.6
Stardist	84.7	81.4	73.9	65.5	56.6	48.0	36.1	24.7	10.5	53.5
no nucleus	88.6	87.0	84.5	79.0	74.3	67.5	54.8	42.6	23.0	66.8
nucleus input	92.1	91.0	88.6	86.5	82.1	72.7	61.9	44.8	22.1	71.3
FPF \odot	94.4	93.2	91.2	88.4	84.8	77.1	61.3	48.0	23.7	73.6
FPF \oplus	93.5	92.5	90.1	88.6	84.2	75.8	63.3	47.9	22.0	73.1
FPF \odot weighted loss	94.4	93.2	89.7	88.3	84.2	76.7	63.3	45.3	22.8	73.1
FPF \oplus weighted loss	94.2	93.2	91.6	89.1	84.1	76.6	64.8	48.4	22.4	73.8

Table 3.8: Detection results for the cells test dataset in terms of AP.

The superior performance of the FPF architectures becomes evident when they are evaluated on the clustered cells subset (Table 3.11 and 3.12): a relative performance improvement of 23.88% (no nucleus) and 4.43% (nucleus input channel) for detection in terms of mean AP. Fig 3.15 visualizes the segmentation results for clustered cells. Fig 3.16 shows a visualization of the segmentation masks predicted by the model without nucleus information, with nucleus channel, and the best performing FPF architecture.

Similarly for segmentation, FPF \oplus with weighted loss performs better. Its relative performance improvement is 23.17% compared to the model without nucleus information, and 4.16% compared to the model with the nucleus input channel, both in terms of mean AP. Fig 3.14

3.2 Detection and Segmentation of Eukaryotic Cells via Feature Pyramid Fusion

	IoU threshold									mAP
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	
U-Net	53.8	47.1	41.4	35.4	29.5	22.4	15.5	7.7	-	47.0
Stardist	86.9	84.3	81.3	77.3	69.8	63.2	50.4	32.2	9.4	61.6
no nucleus	89.8	87.0	85.8	83.8	81.1	75.1	65.7	49.9	22.6	71.2
nucleus input	93.1	90.9	90.8	88.1	86.6	80.7	73.3	58.5	23.6	76.2
FPF \odot	94.3	94.3	92.0	90.5	88.5	83.0	74.6	60.0	28.4	78.4
FPF \oplus	93.5	93.5	90.9	90.8	89.6	84.0	76.2	59.2	24.6	78.0
FPF \odot weighted loss	94.6	93.4	92.1	89.0	87.6	83.5	75.3	60.5	25.3	77.9
FPF \oplus weighted loss	94.1	93.0	93.0	91.1	89.0	84.9	76.4	60.3	24.7	78.5

Table 3.9: Segmentation results for the cells test dataset in terms of AP.

	TP	FP	FN	mAP
Stardist	379	91	129	63.2
no nucleus	409	93	99	75.1
nucleus input	427	79	81	80.7
FPF \odot	441	82	67	83.0
FPF \oplus	443	65	65	84.0
FPF \odot weighted loss	445	69	63	83.5
FPF \oplus weighted loss	450	66	58	84.9

Table 3.10: Detailed cell segmentation results for an IoU threshold of 0.75 on the cells test dataset.

shows an example of a segmentation performed by FPF \oplus with weighted loss on a cluster of macrophages. The experiments show that U-Net is not suitable here and the instance-based FPF architectures perform much better. Although Stardist performs well for nucleus segmentation, the performance is considerably lower for cell segmentation.

Using additional post-processing, the detection mean AP on the dataset of clustered cells is improved to 0.65, while the segmentation mean AP increases to 0.682. For the detection mean AP, this is a relative improvement of 31.58% (no nucleus) and 10.92% (nucleus input channel). For the segmentation mean AP, this is a relative improvement of 24.45% (no nucleus) and 5.25% (nucleus input channel).

The runtime of the FPF model is about 222 ms per image using Tensorflow Serving on a server with an Nvidia Geforce GTX 1080 Ti graphics card.

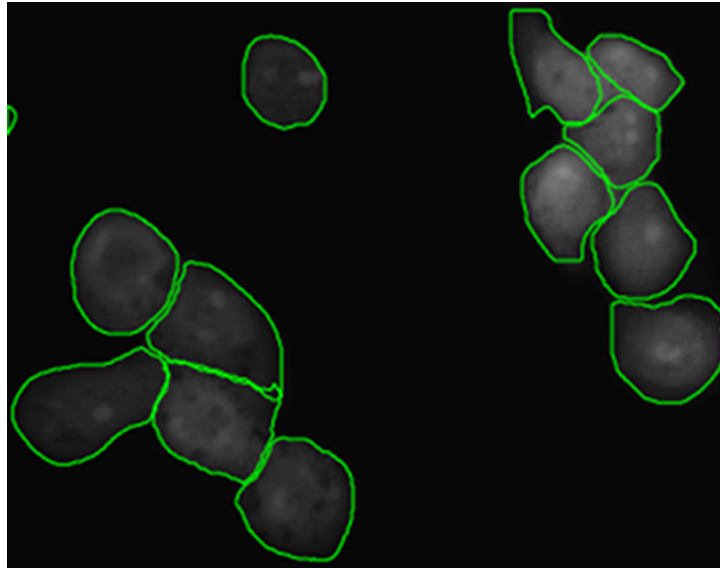


Figure 3.14: Cell segmentation of clustered cells by Feature Pyramid Fusion (FPF) on a 512×512 patch.

	IoU threshold									mAP
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	
no nucleus	77.2	72.6	66.7	62.3	56.6	48.2	33.4	21.8	5.7	49.4
nucleus input	85.2	83.9	79.2	74.7	68.8	58.1	44.7	26.1	6.6	58.6
FPF \ominus	85.6	82.5	82.5	75.6	69.9	64.0	43.8	25.6	7.3	59.6
FPF \oplus	85.1	83.7	81.5	76.3	70.6	59.1	47.3	28.0	6.3	59.8
FPF \ominus weighted loss	85.6	84.7	82.7	78.5	74.7	61.8	47.5	27.1	7.8	61.2
FPF \oplus weighted loss	86.1	84.9	82.5	77.5	69.4	62.5	48.1	26.7	5.9	60.4

Table 3.11: Detection results for clustered cells test dataset in terms of AP.

	IoU threshold									mAP
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	
no nucleus	77.9	72.0	68.4	67.3	63.1	55.8	45.9	31.9	10.4	54.8
nucleus input	83.9	82.6	81.5	77.7	75.4	67.6	58.9	41.6	14.2	64.8
FPF \ominus	85.6	84.6	82.4	80.5	77.8	70.3	61.5	43.4	15.0	66.8
FPF \oplus	86.0	83.9	81.4	79.6	78.5	71.2	63.6	41.9	13.6	66.6
FPF \ominus weighted loss	85.9	84.9	82.9	81.9	78.8	71.8	62.5	42.0	16.0	67.4
FPF \oplus weighted loss	86.0	84.8	83.5	80.7	78.5	72.6	61.9	42.7	16.1	67.5

Table 3.12: Segmentation results for clustered cells test dataset in terms of AP.

3.2 Detection and Segmentation of Eukaryotic Cells via Feature Pyramid Fusion

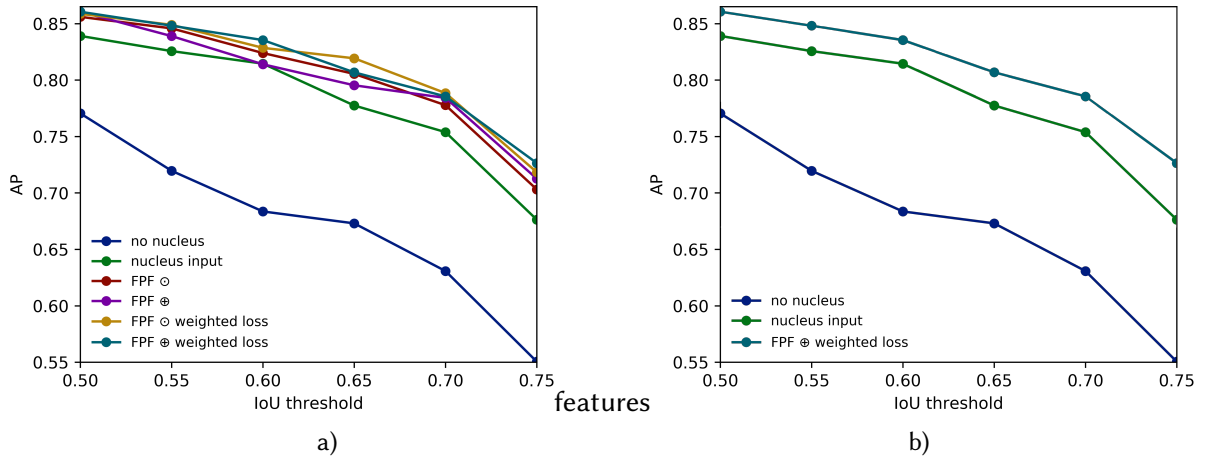


Figure 3.15: APs for cell segmentation on clustered cells. All FPF settings (a) and the best performing setting FPF \oplus with weighted loss (b).

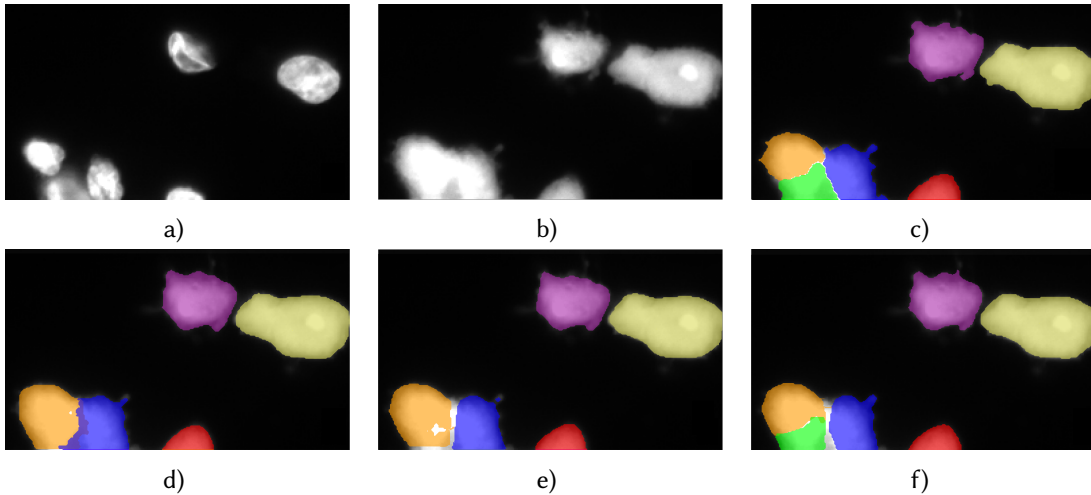


Figure 3.16: Visualization of segmentation of clustered cells. Top: nucleus signal (a), cytoplasm signal (b), and ground truth segmentation (c). Bottom: Instance segmentation predicted by for model without nucleus information (d), with nucleus channel (e), and FPF \oplus with weighted loss (f).

3.2.5 Summary

In this section, we presented a novel deep learning approach for cell detection and segmentation based on fusing previously trained nucleus features on different feature pyramid levels. The proposed feature pyramid fusion architecture clearly outperforms a state-of-the-art Mask R-CNN approach for cell detection and segmentation on our challenging clustered cells dataset with relative mean average precision improvements of up to 23.88% and 23.17%, respectively. Combined with a post-processing step, the results could be further improved to 31.58% for detection and 24.45% for segmentation, respectively.

4

Image Similarity Search

This chapter presents two works in the broad field of image similarity search. A low query time on the one hand and a high retrieval quality on the other hand are requirements that are placed on a good image similarity search system. The works presented in this chapter each deal with one of these aspects. The first work gives answers to the question of how to improve the retrieval quality of similarity search results. It presents methods to better capture a user's search intention. The second work presents a way to efficiently search in large-scale image databases. It introduces a novel two-stage approach that is integrated into Elasticsearch.

4.1 Intentional Image Similarity Search

4.1.1 Introduction

A fundamental problem of content-based image retrieval is to overcome the discrepancy between the information that can be extracted from visual data and the human interpretation of the same data. In the literature, this discrepancy is also known as the semantic gap [Sme+00b]. Using state-of-the-art convolutional neural network (CNN) features has brought us close to the goal of bridging this gap. Image representations learned by deep neural networks can greatly improve the performance of content-based image retrieval systems. They are less dependent on pixel intensities and are better suited for searching semantic content.

In addition to the semantic gap, there is an intentional gap that describes the coincidence between a query and a user's intention. Query-by-example is the most popular, most intuitive, and most expressive strategy to describe a user's search intention in content-based image and video retrieval scenarios. Nevertheless, simply presenting an image as a query is often insufficient to express a user's intention. For example, the pictures in Figure 4.1 show query images presented to the database of the German Broadcasting Archive, i.e., an institution that maintains the cultural heritage of the television broadcasts of the former German Democratic Republic (GDR). In all presented query images, the user's intention is not clear:

In Figure 4.1a, possible search intentions are:

- Crowd
- Person
- Katarina Witt
- Autograph signing session
- Katarina Witt signing autographs
- Katarina Witt in figure skating dress

In Figure 4.1b, possible search intentions are:

- Simson scooter
- Is the woman important?
- Is the layout important (woman sitting on a scooter at a parking area with Trabant cars in front of a house)?

In Figure 4.1c, possible search intentions are:

- Motorbike
- Vintage Motorbike

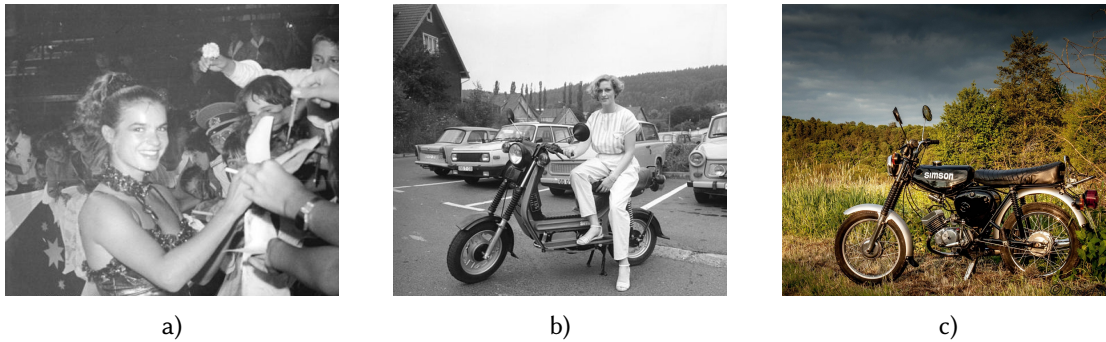


Figure 4.1: Query images.

- Moped
- Simson moped
- Simson S51 moped
- Black Simson S51 moped
- Motorbike on meadow with trees in the background

The mismatch between a user's intention and a query image can be attributed to the following factors: region of interest, layout, and specificity. It is often unclear whether the user is interested in only part of the image or in the whole scene. Furthermore, there is the question of whether the layout is important. For example, is it important in query image 4.1b that the woman is sitting on a scooter in front of parking cars with a house in the background? Finally, the specificity of the query is an important factor to capture the user's search intention. It ranges from the general concept of the query image (e.g., motorbike), to the specific object, person, or scene (e.g., Simson S51), to a duplicate of the image. In this context, color, texture, and shape are further attributes to specify a user's intention. In all these cases, more user interactions are necessary to specify a user's intention.

In this section, a novel similarity search approach is presented that uses intentional constraints to capture regions of interest, layout, and specificity. These constraints are defined by user interactions using region labeling and checkboxes for layout and specificity. Regarding specificity, we have to distinguish between the hierarchy of classes (e.g., car \rightarrow Volkswagen \rightarrow Golf) and the attributes like color, texture, and shape. The best solution for specific classes are fine-grained models trained for subcategories of e.g., persons, cars, dogs, or flowers. However, this solution is not really scalable, since there are thousands of classes that would need to be mapped with fine granularity. Therefore, we realized a hybrid approach where CNN features are combined with handcrafted features (e.g., color moments and SIFT descriptors) to handle specificity constraints, while fine-grained similarity modules can be plugged in for the most important classes such as, e.g., persons.

The hybrid approach operates in two stages. In the first stage, CNN features are used to find semantically similar content. These images are re-ranked in the second stage using handcrafted features. These features are extracted from image regions that are responsible for the high semantic similarity score between the query and the result image. The responsible image

regions (called heat maps) are calculated using a novel technique for visualizing deep similarity networks. These regions are also used to realize the layout constraint by comparing the heat maps.

The contributions are as follows:

- A new query specification scheme is presented that allows to clarify the search intention of the user presenting the query image.
- A hybrid method using CNN and handcrafted features is presented to realize intentional image similarity search.
- A novel analysis technique for deep similarity networks is introduced to find the relevant image regions.

Parts of this section have been published in: Nikolaus Korfhage, Markus Mühling, and Bernd Freisleben. “Intentional Image Similarity Search.” in: *Artificial Neural Networks in Pattern Recognition: 9th IAPR TC3 Workshop, ANNPR 2020, Winterthur, Switzerland, September 2–4, 2020, Proceedings* 9. vol. 12294 LNAI. Springer. Springer Science and Business Media Deutschland GmbH, 2020, pp. 23–35. doi: 10.1007/978-3-030-58309-5_2.

4.1.2 Related Work

In multimedia search, Kofler et al. [KLH16] distinguish between the topical dimension (“what” is the user searching for) and the intent dimension (“why” is the user searching). While the terms “intent” and “intention” are used synonymously in the literature, Kofler et al. [KLH16] distinguish between the “intent” as the “immediate reason, purpose, or goal behind a user’s information need” [HKL12] and the “intention” which describes the information need as a whole. In the case of content-based image similarity search, we consider the intention gap as the coincidence between the query image and the user’s intention. This is often reflected in the ambiguity of the query image. The term “intent” goes deeper and considers, for example, conceptual models of user intent which are built based on click-through data of user sessions, query log analysis, or user profiles exploiting long-term search behaviors. While there is a wide range of intent-aware approaches in the field of text and multimedia information retrieval [KLH16; Cai+15; Den+14] that are mainly based on keyword or text queries, less research effort has been devoted to intentional image similarity search.

In this section, we focus on content-based image retrieval with query-by-example. Query-by-content based on feature representations learned by deep CNNs have greatly increased the performance of content-based image retrieval systems [Wan+14], since they are less dependent on pixel intensities and better represent the semantic content of the images. Thus, they try to bridge the semantic gap between the data representation and the human interpretation.

In addition to the semantic gap, there is an intentional gap that describes the coincidence between a query and a user’s intention. In general, the intentional gap is due to ambiguities in the query image. As already described in Section 4.1.1, a search image is often insufficient to express a user’s intention concerning region of interest, layout, and specificity. The query image, for example, could contain image regions that are not part of a user’s search intention. Furthermore, the specificity of the query image has to be clarified.

Relevance feedback is a commonly used technique to narrow down a user's search intention. In this scenario, a user interacts with the search engine to evaluate an initial retrieval result. The additional relevant and non-relevant labeled images are used in an iterative process to refine the retrieval results. An overview of relevance feedback in image retrieval is given by Zhou and Huang [ZH03].

Bian et al. [Bia+12] use a query suggestion approach for query-by-example image search to specify a user's intention. Given a query image, informative attributes reflecting visual properties of the query image are suggested to the user as complements to the query. By selecting some suggested attributes in a feedback session, a user can clarify his or her search intention.

The approach of Guan and Qui [GQ07] learns a user's intention in an interactive image retrieval process. Given a query image, the relevant image regions are inferred both from the query and from multiple relevance feedback images using local image patch appearance prototypes. These relevant regions are then used to refine the ranking result.

Zhang et al. [Zha+13] present a semantic concept approach. The authors organize the semantic concepts into a hierarchy and augment each concept with a set of related attributes (e.g., round, red, shiny). The queries are mapped onto the concept hierarchy with attributes, and user feedback is collected to refine the ranking results.

Other possibilities of specifying the search intention are query expansions using, for example, multiple query images [AZ12; ABB20], additional keywords, or text descriptions [JB16; PG17].

To the best of our knowledge, there are no deep similarity search approaches dealing with query image ambiguities.

4.1.3 German Broadcasting Archive

Our hybrid feature approach for intentional image similarity search has been applied to historical video recordings of the German Broadcasting Archive (DRA). The DRA maintains the cultural heritage of television broadcasts of the former German Democratic Republic (GDR). It was founded in 1952 as a charitable foundation and joint institution of the Association of Public Broadcasting Corporations in the Federal Republic of Germany (ARD).

The archive contains film documents of former GDR television productions from the first broadcast in 1952 until its cessation in 1991. It includes a total of around 100,000 broadcasts, such as: contributions and recordings of the daily news program *Aktuelle Kamera*; political magazines such as *Prisma* or *Der schwarze Kanal*; broadcaster's own TV productions including numerous films, film adaptations and TV series productions such as *Polizeiruf 110*; entertainment programs (e.g., *Ein Kessel Buntes*); children's and youth programs (fairy tales, *Elf 99*); as well as advice and sports programs.

The DRA provides access to this valuable collection of scientifically relevant videos. The uniqueness and importance of the material fosters a large scientific interest in the video content. Access to the archive is granted to scientific, educational and cultural institutions, to public service broadcasting companies and, to a limited extent, to commercial organizations and private persons. The video footage is often used in film and multimedia productions. Furthermore,

there is a considerable international research interest in GDR and German-German history. International scientists use the DRA for their research in the fields of psychology, media, social, political or cultural science.

The DRA is answering a wide range of time-consuming research requests. However, finding similar images in large multimedia archives is manually infeasible. Therefore, the DRA aims to digitize and index the entire video collection to facilitate search in images and videos. In this context, content-based image retrieval using query-by-example is a powerful tool to make the valuable information in the archive findable.

4.1.4 A Novel Approach to Intentional Image Similarity Search

In this section, the proposed intentional similarity search approach is presented. To better meet a user's search intentions, the approach is based on a plugin mechanism for fine-grained similarity search modules and a hybrid approach based on CNN and handcrafted features. Further query specifications are required to capture the search intention of a user presenting a query image. The query specification scheme and the mapping of queries to similarity search modules is presented below. Then, we present the introduced hybrid approach using deep CNN and handcrafted features. Finally, the plugin mechanism using the example of similarity search for faces is described.

Query Specification

To disambiguate a query, a user is offered several options for further specifying his or her search intention associated with the presented query image. First, the user is allowed to specify the region of interest to exclude irrelevant parts of the query image. Second, the user can choose whether (s)he is looking for a specific concept (e.g., a VW Golf). This option is available if one of the plugins for fine-grained search detects the corresponding general concept (e.g., a car). Third, the user has the possibility to select color, texture, and shape as additional query conditions to clarify the search intention. The selected features are automatically extracted from the region of interest and used to refine the initial ranking results of either the general or the specific deep similarity search model. The hybrid feature approach is described in the next section. Fourth, the user can tell the similarity search system that the layout is important. The layout is considered by comparing the relevant image regions between the query and the retrieved images. The relevant region extraction approach is presented below. Finally, the user can perform a duplicate search. For this purpose, all constraints must be satisfied. Altogether, multiple selected conditions are weighted according to the user's settings in the distance function at the re-ranking stage.

Hybrid Feature Method

The proposed hybrid method operates in two stages. In the first stage, CNN features are used to find semantically similar content. To be scalable to millions of images, this stage relies on compact representations of the images for fast computation of image similarity by the

Hamming distance. The deep similarity search approach used in the first stage is described in the next paragraph. If color, texture or shape are selected as additional query conditions, the results of the first stage are re-ranked in the second stage, using handcrafted features.

Deep Similarity Search

In our work, we rely on the visual modality and focus on large-scale semantic similarity search. Since high-dimensional CNN features are not suitable to efficiently search in very large databases, large-scale similarity search systems focus on binary image codes for compact representations and fast comparisons rather than full CNN features. Binary codes enable fast distance computation in the Hamming space. Furthermore, the distance computation complexity is reduced by Multi-Index Hashing [NPF12]. We use a model that is trained to generate 256 bit binary codes for fast image retrieval. First, a NASNet [ZL16], pretrained on ImageNet [Den+09], is trained on ImageNet and the Places 205 dataset [Zho+17]. Before the final classification layer a *tanh* activation layer is integrated which produces the 256-dimensional codes. Next, the model is trained for a few epochs with a smaller learning rate.

Handcrafted Features

The definition of similarity ranges from pixel-based similarity to semantic similarity. The latter corresponds to human understanding. The definition and optimization of similarity functions is subject to current research [Bla+16; Lia+16]. In the following, color histograms are proposed to compute the similarity of color distributions, GIST features [OT01] to measure texture similarity, and SIFT features [Low99] for shape. To do this, we detect key points and extract SIFT descriptors within the region. Shape similarity is determined by the number of matching key points, their spatial distribution, and their similarity in descriptor space. The handcrafted features are extracted from image regions that are responsible for the high semantic similarity score at the first stage. For this purpose, a new method is introduced to find these regions, as described in the following paragraph. To compute the similarity between the relevant regions of two images, the Euclidean distance of the normalized feature vectors is calculated.

Relevant Region Extraction

To re-rank the retrieval list, we extract relevant regions by a method similar to *Class Activation Maps* (CAM) [Zho+16]. The idea is to use the output activations of the query image to detect relevant regions in images of the retrieval list. The method requires that the last convolutional layer is followed by a global average pooling layer. Global average pooling outputs the spatial average of the feature map of the last convolutional layer. Thus, each output in the final layer is a weighted sum of the pooled vector. CAM can then be generated by applying the weights corresponding to a specific output class to weight each feature map of the last convolutional layer. In contrast to CAM, we use the averaged weights of the 256-dimensional coding layer output of the query image to weight the final feature maps of the retrieval image.

For an image r in the retrieval list, $f_k^r(x, y)$ represents the activation of unit k in the last convolutional layer at spatial location (x, y) . For this unit, global average pooling of the

convolutional layer with depth C results in $F_k^r = \frac{1}{C} \sum_{i=1}^C f_k^r(x, y)_i$. For each output unit in the subsequent deep hashing layer, the score is computed as $\sum_k w_{k,s}^r F_k^r$, where w_k^r is the weight between the output s and unit k . Likewise, w_k^q is obtained a for query image q . As in CAM, the bias term is ignored. Finally, the activation map used for extracting regions is obtained by

$$M(x, y) = \sum_k \sum_s w_{k,s}^q f_k^r(x, y). \quad (4.1)$$



Figure 4.2: Heatmaps visualized in retrieval results.

Figure 4.2 shows a heatmap visualization of the upscaled feature maps for several retrieval images weighted by the query image.

Plugin Mechanism

The best solution for specific classes are fine-grained models trained for subcategories, such as faces, car models, bird species or dog breeds. Since this approach is not scalable to thousands of classes, fine-grained models are only integrated for the most important and most frequently used query contents. The user-defined query image regions are analyzed by the search engine to detect classes for which more fine-grained modules exist. Each module has to provide two components: a detection component and a component that generates the binary codes from the image region. In an interactive user session, as described above, the user specifies whether (s)he is searching for the general class or for the automatically detected specific content.

Each module has its own index. If a module is activated, the general search index is replaced by the specific index of the fine-grained model from the corresponding submodule.

In the following, the fine-grained module for faces/persons is presented in more detail. Similar to the general model, the same deep hashing approach, as described above, is used to generate binary codes for large-scale similarity search. In contrast to the general model, the face module follows a two-stage approach. In the first stage, the faces are detected using a joint face detection and alignment approach based on multitask cascaded convolutional networks [Zha+16a]. For this purpose, a publicly available implementation is used¹. After aligning the face regions, a deep hashing model is used to generate the binary codes. We use the pretrained weights of the publicly available FaceNet model [SKP15a] trained on the CASIA-WebFace dataset. We extended the architecture of this model by a coding layer and fine-tuned it on the same dataset extended by training samples for 100 persons from the DRA dataset to adjust the model to the keyframes of the historical video recordings.

¹<https://github.com/davidsandberg/facenet/tree/master/src/align>

While in the indexing phase the binary codes for all detected faces of an image are calculated and fed into the corresponding index, in the search phase only the largest face within the query image region is used to generate the binary query code.

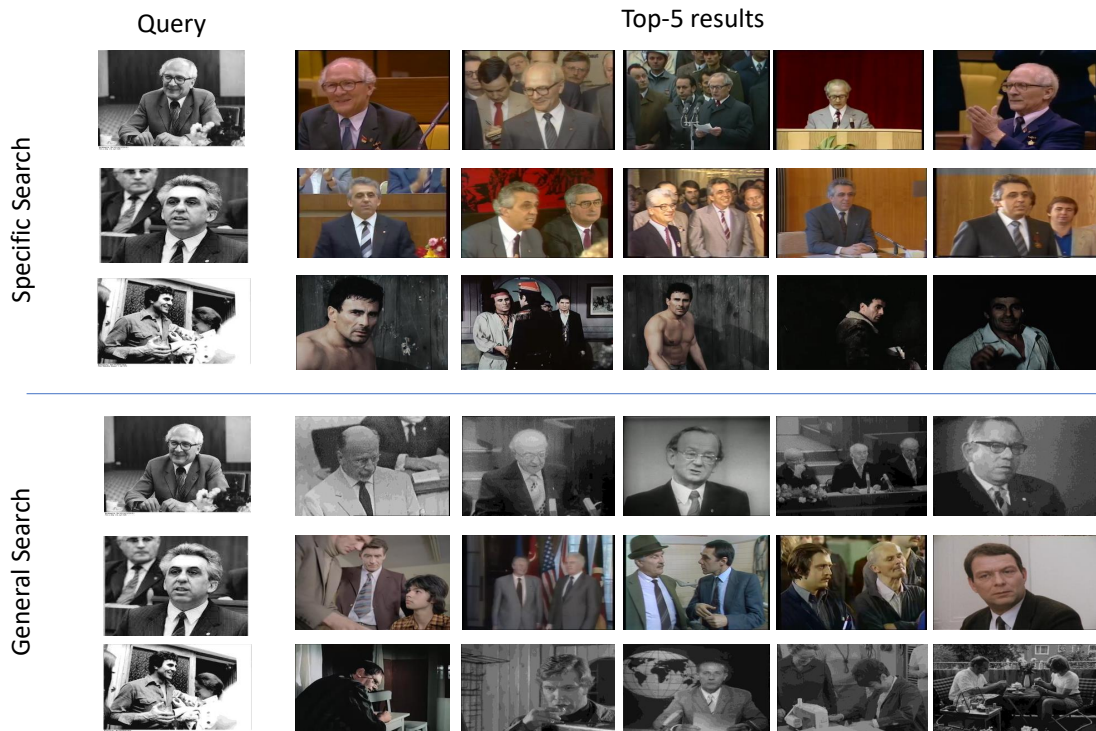


Figure 4.3: Specific retrieval results using the fine-grained face module in comparison to the general search results.

4.1.5 Experimental Results

The proposed intentional image search approach was evaluated experimentally on the keyframes of the video recordings of the German Broadcasting Archive. While the currently digitized index contains more than 10 million keyframes, the retrieval results had to be restricted to 400,000 keyframes due to associated rights of use of the keyframes.

We evaluate our approach qualitatively for two use cases below.

In the first use case, we consider the plugin mechanism and investigate the impact of the face/person similarity module. For this purpose, a face detection and recognition module was integrated into the similarity search system. This allows the user to specify whether general concepts are important, or if (s)he is interested in a specific concept - a person's face in this case.

The experiments were performed on two indices, one for general semantic similarity, another one for face similarity. The general index contains hash codes for all 400,000 keyframes, while the person recognition based index contains about 300,000 image codes, each one representing a detected face.

4 Image Similarity Search

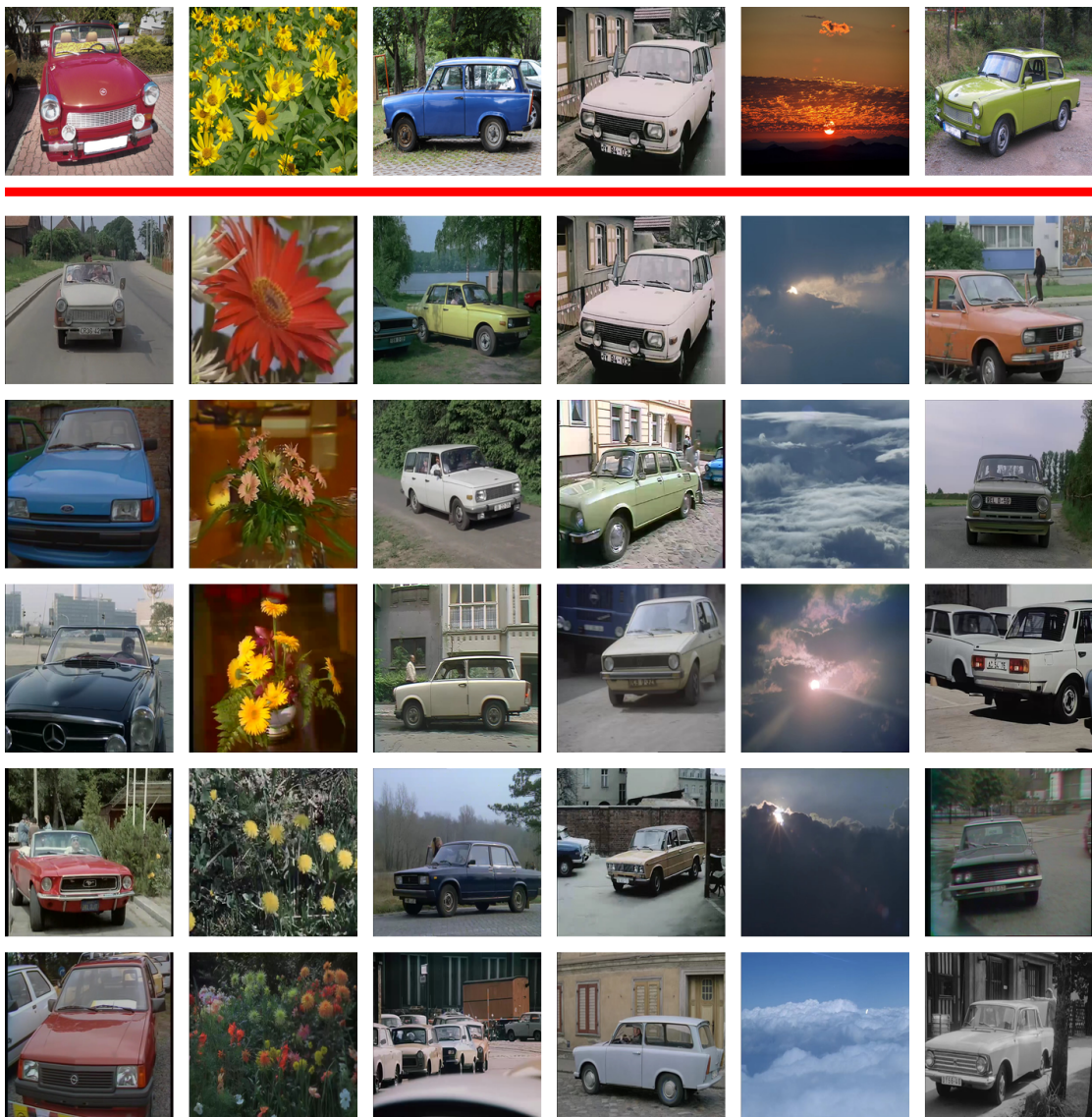


Figure 4.4: Retrieval of deep similarity search (query images in top row).

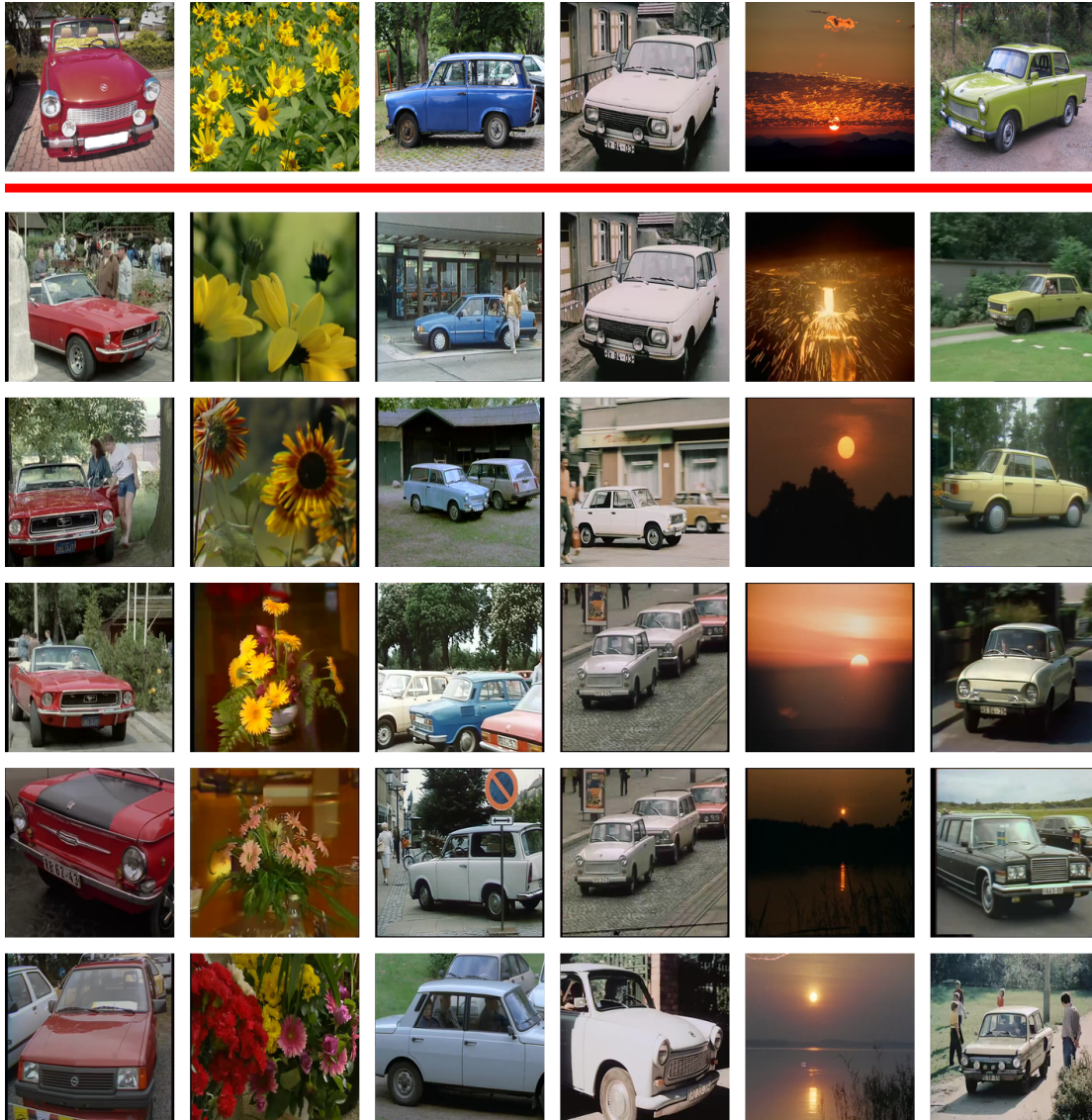


Figure 4.5: Retrieval of deep similarity search, re-ranked by local color histograms (query images in top row).

Figure 4.3 shows the top retrieval results for query images for both the general search and the specific search, which involves a subsequent face detection and recognition step. The detection component of the face plugin reliably detected the faces both in the query and database images. While the general search results contains similar images that contain persons with the same shot size, the specific search delivers the same person in different scenarios with high accuracy.

In the second use case, we evaluated the hybrid feature approach by the example of color histograms. If a user decides that color is an important property in the query image, the retrieval list is re-ranked, as described in Section 4.1.4. Figure 4.4 shows the retrieval for some query images. Figure 4.5 shows the same retrieval list when color is of interest to the user, re-ranked according to the similarity of color histograms extracted from the relevant regions. This example shows how retrieval results of CNNs trained on (high level) semantic concepts can be refined by re-ranking them according to selected low level features.

In both cases, the additional specification of the query leads to results that reflect a user's intention.

4.1.6 Summary

In this section, a novel intentional image similarity search approach was proposed to better meet a user's search intention. For this purpose, a new scheme for specifying a user's intention with respect to a query image was introduced. The best solution for specific classes are fine-grained models trained for subcategories, such as faces. Since this approach is not scalable to thousands of classes, a plugin mechanism was integrated to support specific search for the most important concepts. Furthermore, a hybrid feature method based on CNN and handcrafted features was used to consider color, texture, and shape. In this context, a novel analysis method for deep similarity networks was presented for the purpose of finding relevant image regions. Finally, the proposed system was evaluated qualitatively on video recordings of the German Broadcasting Archive, showing promising results.

4.2 ElasticHash: Semantic Image Similarity Search in Elasticsearch

4.2.1 Introduction

Query-by-content approaches based on feature representations that are learned by deep convolutional neural networks (CNNs) have greatly increased the performance of content-based image retrieval systems. However, state-of-the-art methods in the field of semantic image similarity search suffer from shallow network architectures and small data sets with few image classes in the training as well as in the evaluation phases. Few image classes in the training phase lead to poor generalizability to query images with unknown content in the evaluation phase, i.e., a more fine-grained modeling of the image content is required. Thus, high accuracy for arbitrary search queries, fast response times, and scalability to millions of images are necessary to meet many users' needs both in scientific and commercial applications.

In this section, we present *ElasticHash*, a high-quality, efficient, and scalable approach for semantic image similarity search based on the most popular enterprise full-text search and analytics engine Elasticsearch² (ES). ES processes queries very fast due to inverted indices based on Lucene³, scales to hundreds of servers, provides load balancing, and supports availability and reliability. Apparently, the properties of ES are not only desirable for full-text search, but also for semantic image similarity search. Furthermore, integrating image similarity search into ES allows multi-modal queries, e.g., combining text and images in a single query. Our contributions are as follows:

- We present *ElasticHash*, a novel two-stage approach for semantic image similarity search based on multi-index hashing and integrate it via terms lookup queries into ES.
- We present experimental results to show that *ElasticHash* achieves fast response times and high-quality retrieval results at the same time by leveraging the benefits of short hash codes (better search times) and long hash codes (higher retrieval quality). To the best of our knowledge, we provide the first evaluation of image similarity search for more than 120,000 query images on about 6.9 million database images of the OpenImages data set.
- We make our deep image similarity search model, the corresponding ES indices, and a demo application available at <http://github.com/umr-ds/ElasticHash>.

Parts of this section have been published in: Nikolaus Korfhage, Markus Mühling, and Bernd Freisleben. "ElasticHash: Semantic Image Similarity Search by Deep Hashing With Elasticsearch." in: *Computer Analysis of Images and Patterns: 19th International Conference, CAIP 2021, Virtual Event, September 28–30, 2021, Proceedings, Part II 19*. Springer. 2021, pp. 14–23.

²<https://www.elastic.co>

³<https://lucene.apache.org>

4.2.2 Related Work

Deep learning, in particular deep CNNs, led to strong improvements in content-based image similarity search. With increasing sizes of the underlying image databases, the need for an efficient similarity search strategy arises. Since high-dimensional CNN features are not suitable to efficiently search in very large databases, large-scale image similarity search systems focus on binary image codes for quantization or compact representations and fast comparisons rather than full CNN features.

Recently, several deep hashing methods were introduced [Zhu+16; Eri+15; Wan+15; Wan+18b; Cao+17a; Liu+16a; Cao+18b]. Many of them employ pairwise or triplet losses. While these methods often achieve state-of-the-art performance on their test data sets, they are not necessarily suitable for very large data sets and fine-grained image similarity search based on thousands of classes. Existing deep hashing methods are often trained using small CNNs that usually cannot capture the granularity of very large image data sets. Often, CNN models like AlexNet [KSH12] are used as their backbones, and they are usually evaluated on a small number of image classes [WKC12; Cao+18b; Eri+15; Zhu+16] (e.g., a sample of 100 ImageNet categories [Cao+18b], about 80 object categories in COCO [Lin+14], NUS-WIDE [Chu+09] with 81 concepts, or even only 10 classes as in MNIST or CIFAR). Additionally, the image dimensions in CIFAR and MNIST are very small (32x32 and 28x28, respectively) and thus not sufficient for image similarity search in real-world applications. Many approaches are trained on relatively small training data sets (e.g., 10,000 - 50,000 images [Cao+17b; Cao+18b; Liu+16a]). In addition, there are no standardized benchmark data sets, and each publication uses different splits of training, query, and database images, which further complicates a comparison of the methods. Furthermore, training from scratch can be prohibitively expensive for large data sets. We observed that for large data sets with a high number of image classes, a transfer learning approach that combines triplet loss and classification loss leads to good retrieval results. To the best of our knowledge, *ElasticHash* is the first work that presents a deep hashing model trained and evaluated on a sufficiently large number of image classes.

The currently best performing approaches for learning to hash image representations belong either to product quantization (PQ) methods [JDS10; JDJ19] and methods based on deep hashing (DH) [Wan+15; Eri+15]. Amato et al. [Ama+18] present PQ approaches that transform neural network features into text formats suitable for being indexed in ES. However, this approach cannot match the retrieval performance of FAISS [JDJ19]. Therefore, we focus on deep hashing that in combination with multi-index hashing (MIH) [NPF12] can circumvent exhaustive search in Hamming space and achieve low search latency while maintaining high retrieval quality.

ElasticHash is related to other image similarity search methods integrated into ES. For example, FENSHSES [Mu+19] integrates MIH into ES and has a search latency comparable to FAISS. The method works efficiently for small radii of the Hamming ball and relatively small data sets (500,000 images). Small hamming radii, however, often produce too few neighbors for a query [NPF12]. MIH like FENSHSES is thus not suitable for our scenario of large-scale image retrieval in ES with long binary codes (256 bits), where we require sub-second search latency on a data set of about 7 million images. Furthermore, we solve the shortcomings of FENSHSES using only a subset of bits rather than the whole hash codes to perform our MIH-based coarse

search. While other works extend ES for image similarity search by modifying the Lucene library [Gen+10], our approach is seamlessly integrated into ES without modifying its code base.

4.2.3 *ElasticHash*

ElasticHash consists of several components as shown in Figure 4.6: a deep hashing component, an ES cluster, and a retrieval component. The deep hashing component is realized as a web service using Tensorflow Serving where the integrated deep hashing model is applied to images and the corresponding binary codes are returned. In the first phase, the binary codes are extracted from the database images in the indexing phase using the deep hashing component and stored into the ES cluster. After initially building the index, the retrieval component handles incoming query images and visualizes the retrieval results. For this purpose, the binary codes are extracted from the query images using the web service, the corresponding ES queries are assembled and sent to the ES cluster that returns the final list of similar images. The entire similarity search system can be easily deployed for production via Docker.

The deep hashing model is described in more detail in Section 4.2.3, including the training strategy and network architecture. In Section 4.2.3, the ES integration is presented.

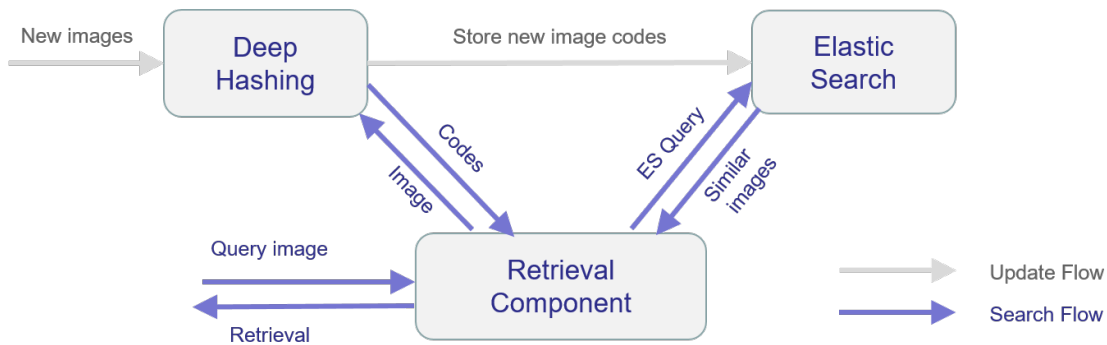


Figure 4.6: Overview of the workflows for image similarity search in ES.

Deep Hashing Model

We now describe our deep hashing model and how it is used to extract both short and long hash codes. The model training consists of two phases that both use ADAM as the optimization method. First, an ImageNet-pretrained EfficientNetB3 [TL19] model is trained on a data set with a larger number of classes in order to obtain a more fine-grained embedding. In contrast to the original ImageNet dataset, it contains all ImageNet classes with more than 1000 training images and all classes of the Places2 [Zho+17] data set, which results in a total number of classes of 5,390. The model is trained with cross-entropy loss on a Softmax output. After two epochs of training the final layer with a learning rate of 0.01, all layers are trained for another 16 epochs with a learning rate of 0.0001.

In the second phase, the classification model's weights are used to initialize the deep hashing model. This model includes an additional 256-bit coding layer before the class output layer with \tanh activation and 256 outputs. This model is trained for 5 epochs with a learning rate of 0.0001. It is trained on the same data set as before, however, by combining cross-entropy loss on the output and hard triplet loss [SKP15b] on the coding layer.

With the classification loss

$$\mathcal{L}_c = \sum_{i=1}^K y_i \log p_i \quad (4.2)$$

for K classes with labels y_i and predictions p_i , and the triplet loss

$$\mathcal{L}_t = \max(d(a, p) - d(a, n) + \gamma, 0) \quad (4.3)$$

for Euclidean distance d between the 256-dimensional output of the coding layer for anchor image a and positive example p and between a and a negative example n , respectively, the combined loss function is given by:

$$\mathcal{L} = \alpha \mathcal{L}_c + \beta \mathcal{L}_t, \quad (4.4)$$

where we set margin $\gamma = 2$ and weights $\alpha = 1$ and $\beta = 5$. We first sample a batch of size $b = 128$ images from a uniform distribution of the classes. This batch is used for both computing the classification loss and generating b hard triplets. To make the similarity search more robust, we used heavy data augmentation in both phases, which in addition to standard augmentation methods includes inducing JPEG compression artifacts. After training, the model generates 256-bit codes. These codes can be decomposed into four 64-bit codes for fast computation of Hamming distance on long integers. However, using codes of this length on a corpus of about 10 million images is too expensive, even when using multi-index hashing. We therefore extracted 64-bit codes from the original 256-bit codes to perform the filtering on shorter codes and thus smaller Hamming ball radii. To extract the 64 most important bits from the 256-bit codes, we first compute the correlations between bit positions (see Figure 4.7) and select the bits with the lowest correlation to all remaining bits. Then we partition the 64-bit codes into four partitions with 16 bits by applying the Kernighan-Lin algorithm [KL70] on the bit correlations.

Integration into ES

Before describing our image similarity search integration into ES, we will shortly review MIH in Hamming space [NPF12]. The idea of MIH is based on the following observation: for two binary codes $h = (h^1, \dots, h^m)$ and $g = (g^1, \dots, g^m)$ where m is the number of partitions, h^k and g^k are the k^{th} subcodes and H is the Hamming norm, the following proposition holds:

$$\|h - g\|_H \leq r \Rightarrow \exists k \in \{1, \dots, m\} \left\| h^k - g^k \right\|_H \leq \left\lfloor \frac{r}{m} \right\rfloor \quad (4.5)$$

For the case of 64-bit codes that are decomposed into $m = 4$ subcodes, this means that a code is in a Hamming radius $r < 12$ if at least one of the subcodes has a distance of $d \leq \lfloor \frac{r}{m} \rfloor = 2$ from the query subcode. The performance of MIH can be increased if the subcodes are maximally

4.2 ElasticHash: Semantic Image Similarity Search in Elasticsearch

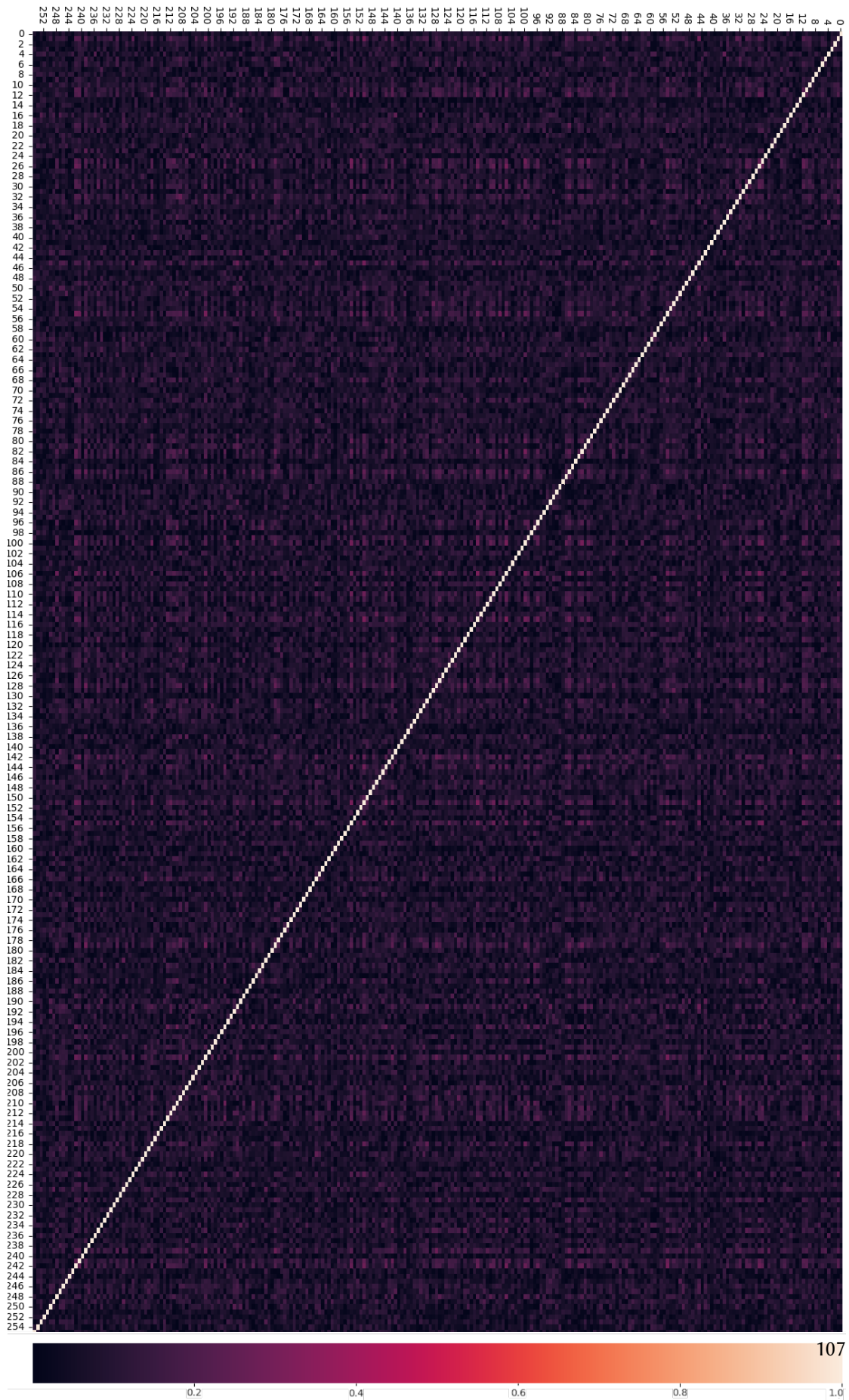


Figure 4.7: Bit correlations of 10,000 hashcodes.

4 Image Similarity Search

independent of each other [Wan+13a], especially for shorter codes [Mu+19]. Thus, after training a deep hashing model, the bit positions should be permuted accordingly.

The ES index used for retrieval contains four short codes ($f_0 - f_3$) and four long subcodes ($r_0 - r_3$) for each image. The short codes are used for MIH and efficiently utilize the reverse index structure of ES and are thus separated into four subcodes of type "keyword". The long codes are also separated into four subcodes in order to allow fast computation of Hamming distances for values of type long as shown in Listing 4.1.

```
PUT /es-retrieval/default/_mapping
```

```
1 {
2   "properties": {
3     "image": {"type": "text"},
4     "f_0": {"type": "keyword"},
5     "f_1": {"type": "keyword"},
6     "f_2": {"type": "keyword"},
7     "f_3": {"type": "keyword"},
8     "r_0": {"type": "long"},
9     "r_1": {"type": "long"},
10    "r_2": {"type": "long"},
11    "r_3": {"type": "long"}
12  }
13 }
```

Listing 4.1: Mapping of retrieval index.

An additional index is used for fast lookup of neighboring subcodes within the retrieval query. The neighbors index does not change once it has been created and merely serves as an auxiliary index for term queries. It requires pre-computing all nearest neighbors for all possible 16-bit subcodes. Thus, the index of neighbors contains 2^{16} documents. The document id corresponds to the unsigned integer representation of a 16-bit subcode and can therefore be accessed within a term query. It contains a single field "nbs" that is assigned to a list of all neighboring 16-bit codes within a Hamming radius of d of the corresponding query subcode. Since this index basically works as a lookup table, it could also be realized somewhere else, i.e., not as an ES index. However, integrating the lookup table this way eliminates the need for external code and enables fast deployment of the whole system. All documents representing all possible 16-bit subcodes are inserted according to the query in Listing 4.2.

```
POST /nbs-d2/_doc/<16 bit subcode>
```

```
1 {
2   "nbs": [<d2 neighbors of 16 bit subcode>]
3 }
```

Listing 4.2: Query for adding an entry to neighbor lookup index.

In this stage, MIH is realized by querying the additional index of neighbors for fast neighbor lookup. Even with MIH, using the full code length of the deep hashing model trained for 256-bit codes is too expensive for larger databases. We therefore limit the code length for the filtering stage to 64-bit codes. To obtain a sufficiently large set of candidate hash codes in the first stage, we need to search within a Hamming ball with a correspondingly large radius. We set $d = 2$,

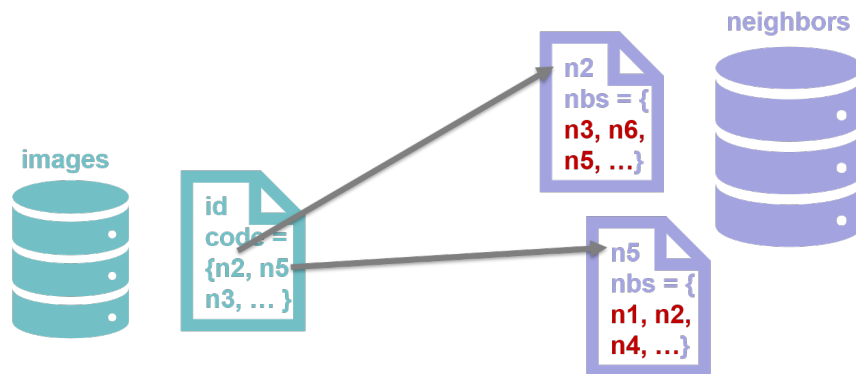


Figure 4.8: A document representing an image (green) is related via subcodes to documents holding lists of neighboring subcodes in the neighbors index (violet).

which will return at least all codes within $r = 11$ of a 64-bit code. In our setting with $d = 2$, this results in 137 neighbors per subcode, i.e., 548 neighbors in total.

In ES, we realize MIH by using a terms lookup. It fetches the field values of an existing document and then uses these values as search terms (see Listing 4.2). In contrast to putting all neighbors into the query, using a dedicated index for subcode neighbors has the advantage that the retrieval of neighboring subcodes is carried out within ES. Thus, the query load is small, and no external handling of neighbor lookup is necessary. The relation between a document and the neighbors index are shown in Figure 4.8.

In the second stage, all codes obtained by MIH are re-ranked according to their Hamming distance to the long code. To compute the Hamming distance of the 256-bit code, the Painless Script in Listing 4.3 is applied to each of the four subcodes.

POST `_scripts/hd64`

```

1 {
2   "script": {
3     "lang": "painless",
4     "source": "64-Long.bitCount(params.subcode~doc[params.field].value)"
5   }
6 }
```

Listing 4.3: Query for adding a Painless Script.

The query in Listing 4.4 combines the MIH step as a filter with a term query and the re-ranking step as an application of the painless script from Listing 4.3 on the filtered retrieval list.

GET /es-retrieval/_search

```

1 {
2   "query": {
3     "function_score": {
4       "boost_mode": "sum",
5       "score_mode": "sum",
6       "functions": [
7         ...,
8         {
9           "script_score": {
10            "script": {
11              "id": "hd64",
12              "params": {
13                "field": "r_<i>",
14                "subcode": "<64 bit subcode for re-ranking>"
15              }
16            }
17          },
18          "weight": 1
19        },
20        ...
21      ],
22      "query": {
23        "constant_score": {
24          "boost": 0.,
25          "filter": {
26            "bool": {
27              "minimum_should_match": 1,
28              "should": [
29                ...,
30                {
31                  "terms": {
32                    "f_<j>": {
33                      "id": "<16 bit subcode for lookup>",
34                      "index": "nbs-d2",
35                      "path": "nbs"
36                    }
37                  }
38                },
39                ...
40              ]
41            }
42          }
43        }
44      },
45    }
46  }
47 }

```

Listing 4.4: Query for performing two-stage similarity search.

4.2.4 Experimental Evaluation

To determine the search latency and retrieval quality of *ElasticHash*, we evaluate three settings for using the binary hash codes generated by our deep hashing model for large-scale image retrieval in ES: (1) short codes, i.e., 64 bits for both filtering and re-ranking, (2) long codes, i.e.,



Figure 4.9: Top-10 retrieval results for (a) short codes, (b) long codes, and (c) *ElasticHash* for the same query image (first on the left); green: relevant result; red: irrelevant result.

256 bits for both filtering and re-ranking, and (3) *ElasticHash*, i.e., 64 bits for filtering, 256 bits for re-ranking. Settings (1) and (2) are similar to the MIH integration of Mu et al. [Mu+19].

To evaluate our approach, we use OpenImages [Kuz+20], which is currently the largest annotated image data set publicly available. It contains multi-label annotations for 9.2 million Flickr images with 19,794 different labels and is partitioned into training, validation, and test data set. On the average, there are 2.4 positive labels for the training split, while the validation and test splits have 8.8. As our database images we use all training images being available when downloading the data set, i.e., 6,942,071 images in total. To evaluate the retrieval quality, we use all downloaded images from the OpenImages test and validation set as query images (121,588 images in total). From these images, we draw a sample of 10,000 images to measure the search latencies for the three different settings.

The quality of the retrieval lists is evaluated using the average precision (AP) score, which is the most commonly used quality measure in image retrieval. The AP score is calculated from the list of retrieved images as follows:

$$AP(\rho) = \frac{1}{|R \cap \rho^N|} \sum_{k=1}^N \frac{|R \cap \rho^k|}{k} \psi(i_k), \text{ with } \psi(i_k) = \begin{cases} 1 & \text{if } i_k \in R \\ 0 & \text{otherwise} \end{cases} \quad (4.6)$$

where N is the length of the ranked image list, $\rho^k = \{i_1, i_2, \dots, i_k\}$ is the ranked image list up to rank k , R is the set of relevant documents, $|R \cap \rho^k|$ is the number of relevant images in the top- k of ρ and $\psi(i_k)$ is the relevance function. We consider an image as relevant, if it has at least one label in common with the query image. To evaluate the overall performance, the mean AP score is calculated by taking the mean value of the AP scores over all queries.

Results

We first evaluate the search latency for queries. Next, we compare the retrieval quality in terms of AP. The experiments were performed on a system with an Intel Core i7-4771 CPU @ 3.50GHz and 32 GB RAM.

Table 4.1 shows that for a k up to 250 there is no notable decrease in retrieval quality when using *ElasticHash* rather than using long codes for both stages. Figure 4.9 shows examples of

4 Image Similarity Search

	top k						
	10	25	50	100	250	500	1000
ES 64 MIH (short)	87.94	86.08	84.44	82.54	79.41	76.44	72.86
ES 256 MIH (long)	95.35	94.72	94.23	93.71	92.90	92.09	90.95
<i>ElasticHash</i>	95.21	94.48	93.90	93.22	92.02	90.61	88.42

Table 4.1: Retrieval quality (mean AP) for different thresholds of k on 121,588 query images.

		top k						
		10	25	50	100	250	500	1000
FAISS 64 ex.	μ	22.55	22.59	22.62	22.61	22.68	22.74	22.93
	σ	1.92	1.89	1.93	1.90	1.93	1.90	2.04
FAISS 256 ex.	μ	160.90	160.54	161.11	161.26	161.44	161.24	161.26
	σ	17.14	17.06	17.33	17.99	18.03	17.78	17.38
FAISS 64 MIH	μ	31.05	34.02	29.52	29.88	29.58	29.89	30.17
	σ	13.46	14.81	12.42	12.46	12.40	12.46	12.48
FAISS 256 MIH	μ	209.31	225.63	234.05	219.78	217.57	221.32	222.43
	σ	43.03	46.78	45.33	42.99	43.27	42.74	42.20
FAISS IVF 64	μ	45.74	50.40	46.04	48.36	45.90	45.74	49.16
	σ	2.59	6.45	2.61	4.62	2.49	2.46	5.17
FAISS IVF 256	μ	110.61	108.66	108.28	108.31	109.11	109.13	109.94
	σ	4.58	3.35	2.99	3.11	3.12	3.02	4.48
ES 64 MIH (short)	μ	23.09	23.98	24.45	25.58	28.38	33.09	42.20
	σ	4.74	4.65	4.70	4.72	4.86	5.20	6.07
ES 256 MIH (long)	μ	111.83	111.58	111.99	113.05	116.77	121.98	132.60
	σ	16.50	16.58	16.72	16.54	17.04	17.13	17.99
<i>ElasticHash</i>	μ	36.12	36.75	37.28	38.17	40.88	45.73	55.23
	σ	7.80	7.96	7.81	7.89	7.93	8.12	8.64

Table 4.2: Search latencies for ES and FAISS queries (ms) with standard deviation for different thresholds of k on 10,000 query images.

the top-10 retrieval results for the three settings. The retrieval quality of *ElasticHash* is similar to using long codes, and both are superior to using short codes. On the other hand, Table 4.2 indicates that the average retrieval time only slightly increases compared to using short codes for both stages. This suggests that *ElasticHash* is a good trade-off between retrieval quality and search latency. Although our deep hashing model was trained on 5,390 classes, but almost 20,000 classes occur in the validation data set, high AP values are achieved for *ElasticHash*. For comparison, Table 4.2 also shows the query times for several approaches in FAISS [JDJ19] with

binary indices of length 64 and 256, respectively. We evaluate FAISS with exhaustive search (ex.), multi-index hashing (MIH), and inverted index file (IVF). For IVF in FAISS we perform a range search within a radius < 12 for short codes, and a radius < 48 , respectively. The unordered results are then re-ranked.

4.2.5 Summary

We presented *ElasticHash*, a novel two-stage approach for semantic image similarity search based on deep multi-index hashing and integrated via terms lookup queries into ES. Our experimental results on a large image data set demonstrated that we achieve low search latencies and high-quality retrieval results at the same time by leveraging the benefits of short hash codes (better search times) and long hash codes (higher retrieval quality).

5

Segmentation-based Image Similarity Search via Region Prompts

5.1 Introduction

Image similarity search is used to find visually similar images to a given query image in large image collections. It is employed in various application areas, such as e-commerce [Ak+18], healthcare [Sil+22; KRS20; Qay+17], art-historical research [Spr+21], research in historical video archives [Müh+19], or as part of a visual data acquisition workflow [Müh+22].

Image similarity search systems allow users to efficiently search in large sets of images by storing images as compact feature vectors. This means that usually entire images are indexed. However, a desirable property of an image-based search system is to not only enable whole-image search, but to search for certain image areas or objects that are of interest within an image. A straightforward solution is to use an object detector, detect and classify all objects in all images of the image collection, and set up an index for all detected objects. This approach is disadvantageous for the following reasons. First, it only works for objects, not arbitrary regions

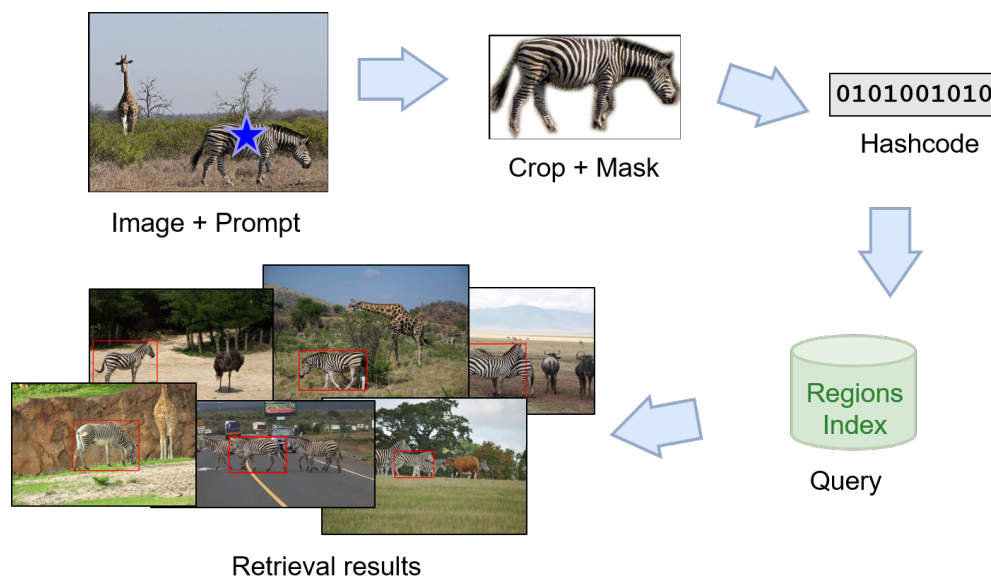


Figure 5.1: Example query image with a point prompt for segmentation-based similarity search.

within an image. Second, it is limited to a set of object classes used for training, i.e., it is biased towards the trained object categories. Even if the object detector is replaced by a panoptic segmentation approach [Kir+19], the search would still be limited to predefined categories, and robustness to distribution shifts is limited.

In this section, we present a novel approach to perform image similarity search that is not limited to a dictionary of object categories, but is based on publicly available foundation models that we use for image encoding and image segmentation. Our approach allows users to better express their search intent by enabling a more refined search for objects or image regions. This search intent can be conveyed via prompts, such as point, box, and text prompts. Fig. 5.1 shows how a query is processed in our approach, called *Search Anything*. First, a user selects a query image. Then, the user selects an image region of this query image via a prompt. The selected image region is automatically segmented. Next, a binary feature vector is extracted based on the masked region. This feature vector is used as a query for an image region index, and the user gets back as a result the images that contain the corresponding region.

Search Anything is trained in a self-supervised manner on mask features extracted by the FastSAM foundation model [Zha+23] and semantic features for masked image regions extracted by the CLIP foundation model [Rad+21] to learn binary hash code representations for image regions. In our experimental evaluations based on several datasets from different domains in a zero-shot setting, we show that combining region masks, segmentation features, and semantic features improves region retrieval performance. If objects or regions are sufficiently large, an improvement in retrieval quality can be achieved if they are masked and thus only features from the image region are considered. For small objects, on the other hand, context is important, and useful features can be extracted from the context.

Our contributions are as follows:

- We present *Search Anything*, a novel approach to use both segmentation features and semantic features for region-based image similarity search, relying on image-based prompts, such as point and box prompts, originally used in image segmentation.
- We experimentally evaluate that masking and the use of a combination of segmentation features and semantic features leads to better search results.
- To the best of our knowledge, our approach is the first approach to apply and evaluate region-based image similarity search in a zero-shot setting.

Parts of this section are based on: Nikolaus Korfhage, Markus Mühling, and Bernd Freisleben. “Search Anything: Segmentation-based Similarity Search via Masked Region Prompts.” in: *Submitted; Under Review*. 2024.

5.2 Related Work

5.2.1 Foundation Models

Foundation models [KRS20] are large neural network models trained on large amounts of data. In our work, we combine foundation models trained for different tasks in the vision domain to

create a general-purpose image-based similarity search approach. Popular foundation models for image segmentation are CLIPSegment [LE22], SAM [Kir+23], and more recently, FastSAM [Zha+23]. We make use of FastSAM, since inference with FastSAM is an order of magnitude faster than with SAM, while its segmentation performance is similar to the original SAM model. SAM was not trained with explicit semantic supervision, but it can capture some semantics in its representation [Kir+23]. However, it still lags behind the predictive performance of models for image encoding trained with explicit semantic information, i.e., text. We therefore enhance mask features by combining them with semantic features. We use CLIP [Rad+21] for extracting features of (masked) image regions, since it was trained contrastively on text. In particular, CLIP’s image encoder has shown a good zero-shot retrieval performance due to its contrastive training on image-text pairs. We argue that we can improve the retrieval quality by combining these opposing features. Models trained for general purpose segmentation tend to encode more low-level information such as shape and color, rather than semantic categories. In contrast, models trained with contrastive learning and strong data augmentation are explicitly trained to ignore such information. We propose a novel approach that allows users to search for regions of an arbitrary natural image, taking into account both semantic and segmentation features. The combination of two foundation models allows us to accomplish this goal without a large training effort and without the need for any manual annotations.

5.2.2 Image Similarity

What ‘similarity’ means for images depends on the problem to be solved and its context. Similarity may range from a general concept present in a query image, layout, specific instances of objects [CAS20; Che+22], or attributes such as color, texture, and shape, and is ultimately determined by a user’s search intention [KMF20].

Most current approaches focus on one specific notion of similarity by learning a similarity function that is either determined by specific concepts used during training or by the underlying data distribution, as in self-supervised learning [VCM23]. However, there are often several types of competing semantic aspects that are of interest, such as the general concept, color, and shape. Conditional Similarity Networks (CSN) [VBK17] were introduced to handle contradicting notions of similarity within a single model by learning embeddings, i.e., different similarities are encoded in separate subspaces. In contrast, we combine two notions of similarity that are in general not mutually exclusive: the general concept of a region and its exact visual characterization given by attributes such as shape, color, and texture.

We implicitly combine two different measures of similarity as follows: (a) the implicit similarity function learned by vision-text contrastive models such as CLIP, which we refer to as semantic similarity, and (b) similarity between features learned by general purpose segmentation models such as SAM, which we refer to as segmentation similarity. This distinction is somewhat arbitrary, since, for example, information on attributes such as color and shape can also be implicitly encoded in the features of vision-language models, e.g., if they are specific to a semantic concept, and conversely, some semantic information is learned by segmentation models trained for general purpose segmentation [Kir+23]. Nevertheless, the corresponding feature space differs fundamentally due to the underlying training data and method. By combining semantic similarity and segmentation similarity, we show that we can improve

zero-shot region retrieval performance. We assume that (a) images are natural images, (b) certain objects or regions in a query image are of interest, and (c) both semantic similarity and the exact visual similarity, given by attributes such as shape, color, texture, etc., should be reflected in the retrieval list.

5.2.3 Prompts

The use of prompts originally refers to user interaction with large language models such as GPT-3 [Bro+20] via text prompts. CLIP supports text prompts for large vision-language models. In CLIPSegment, text prompts are used to segment images. More recently, image-specific prompts were introduced for general image segmentation by SAM. They include box and point prompts and seem particularly appealing to be used for image-based similarity search to refine a user’s search intent. These kinds of intuitive prompts complement existing image-based search input methods such as, i.e., drawing sketches in sketch-based image retrieval [Yel+18; Lin+23; Tur+22] or drawing (partial) segmentation maps [FIY19]. Enabling region-based prompts could also be useful for content-based image retrieval systems that already have a variety of search options (e.g. iArt [Spr+21]), but still lack the ability to search for specific image regions.

5.2.4 Zero-shot Region Retrieval

Our approach is related to some approaches proposed for region retrieval [Jin+04; HGC10; VFE15] and object retrieval [Hoi+04; KPK03]. In contrast to previous region retrieval approaches that consider the spatial configuration of semantic regions in a query image and retrieve similar images [FIY19; Mai+17; HMS17], we consider database images and query images as collections of regions that can be searched for. Available region retrieval methods are usually trained on labeled training data and are not applicable to a general-purpose region retrieval approach. In contrast, our approach leverages the generalization capability of foundation models and acts as a zero-shot image region retrieval system. This is also different from recent zero-shot sketch-based image retrieval (ZS-SBIR) approaches [Lin+23; Sai+23; Yel+18; Tur+22] that use a sketch as its input, whereas our system uses region prompts on a query image as its input.

5.2.5 Large-scale Similarity Search

While the problem of extracting features quickly can be solved by efficient architectures (e.g., Convolutional Neural Network (CNN) architectures such as FastSAM), a significant fraction of query time is spent on nearest neighbor search. Depending on the method used to identify regions in an image, there easily can be more than 100 regions per image, which means that the index becomes orders of magnitude larger than for conventional image similarity search. This requires an efficient representation of the feature vectors. Besides using methods based on product quantization [Guo+20; JDS10], binary representations can be learned using deep hashing [Eri+15; Lin+15; Luo+23]. Since we do not use class-labeled data for training our hashing model, we have to rely on semantic information encoded in the features extracted from the foundation models. In unsupervised or self-supervised deep hashing [She+18; Yan+19], a small neural network or single layer is trained to learn the distance from feature vectors on unlabeled

training data. Although not yet applied to the problem of region retrieval, existing approaches [Luo+23] have been extensively evaluated on features extracted from image classification models. In our similarity search approach, we use FAISS [JDJ19] for indexing binary vectors for image regions.

5.3 Approach

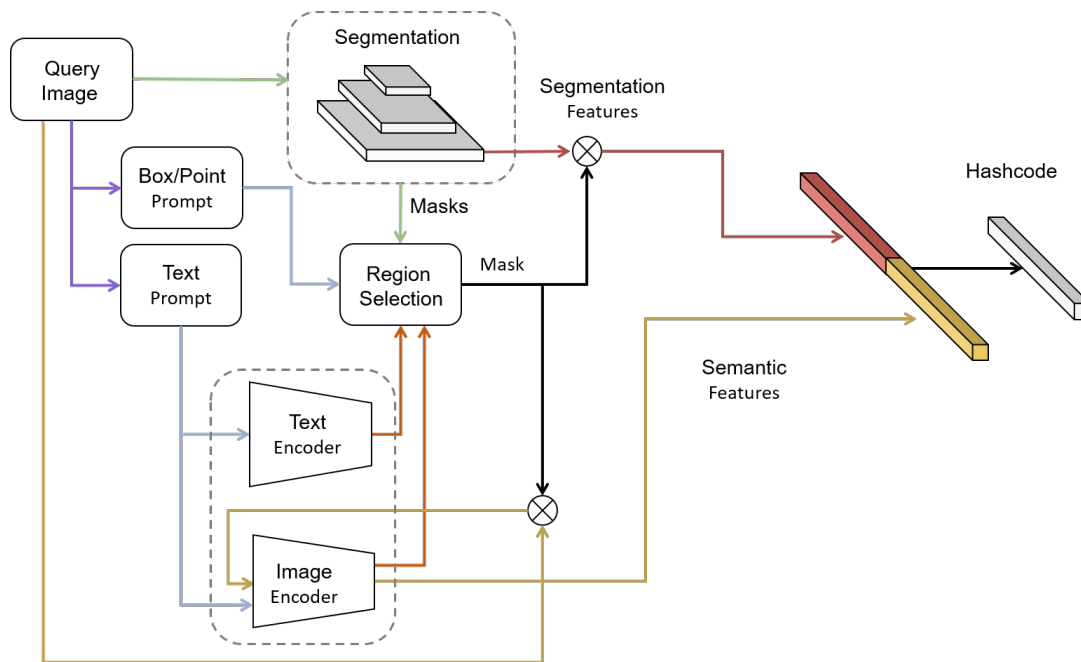


Figure 5.2: *Search Anything*: similarity search with region prompts. First, a user selects a query image (1). Then, the user refines the search with a region prompt (2). Segmentation features and semantic features are masked and concatenated. The combined feature vector is compressed by deep hashing.

Figure 5.2 shows our approach, called *Search Anything*, for generating compact binary codes for image regions in a query image via prompts. First, a user chooses a query image that contains an object or region of interest. This query can then be refined by the user via various prompts, such as point prompt, box prompt, or text prompt (violet). The query image is then segmented by a foundation model for image segmentation. The segmentation masks of the query image are used in the subsequent region selection step (green). It selects the corresponding query object or region according to the user’s prompt (blue). In case of a text prompt, an image-to-text foundation model is used to match the text encoding (brown) to one of the query image’s segmentation masks. The image encoder of the image-to-text model is then used again for obtaining generalized image features (yellow) for the selected region. Depending on the size of the region, the input to the model is masked with the selected segmentation mask (\otimes in Fig. 5.2 denotes pixel-wise multiplication). The same mask is used when creating a segmentation feature vector (red), but here, instead of masking the input, the feature map extracted from the feature pyramid [Lin+17a] of the segmentation model is masked. Then, the semantic feature

vector is concatenated with the extracted segmentation feature vector of the segmentation model. Finally, the concatenated feature vector is fed into a deep hashing neural network that returns a compact binary representation of the combined mask and semantic image features for the image region. Finally, the obtained region-of-interest hashcode is used to efficiently search the database of image regions.

5.3.1 Region Prompts

To process the prompts and extract the segmentation masks for selected regions, we use the FastSAM [Zha+23] approach for dividing the segment anything task into two stages. In the first stage, the masks for all regions in the query image are generated using a CNN-based object detector (i.e., All Instance Segmentation). In the second stage, the corresponding mask is assigned to the prompt (i.e., Prompt-guided Selection). The generation of segmentation masks for all instances in the image is performed by the segmentation model’s backbone, which is a YOLOv8 object detector [JCQ23]. Both stages are summarized in Fig. 5.2 as *Region Selection*; they enable the utilization of point prompts, box prompts, and text prompts. While in SAM prompts are part of the transformer-based architecture as inputs, in FastSAM prompts are processed after segmentation has been performed. But image-based prompting works the same way: as with SAM, foreground and background points can be set. If a foreground point lies in several masks, background points are used to exclude irrelevant masks, and multiple foreground points are used to merge segmentation masks into a single mask. With a box prompt, a user can draw a box around a region-of-interest. By matching the Intersection over Union (IoU), the corresponding mask is assigned. Finally, text prompts offer text-based queries to specify a region-of-interest within the query image. Text and image embeddings are extracted from CLIP, and the masked features with the highest similarity score to the text embedding are selected. While text prompt processing is the most expensive step at inference time, the image embeddings can be used directly for the following semantic feature extraction step. In the context of region-based retrieval, these different prompts are used to specify regions as queries, and we therefore refer to them as *region prompts*. Based on the selection of a segmentation mask for the region prompt, we extract segmentation features for the corresponding prompt region, from the highest resolution level of the feature pyramid of the segmentation model and semantic features from the image encoder.

5.3.2 Deep Hashing

We use a hashing layer for compressing masked region features. It generates a compact representation of length L for a given image region $x \in \mathbb{R}^{H \times W \times 3}$ as follows:

$$\text{hash}_L(x) = \tanh(W(f_M(x) \oplus f_I(x)) + b), \quad (5.1)$$

where \oplus denotes concatenation. The segmentation features f_M are obtained as follows. From the highest resolution layer of the feature pyramid of the segmentation model, each channel is multiplied with the predicted segmentation mask and then average-pooled. This results in a 320-dimensional feature vector (corresponding to the number of channels of the highest resolution layer). The semantic feature vector f_I is extracted from the image encoder of the

CLIP model by feeding it with the masked image crop. To obtain robust image features, we use OpenCLIP ViT-H-14 [Ilh+21] that showed a high zero-shot performance and is robust to natural distribution shifts [Rad+21]. The image feature vector is 1024-dimensional.

5.3.3 Training

To train our deep hashing model in a self-supervised manner, we use the Segment Anything 1 Billion (SA-1B) [Kir+23] dataset. It contains class-agnostic mask annotations for 11M images, i.e., segmentation masks without labels. SA-1B was originally used for training foundation models for general-purpose object segmentation. Masks can contain regions, objects or parts, and overlapping masks occur. It is the largest segmentation dataset available. It contains an average of 100 segmentation masks per image.

Similar to generating the inputs for the hashing network, the concatenated feature vector used for computing the target cosine distances is extracted directly from the two foundation models. From the highest resolution layer of the feature pyramid of the segmentation model, each feature map is multiplied with the groundtruth segmentation mask and then spatially averaged. The semantic features are obtained from the image encoder by masking the input crop. Both vectors are then concatenated to form the 1344-dimensional vectors used for computing the inner-batch target cosine distances.

To learn hash codes in a self-supervised manner, we use

$$\mathcal{L} = \|\cos(t_1, t_2) - \cos(\text{hash}_L(x_1), \text{hash}_L(x_2))\|_2^2 \quad (5.2)$$

as our loss function to minimize the difference of the cosine distances within a batch, similar to Su et al. [Su+18], where t_1 and t_2 are the concatenated mask and semantic image feature vectors of a region, as described above for two regions x_1 and x_2 .

We train with batches of 24 images and sample 32 groundtruth masks per image. The effective mask batch size is then 768 masks. With AdamW [LH18] we train for about 70K batches (about 50M regions). This corresponds to about 5% of the whole SA-1B dataset.

5.3.4 Region Indexing

Adding an image region to the index requires segmenting the image. One possibility could be to use panoptic segmentation for this purpose. However, this would fall short if, for example, parts of objects are formulated as a query region. In this case, it is necessary that segmentations of partial areas are also indexed. We therefore use a more general approach and apply the *All Instance Segmentation* step proposed for FastSAM. This typically results in a number in the order of hundreds of regions. To decrease the number of regions per image in the database, if necessary, we can, for example, order the objects by size and either drop small objects below a certain threshold or only keep a maximum of top k masks per image.

5.4 Experiments

We first evaluate how the proposed combination of semantic features and segmentation features together with masking improves the performance of *Search Anything* over the baseline using CLIP features alone. Next, we evaluate the zero-shot region retrieval performance of our trained deep hashing model. We apply it to multiple datasets of different domains and use the datasets’ bounding box and class annotations for indexing and as box prompts for query images, respectively. Finally, we present qualitative results where we perform several box queries on the query image. In contrast to our zero-shot retrieval evaluation, in this scenario we also apply region extraction in the indexing phase. The zero-shot region retrieval performance is evaluated in terms of mean average precision (mAP). For all experiments, we index binary hash codes with FAISS [JDJ19].

5.4.1 Datasets

To evaluate our zero-shot object and region retrieval performance, we split the available segmentation datasets into a set of query regions and a set of database regions, respectively. For this purpose, we took already existing splits of the datasets (training, validation, test) and assigned them to a query set and a database set accordingly, whereas the database set is always larger than the query set. Although all datasets contain natural images, they each differ in their underlying problems. All datasets are detection datasets, i.e., they have at least one bounding box annotation per image. These box annotations are used as queries to simulate user box prompts, as well as for indexing the database of objects. The individual partitions and dataset-specific challenges are described in more detail below.

COCO 44K is a dataset that we created from COCO [Lin+14]. It is a subset of COCO and contains 44k images of objects. Since the computations with uncompressed feature vectors are expensive, instead of using all objects occurring in COCO, we use COCO 44K to evaluate different feature extraction strategies. The dataset consists of 29,488 query objects (taken from 4,000 validation images) and 289,403 database objects (taken from 40,000 training images). To evaluate the retrieval performance depending on the object size, we follow the definition of the COCO dataset to group objects into *small* (box area smaller than 32×32 pixels), *medium* (smaller than 96×96 pixels) and *large* (all other objects). The query set contains 8,773 small, 10,547 medium, and 10,168 large objects. The database set contains 86,532 small, 103,168 medium, and 99,703 large objects.

COCO Stuff 25K is a subset we created from COCO Stuff [CUF18]. COCO Stuff extends the original COCO dataset to a panoptic segmentation dataset by adding 91 stuff categories. All stuff annotations in the validation set are used as query regions, resulting in 32,801 regions corresponding to 5,000 images. From the stuff annotations in the COCO training dataset, we use 126,720 regions from 20,000 images database. By considering only the stuff annotations, this dataset allows us to more closely examine the retrieval performance of our approach for regions that are not objects.

PASCAL VOC contains 20 object categories and 1,464 images for training, 1,449 images for validation. It is widely used as a benchmark for object detection, semantic segmentation, and

classification. From the training images, we use all 5,823 objects as queries to the database of 5,717 objects.

Oxford-IIIT Pet Dataset [Par+12] contains 37 pet categories corresponding to breed, with roughly 200 images per category. We use it to evaluate our fine-granular retrieval performance. There is one bounding box annotation of the head and one segmentation map for the whole animal for each image. Instead of using the available head annotations, we created bounding boxes of the segmentation map for evaluating our zero-shot performance, which we consider as a hard task, since it has a higher intra-class variance. We use the datasets’ training and validation split to extract 3,669 query regions and 3,680 database regions.

Stanford Cars [Kra+13] has 16,185 images of 196 classes of cars with bounding box annotations. Class annotations are at the level of make, model, and year. The data is split into 8,144 training images, which we use as our database set, and 8,041 testing images that we use for querying.

WIDER FACE [Yan+16] has 32,203 images with 393,703 faces. There are multiple annotations per face: blur, expression, illumination, occlusion, pose, each with 2-3 characteristics, resulting in 72 possible labels. To be counted as positive, all annotated characteristics must match. We use faces with an area of at least 96×96 pixels, resulting in 2,120 faces (from 3,226 images) as queries and 8,647 faces (from 12,880 images) as database.

5.4.2 Region Features

The performance of our self-supervised deep hashing model crucially depends on the choice and extraction of suitable features, since these are used as targets in the cost function. Before training a deep hashing model on the mask and semantic image features for objects, we verified that these features are indeed useful for zero-shot image retrieval. Additionally, we confirmed that zero-shot image retrieval benefits from both mask and semantic image features. For small objects, we found that it is beneficial to consider the whole area around a query object rather than masking it. This is reasonable since very small objects often can be visually assigned to a class when considering its context (see Figure 5.3). The following experiments for extracting suitable features were conducted on *COCO 44K*.

Masking

We first evaluated the effect of masking input and feature maps, respectively. We can mask the features extracted from the feature pyramid of the segmentation model directly. If the semantic features are obtained from CLIP, the image encoding feature vector has no spatial dimensions and thus cannot be masked. However, we can mask the input image crop to ignore the unimportant image regions when computing the semantic features. Table 5.1 shows the object retrieval results on *COCO 44K* grouped by object sizes. Masking the input generally improves performance, however, for small and medium-sized objects we see a slight decrease in performance, if we mask the object instead of using the crop around the object. Masking the segmentation features has a higher impact on the quality of retrieval results, as shown in Table 5.2. The results also show that features of models trained for general-purpose segmentation, i.e., without class labels, contain some semantic information, even if the performance is

Size	Area	mAP@k (%)						
		10	25	50	100	250	500	1000
large	box	84.00	81.37	79.20	76.91	73.45	70.23	66.35
	mask	86.77	84.95	83.36	81.61	78.81	76.15	72.86
medium	box	79.76	77.11	74.92	72.60	69.09	65.89	62.21
	mask	74.56	71.74	69.34	66.89	63.44	60.53	57.38
small	box	60.14	56.59	53.64	50.75	47.10	44.42	41.76
	mask	56.10	52.72	49.85	47.04	43.63	41.19	38.79

Table 5.1: Input masking for semantic features (CLIP).

Size	Area	mAP@k (%)						
		10	25	50	100	250	500	1000
large	box	51.77	48.23	44.96	41.82	38.10	35.58	33.24
	mask	59.09	55.33	52.05	48.80	44.72	41.84	39.12
medium	box	39.24	36.19	33.45	30.95	27.98	26.10	24.44
	mask	43.19	39.63	36.73	33.97	30.62	28.43	26.44
small	box	38.22	35.45	32.91	30.51	27.79	26.01	24.40
	mask	38.76	35.52	32.87	30.33	27.39	25.46	23.74

Table 5.2: Feature masking of segmentation feature maps.

still significantly behind image encoders trained with annotated data such as CLIP’s image encoder.

Context

While masking objects seems to improve retrieval performance, Table 5.1 also shows that this is not the case for small and medium-sized objects. A possible cause for this effect is evident in Fig. 5.3. It shows small objects that are visually hard to recognize given only the cropped object, but with a larger context around the object, it is easier to recognize the object. This motivated us to consider the context around small objects, and we found that for small and medium-sized objects, context is generally beneficial. Thus, we next investigate how large the context boxes around the objects should be to obtain better results for small and medium-sized objects. From Table 5.3 it is evident that for small objects a context of 3 times the object’s bounding box gives the best results. Similarly, for medium-sized objects, using a crop box without masking improves the retrieval performance the most.

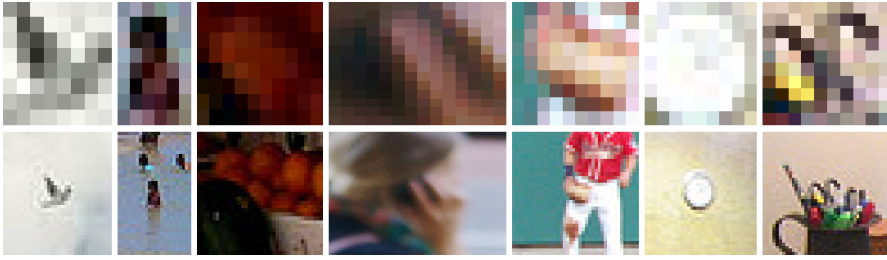


Figure 5.3: Small objects without context (top) and with a $3\times$ larger crop around the object (bottom). Labels (left to right): *bird, person, orange, cell phone, baseball glove, clock, scissors* [Lin+14].

Size		mAP@k (%)						
		10	25	50	100	250	500	1000
medium	mask	74.56	71.74	69.34	66.89	63.44	60.53	57.38
	1.0	79.76	77.11	74.92	72.60	69.09	65.89	62.21
	1.5	79.35	76.43	74.06	71.53	67.79	64.44	60.57
	2.0	75.37	71.92	69.19	66.42	62.47	59.10	55.38
small	2.5	71.59	68.17	65.38	62.58	58.64	55.44	52.00
	3.0	72.13	68.66	65.92	63.08	59.18	55.96	52.49
	3.5	71.91	68.44	65.64	62.83	58.97	55.78	52.33

Table 5.3: Context for semantic features.

Concatenated Features

Finally, we selected the best setting for feature extraction for both semantic features and segmentation features and compared it to combining both, i.e., concatenating the semantic and segmentation feature vectors. We showed that for large objects the retrieval performance can be improved when using masks and thereby effectively filtering out unimportant image regions. By using a foundation model to obtain image regions, we also obtain segmentation features without the need for additional computations. We therefore decided to use segmentation features by concatenating them with the semantic feature vector, although the results in Table 5.4 indicate that there is only a minor improvement in object retrieval performance in terms of average precision. However, some qualitative evaluations show that masked inputs together with segmentation features lead to better retrieval results, but these do not always translate into a better mAP value. Figure 5.4 shows the query regions (with red frames) and the top 5 results. The retrieval results in Figure 5.4 obtained by extracting only CLIP features mostly do not contain wrong classes according to the COCO annotations, but the results using *Search Anything* with masking often seem visually closer to the query region-of-interest than those of CLIP features alone. Our approach can also eliminate ambiguities, as shown in the top row.

5 Segmentation-based Image Similarity Search via Region Prompts

Size		mAP@k (%)						
		10	25	50	100	250	500	1000
all	segment	47.69	44.14	41.15	38.31	34.87	32.55	30.41
	semantic	78.95	76.30	74.18	71.98	68.86	66.16	62.96
	concat	79.07	76.43	74.29	72.10	68.98	66.28	63.09
large	segment	59.09	55.33	52.05	48.80	44.72	41.84	39.12
	semantic	86.77	84.95	83.36	81.61	78.81	76.15	72.86
	concat	86.78	84.96	83.37	81.62	78.84	76.19	72.90
medium	segment	43.19	39.63	36.73	33.97	30.62	28.43	26.44
	semantic	79.76	77.11	74.92	72.60	69.09	65.89	62.21
	concat	79.71	77.09	74.90	72.59	69.08	65.89	62.22
small	segment	38.76	35.52	32.87	30.33	27.39	25.46	23.74
	semantic	72.13	68.66	65.92	63.08	59.18	55.96	52.49
	concat	72.20	68.74	66.01	63.18	59.29	56.08	52.61

Table 5.4: Concatenated segmentation and semantic features.

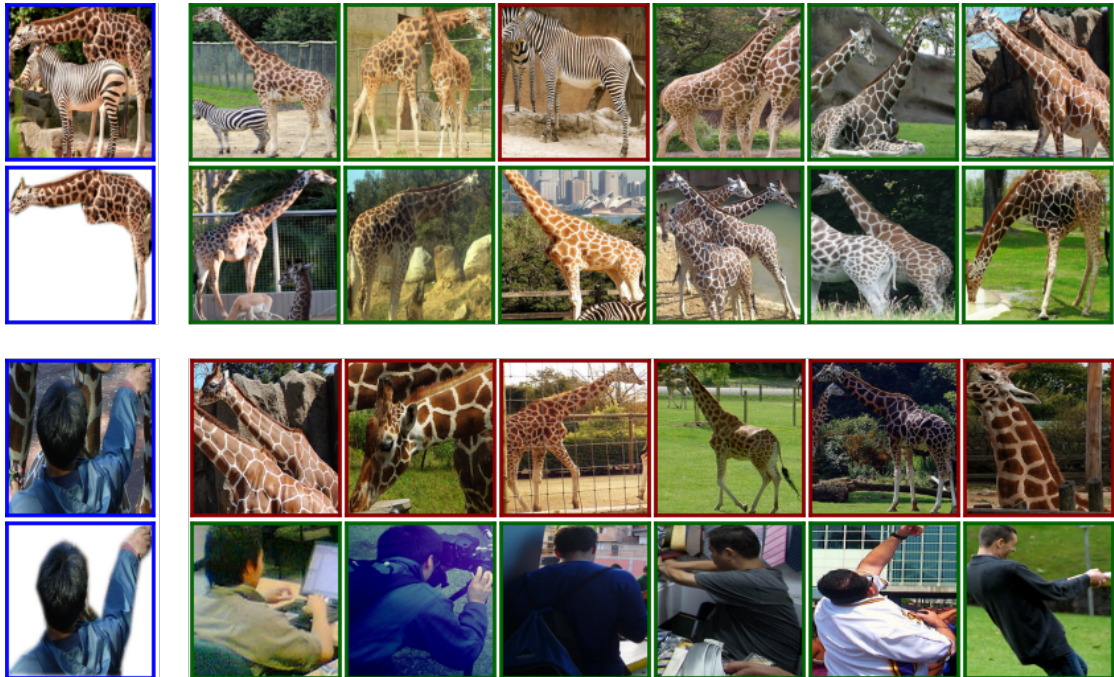


Figure 5.4: Using masked input and segmentation features can improve retrieval results. *Images with frames: query images; other images in each row: 6 retrieval results; upper rows: semantic features from the crop box (CLIP baseline). lower rows: masked input and concatenated segmentation features of our approach. Labels: giraffe, person [Lin+14].*

Dataset	mAP@3 (%)			mAP@5 (%)			mAP@10 (%)		
	256 bit	64 bit	32 bit	256 bit	64 bit	32 bit	256 bit	64 bit	32 bit
COCO 44K	67.61	61.52	39.66	67.34	61.26	39.42	65.99	60.07	39.02
COCO Stuff 25K	28.11	24.63	12.95	28.84	25.29	13.05	28.10	24.74	13.19
PASCAL VOC	85.64	80.19	56.73	85.05	79.67	56.61	83.61	78.04	55.28
Oxford-IIIT Pets	66.99	48.02	11.32	65.53	47.47	12.05	61.20	44.16	2.97
Stanford Cars	67.58	43.13	4.14	65.62	42.29	4.22	60.23	38.69	4.16
WIDER FACE	68.07	64.72	55.55	67.37	64.58	56.05	66.11	63.93	56.41

Dataset	mAP@100 (%)			mAP@1000 (%)		
	256 bit	64 bit	32 bit	256 bit	64 bit	32 bit
COCO 44K	58.87	53.51	36.17	50.36	45.96	32.39
COCO Stuff 25K	20.84	18.79	11.89	15.09	13.96	10.03
PASCAL VOC	75.94	70.05	48.06	63.24	57.69	38.63
Oxford-IIIT Pets	40.65	28.17	11.05	24.64	16.58	7.27
Stanford Cars	36.78	22.50	3.29	22.23	12.41	1.98
WIDER FACE	60.97	60.35	55.00	57.39	57.62	53.16

Table 5.5: Zero-shot retrieval performance on different datasets with different code lengths.

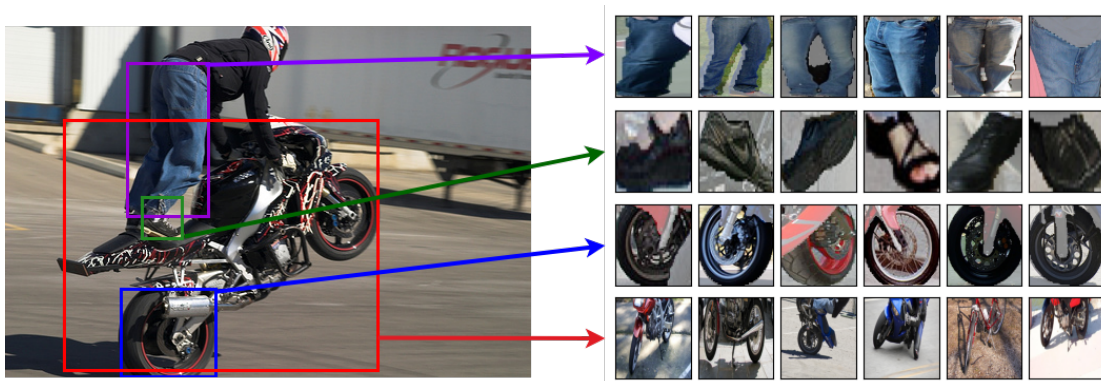


Figure 5.5: Example query image with region prompts and top 6 results performed on PASCAL VOC [Eve+10].

5.4.3 Region Retrieval

Zero-shot Region Retrieval

We follow previous work and evaluated the zero-shot retrieval performance in terms of mean average precision (mAP@k) for $k \in \{3, 5, 10, 100, 1000\}$. Depending on query time require-

ments or the size of the dataset, the length of the binary vectors is important. We therefore evaluated models for the binary code lengths 32, 64, and 256. We did not investigate shorter bit lengths, as the performance already decreased significantly at 32 bits. We used all annotated bounding boxes available for the corresponding dataset for indexing and queries. The model configuration is the one we found before in feature analysis (see Section 5.4.2), where semantic and mask features are concatenated and mask features are always masked, whereas for semantic features of medium-sized regions context within the bounding box is considered and for small regions a $3\times$ larger bounding box is used for features extraction. The results in Table 5.5 show the zero-shot retrieval results for different data sets and different code length. In general, our approach works well if the length of the binary codes is not too short, i.e., not less than 64 bits. Compared to the uncompressed features on COCO, there is still room for improvement even with the 256-bit model. The results for object-based datasets (i.e., COCO and VOC) are significantly better compared to the results for COCO Stuff that contains no objects at all. In this case, the use of masks and segmentation features does not seem to be helpful. The results are good for fine-granular object classes (i.e., Stanford Cars and Oxford IIIT Pets), if the code length is not too short, which is the case for Stanford Cars for 32-bit codes.

Region Prompts

Fig. 5.5 demonstrates that our approach can be used to perform a fine-granular region similarity search. The queries are performed on the PASCAL VOC dataset. For each image, the 25 largest regions were used for indexing regions with 256-bit codes. Object classes such as *wheel* or *jeans* are not part of the PASCAL VOC class lexicon, but can be searched for via region prompts using our approach. Only box prompts for image regions are considered in this example, but other types of region prompts work in the same way.

5.4.4 Efficient Region Retrieval

The most expensive step is the use of the image encoder, which is applied after segmentation and region selection, and in case of text prompts, additionally before region selection.

Between segmentation and retrieval of similar regions, there is a user interaction (e.g., box or point prompt), which splits the total time into two phases. In the first phase, the time for segmentation corresponds to the inference time for point prompts of 40 ms on a single NVIDIA GeForce RTX 3090 reported for FastSAM [Zha+23]. The second phase requires the application of the semantic feature extraction method, which accounts for a large part of the computation time, and the lightweight hashing part, which is negligibly small. Hashcode extraction takes about 40-50 ms on a single NVIDIA A100 80 GB.

In this section, we take a closer look at the retrieval performance and query time. Therefore, we evaluate query time and retrieval quality using the FAISS library [JDJ19]. We run the experiments on a system with an NVIDIA A100 80 GB GPU, 128 AMD EPYC 7713 CPUs, and 225 GB of memory. We use 36,781 objects from the COCO validation split as queries and 114,925 objects from the training split as our database. We evaluate the following settings:

- FAISS on 256 bit vectors (CPU)

	1	2	3	5	10	50	100	200	500	1000
FAISS float (GPU)	69.27	73.39	74.11	73.88	72.26	66.11	63.16	60.04	55.61	52.15
FAISS binary (CPU)	62.26	65.20	65.75	65.49	64.05	58.57	55.89	53.10	49.27	46.37

Table 5.6: Retrieval performance binary vs. float in FAISS (mAP).

	1	2	3	5	10	50	100	200	500	1000
FAISS float (GPU)	2.35	5.90	5.92	5.97	6.06	6.45	6.82	7.41	8.87	11.22
FAISS binary (CPU)	2.81	2.84	2.83	2.85	2.88	3.20	3.62	4.46	6.94	10.98

Table 5.7: Query time binary vs. float in FAISS (ms).

- FAISS on 256-d float vectors of output layer without thresholding (GPU)

Table 5.7 shows the results in terms of query time. For longer retrieval lists, using binary vectors is more efficient, but as shown in Table 5.6, the retrieval quality decreases due to the quantization error.

5.4.5 Qualitative Results

In this section, we give several examples where we qualitatively evaluate the effect of using masking and mask features compared to using CLIP features alone. We use the uncompressed features here, corresponding to the settings in Table 5.4 for semantic and combined features.

In the visualizations of the retrieval results shown below, the query region has a blue frame. The rows labeled as "CLIP" correspond to using semantic features without masking, as described in Section 5.4.2. The rows labeled as "Proposed" correspond to additionally using mask features and masking, respectively. The **blue** label above the query region is the label of the corresponding object in COCO. Retrieval images with a **green** frame have the same label as the query image, images with **red** frames have different labels.

Images and box annotations are taken from COCO 44K (see Section 5.4.1). Query regions are taken from the validation set, where the database images correspond to the training dataset.

Ambiguities

The retrieval results in Figures 5.6, 5.7, and 5.8 show examples of box queries, where there is more than one possible region within the box. If features are extracted from the whole box, those of the irrelevant objects and background clutter have an impact on the retrieval result list. In contrast, when using masked box queries and segmentation features, the retrieval results often look better. This is also reflected in a higher zero-shot region retrieval performance, since there are more matches between query label and retrieval label.

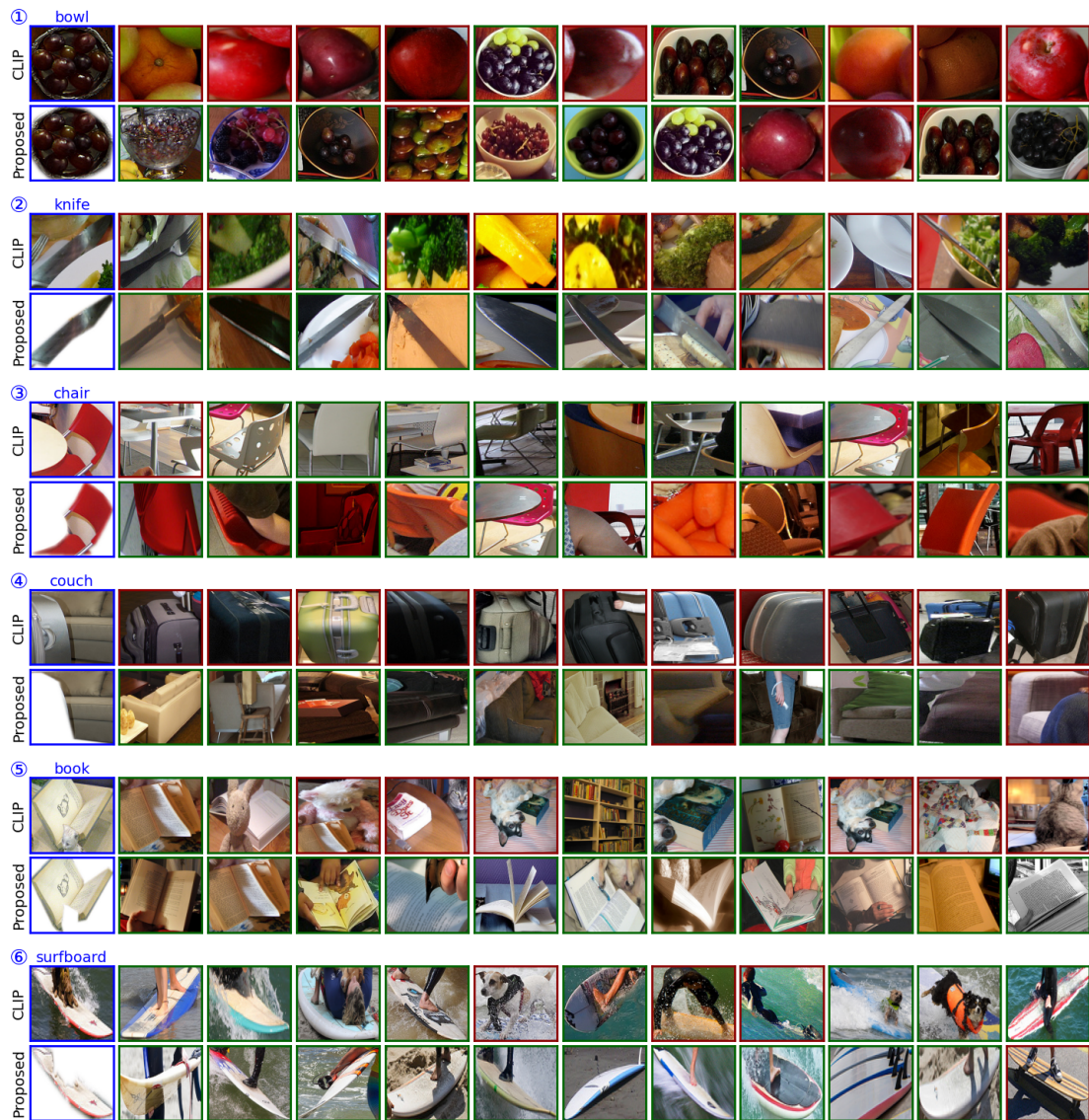


Figure 5.6: Ambiguous box prompt 1/3.

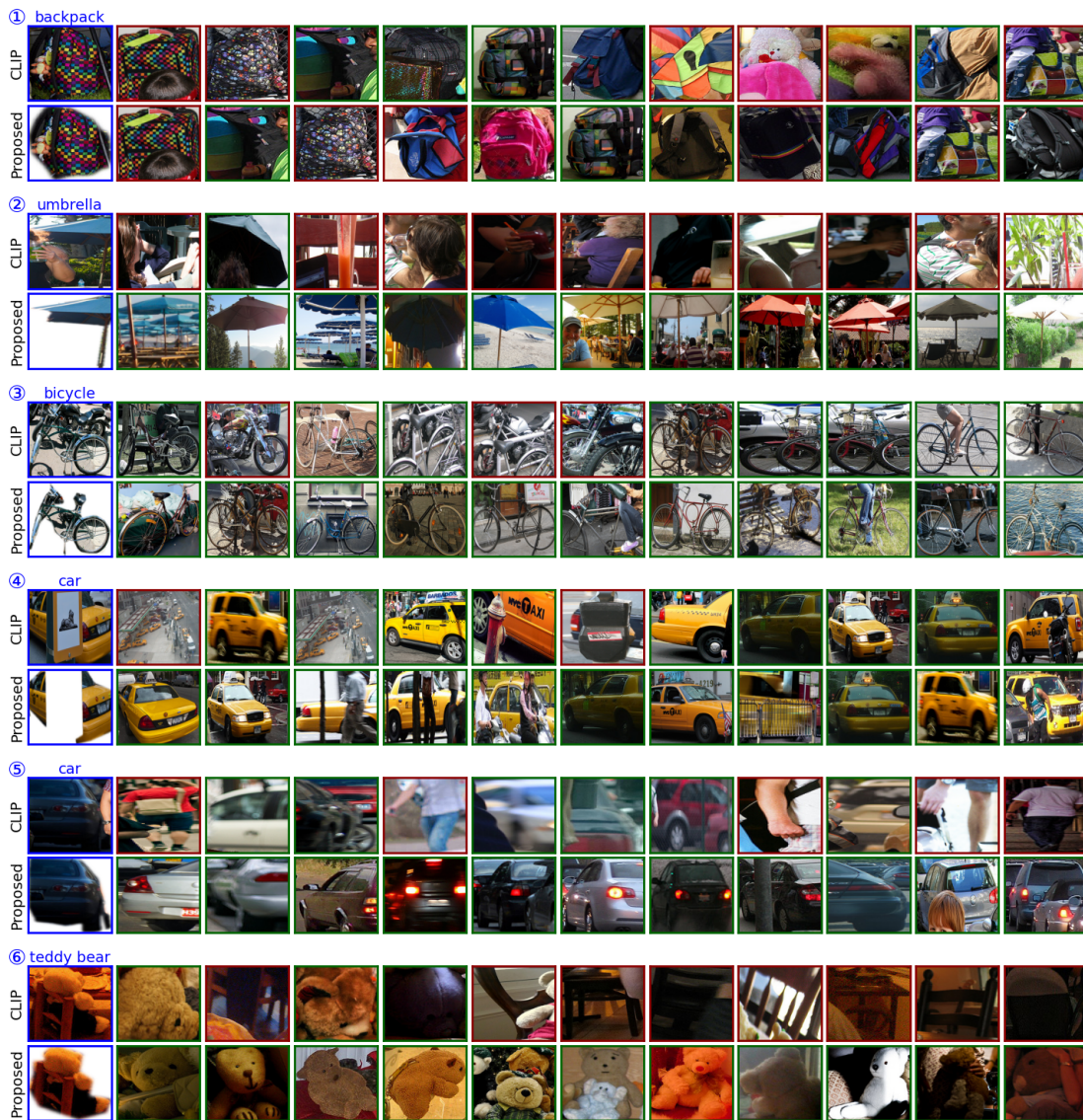


Figure 5.7: Ambiguous box prompt 2/3.

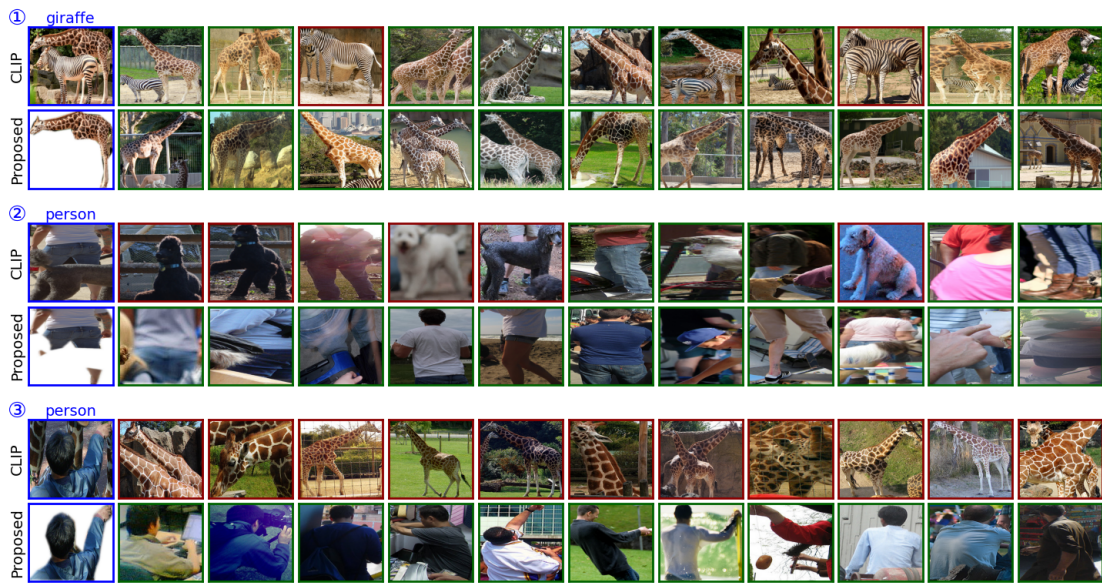


Figure 5.8: Ambiguous box prompt 3/3.

For example, in Figure 5.6 (2), there are plate, knife, food, and fork within the selected query region. Accordingly, there is food and other cutlery in the retrieval list, if we only use CLIP features here. In contrast, by using a region prompt that selects the mask of the knife, we obtain mostly knives in the retrieval list. The objects in the retrieval list have also been indexed according to their masks. Similarly, even if there are only small interfering objects as the cat in Figure 5.6 (5) or luggage in Figure 5.6 (4), this has a strong impact on the retrieval list without using masks and mask features.

Region Appearance

Other scenarios in which the use of masks and mask features can be advantageous are shown in Figures 5.9-5.11. Although no improvement in terms of correct class labels is usually visible here, the objects in the retrieval list are often closer to the image of the query region in terms of shape, color or perspective, if masks and mask features are used. For example, in Figure 5.10 (1), both retrieval lists show laptops, but if applying masks and using segmentation features, the top results are closer to the query object, i.e., an opened laptop seen from behind with a brand logo. Similarly, the returned crops of elephants shown in Figure 5.9 (1) are viewed from the same perspective as the query crop. Using masks and mask features for the blue chair in Figure 5.10 (7) returns two blue chairs as top results in the retrieval list, which are missing when using CLIP features alone.

Context Boxes

Table 5.8 shows more evaluated context box sizes for small and medium sized objects of COCO 44K. Crop boxes around objects are evaluated for factors of steps of 0.5 for semantic

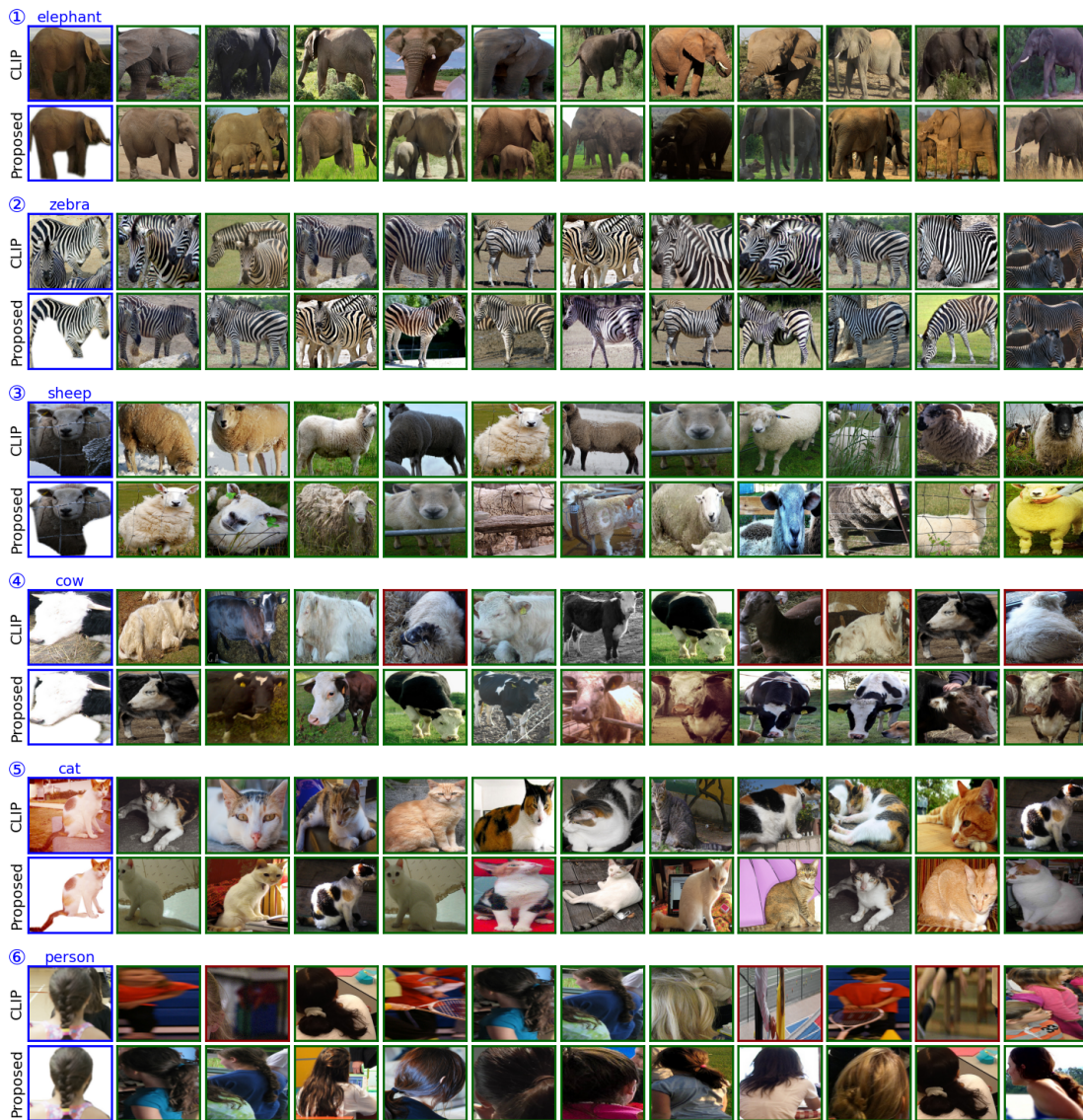


Figure 5.9: Region appearance 1/3.

5 Segmentation-based Image Similarity Search via Region Prompts



Figure 5.10: Region appearance 2/3.

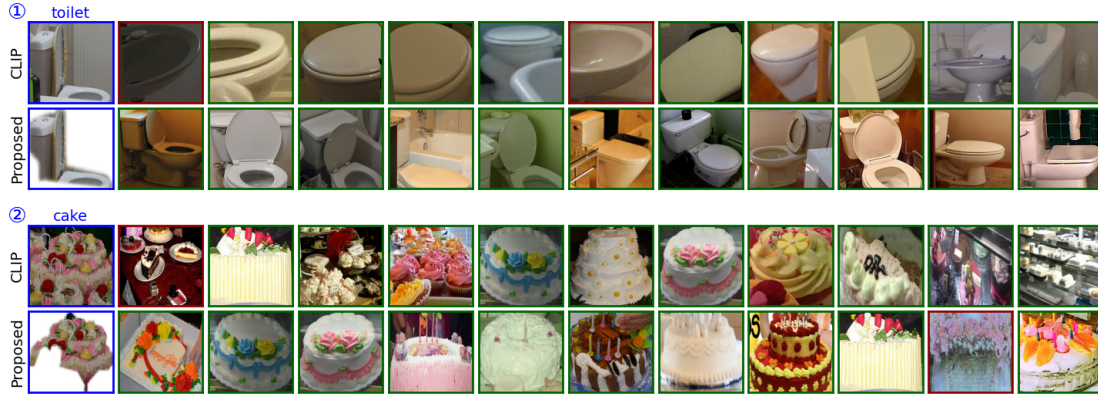


Figure 5.11: Region appearance 3/3.

		mAP@k (%)						
Size		10	25	50	100	250	500	1000
medium	mask	74.56	71.74	69.34	66.89	63.44	60.53	57.38
	1.0	79.76	77.11	74.92	72.60	69.09	65.89	62.21
	1.5	79.35	76.43	74.06	71.53	67.79	64.44	60.57
	2.0	75.37	71.92	69.19	66.42	62.47	59.10	55.38
	2.5	71.55	67.96	65.12	62.26	58.27	55.01	51.47
	3.0	68.63	64.90	61.98	59.12	55.20	52.02	48.61
	3.5	65.91	62.02	59.12	56.20	52.31	49.22	45.91
small	masked	56.10	52.72	49.85	47.04	43.63	41.19	38.79
	1.0	60.14	56.59	53.64	50.75	47.10	44.42	41.76
	1.5	66.84	63.48	60.63	57.80	53.93	50.91	47.75
	2.0	70.05	66.67	63.83	60.97	57.00	53.82	50.43
	2.5	71.59	68.17	65.38	62.58	58.64	55.44	52.00
	3.0	72.13	68.66	65.92	63.08	59.18	55.96	52.49
	3.5	71.91	68.44	65.64	62.83	58.97	55.78	52.33

Table 5.8: Context for semantic features.

features from CLIP. For medium sized objects, there is only an improvement if a context around the object is used. Increasing the box size results in decreasing mAP. For small objects, the performance increases for boxes of a size up to $3 \times$ the object’s bounding box. Larger boxes decrease the performance.

5.5 Summary

We presented *Search Anything*, a novel approach to perform similarity search in images, based on region queries using intuitive prompts. *Search Anything* is trained in a self-supervised manner on mask features extracted by the FastSAM foundation model and semantic features for masked image regions extracted by the CLIP foundation model to learn binary hash code representations for image regions. By coupling these two foundation models, we can index and search images at a more fine-grained level than with full image similarity search.

In our experimental evaluations based on several datasets from different domains in a zero-shot setting, we have shown that combining region masks, segmentation features, and semantic features improves region retrieval performance. If objects or regions are sufficiently large, an improvement in retrieval quality can be achieved if they are masked and thus only features from the image region are considered. For small objects, context is important, and useful features can be extracted from the context.

6

Image Similarity Search in Applications

In this chapter, we present image similarity search in different applications and use cases. Recently, systems that use image similarity searches have opened up new ways of exploring large image or video datasets. For example, iArt [Spr+21] is a system that allows to visually search a large database of art-historical images. It integrates several deep learning models for content-based image retrieval and offers keyword-based image retrieval. The integrated models are trained on different datasets and the system enables a facet search. In contrast, the systems presented in this chapter use a single model based on training data for classification, and the focus is on query speed by using deep hashing methods and efficient index structures.

We first present two systems in which image similarity search is used to explore large datasets from different domains. In the first system, image similarity search helps to make a large corpus of video recordings from GDR television accessible. The first system is used for content-based image retrieval in media and TV production. Finally, we present a system in which image similarity search is used to efficiently acquire qualitative training data within this corpus. We will first highlight the use cases and proposed similarity search methods in the respective systems, and then present the overall systems in Section 6.1, 6.2, and 6.3.

Image Similarity Search in Archives

In this system, image similarity search is part of a tool set for accessing a large corpus of digitized videos of the GDR television. Together with video OCR, face recognition and concept detection, image similarity allows to find similar videos in this large corpus of 3,000 hours of historical GDR television recordings.

The image similarity search part within the system analyzes keyframes from video shots (the first, last, and three intermediate frames). These keyframes are processed through a CNN, converting them into binary codes stored in the database. We use a two-stage approach based on deep hashing. First, a coarse search with 64-bit binary codes narrows down potential results using Hamming distance and a vantage point tree for efficient nearest neighbor search. The short list from this stage is then refined using 256-bit binary codes, ranking images based on their Hamming distance to the query image. By using hash codes in the second stage as well, this approach eliminates the need for time-consuming distance computations on high-dimensional float features during testing, and comes at no additional cost, since the CNN is trained on both short and long codes, thus sharing CNN parameters for both coarse and refined search.

Image Similarity Search in TV and Film Production

In the presented system, image similarity search is used to complement textual content-based queries, providing more flexibility by allowing arbitrary images to be used to query the video database for similar content.

The proposed method for image similarity search uses binary hash codes for more efficient searching in a two-stage approach. It integrates two coding layers (64-bit and 256-bit) into the same architecture, allowing for concurrent computation at inference time. Keyframes from the video collection are processed, and the resulting binary hash codes are stored in the database. For comparing hash codes, the Hamming distance is used, along with a vantage point tree as an additional index structure to accelerate the search. The method for image similarity search extends the method proposed in Section 6.2. Additionally, the CNN for similarity search and for concept detection are merged within one architecture for sharing weights and thus sharing the computation of binary hash codes for similarity search and the computation of class probabilities for image classification. When indexing or adding new videos, this halves the computation time for a keyframe.

Image Similarity Search for Data Acquisition

Within the presented system, image similarity search is part of the deep learning cycle of iteratively improving a model for concept detection. Initially, image similarity search assists in the collection of training samples by identifying images that are visually similar to query images from existing concepts, thereby enriching the dataset and improving the deep learning model. The idea is, as the model is improved and new concepts are introduced, the image similarity search system itself is updated as well and refined to recognize new concepts, improving its ability to find more diverse and relevant training samples in subsequent iterations. In the context of this system, image similarity search offers a way to extract training data very quickly from the corpus itself. Additionally, the proposed method for image similarity search drastically reduces query time by integrating a two-stage approach based on multi-index hashing into Elasticsearch (see Section 4.2). The approach for image similarity search generates 256-bit codes for which a two-stage method is proposed. It first uses 64-bit codes for a coarse search and then 256-bit codes for re-ranking. The 64-bit codes are derived from the 256-bit codes using the Kernighan-Lin algorithm and multi-index hashing.

6.1 Content-Based Video Retrieval in Historical Collections of the German Broadcasting Archive

6.1.1 Introduction

Digital video libraries become more and more important due to new potentials in accessing, searching and browsing the data [Chr+95; MG02; AJ15]. In particular, content-based analysis and retrieval in large collections of scientific videos is an interesting field of research. Examples are Yovisto¹, ScienceCinema² and the TIB|AV portal³ of the German National Library of Science and Technology (TIB). The latter provides access to scientific videos based on speech recognition, visual concept classification and video OCR (optical character recognition) [SP14; HBS13]. The videos of this portal stem from the fields of architecture, chemistry, computer science, mathematics, physics, and technology/engineering.

The German Broadcasting Archive (DRA) in Potsdam-Babelsberg provides access to another valuable collection of scientifically relevant videos. It encompasses significant parts of the audio-visual tradition in Germany and reflects the development of German broadcasting before 1945 as well as radio and television of the former German Democratic Republic (GDR). The DRA was founded in 1952 as a charitable foundation and joint institution of the Association of Public Broadcasting Corporations in the Federal Republic of Germany (ARD). In 1994, the former GDR's radio and broadcasting archive was established. The archive contains film documents of former GDR television productions from the first broadcast in 1952 until its cessation in 1991, including a total of around 100,000 broadcasts, such as contributions and recordings of the daily news program *Aktuelle Kamera*, political magazines such as *Prisma* or *Der schwarze Kanal*, broadcaster's own TV productions including numerous films, film adaptations and TV series productions such as *Polizeiruf 110*, entertainment programs (e.g., *Ein Kessel Buntes*), children's and youth programs (fairy tales, *Elf 99*) as well as advice and sports programs. Access to the archive is granted to scientific, educational and cultural institutions, to public service broadcasting companies and, to a limited extent, to commercial organizations and private persons. The video footage is often used in film and multimedia productions. Furthermore, there is a considerable international research interest in GDR and German-German history. Due to the uniqueness and importance of the video collection, the DRA is the starting point for many scientific studies.

International scientists, particularly from the USA and UK, followed by the Netherlands, Japan, Sweden and Switzerland, use the DRA for their research in the fields of psychology, media, social, political or cultural science. These studies are, for example: *Heavies in East Germany* (Humboldt University Berlin), *Space Travel in the GDR* (Harvard University, USA), *The Jewish in TV* (Ludwig Maximilian University of Munich), *Socialism on the Screen* (Loughborough University, UK), *Self-made in Consumer Society* (University of Mannheim), and *Child and Youth Education in Fictional Subjects* (Shizuoka University, Japan). The DRA is answering a wide range of research requests concerning the life of GDR citizens and social perceptions. The number of comprehensive and time-consuming requests is considerably increasing, e.g., *youth fashion in*

¹<http://www.yovisto.com>

²<http://www.osti.gov/sciencecinema>

³<http://av.tib.eu>

the GDR, especially for punks and bluesers, living in East Germany, in particular home furnishings from Deutsche Werkstätten Hellerau or the socialist city as a model of urban development in the GDR, specifically pictures including socialist classicism, buildings made with precast concrete slabs, demolition and spectacular buildings.

Due to the time-consuming task of labeling videos manually, human annotations focus on larger video sequences and contexts. Furthermore, finding similar images in large multimedia archives is manually infeasible. Thus, the DRA aims to digitize and index the entire video collection to facilitate search in videos.

In this section, an automatic video analysis and retrieval system for searching in historical collections of GDR television recordings is presented. It consists of novel algorithms for visual concept classification, similarity search, person and text recognition to complement human annotations and to support users in finding relevant video shots. In contrast to manual annotations, content-based video analysis algorithms provide a more fine-grained analysis, typically based on video shots. A GDR specific lexicon of 91 concepts including, for example, *applause*, *optical industry*, *Trabant*, *military parade*, *GDR emblem* or *community policeman*, is used for automatic annotation. An extension of deep convolutional neural networks (CNN) for multi-label concept classification, a comparison of a Bag-of-Visual Words (BoVW) approach with CNNs in the field of concept classification, and a novel, fast similarity search approach are presented. The results of automatically annotating 2,500 hours of GDR television recordings are evaluated from a technical and an archival perspective.

Parts of this section have been published in: Markus Mühling, Manja Meister, Nikolaus Korfhage, Jörg Wehling, Angelika Hörth, Ralph Ewerth, and Bernd Freisleben. “Content-based Video Retrieval in Historical Collections of the German Broadcasting Archive.” in: *International Journal on Digital Libraries* 20 (2019), pp. 167–183.

6.1.2 A Content-Based Video Retrieval System

In this section, a content-based video retrieval system to support search in historical GDR television recordings is presented. Its aim is to automatically assign semantic tags to video shots for the purpose of facilitating content-based search and navigation.

Figure 6.1 shows an overview of the developed video retrieval system. First, the videos are digitized and preprocessed. The preprocessing step mainly consists of shot boundary detection with additional tasks, such as video transcoding or thumb generation, which are required for later visualization purposes. Based on video segmentation, the following automatic content-based video analysis algorithms are applied: concept classification, similarity search, person and text recognition. The resulting metadata are written to a database. Given this semantic index, arbitrary search queries can be processed efficiently. The query results are returned to the user as a list of video shots, ranked according to the probability of the presence of the desired content. Due to the large amount of video data and the associated computational requirements, distributed content-based video analysis is performed in a service-oriented architecture.

In the following, the digitization process, the analysis algorithms, the service-oriented architecture and the workflows are described.

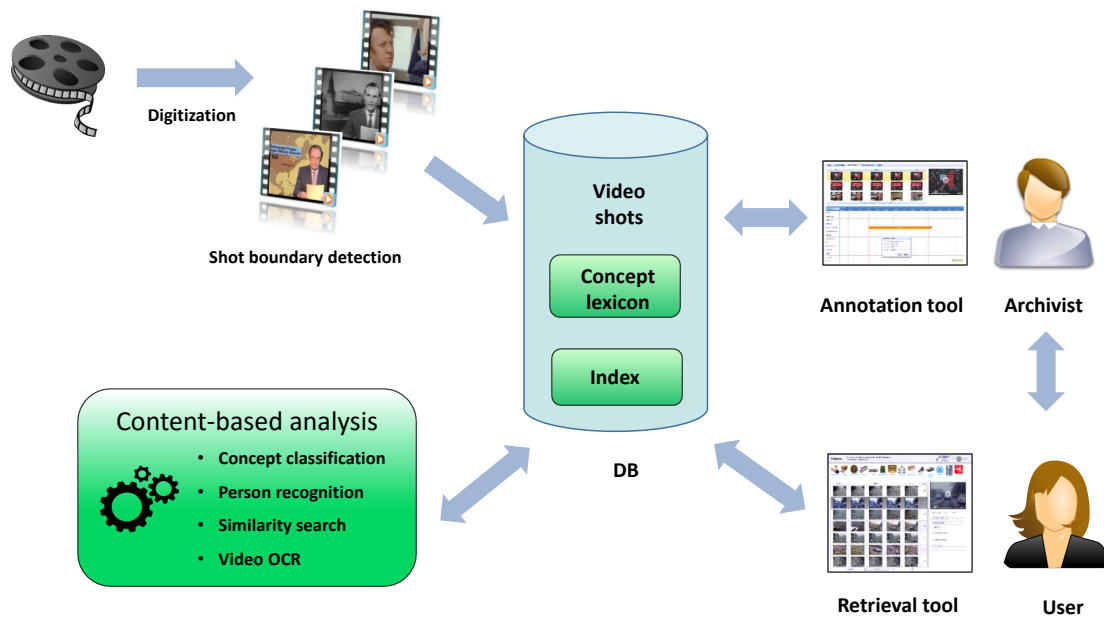


Figure 6.1: Video retrieval system.

Video Digitization

Overall, 3,100 hours of the analog, historical GDR television recordings have been digitized. The digitized material has been selected from the large amount of available analog video material according to its relevance for research. In particular, it consists of socio-political magazines, the daily news program *Aktuelle Kamera* and several TV productions. The collection provides a wide range of themes and reflects everyday life in the former GDR.

The digitization process has been carried out both by the DRA itself and by an external provider. In a manual preprocessing step, the tapes have been technically prepared and the content has been reconciled using the FESAD database of the DRA. FESAD (“Fernseharchivdatenbank”) is a TV database jointly used in the ARD, which is a consortium of public-law broadcasting institutions of Germany. For the external digitization, technical parameters such as the video format (e.g., Betacam SP), the duration, and the time codes have been reviewed. About 100 hours of video data have been digitized by the DRA. Depending on the video format, different digitization devices have been used, such as an AVID workstation, DVS Fuze 5.10, DVS Venice 3.2 or Digital Vision Phoenix 2015. Finally, the digitized videos have been corrected with respect to their colors.

However, most of the video tapes have been digitized by the worldwide unique automated digital archiving system ADAM (Automated Digital Archive Migration). ADAM was developed by the Swiss JORDI AG⁴ in collaboration with the WDR media group⁵ according to archival requirements. The core of the system is an industrial robot that transfers the content of the video tapes automatically into a digital file-based archive. It grabs video tapes from a carousel

⁴<http://www.jordicom.ch/tv-media/>

⁵<http://wdr-mediagroup.com>

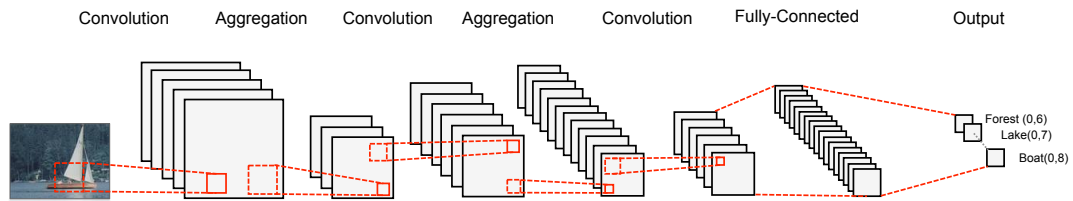


Figure 6.2: Deep convolutional neural network.

and places them in a pool of up to 740 parking slots. The tapes are passed to a video tape recorder (Sony MSW 2100 EP), and dirty or defective tapes are automatically cleaned.

Content-based Video Analysis

The aim of the content-based video analysis algorithms is to automatically assign semantic tags to videos for the purpose of facilitating content-based search and exploration. The fundamental problem is to overcome the discrepancy between the extracted features and the human interpretation of the (audio-)visual data. In the literature, this discrepancy is also known as “semantic gap”. Smeulders et al. [Sme+00b] describe the semantic gap as “the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation”.

Typically, automatically generated labels are assigned to video shots. Therefore, shot boundary detection has to be performed. The aim of shot boundary detection is the temporal segmentation of a video sequence into its fundamental units, the shots. A shot is generally understood as a video sequence recorded continuously without any interruption. The transitions between shots can be abrupt (cuts) or gradual (fade in/out, dissolves, wipes). Shots are detected using our shot boundary detection algorithms [EF04; EF09], some of which belonged to the top approaches at the TRECVID challenge 2007⁶. To detect gradual transitions more reliably, camera motion estimation (e.g., see [Ewe+07]) is leveraged for false alarm removal.

Based on the results of the temporal video segmentation, concept classification, person and text recognition are applied for video analysis to automatically extract high-level content-based metadata. Furthermore, an index is generated for fast semantic similarity search in large video databases.

In the following sections, the content-based video analysis algorithms as well as the similarity search approach are described in more detail.

Visual Concept Classification

The classification of visual concepts is a challenging task due to the large complexity and variability of their appearance. Visual concepts can be, for example, objects, sites, scenes, personalities, events or activities. The definition of our GDR specific concept lexicon is based on

⁶<http://trecvid.nist.gov>

the analysis of user search queries with a focus on queries that were experienced as difficult and time-consuming to answer manually. Considering the utility or usefulness for search queries, the observability by humans and the feasibility in the sense of automatic detection, a lexicon of 91 concepts was defined after analyzing more than 36,000 user queries received within a five-year period from 2008 to 2013. Therefore, user queries that are assumed to be of future research interest were summarized thematically and ordered by frequency. The concept lexicon comprises events such as *border control* and *concert*, scenes such as *railroad station* and *optical industry*, objects like *Trabant* or activities such as *applauding*. To build the concept models, training data has to be annotated manually. For this purpose, a client-server based annotation tool was built to facilitate the process of training data acquisition and to select a sufficient quantity of representative training examples for each concept. We have defined a minimum number of 100 positive samples per concept.

Recently, deep learning algorithms fostered a renaissance of artificial neural networks, enabled by the massive parallel processing power of modern graphics cards. Deep learning approaches, especially deep CNNs, facilitated breakthroughs in many computer vision fields [KSH12; GMH13; Bre+13; Tai+14; SKP15b]. Instead of using hand-crafted features such as SIFT descriptors [Low99], CNNs learn the features automatically during the training process. A CNN consists of several alternating convolution and aggregation (i.e., max-pooling) layers with increasingly complex feature representations and typically has several fully connected final layers, as shown in Figure 6.2.

Most state-of-the-art network architectures for image recognition [KSH12; Sze+15; He+16] as well as the current datasets [Den+09; Zho+14] consider only a single concept per image (“single-label”). In contrast, real world concept classification scenarios are multi-label problems. Several concepts, such as *summer*, *playground* and *teenager*, may occur simultaneously in an image or scene. While some approaches use special ranking loss layers [Gon+13], we have extended the CNN architecture of the GoogleNet [Sze+15] using a sigmoid layer instead of the softmax layer in combination with a cross entropy loss function.

Since the training of a deep CNN model from scratch requires millions of training images and due to the relatively small amount of available training data, we have adapted a pre-trained CNN classification model (GoogleNet [Sze+15] trained on ImageNet [Den+09]) to the new GDR concept lexicon using our multi-label CNN extension and performed a fine-tuning on the GDR television recordings. The models were trained and fine-tuned using the deep learning framework Caffe [Jia+14].

Similarity Search

Since the DRA offers researchers a large number of video recordings containing several millions of video shots, the need for a system that helps to rapidly find desired video shots emerges. While scanning through the whole video archive is practically infeasible for humans, a possible solution is to index the videos via concepts as described in Section 6.1.2. However, this approach requires manually annotated training images for learning the concept models. Additionally, search queries are restricted to the vocabulary of predefined concepts and new concept models have to be developed on demand. In contrast to textual concept-based queries, image-based queries provide users more flexibility and a new way of searching.

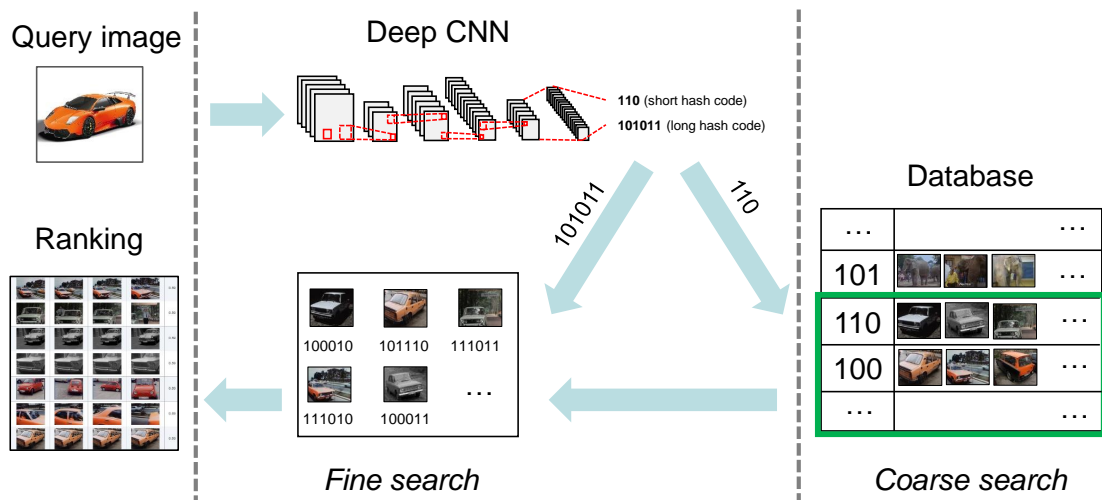


Figure 6.3: Content-based similarity search.

While query-by-content based on low-level features turned out to be insufficient to search successfully in large-scale multimedia databases, image representations learned by deep neural networks greatly improved the performance of content-based image retrieval systems [Wan+14]. They are less dependent on pixel intensities and are clearly better suited for searching semantic content. However, high-dimensional CNN features are not well suited for searching efficiently in large video collections. Fast search in large databases is an essential requirement for practical use. For this purpose, proposals for learning binary image codes for compact representations and fast matching of images have been made. Krizhevsky and Hinton [KH11], for example, used deep autoencoders and Lin et al. [Lin+15] extended a CNN to learn binary hash codes for fast image retrieval.

In this section, an approach for fast content-based similarity search in large video databases is presented. To efficiently store and match images, the approach is based on learning binary codes by deep CNNs. This mapping of images to binary codes is often referred to as “semantic hashing” [SH09]. The idea is to learn a “semantic hash function” that maps similar images to similar binary codes. To learn the hash function, a method similar to the approach described by Lin et al. [Lin+15] is used. Based on a pre-trained CNN classification model, we have devised a coding layer and an appropriate loss layer for error propagation for the network architecture. The best results were obtained fine-tuning for one epoch on Chatfield et al.’s VGG-16 network [Cha+14] trained on the PLACES dataset [Zho+14]. The learning rate is decreased following a polynomial decay with a power of 4, resulting in a fast decrease of the learning rate early in the training process. As optimization method, we used SGD with a momentum of 0.9. An advantage of using pre-trained classification models is the speed-up in training time. To obtain high-level features, we have built the coding layer on top of the last fully-connected layer. Furthermore, the hash function can be adapted to unlabeled datasets by using the predictions of the pre-trained classification model for error propagation.

The overall system for similarity search is based on the analysis of keyframes, i.e., representative images. In our approach, five frames (the first, the last and three in between) per video shot are used as keyframes for indexing. Given the hash function, the keyframes of the video collection

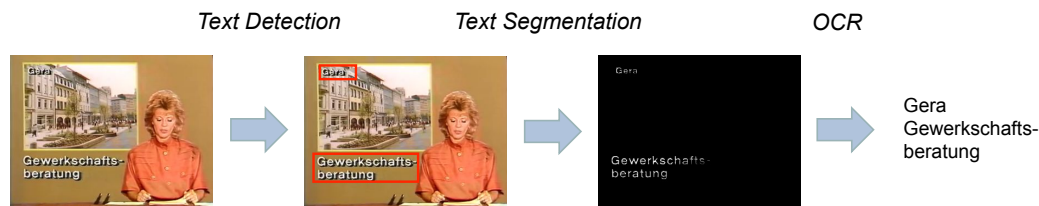


Figure 6.4: Text recognition pipeline.

are fed into the deep CNN and the mapped binary codes are stored in the database. Based on the resulting index, queries-by-image can be answered by matching the binary code of the given image to the database. The overall retrieval process is shown in Figure 6.3. Given the query image, the binary codes are extracted using the learned deep CNN. We use a two-stage approach based on semantic hashing. First, a coarse search is performed using 64-bit binary codes, resulting in a comparatively short list of potential results. The Hamming distance is applied to compare the binary codes. A vantage point tree [Yia93] is used as an additional index structure that recursively partitions the binary codes in the Hamming space into close and distant points by choosing so called “vantage points” to significantly accelerate the nearest neighbor search. The resulting short list consists of 10,000 nearest neighbors.

The longer the binary codes are, the more accurate the image representations are. Therefore, in the second stage, a refined search using 256-bit binary codes is performed on the short list. The images are ranked according to the Hamming distance to the query image.

Differently from Lin et al. [Lin+15], hash codes are used for the refined search as well. In our two-stage approach, both coding layers, for 64-bit and 256-bit binary codes, are integrated into the same architecture and trained concurrently. Hence, sharing the CNN parameters for coarse and refined search completely eliminates the time for a fine search formerly required for distance computations on high dimensional float features at test time. Finally, the resulting images are mapped to video shots.

Person Recognition

Based on the analysis of user search queries, our GDR specific concept lexicon has been extended by 9 personalities, including *Erich Honecker*, *Walter Ulbricht*, *Hilde Benjamin*, *Sieg-mund Jähn*, *Hermann Henselmann*, *Christa Wolf*, *Werner Tübke*, *Stephan Hermlin*, and *Fritz Cremer*. Instead of using concept classification, persons are recognized using a face recognition approach [EMF07]. For this purpose, feature representations of known persons are stored in a face database. A face recognition system has been built that scans the video shots and recognizes the identity of a detected face image by comparing it to the face database. Finally, the resulting index of person occurrences can be used in search queries.

The face processing pipeline consists of several components: face detection, face alignment, and face recognition. For the face recognition component, we evaluated the following approaches: Fisherfaces [BK97], Local Binary Pattern Histograms [AHP04] and a commercial library, called

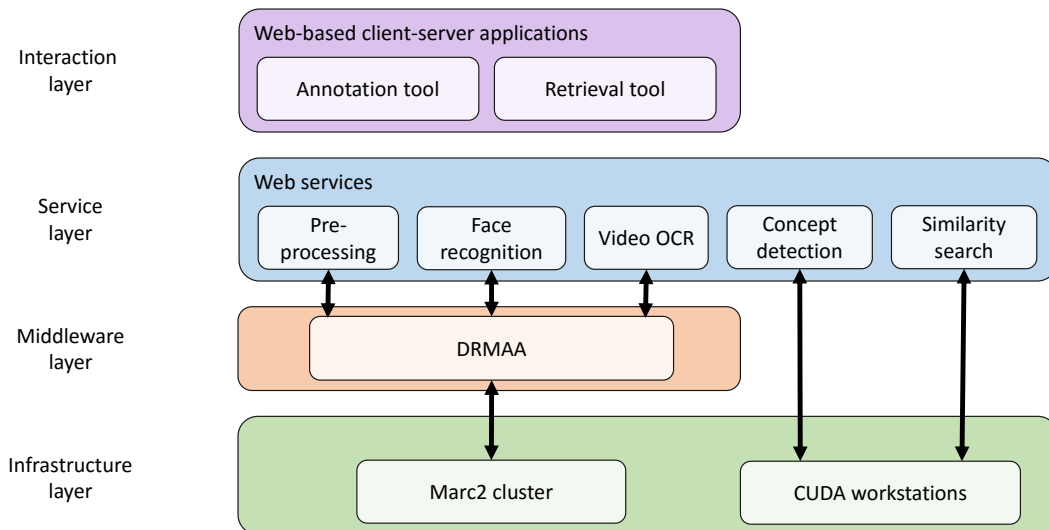


Figure 6.5: Service-oriented architecture for CBVR.

FaceVACs⁷. Furthermore, we evaluated whether a preprocessing step using grayscale histogram equalization, training data augmentation using Google search queries, or a face tracking component improve the recognition accuracy. Based on the results of our evaluations, we finally used the method of Viola and Jones [VJ01] for face detection, and FaceVACs for face alignment and recognition.

Text Recognition (Video OCR)

Superimposed text often hints at the content of a video shot. In news videos, for example, the text is closely related to the current report. In silent movies, it is used to complement the screen action with cross headings. The involved algorithms can be distinguished by their objectives, whether it is text detection, also called text localization, text segmentation, or optical character recognition [GE04] (see Figure 6.4).

We have developed a text recognition system that allows users to search for in-scene and overlaid text within the video archive. For this purpose, the I-frames of the videos are analyzed, and the recognized ASCII text is stored in the database. Based on the resulting index, OCR search queries can be answered by a list of video shots ranked according to the similarity to the query term. Due to low technical video quality and low contrast of text appearances, the similarities between the query term and the words in the database are calculated using the Levenshtein distance [Lev66].

For text detection, localization and segmentation in video frames, a method based on Maximally Stable Extremal Regions (MSER) has been employed [Mat+04; NM12] using the Open Computer Vision (OpenCV) Library⁸. It can detect both overlaid text, as well as text within the scene, for example on banners. Experimental results have revealed that the text segmentation component

⁷<http://www.cognitec.com>

⁸<http://opencv.org>

plays an important role in the case of videos of low technical quality. Text segmentation crops the detected and as characters classified extremal regions out of the image to yield black letters on a white background. This step is necessary to feed the result into an OCR algorithm that transforms the image into machine-readable text. A non-uniform background would normally impair this process. For OCR, we evaluated two open source libraries: a Long Short-Term Memory network (LSTM) approach⁹ [Bre+13] and Tesseract¹⁰. Tesseract outperformed the LSTM-based approach on the GDR television recordings.

Service-Oriented Architecture

A service-oriented architecture is used to deal with the requirements of the content-based video retrieval system. Due to the large amount of video data and the computationally expensive video analysis algorithms a distributed heterogeneous architecture is employed to provide scalability. An overview of the architecture is given in Figure 6.5. User interaction is handled via web-based client-server applications providing interfaces to use the following web services: preprocessing, person recognition, video OCR, concept classification and similarity search. The GUIs of the annotation and retrieval tool are described in more detail in Sections 6.1.2 and 6.1.3.

The web services are executed on different hardware architectures. Here, we have to distinguish between CPU and GPU algorithms. While face recognition, video OCR and the preprocessing steps including shot boundary detection are CPU intensive, concept classification and similarity search mainly use GPU resources.

For the CPU intensive algorithms, the MARC2 computing cluster at the University of Marburg, Germany, has been used. The MARC2 cluster has 96 compute nodes, each consisting of 4 AMD Opteron 6276 or 6376 with 16 cores@2.3 GHz each, i.e., 6144 cores. In addition, there are two head nodes with the same specification. In total, MARC2 has 24 TB RAM and 192 TB of disk storage space. The operating system is Red Hat Enterprise Linux for the headnodes and CentOS for the compute nodes. The Sun Grid Engine 6.2u5 (SGE) is used as the job scheduler. For this purpose, specific interfaces have been defined. The CPU intensive algorithms have been ported to the MARC2 infrastructure and have been encapsulated in separate jobs. Web services for preprocessing, face recognition, and video OCR are provided. The preprocessing web service starts several jobs for transcoding the videos, for shot boundary detection and for extracting images and thumbs. The web services submit jobs to the SGE, control job execution and provide status information. For management purposes, the distributed resource management application API (DRMAA) is used. Instead of transferring user data directly, only references are sent via parameters during the service call. The images and videos are stored on a data server and the actual data transport is handled via network file system shares. The advantage is an overlap of data transfer and service execution, which contributes to the improvement of the overall runtime performance.



The web services for concept classification and similarity search use GPU resources and are installed on dedicated servers with Nvidia Geforce GTX 770 graphics cards with 4GB of memory.















⁹<https://github.com/tmbdev/ocropy>

¹⁰<http://code.google.com/p/tesseract-ocr/>

6 Image Similarity Search in Applications

Videana Content-based search in historical collections of GDR television recordings in the German Broadcasting Archive

Start	Middle	End	score
			1.00
			1.00
			1.00
			1.00
			1.00
			1.00

Years: All From to

Concept detection

Demonstration

Search

Person recognition

Select person

Search

Similarity search

Bild auswählen

Low-level High-level (100%)

Search

OCR-Search

2 of 6 1 2 3 4 5 6 100

Figure 6.6: GUI of the retrieval tool showing the results for the concept *demonstration*.

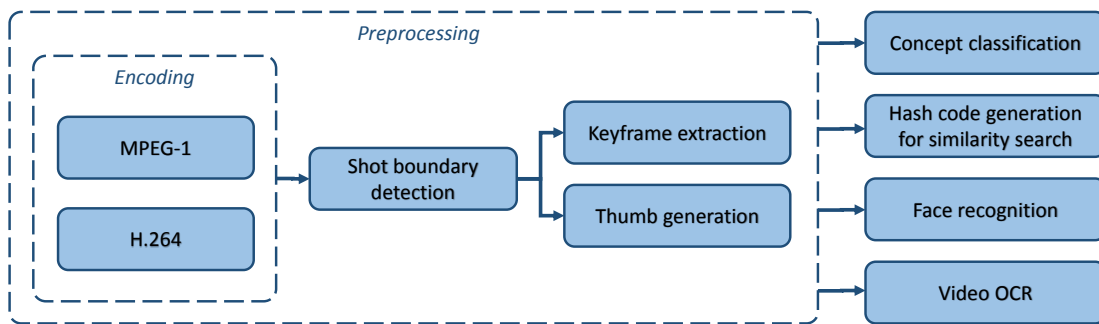


Figure 6.7: Web service dependencies and workflows.

Two similarity search related web services exist. The first one is responsible for hash code generation from videos, for uploading the extracted codes to the database and for updating the index structures. The second one provides similarity search, taking a query image as input and returning a sorted list of the most similar video shots.

While the execution of models for concept classification and similarity search is really fast on GPUs (only a few milliseconds), training of such deep CNNs is computationally expensive even on graphics cards. Considering the hardware requirements concerning processing power, GPU memory, main memory and hard disk capacities, we have built a highly optimized system for deep learning similar to the Nvidia DevBox¹¹ to train deep neural network models. The system consists of four GeForce GTX Titan X GPUs with 12GB RAM and 3072 CUDA cores at 1000/1075 MHz, an Intel Core i7-5930K CPU with six cores at 3.50GHz, 64 GB of DDR-4 RAM, 8 TB of disk space for large datasets and a 250 GB SSD for fast I/O operations.

Workflows

The client-server-based web applications, the annotation and the retrieval tool provide simple interfaces to perform content-based video analysis and retrieval in a distributed, heterogeneous environment.

The retrieval tool provides image and scene search in the metadata enriched video collection. A web-based GUI has been developed to automatically respond to user queries related to concepts, persons, similar images, or text. The retrieval results are presented to the user in the form of a ranked list of video shots (see Figure 6.6) where each video shot is represented by five key frames and a probability score indicating the relevance of the shot. Furthermore, a video player allows to visually inspect the video shots.

Administrative tasks, video uploads, manual annotations and orchestration of the content-based video analysis jobs are handled via the annotation tool. It provides an upload page where videos can be selected and uploaded to the system. The video upload automatically triggers the preprocessing jobs for shot boundary detection and visualization purposes by considering algorithm and data parallelism to run as many processes as possible concurrently. The preprocessing workflow consists of the following jobs: video transcoding, shot boundary

¹¹<https://developer.nvidia.com/devbox>

detection, keyframe extraction and thumbnail generation (see Figure 6.7). While MPEG-1 videos are necessary as a standardized input for the shot boundary detection algorithm, the MP4 videos are compressed using the H.264 codec to generate small videos suitable for web applications. The keyframes are extracted for the subsequent content-based analysis jobs, and the thumbnails are generated for the visualization of shots.

After the videos have been uploaded and preprocessed, they can be manually annotated for training data acquisition based on the predefined lexicon of visual concepts. The application provides GUIs for shot-based and interval-based labeling. The GUI for annotating intervals mainly consists of a video player and a video slider where concept occurrences can be inserted, deleted and edited. The interval-based labeling is more accurate than the shot-based labeling, but also more time-consuming.

As soon as the models for concept classification and person recognition have been built and installed, the job management page of the annotation tool can be used to start content-based analysis jobs to generate meta-data for concepts, persons and text as well as hash codes. The dependencies of the different web services and jobs are visualized in Figure 6.7.

6.1.3 Experimental Results

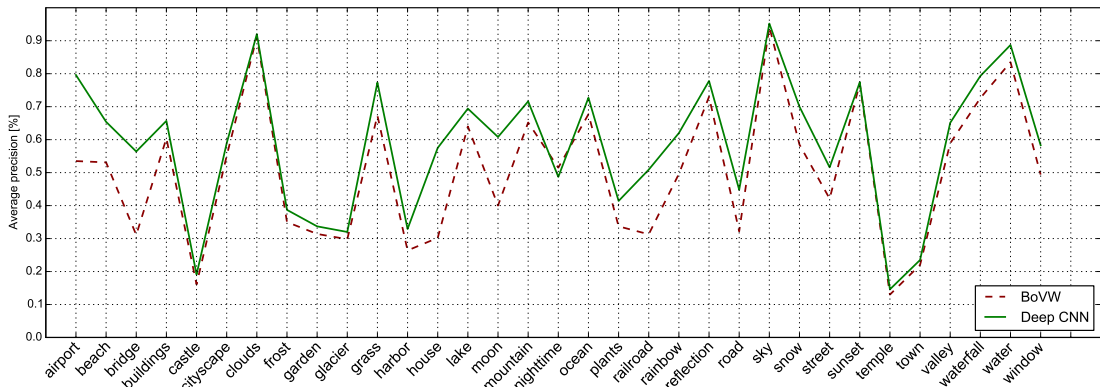


Figure 6.8: Performance comparison between the BoVW and the deep multi-label CNN approach on the NUS-WIDE scene dataset.

Several experiments have been performed. First, the performance of the deep multi-label CNN-based approach for concept classification has been compared to a BoVW approach, motivating the use of CNN-based algorithms in the final system. Second, the content-based video retrieval system has been evaluated on historical GDR television recordings.

The results are evaluated using the average precision (AP) score:

$$AP(\rho) = \frac{1}{|R \cap \rho^N|} \sum_{k=1}^N \frac{|R \cap \rho^k|}{k} \psi(i_k) \quad (6.1)$$

$$\text{with } \psi(i_k) = \begin{cases} 1 & \text{if } i_k \in R \\ 0 & \text{otherwise} \end{cases}$$

where N is the length of the ranked shot list, $\rho^k = \{i_1, i_2, \dots, i_k\}$ is the ranked shot list up to rank k , R is the set of relevant documents, $|R \cap \rho^k|$ is the number of relevant video shots in the top- k of ρ and $\psi(i_k)$ is the relevance function. Generally speaking, AP is the average of the precisions at each relevant video shot. To evaluate the overall performance, the mean AP score is calculated by taking the mean value of the AP scores from different queries.

BoVW vs. Deep Multi-Label CNN

To investigate the performance of the proposed deep multi-label CNN for concept classification, a comparison between a BoVW approach based on state-of-the-art handcrafted features and the deep CNN-based approach has been performed on the fully annotated, publicly available NUS-WIDE scene dataset [Chu+09]. The NUS-WIDE scene subset covers 33 scene concepts and consists of 34,926 images in total. Half of the images are used as the training set and the rest as the test set. For the deep multi-label CNN classifier, a GoogLeNet pre-trained on the ILSVRC 2012 dataset has been fine-tuned on the NUS-WIDE scene training images.

Using the BoVW approach, an image or a video shot is represented as a histogram of visual words by mapping the local descriptors to a precalculated visual codebook. The BoVW approach is based on a combination of different feature representations relying on optimized SIFT variants. These SIFT variants use different sampling strategies: a dense sampling strategy and a Difference of Gaussians (DoG) keypoint detector [Low99]. Color information is integrated using concatenated SIFT descriptors from different color channels (transformed color SIFT, RGB-SIFT) and a combination of SIFT descriptors and local color moments. Furthermore, spatial information is captured using a spatial pyramid of 1x1 and 2x2 equally sized subregions. Altogether, four different SIFT variants are used.

The visual codebooks are generated using a K-means algorithm and consist of 5000 visual words. Instead of mapping a SIFT descriptor only to its nearest neighbor or to all visual words, the codebook candidates are locally constrained to the five nearest visual words. This locality constraint has been shown to be superior for BoVW approaches [LWL11].

The different feature representations are combined in a support vector machine (SVM) classifier using multiple kernel learning [EMF11; Ewe+12; MEF11; Müh+11; MEF15]. For all feature representations, the χ^2 -kernel is used to measure the similarities between the data instances.

Figure 6.8 shows the comparison between the BoVW and the multi-label deep CNN approach on the NUS-WIDE scene dataset for each concept. While the BoVW approach achieves a mean AP of 50.3%, the deep CNN approach obtains 58.6%. Thus, the novel deep multi-label CNN approach significantly outperforms the BoVW approach on the NUS-WIDE scene dataset by a relative performance improvement of almost 20%. Results on the other NUS-WIDE subsets are 56.3% and 79.62% on NUS-WIDE objects and 40.83% and 55.24% on NUS-WIDE lite for BoVW and CNN, respectively.

In contrast to binary SVM classifiers, deep neural networks are inherently capable of processing multiple classes, such that only a single compact model has to be built for all concept classes. While the runtime of the BoVW approach is already 1.97 seconds for feature extraction on the CPU and the classification runtime depends linearly on the number of concepts, the multi-label

CNN takes less than a second on the CPU (Intel Core i5) and is even considerably faster on the GPU.

Although deep neural networks are computationally expensive in the training phase, they are very efficient during classification. Altogether, multi-label CNNs provide clearly better recognition quality, compact models and significantly faster classification runtimes.

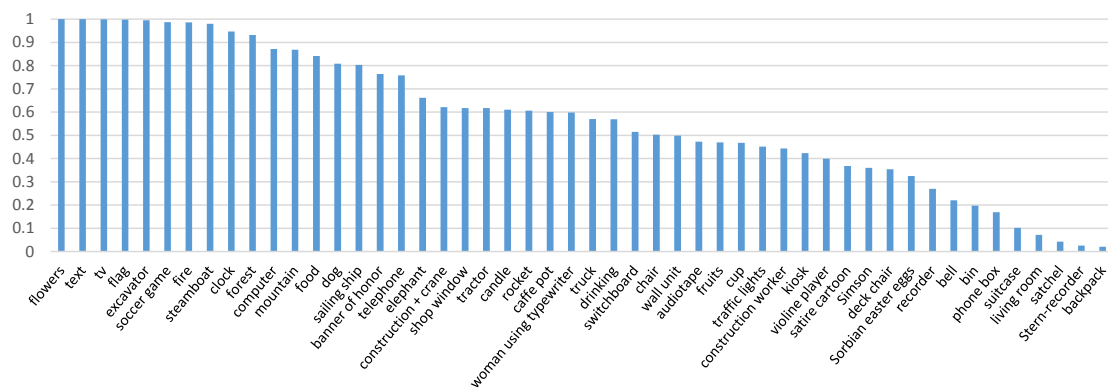


Figure 6.9: Similarity search results for 50 query images from the Internet in terms of average precision evaluated up to the first 100 video shots.

Historical GDR Television Recordings

In this section, the content-based video retrieval algorithms for concept classification, similarity search, person recognition and video ocr are quantitatively and qualitatively evaluated based on the video collection of historical GDR television recordings. In total, more than 3,000 hours of historical GDR television recordings have been digitized. The video footage is technically quite challenging. Many recordings are in grayscale and of low technical quality; the older the recordings, the poorer the video quality. The temporal segmentation of the videos resulted in approximately 2 million video shots. From these shots, 416,249 have been used for the training process and 1,545,600 video shots, corresponding to about 2,500 hours of video data, for testing.

The developed retrieval tool provides a web-based GUI to submit user queries related to concepts, persons, similar images and text. The retrieval results are presented to the user in the form of a ranked list of video shots (see Figure 6.6) where each video shot is represented by five key frames and a probability score indicating its relevance. Furthermore, a video player allows to visually inspect the video shots. In the following, the results for concepts and persons as well as sample queries for similar images and text are presented.

Query image:



Example: A user is searching for material for the film production *Flavors in the GDR* and uploads an image of a meal. By using this query image, the user can carry out the search without words or meta-data while the ranked results contain a large number of relevant shots.

Start	Middle	End	score
			0.92
			0.91
			0.91
			0.91
			0.91
			0.90
			0.90

1 of 10 1 2 3 4 5 6 7 8 9 10 100

Years: All From to

Select concept

Bild auswählen

Low-level High-level (100%)

Figure 6.10: A similarity search result for a query image showing a meal.

Concept Classification and Person Recognition

In total, 86 concepts, consisting of 77 concepts and 9 persons, were evaluated. From the original 91 concepts, 14 were dismissed due to an insufficient number of training images. However, another 14 of the 77 evaluated concepts have less than 100 training images. Altogether, 118,020 positive training examples were gathered for learning the concept models. The retrieval results for concepts and persons were evaluated based on the top-100 and top-200 ranked video shots.

Concept	Top 100	Top 200
Erich Honecker	100 %	100 %
Walter Ulbricht	100 %	100 %
Hilde Benjamin	98.6 %	96.2 %
Siegmund Jähn	98 %	98 %
Hermann Henselmann	85.6 %	85.7 %
Christa Wolf	76.4 %	76.4 %
Werner Tübke	65 %	65 %
Stephan Hermlin	64.3 %	47.1 %
Fritz Cremer	61.6 %	61.6 %

Table 6.1: Face recognition results (mAP).

Although 14 concepts have less than 100 training images, and despite poor video quality, we obtained mean AP scores of 62.4% and 58.0% for the top-100 and top-200, respectively. Even concepts occurring predominantly in grayscale shots of low video quality yielded good results, such as *daylight mining* with 84.3% AP. These results reveal the high robustness of the proposed multi-label deep CNN approach with respect to the low quality historical video data.

For person recognition, we achieved a very good result of 83.3% mean AP on the top-100 and 81.1% mean AP on the top-200 video shots, and even 100% AP for distinctive and frequently occurring personalities, such as Erich Honecker and Walter Ulbricht, as shown in Table 6.1. In total, we achieved a mean AP of 64.6% and 60.5% on the top-100 and top-200, respectively, for both concepts and persons.

Similarity Search

The interpretation whether two images are similar is subjective and context specific. The definition of similarity ranges from pixel-based similarity to image similarity based on the semantic content. How much low-level and semantic similarity contribute to the retrieval results can be individually adjusted in the GUI. Furthermore, two use cases have been implemented: searching by video frames selected from the corpus and searching by external images, e.g., downloaded from the Internet. In our evaluation shown in Figure 6.9, we focus on the more difficult task of semantic similarity using 50 external query images from the Internet chosen collaboratively by computer scientists and archivists. Each retrieval result has been evaluated up to the first 100 video shots. Altogether, we obtained a mean AP of 57.5%. Furthermore, we achieved a very fast response time of less than 2 seconds per similarity search query based on an image corpus of more than 7 million keyframes.

An example result for a query image showing a meal is presented in Figure 6.10. More retrieval results are visualized in Figure 6.11 where the first column shows the query images downloaded from the Internet followed by the first six highest ranked keyframe images.

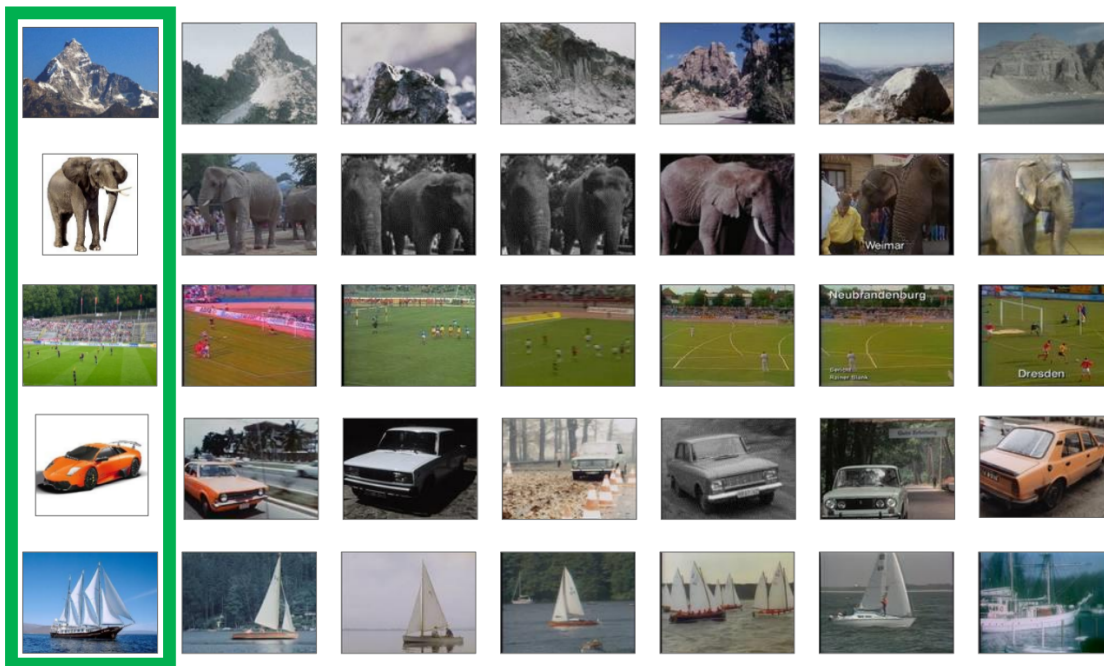


Figure 6.11: Retrieval results for query images downloaded from the Internet. The first column shows the query images followed by first six highest ranked images from the database.

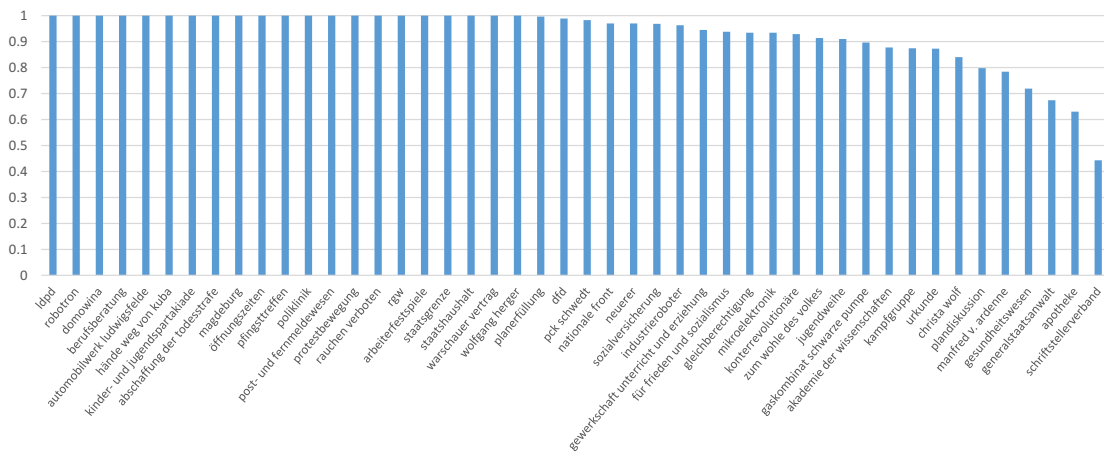


Figure 6.12: OCR retrieval results for 46 text queries in terms of average precision evaluated up to the first 100 video shots.

Video OCR

For the task of text retrieval, 46 query terms according to previously observed search query preferences of DRA users have been evaluated. Based on these 46 search queries, like *Ab-schaffung der Todesstrafe* (abolishment of death penalty), *Mikroelektronik* (microelectronics),

Öffnungszeiten (opening hours), *Protestbewegung* (protest movement), *Rauchen verboten* (no smoking), *Warschauer Vertrag* (Treaty of Warsaw), *Planerfüllung* (plan fulfillment), *Gleichberechtigung* (equal rights), *Nationale Front* (national front), *Staatshaushalt* (national finances), or *Kinder- und Jugendspartakiade* (children and youth spartakiad), a very satisfying retrieval performance of 92.9% mean AP has been obtained. The average precision results for the 46 queries are shown in Figure 6.12. As expected, the results for overlaid text are significantly better than for text within the scene.

Archivist's Perspective

In this section, the presented content-based video retrieval system is evaluated from an archivist's perspective with a focus on usability and usefulness for archivists and DRA users. Since users of the archive are often looking for everyday scenes in the former GDR, concepts such as *pedestrian*, *supermarket*, *kitchen*, *camping site*, *allotment* or *production hall*, are valuable contributions to help researchers finding appropriate scenes. Concepts with an AP score of more than approximately 50% turned out to be very useful in practice. 66% of the concepts achieved an AP score of more than 50%.

Furthermore, searching manually for persons in videos is a quite time consuming task, particularly for less known members of the *Politbüro* and Ministers of the GDR. Thus, the high quality of the provided automatic person indexing algorithms is a great benefit for archivists as well as for users of the archive.

The implemented similarity search system significantly extends the accessibility to the data in a flexible way. It provides complementary search queries that are often hard to verbalize. In addition, it facilitates incremental search. Previous results may serve as a source of inspiration for new similarity search queries for refining search intentions.

Another useful search option is offered by video OCR. OCR search results are very helpful since overlaid text is often closely related to the video content. The system recognizes the majority of slogans, locations, and other terms correctly.

In the following, the benefits of our content-based video retrieval system are illustrated based on search requests that arose in the context of the TV production *Das Erbe der Nazis* (heritage of the Nazis). While these queries are manually difficult and time-consuming to answer, the content-based video retrieval system is able to yield fast and practically useful results. Example queries are:

- *Every-day scenes in the GDR across the decades 1950s/60s/70s/80s*:
Using the content-based video retrieval system, this query can be answered by specifying the years and searching for concepts concerning every-day life in the GDR, e.g., *pedestrian* or *kitchen*. These results may serve as a starting point for further similarity search queries.
- *Monday demonstrations in 1989, crowds, banners "Wir sind das Volk" ("We are the people")*:
The retrieval result for this query can be obtained by combining queries for the concepts *demonstration* and *banners/slogans*, an OCR query for in-scene text and a restriction to the years 1989.

- *Erich Honecker, parade, 1970s:*
Useful results for this query can be found by combining a query for the person *Erich Honecker*, a query for the concept *military parade* and restricting the years to the 70s.

Altogether, the fine-grained automatic annotation is a very valuable supplement to human-generated meta data. Due to the variability of the content-based video retrieval system, different user needs are taken into account. The combination of different search modalities allows the DRA to answer a wide range of user queries leading to more precise results in significantly less time.

6.1.4 Summary

The DRA maintains the cultural heritage of television of the former GDR. The uniqueness and importance of this archive causes a great interest in the video content. In this section, we have presented a content-based video retrieval system for searching in historical collections of GDR television recordings. The recordings have been digitized, a service-oriented architecture for dealing with a large amount of video data has been implemented and novel algorithms for visual concept classification, similarity search, person recognition and video OCR have been developed to complement human annotations and to support users in finding relevant video shots. Experimental results on about 2,500 hours of GDR television recordings have indicated the excellent video retrieval quality in terms of mean average precision as well as in terms of usability and usefulness from an archivist's perspective.

6.2 Deep Learning for Content-based Video Retrieval in Film and Television Production

6.2.1 Introduction

Digitization has fundamentally changed the workflow of professional media production. Today, recording and production as well as processing and distribution of video contents are accomplished in a convenient and efficient manner. However, a meaningful annotation of multimedia contents still requires a large amount of manual effort provided by human annotators. This often entails that multimedia material is annotated only superficially, if at all. Due to this time-consuming annotation process, video annotation is mostly carried out for an entire video, whereas a frame- or shot-based annotation is required for making the video material searchable and thus useful for film or television production beyond the scope of the underlying project. As a consequence of the lack of automatic annotation systems in the domain of media production, a vast amount of produced high-quality video data remains inaccessible.

Recently, deep learning approaches, particularly deep convolutional neural networks, have led to breakthroughs in many computer vision fields. Integrating these new technologies into the media production workflow and leveraging automatic video annotation would allow media production firms, broadcasting companies, television stations, media archives and footage agencies to utilize the enormous potential of their stored multimedia data.

In collaboration with a film and television production company, taglicht media Film- & Fernsehproduktion GmbH¹², the requirements of automatic content-based video analysis in the field of media production have been identified. Taglicht media is a leading German production company of high-quality documentary films with a focus on history, science, nature, wildlife and societal affairs. The main goal of using content-based video analysis in the field of media production is to support the process of video cutting and distribution. Since the time intervals between video shooting, cutting and distribution are often small, the automatic content-based video analysis process has to be fast and efficient. The process of video cutting will not be fully automated in the foreseeable future, but the video cutter can be supported by providing proposals and a fast overview of the underlying video footage. Therefore, meta-information about the occurring persons and general visual concepts are very useful. Furthermore, content-based similarity search increases the accessibility to the video data. This is also important for media distribution and sales.

In this section, we present new deep learning algorithms for visual concept detection, similarity search, face detection, face recognition and face clustering in the context of media production, bundled in a multimedia tool for fast video inspection and retrieval. Furthermore, a novel multi-task learning approach for combining concept detection and similarity search, a new concept lexicon tailored to media production, and novel visualization components, developed in collaboration with the taglicht media company, are introduced. Experimental results show the quality of the proposed approaches. The concept detection approach achieves a mean average precision of approximately 90% on the top-100 video shots, and the face recognition approach

¹²<http://taglichtmedia.de>

outperforms the baseline on the public *Movie Trailers Face Dataset* [OWS13] by about 20% in terms of average precision.

Parts of this section have been published in: Markus Mühling, Nikolaus Korfhage, Eric Müller, Christian Otto, Matthias Springstein, Thomas Langelage, Uli Veith, Ralph Ewerth, and Bernd Freisleben. “Deep Learning for Content-based Video Retrieval in Film and Television Production.” in: *Multimedia Tools and Applications* 76 (21 Nov. 2017), pp. 22169–22194. ISSN: 15737721. DOI: 10.1007/s11042-017-4962-9.

6.2.2 Content-based Video Analysis for Media Production

In recent years, the media production workflow has been digitized, starting from video shooting through video cutting and editing up to video distribution. Although digital videos can be used conveniently as inputs to automatic content-based video analysis algorithms, the content-based labeling process is typically still manual, time-consuming and inefficient compared to the remaining workflow. Hence, video data is only annotated if it is necessary for the underlying task. In the video shooting stage, for example, video data is labeled only to a very limited extent to indicate its film or program affiliation. The utilization of video footage beyond the scope of a current project is impossible without a high manual effort. For video marketing activities of footage agencies, a fine-grained content-based labeling of videos is quite important to increase the size of the material that shows great promise for sale. Even in video archives, human annotators typically focus on larger video sequences and contexts, making search for dedicated video content in large video archives difficult and time-consuming. Searching for unlabeled content or finding similar video content manually is currently not feasible.

Our multimedia tool is called GoVideo and has been developed in the context of media production. It bundles novel deep learning algorithms for concept detection, similarity search, face detection, recognition and clustering.

For the purpose of media production, a novel visual concept lexicon has been developed, which will be introduced in Section 6.2.2 together with the concept detection approach. To find similar video content, a fast and scalable algorithm for similarity search is proposed in Section 6.2.2. Additionally, a combined multi-task model for concept detection and similarity search is presented in Section 6.2.2, which helps in reducing memory and runtime requirements. Furthermore, acting persons play an important role in sorting video footage both for getting an overview of the occurring persons (i.e., who occurs when in the video) and for searching for a particular person in the overall data. Section 6.2.2 presents the algorithms for finding persons in videos.

The labels automatically generated by these algorithms are assigned to the fundamental units of a video sequence, i.e., video shots, which are generally understood as video sequences recorded continuously without any interruption. This temporal segmentation is the result of a shot boundary detection algorithm. We use shot boundary detection algorithms without thresholds [EF04; EF09] that belonged to the top three approaches at the TRECVID challenge 2007¹³. Concept detection and similarity search are performed on a maximum of five equally distributed keyframes per video shot (depending on the length of the shot) due to performance

¹³<http://trecvid.nist.gov>

considerations. Frames in one shot usually look very similar to each other and the benefit of considering all of them is small compared to the high increase of computation costs.

Concept Detection

The fundamental problem of visual concept detection is to overcome the discrepancy between the extracted features and the human interpretation of the (audio-)visual data. In the literature, this discrepancy is also known as the “semantic gap”. Smeulders et al. [Sme+00b] describe the semantic gap as “the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation”.

An important building block of the proposed concept detection approach is a suitable concept lexicon. In collaboration with media producers, a new concept lexicon tailored to media production has been developed. The lexicon consists of 58 concepts and is divided into five different categories: “who”, “what”, “where”, “when” and “how”. An overview of the concepts is shown in Table 6.2. The lexicon consists of general concepts with a high coverage of video content and media production specific concepts, such as “full shot”, “medium shot” and “close-up”. Depending on the time of day, some television programs try to adjust their video content. Therefore, a distinction between daytime and nighttime is important. Overall, the new concept lexicon helps to naturally understand the video content for accelerating the process of video cutting and presorting: “who” indicates how many persons or animals are occurring in a video scene, “what” is related to actions and events, “where” gives insights into locations and places, “when” distinguishes between day and night and “how” provides information about shot sizes, where close-ups of hands are also an important concept since they strongly correlate with specific actions.

The automatic classification of these concepts is quite challenging due to the large complexity and variability of their appearance. Recent advances in deep learning have led to a renaissance of neural networks, in particular deep convolutional neural networks (CNNs), in the field of computer vision [KSH12; GMH13; Bre+13; Tai+14]. Instead of using hand-crafted features such as SIFT descriptors [Low99], CNNs learn the features automatically during the training process. A CNN consists of several alternating convolution and max-pooling layers with increasingly complex feature representations and typically has several fully connected classification layers.

State-of-the-art neural network architectures for image recognition [KSH12; Sze+15] as well as most current datasets [Den+09; Zho+14] consider only a single concept per image (“single-label”). In contrast, real world concept classification scenarios are multi-label problems. Several concepts, such as “person”, “outdoor” and “total view”, may occur simultaneously in a scene. Wei et al. [Wei+14] introduced a flexible deep CNN architecture, called Hypotheses-CNN-Pooling, which produces multi-label predictions. The approach is similar to region-based CNNs, because it takes object segment hypotheses as inputs. However, the results of the different hypotheses are finally aggregated using a max-pooling layer. This approach is computationally very expensive and does not scale to large multimedia archives.

In this section, two approaches are pursued for concept detection. Both of them are based on adapting pre-trained CNN classification models to the novel concept lexicon and fine-tuning them for the new training data. Up to 1,000 positive training instances plus negative examples

have been collected from video footage, documentary films and Google search. The models were trained and fine-tuned using the deep learning framework Caffe [Jia+14].

The first approach uses a multi-label CNN. While other authors employ special ranking loss layers to handle multi-label predictions [Gon+13; Cha+14], we have extended the CNN architecture of a Deep Residual Neural Network [He+16] using a sigmoid layer instead of the softmax layer and a cross entropy loss function. The model is pre-trained on ImageNet [Den+09] and fine-tuned to the collected data and the new concept lexicon.

The second approach uses a single-label CNN. Here, we use a pre-trained model based on ImageNet [Den+09] and Places [Zho+14], providing 1,365 concepts including 365 scene categories and 1,000 object categories. These concepts are mapped to the new concept lexicon, and the final probabilities are the aggregated scores of all related concept categories. For example, all animal related concepts are aggregated for the concept “animal”. Concepts and their positive training examples not covered by the mapping are added to the single-label CNN, and the training process is continued on the overall training data. Several concepts are ignored in the single-label model due to a large overlap with existing concepts. The concept “full shot”, for example, always belongs to several concepts, thus cumbering the training process.

Finally, the single-label and multi-label models are combined in a post-processing step. Based on an evaluation of the models, the probabilities per concept are averaged if the evaluated performance measures are similar, otherwise the better model is used.

Similarity Search

Similarity search in videos addresses a variety of modalities and similarity measures. For example, Meddeb et al. [MKA16] perform speech similarity search. Hudelist et al. [Hud+16] provide interactive search by combining video retrieval with human-based visual inspection. In our work, we rely on the visual modality and focus on large-scale semantic similarity search. The definition of similarity ranges from pixel-based similarity to semantic similarity that corresponds to human understanding. The definition and optimization of similarity functions is subject to current research [Wan+16; Bla+16].

In contrast to textual concept-based queries that are limited to a set of predefined concepts, similarity search provides more flexibility. Arbitrary images can be used to query the video database for similar content.

With an increasing size of a video archive, the need for an efficient image-based search becomes important. Query-by-content based on low-level features is often insufficient to search successfully in large-scale multimedia databases. However, image representations learned by deep neural networks have greatly increased the performance of content-based image retrieval systems [Wan+14], since they are less dependent on pixel intensities. Although fast retrieval in large databases is an essential requirement for practical use, these high-dimensional CNN features are not suitable to efficiently search in large video databases. Therefore, we have developed a semantic hashing approach [KH11; Lin+15; SH09] that uses binary image codes for compact representations rather than full CNN features. Our method extends the approach introduced by Lin et al. [Lin+15]. We fine-tuned a VGG-16 CNN architecture [Cha+14], pre-trained on the Places dataset [Zho+14], with an additional coding layer before the final

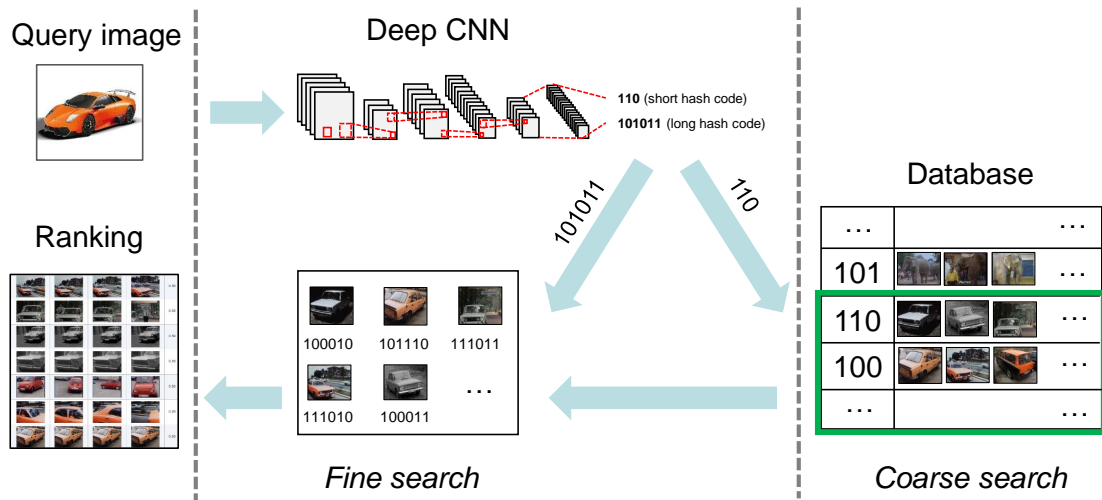


Figure 6.13: Content based similarity search.

classification layer. In contrast to the approach of Lin et al. [Lin+15], hash codes are used for refined search as well. Our two-stage approach integrates two coding layers, a 64-bit and a 256-bit binary coding layer, into the same architecture, enabling concurrent computation at testing time. The returned hash codes for the keyframes of the video collection (five per shot) are fed into the deep CNN and the returned binary hash codes are kept in the database. In order to compare the hash codes, the Hamming distance is used together with a vantage point tree [Yia93] as an additional index structure to speed up the search. Figure 6.13 shows the stages of processing a query image.

Multi-Task Learning

Concept detection and similarity search are related tasks. Both use pre-trained CNN models based on visual recognition tasks. To reduce memory and runtime requirements, a multi-task CNN architecture is proposed to enable concurrent computation of concept predictions and image codes at testing time. Therefore, the single-label concept detection CNN architecture (see Section 6.2.2) is extended by two further branches for 64 and 256-bit codes, respectively. These branches contain corresponding encoding and decoding layers with additional classification loss layers, as already described in the previous section. Altogether, the multi-task learning setting contains three loss layers equally weighted for error back-propagation: one for learning concept predictions and two for learning short and long binary image codes. The shared CNN architecture is a deep residual neural network [He+16] pre-trained on ImageNet [Den+09] and Places [Zho+14]. The new CNN architecture with two tasks and three classification losses is depicted in Figure 6.14. This architecture efficiently shares the computation of binary hash codes for similarity search and the computation of class probabilities for image classification. The CNN is fine-tuned for about one epoch on the training data set. More precisely, the model is trained for 230,000 iterations with a batch size of 24. We set the initial learning rate to 0.00033 for the three final layers and an order of magnitude smaller for the remaining layers. The learning rate is decreased following a polynomial decay with a power of 4, resulting in a fast

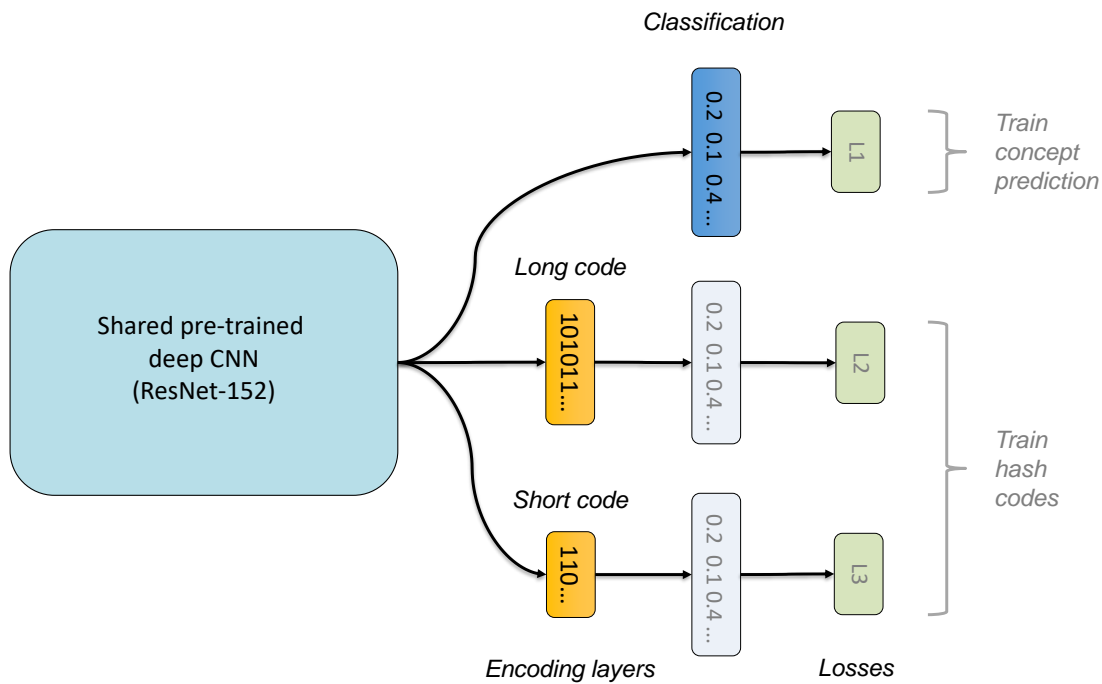


Figure 6.14: Multi-task learning of binary image codes and concept predictions.

decrease of the learning rate early in the training process. As an optimization method, we use Nesterov's accelerated gradient method [Sut+13; Nes83] with a momentum of 0.9.

Face Recognition

Convolutional neural network approaches for face recognition have recently achieved and surpassed human performance [SKP15b; PVZ15]. However, for large video archives face recognition is still quite challenging because usually thousands of different persons must be captured, sometimes over long periods of time, with a large variety of occurrences. Furthermore, the increasing size of the archives needs to be handled. In the following, we present our face recognition framework to satisfy these challenges. In a first step, faces are detected and tracked to extract the relevant video content for face recognition. Subsequently, the facial features of every image within each track are computed by a convolutional neural network. Based on the extracted facial features, three tasks are performed: face identification, face retrieval, and face clustering.

Face Detection and Tracking

To quickly detect faces in a large amount of video data with high precision and recall, we investigated several approaches. Basic approaches, such as the approach of Viola and Jones [VJ01], are very fast, but are not able to reliably find non-frontal faces. With the rise of convolutional neural networks, face detection approaches have been significantly improved in terms precision

and recall. However, first CNN approaches [FSL15] rely on computationally expensive CNN calculations that are applied several times in a multi-scale and sliding window fashion. The task of locating pre-defined concepts in visual content has been improved by utilizing region proposal networks [Ren+15] (Faster R-CNN) that automatically propose regions and share their weights with the classification network or more recently without any object proposals [Liu+16b] resulting in a significant speed-up. The former motivated Jiang and Learned-Miller [JL17] to apply a R-CNN approach to face detection. Our approach to face detection is based on this method because it yields state-of-the-art results on the public *Face Detection Data set and Benchmark* (FDDB) [JL10] and the *IARPA Janurs Benchmark A* (IJB-A) [Kla+15]. Similar to Jiang and Learned-Miller [JL17], we fine-tuned the R-CNN Pascal VOC (Visual Object Classes) model [Ren+15] on the database *WIDER FACE* [Yan+16]. We omitted training faces with a size below a particular size (width or height of ground truth below 10 px), because we found that those faces are not relevant to face identification tasks since they are most likely not depicting a person of interest. For a faster computation, we utilize the more simplistic Zeiler & Fergus [ZF14] CNN architecture instead of the VGG16 [SZ14] architecture, while producing comparable results.

Based on the high quality results of our face detection approach, a simple set of conditions is sufficient for reliable face tracking. Similar to Ortiz et al. [OWS13], we adapt the face overlap as well as the RGB histogram intersection to decide if the current detection belongs to an already existing face track or depicts a new identity. During our investigation, we observed that most of the errors occur in close-up dialogs. Due to the insignificant changes in lighting and face location, cut detection is often unsuccessful, which leads to face trackings treating both persons as one face track. To counteract this observation, a simple and fast Local Binary Pattern (LB) [OPM02] feature vector is additionally extracted for every face to help to distinguish between different persons. If the face tracker can not establish a connection to a face in the last 20 frames of the video, the face track is finalized and saved under the condition that it has a minimal length of 5 detections.

Face Representation

Convolutional neural networks typically require large amounts of training data in order to learn reliable models. Hence, as mentioned by Masi et. al [Mas+16], current state-of-the-art approaches on *Labeled Faces in the Wild* [Lea14] like FaceNet [SKP15b], VGG Face [PVZ15] or DeepID3 [Sun+15] use millions of face images and very deep CNN architectures. However, before *MS-Celeb-1M* [Guo+16] has been released recently, the largest publicly available dataset *CASIA-WebFace* [Yi+14] contained only around 500.000 images covering about 10.000 person identities. For this reason, we use the network of Yi et al. [Yi+14], which performs well on public benchmarks like *Labeled Faces in the Wild* [Lea14] exploiting *CASIA-WebFace*. This approach yields very good results considering the amount of training data. Another advantage is its low computational cost, due to the smaller CNN structure with only eleven weight layers, as well as the relatively small 320-dimensional feature vector extracted from the last pooling layer. For every track t consisting of $|t|$ images in our video archive, we extract the feature vector f_i of every image i , resulting in a set of feature vectors $F_t = \{f_1, \dots, f_{|t|}\}$. All these sets are stored in a track dictionary $D_T = \{F_1, \dots, F_{|T|}\}$, where $|T|$ refers to the current number of tracks in the video archive. This workflow is visualized in Figure 6.15.

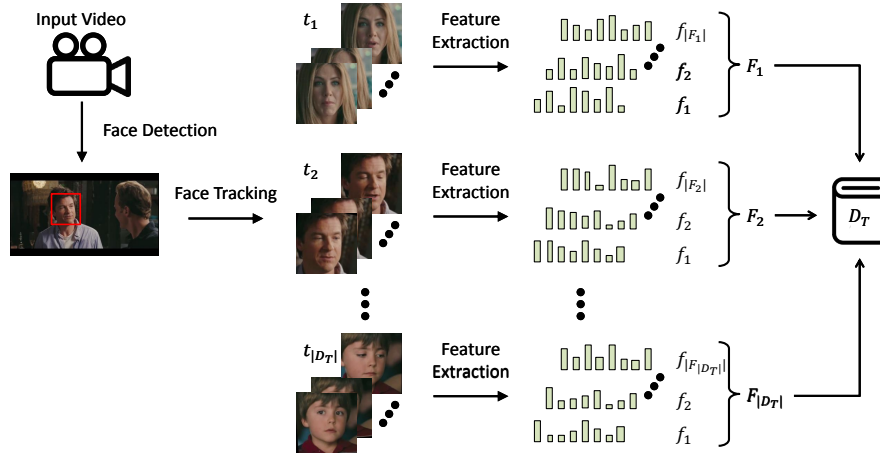


Figure 6.15: An overview of our video indexing workflow.

Face Identification

Face identification algorithms operate in a supervised manner, i.e., they require a dictionary $D_P = \{F_1, \dots, F_{|P|}\}$ of persons P that the user wants to identify in the underlying video footage. Given a person p , we first extract the facial features F_p of every corresponding image using the CNN model described in Section 6.2.2. As a result, the person dictionary $D_P = \{F_1, \dots, F_{|P|}\}$ for each person $p \in P$ can be computed.

Based on the extracted facial features, we use two different ways of representing a given face track t when comparing it to a specific person. The first track representation (*tr-single*) calculates the cosine similarity of each feature vector $f_i \in F_t, i = 1, \dots, |F_t|$ to every $f_j \in F_p, j = 1, \dots, |F_p|$. The final similarity value is defined by calculating the mean of all $|F_t| \cdot |F_p|$ comparisons per track. However, this is computationally expensive. For this reason, the second track representation (*tr-mean*) uses the mean feature vector of the given face track:

$$\bar{f}_t = \frac{1}{|F_t|} \sum_{k=1}^{|F_t|} f_k \quad (6.2)$$

and compares it to all $f_j \in F_p$ for each character p . This reduces the number of comparisons to $|F_p|$ per track, making it more suitable for big data applications.

Face Retrieval

In contrast to face identification, face retrieval focuses on extracting similar face tracks in the video content by providing a query image showing an arbitrary person. The image i is uploaded by the user (similar to Figure 6.13) and subsequently analyzed by the face detection approach described in Section 6.2.2. If a face is found, the feature vector f_i is extracted. Similar to Section 6.2.2, we compare this feature vector with the track representations *tr-single* and *tr-mean* to measure the similarity to every face track in D_T . While the similarity of the query image to a specific track t can be calculated by only one computation with *tr-mean*, the mean cosine similarity value of all $|F_t|$ comparisons is calculated for *tr-single*.

Face Clustering

To summarize the content of a video in an unsupervised manner, it is useful to cluster the appearances of the persons in the video. This can be used for annotation purposes or to transform raw video footage into actual television broadcasts. Therefore, our framework uses an agglomerative clustering approach [EMF07] for the face tracks in the dictionary D_T for a given movie. For this purpose, the cosine similarities between the tracks are computed using the mean feature vectors (*tr-mean*). As with all hierarchical clustering algorithms, a threshold is necessary to terminate the combination of clusters when a certain similarity minimum is reached. Selecting this threshold is one of the perennial issues with data clustering and is discussed in Section 6.2.4. To avoid single track clusters in our web interface, we reduce the number of results by omitting those that only contain a single face track and therefore most likely show an unknown person. As a representation of a cluster in the web interface, we use the most frontal face of any face track in a given cluster, determined by a feature comparison with a computer generated standard frontal face.

6.2.3 Video Retrieval Tool

The previously described algorithms for concept detection, similarity search, face detection, recognition and clustering are bundled in a video analysis and retrieval tool, called GoVideo. It is based on a service-oriented architecture, as described in Section 6.2.3. User interaction is handled via a web-based client-server application. The GUI and the novel components are described in more detail in Section 6.2.3.

Service-oriented Architecture

Due to the computationally expensive video analysis algorithms, a distributed, heterogeneous architecture is used to provide scalability. An overview of the architecture is shown in Figure 6.16. The web services are executed on different hardware architectures. Here, we have to distinguish between CPU and GPU intensive algorithms.

For the CPU intensive preprocessing algorithms, a compute cluster with a head node and four compute nodes is used, each node consisting of 4 AMD Opteron processors 6212 with 16 cores@1.4 GHz and 64 GB main memory each. The Sun Grid Engine (SGE) is used as the job scheduler. Therefore, specific interfaces have been defined. The preprocessing algorithms have been encapsulated in separate jobs (bash scripts). The preprocessing web service starts several jobs for transcoding the videos, for shot boundary detection and for extracting images and thumbnails. The web service submits jobs to the SGE, controls job execution and provides status information. For connecting the computational resources, the distributed resource management application API (DRMAA) is used. Instead of transferring user data directly, only references are sent via parameters during the service call. The images and videos are stored on a data server and the actual data transport is handled via network file system shares. The advantage is an overlap of data transfer and service execution that contributes to the optimization of the overall runtime performance.

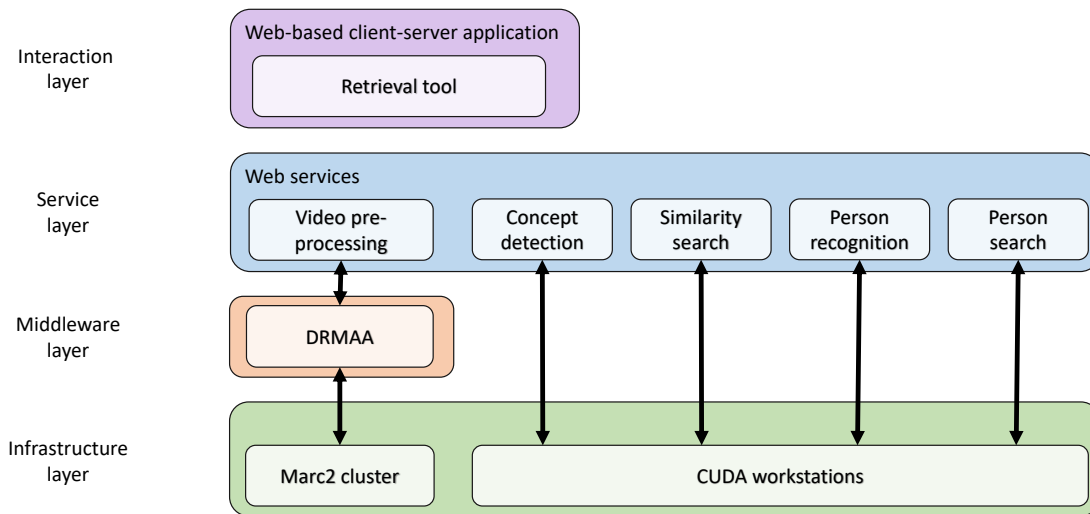


Figure 6.16: Service-oriented architecture for content-based video retrieval.

After the videos have been preprocessed, content-based meta-data generation jobs for concepts, persons and similarity search are started. The resulting meta-data and indices are used by the retrieval tool to respond to arbitrary search queries.

The concept detection, similarity search and face processing related web services intensively use the GPU and are installed on dedicated servers with Nvidia Geforce GTX 770 graphics cards with 4GB memory and GeForce GTX Titan X GPUs with 12GB RAM, respectively.

Visualization

The client/server-based web application provides an intuitive user interface to perform content-based video analysis and retrieval in a distributed, heterogeneous environment. The user management ensures that each user sees only its own video collection.

The GUI (see Figure 6.17) is divided into several areas. The video table at the top of the GUI handles video selection and uploads. While analyzing a video, the video table additionally provides progress information. Selecting a video opens a video specific timeline where the brightness of the events indicates the confidence of the system. The results in the different components are arranged according to the new concept lexicon. The pages in the timeline, for example, correspond to the categories of the concept lexicon plus a person clustering. The person clustering provides a fast overview of the acting persons within a video. Furthermore, a bar chart is displayed at the bottom of the video player to clearly present the occurring concepts according to the new concept lexicon. If an event in the timeline is selected, the video player jumps at the position with the highest confidence within the shot.

Using the search buttons for concepts, persons and similarity on the right hand side forces the GUI to switch from the timeline view to the retrieval view. The retrieval results are presented to the user in the form of a ranked list of video shots where each video shot is represented by the video frame with the highest probability of showing the requested content (see Figure

6 Image Similarity Search in Applications

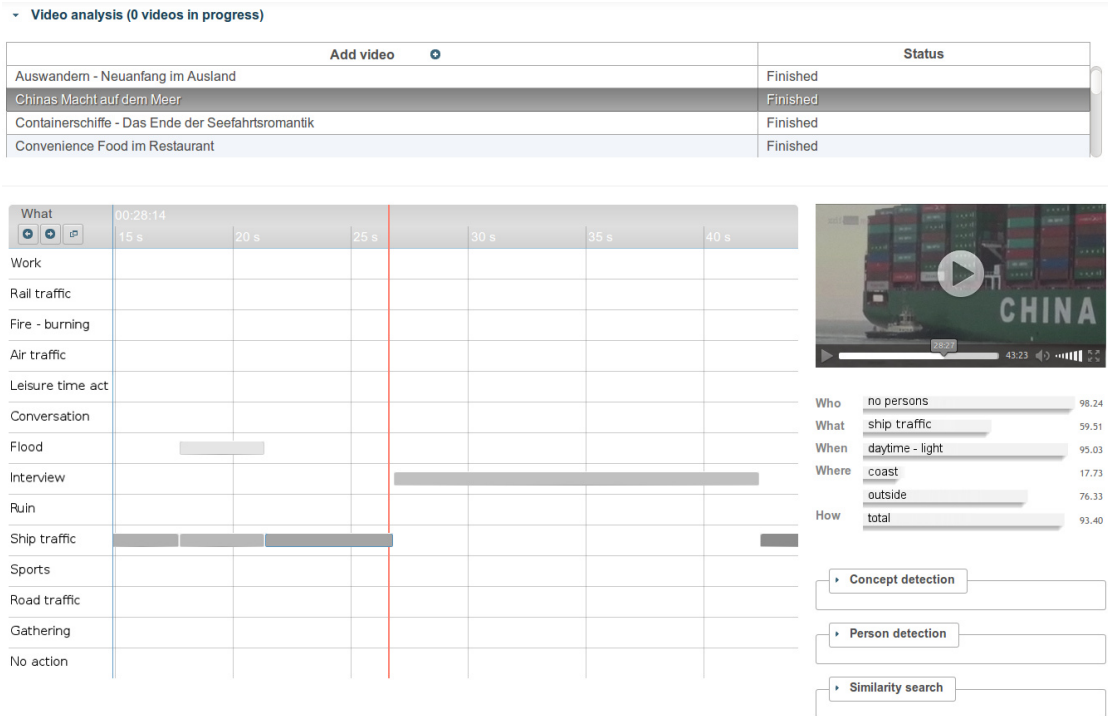


Figure 6.17: GUI of the video analysis and retrieval tool.

6.23 for an example of similarity search). The thumbnails are arranged in a data grid with the probability provided at the bottom of the image. The number of rows in the data grid can be adjusted to see as much as possible results on a single page.

6.2.4 Experimental Results

In this section, experimental results are presented for concept detection (Section 6.2.4), similarity search (Section 6.2.4), and face recognition (Section 6.2.4). For the face recognition task, the scenarios face identification (Section 6.2.4), face retrieval (Section 6.2.4) and face clustering (Section 6.2.4) are evaluated.

Concept Detection and Similarity Search

Taglicht media uploaded 94 videos, mainly consisting of documentaries. In total, the test set consists of 41 hours of video data. The concept detection results are evaluated using the average precision (AP) score that is the most commonly used quality measure in video retrieval. We calculated the AP score from the list of ranked video shots up to rank N as follows:

$$AP(\rho) = \frac{1}{|R \cap \rho^N|} \sum_{k=1}^N \frac{|R \cap \rho^k|}{k} \psi(i_k) \quad (6.3)$$

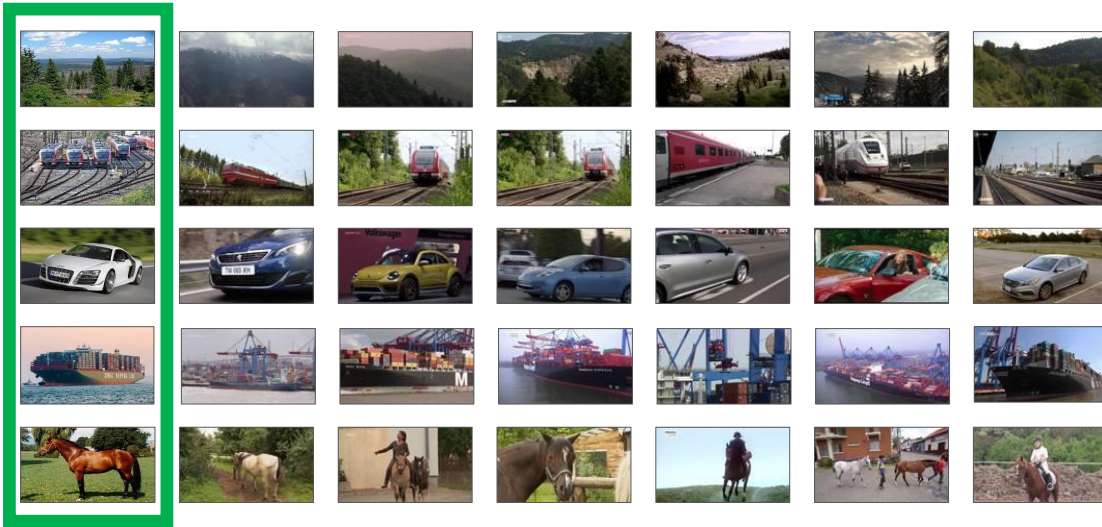


Figure 6.18: Visualization of similarity search results for five query images.

$$\text{with } \psi(i_k) = \begin{cases} 1 & \text{if } i_k \in R \\ 0 & \text{otherwise} \end{cases}$$

where $\rho^k = \{i_1, i_2, \dots, i_k\}$ is the ranked shot list up to rank k , R is the set of relevant documents, $|R \cap \rho^k|$ is the number of relevant video shots in the top- k of ρ and $\psi(i_k)$ is the relevance function. Generally speaking, AP is the average of the precisions at each relevant video shot. To evaluate the overall performance, the mean AP score is calculated by taking the mean value of the AP scores from different queries. Table 6.2 shows the results of the multi-label and single-label approach. While the multi-label approach also uses negative training examples, it achieves an excellent retrieval performance of 87.5% mean AP. Combining the models as proposed in Section 6.2.2 even results in approximately 90.3% mean AP.

The semantic similarity search also delivers very good retrieval performance. We evaluated the retrieval results for 21 query images based on the multi-task CNN model on the top-100 video shots and obtained a mean AP score of 93.5%. Detailed results are shown in Table 6.3. Five query images downloaded from the WWW together with the top six retrieved video shots are visualized in Figure 6.18. Another example for similarity search is depicted in Figure 6.23 where a keyframe of a boat is selected as query image and the results are visualized in the retrieval view of the application.

While the similarity search results based on the multi-task CNN architecture perform very well and the concept detection results achieve similar quality (81.8% mean AP (single-task) vs. 78.5% mean AP (multi-task)), the multi-task approach saves memory and runtime. Concept predictions and image codes are concurrently computed so that the image codes for similarity search are obtained without considerable extra computational effort. The runtime of the combined webservice for concept detection and image code generation takes about 0.22 seconds per image on a Nvidia Geforce GTX 770 graphics card including image transfer to the webservice and preprocessing on the CPU. For analyzing a minute of video data, the runtime depends on

6 Image Similarity Search in Applications

		Multi Label	Single Label			Multi Label	Single Label
Who	one person	0.901	-	Where cont.	airport	0.955	0.620
	two persons	0.947	-		train station	0.870	0.940
	several persons	1.000	-		inside	0.976	0.984
	crowd	1.000	-		inside train station	0.960	1.000
	no persons	1.000	-		inside airport	0.952	0.775
	animal - animals	1.000	0.963		inside train	0.906	0.804
What	rail traffic	0.988	0.985		inside airplane	0.942	0.771
	ship traffic	0.991	0.992		inside car	0.940	0.618
	air traffic	0.995	1.000		inside living room	0.980	0.696
	road traffic	0.961	0.931		inside kitchen	0.995	0.689
	work	0.626	0.959		inside office	0.363	0.982
	sport	0.979	0.983		inside repair shop	0.266	0.229
	leisure time activity	0.901	0.731		inside factory building	0.805	0.932
	fire - burning	0.975	0.983		inside school - university	0.454	0.115
	flood - inundation	0.922	0.911		inside laboratory	1.000	1.000
	ruin - destruction	0.877	0.980		inside shop - supermarket	0.928	0.974
	interview	0.977	0.983		inside ship	0.958	0.103
	conversation	0.818	0.928		When	in the daytime - light	1.000
gathering - deployment	0.540	0.955	at night - dark	1.000		0.984	
Where	no action	0.000	0.134	How	total	1.000	-
	outside	1.000	0.969		medium shot	1.000	-
	detached house	0.816	0.637		close-up	0.937	-
	settlement	0.962	0.551		face	0.873	1.000
	village	0.715	0.330		hands	0.965	0.860
	city	1.000	0.869	Mean	0.875	0.818	
	open landscape	0.977	1.000				
	land - field	0.946	0.933				
	mountains	0.944	0.996				
	coast - ocean	0.971	0.935				
	river	0.681	0.674				
	lake	0.827	0.771				
park - garden	0.518	0.857					
sports venue	1.000	0.886					
forest	0.979	0.985					

Table 6.2: Concept detection results in terms of average precision based on the top 100 shots per concept.

Query image	AP	Query image	AP
rail traffic	0.942	inside kitchen	0.986
fire	0.945	inside car	0.594
air traffic	0.991	city	0.982
river	0.892	sports venue	1.000
mountains	0.913	inside office	0.998
hands	0.720	village	0.836
coast - ocean	0.978	detached house	0.976
ship traffic	0.994	face	0.998
road traffic	1.000	land - field	0.943
animal	1.000	crowd	0.943
forest	0.999	Mean	0.935

Table 6.3: Similarity search results in terms of average precision based on the top 100 shots per query image.

the video resolution and the number of video shots or rather keyframes. The average number of keyframes on the dataset that mainly consists of documentary films is about 30 keyframes per minute resulting in a computational runtime for concept detection and hash code generation of about 7 seconds for a one minute video. Once the videos have been analyzed and the concept probabilities per shot have been stored in the database concept queries can be answered extremely fast based on an appropriate database index. Even the response time for similarity search queries based on the comparison of image codes achieves an impressive performance of less than 2 seconds for searching in up to 10 million keyframes which corresponds in the case of documentary films to about 5555 hours of video data.

Face Recognition

Since the proposed face recognition framework is designed for large and continuously growing video archives, the system is evaluated on a suitable test dataset called *Movie Trailers Face Dataset* [OWS13] containing 113 movie trailers from 2010. To compare our results to other approaches, we use the face tracks provided by the authors of the *Movie Trailers Face Dataset*. The dataset covers 145 actors who are partially represented in the person dictionary D_P extracted from *PublicFig+10* [Kum+09; OWS13]. The fact that only half of the actors actually appear in the movie trailers offers a great challenge and forces the face identification framework to find a certain person in a large pool of possible candidates. Another advantage of the *Movie Trailers Face Dataset* is that the 4.485 face tracks are separated in 35% known identities and 65% unknown identities which is a realistic representation of a real television broadcast. As the input for each face recognition task we first extract the image feature vectors of every face track in the *Movie Trailers Face Dataset* and store them in the track dictionary D_T . In the following, the experimental setups for face identification (Section 6.2.4), face retrieval (Section 6.2.4) and face clustering (Section 6.2.4) are explained.

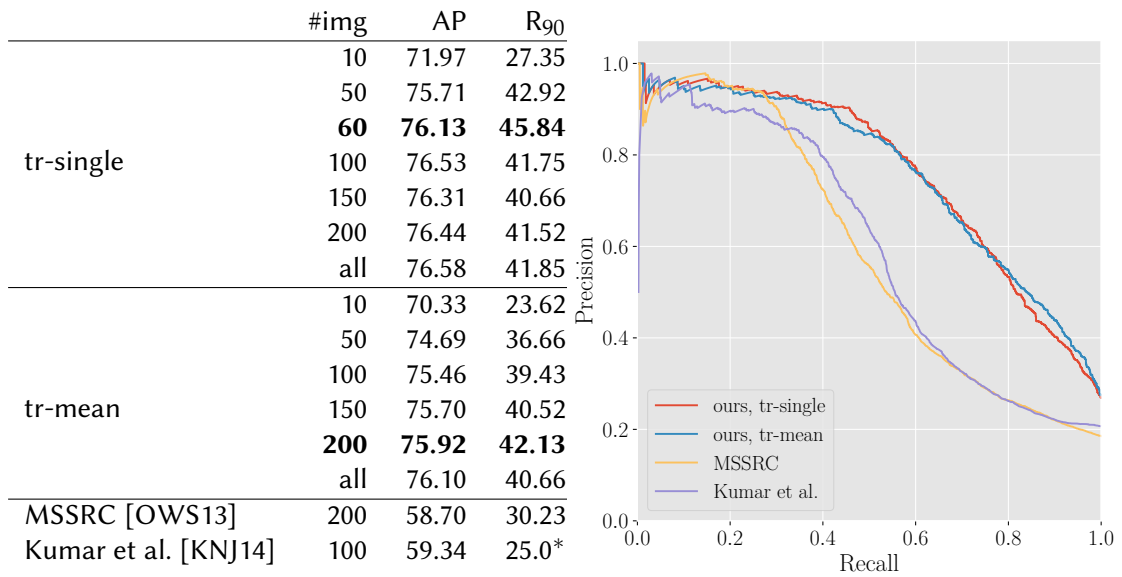


Figure 6.19: Left: Average precision (AP) and recall at 90% precision (R_{90}) of the face identification algorithm for different maximum numbers of images per character (#img) and the baseline. Right: Corresponding precision-recall curves for the best results (marked bold). Values marked with * have been estimated from the figures of the original papers.

Face Identification

For face identification, the same setup as used by the authors of the *Movie Trailers Face Dataset* [OWS13] is used for reporting the precision and recall of the system. As explained in Section 6.2.2 we use the *PublicFig+10* dataset to create the person dictionary D_P as the input. Similar to Ortiz et al. [OWS13] we capped the maximum number of images per person in D_P to avoid skewing the results towards people with more images and to find a good tradeoff between accuracy and computation time. To analyze the influence of the amount of images, we conducted the experiments with a maximum of 10, 50, 100, 150, 200 images as well as with all available images of a person. The results of our approach with the track representations *tr-single* and *tr-mean* together with the baselines *Mean Sequence Sparse Representation-based Classification* (MSSRC) of Ortiz et al. [OWS13] and Kumar et al. [KNJ14] are shown in Figure 6.19.

It is evident that our proposed approach clearly outperforms these baselines in both average precision and recall at 90% precision. Another observation is that there is no significant improvement for a maximum number of images greater than 50. Comparing the results of *tr-single* and *tr-mean*, no difference in the average precision is visible, but the recall is slightly better for *tr-single*. Based on these results, we decided to use a maximum number of 50 images per person and to use the faster, second version of the algorithm due to the insignificant difference in the performance.

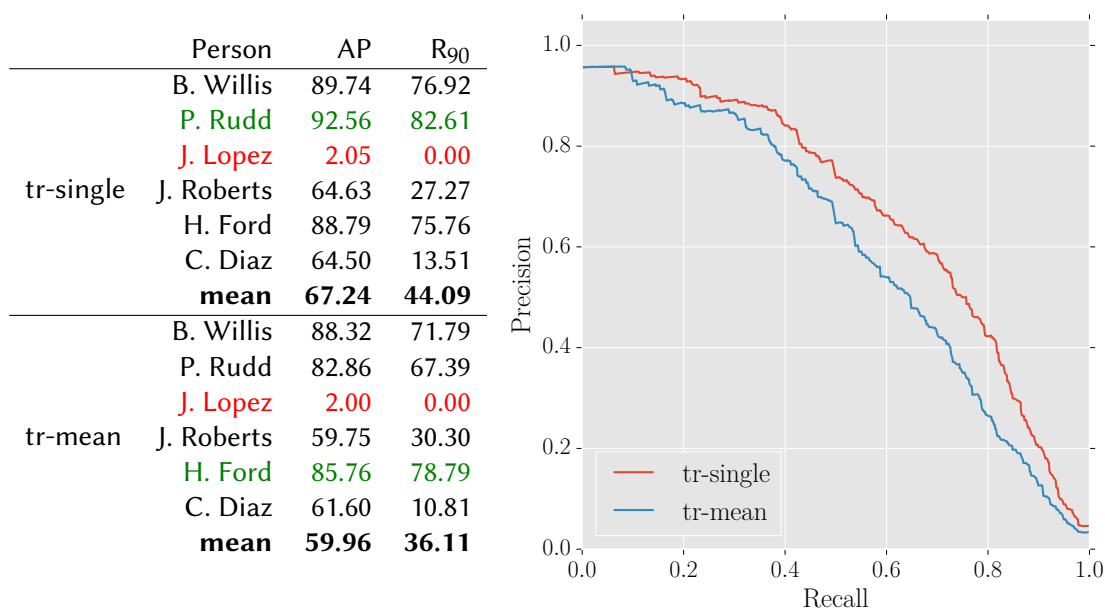


Figure 6.20: Left: Average precision (AP) and recall at 90% precision (R_{90}) for six exemplary characters (best: green, worst: red) as well as the mean AP (MAP) and mean R_{90} for the face retrieval algorithm. Right: Precision-recall curve averaged over all 23 queries.

Face Retrieval

To evaluate the performance of the single-image based face retrieval method described in Section 6.2.2, we manually selected one image from the *PublicFig+10* dataset for each person which also occurs in the *Movie Trailers Face Dataset*. We ended up selecting 85 images to measure precision and recall for each query. The results are shown in Figure 6.20.

The figure shows that *tr-single* that uses every single image of a face track for the comparison with the query image yields better results. This can be explained by variations like pose and expression which will more likely be matched by comparing the query image with every image in the face track separately. By averaging the feature vectors in the track with *tr-mean* this descriptive information can get lost, so we dismissed it for our final framework. An example output of our web application can be seen in Figure 6.21.

Face Clustering

Comparing results of face clustering algorithms is a difficult task, since there are no datasets designed solely for this purpose. We decided to again use the *Movie Trailer Face Database* for this purpose with some augmentations. First, to find videos that are useful for a clustering algorithm, we only consider trailers with at least two known actors who occurred in at least two different tracks. Furthermore, the majority of the face tracks in the data show unknown persons who we could not cluster. The reason is that the *Unknown* label describes more than one individual in contrast to the other labels. Thus, in our evaluation we omit clusters where

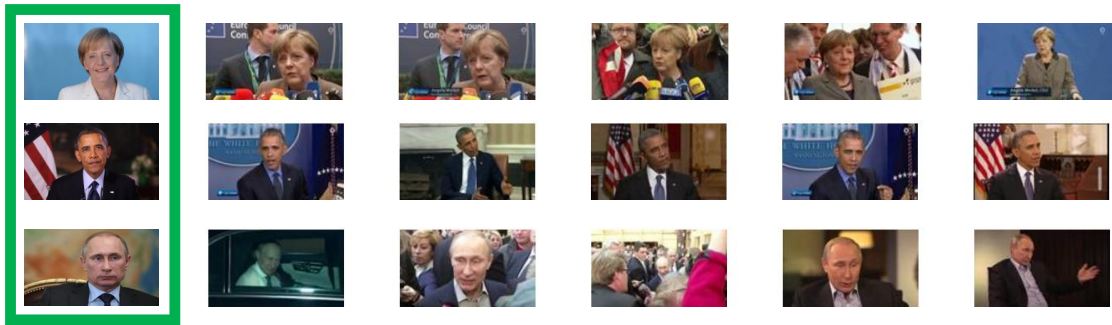
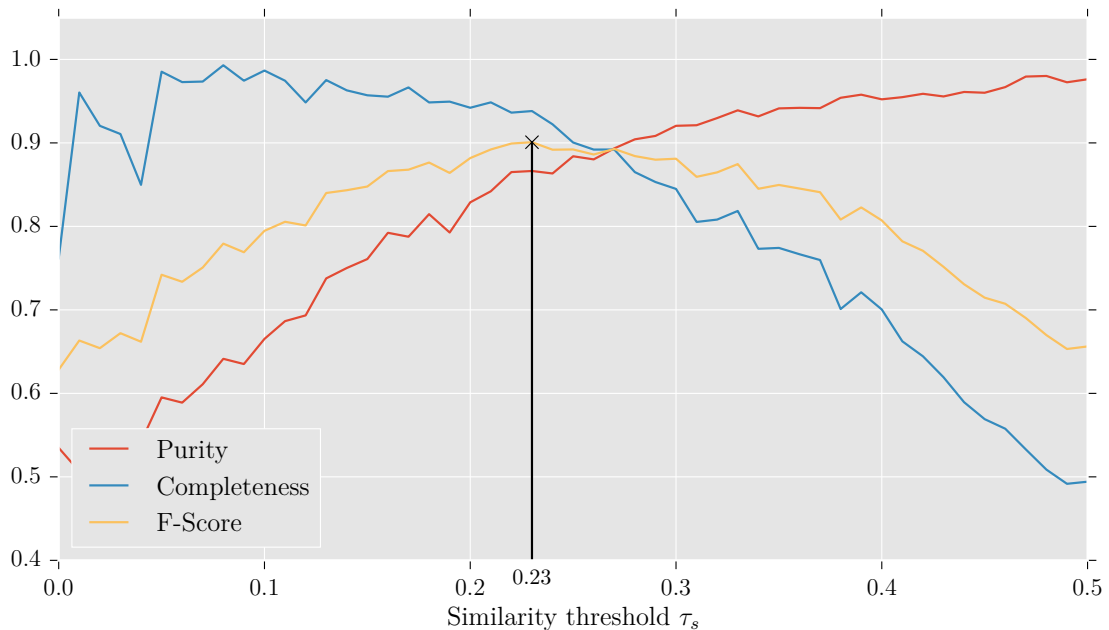


Figure 6.21: Person retrieval results.

Figure 6.22: Purity (blue), completeness (green) and f-score (yellow) for different thresholds τ_s .

unknown persons are the majority of tracks. The *purity* score [Son08] of a cluster will be punished if unknown labels occur in it. The purity of a cluster is the number of face tracks of the most occurring person divided by the size of the cluster. The sum of purities of all clusters divided by the number of clusters gives the overall purity that we report. Nevertheless, a clustering result dividing every face track in its own cluster would yield a perfect purity of 1. Hence, a second score is necessary, the so called *completeness*. It compares the number of face tracks of the most occurring person to its total count of face tracks in the movie trailer, yielding only a perfect result if all face tracks of an individual belong to the same cluster. Selecting a threshold τ_s for the agglomerative clustering algorithm is one of the key issues in data clustering, since it highly depends on the quality of the given face images. We evaluated a range of possible thresholds with their corresponding purity and completeness results in Figure 6.22.


The two contradicting scores purity and completeness are combined via the f-score [OWJ16]. It allows us to find an optimal value of $\tau_S = 0.23$ for our data. Since our face detector provides more face images, especially in non-frontal poses, due to a higher recall and precision compared to the one used by Ortiz et al. [OWS13], we have to deal with more face variations. For this reason, we increased τ_S to a value of 0.4 in our final system to enhance cluster purity while still maintaining good completeness.

6.2.5 Summary


Digitization has fundamentally changed the workflow of professional media production, but the content-based labeling of image sequences and video footage is still performed manually and thus quite time-consuming. In this section, an automatic content-based labeling system, necessary for all subsequent stages of film and television production, archival or marketing, has been presented. Novel deep learning algorithms for visual concept detection, similarity search, face detection, face recognition and face clustering are combined in a multimedia tool for efficient video inspection and retrieval. A new visual concept lexicon and novel visualization components have been developed to support media production activities. Furthermore, a novel multi-task learning CNN for concurrent concept detection and similarity search has been introduced. Experimental results have demonstrated the excellent quality of the proposed approaches. The 58 concepts are detected with a mean average precision of approximately 90% on the top-100 video shots, the similarity search results even achieve 93.5% mean AP on the top-100 video shots for 21 sample query images. The face recognition approach clearly outperforms the baseline system on the public Movie Trailers Face Dataset.

GoVideo


Semantic Search
in Videos



Philipps
Universität
Marburg



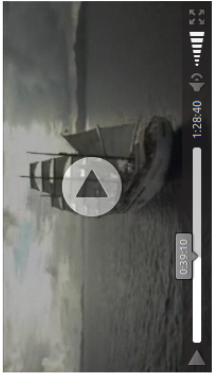
taglicht media



Ernst-Abbe-Hochschule Jena
University of Applied Sciences

GERMAN RESEARCH
UNIVERSITY ALLIANCE
DFG

▶ Videanalysis (0 processing)



▼ Concept search

Select concept

Search [extended search](#)


























▼ Person search

Select person

Search

▼ Similarity search

Search Persons

 0.92	 0.91	 0.91	 0.91	 0.9
 0.92	 0.91	 0.91	 0.91	 0.91
 0.92	 0.92	 0.91	 0.91	 0.91
 0.93	 0.92	 0.91	 0.91	 0.91
 1.0	 0.92	 0.91	 0.91	 0.91

1 of 1470

1 2 3 4 5 6 7 8 9 10

25

Figure 6.23: The image shows the similarity search results for the keyframe displayed in the video player. The retrieval results are presented to the user in the form of a ranked list of video shots where each video shot is represented by the video frame with the highest probability of showing the requested content.

6.3 Visual Information Retrieval in Video Archives

6.3.1 Introduction

Automatic content-based video retrieval methods relying on deep learning, in particular deep convolutional neural networks (CNN), have great potential to unveil the hidden treasures lying in large digital video archives around the world. A particularly interesting video archive is the German Broadcasting Archive (Deutsches Rundfunkarchiv, DRA¹⁴). It maintains the cultural heritage of radio and television broadcasts of the former German Democratic Republic (GDR). The DRA preserves, among other things, the video collection of GDR television recordings from the years 1952 to 1991. Up to now, more than 90% of the historical GDR television recordings (about 34,000 hours of video footage) are digitized. The archive includes a total of around 100,000 broadcasts, such as contributions and recordings of the daily news program *Aktuelle Kamera*, political magazines such as *Prisma* or *Der schwarze Kanal*, broadcaster's own TV productions including numerous films, film adaptations and TV series productions such as *Polizeiruf 110*, entertainment programs (e.g., *Ein Kessel Buntes*), children and youth programs (fairy tales, *Elf 99*) as well as advice and sports programs. Access to the archive is granted to scientific, educational and cultural institutions, to public service broadcasting companies and, to a limited extent, to commercial organizations and private persons. The uniqueness and importance of the archive has sparked considerable international research interest in GDR and German-German history.

International scientists use the DRA for their research in the fields of psychology, media, social, political or cultural science. These studies are, for example: *Heavies in East Germany* (Humboldt University Berlin), *Space Travel in the GDR* (Harvard University, USA), *The Jewish in TV* (Ludwig Maximilian University of Munich, Germany), *Socialism on the Screen* (Loughborough University, UK), *Self-made in Consumer Society* (University of Mannheim, Germany), or *Child and Youth Education in Fictional Subjects* (Shizuoka University, Japan). Furthermore, DRA video footage is often used in film and multimedia productions.

The DRA is answering a wide range of research, cultural, private, and commercial requests concerning the life of GDR citizens and social perceptions. The number of comprehensive and time-consuming requests is considerably increasing, e.g., youth fashion in the GDR, youth cultures, personalities (e.g., *Angela Merkel* or *Joachim Gauck*), sports, music events, sights (e.g., *Sanssouci* in Potsdam or the *Brandenburg Gate*), living in East Germany, in particular home furnishings or the socialist city as a model of urban development in the GDR, specifically pictures including socialist classicism, buildings made with precast concrete slabs, demolition and spectacular buildings. Besides video recordings, the DRA collection contains numerous photographs of unidentified persons. Similarity search in the overall archive can provide important information about the identity of the person. Furthermore, copyright clearance is an important task of the DRA.

The following four use cases can be derived from the user studies and search requests described above: (i) visual concept classification for recurring, frequent terms in search queries, (ii) person recognition for famous personalities, and (iii) image similarity search as well as (iv) person

¹⁴<https://www.dra.de>

similarity search for queries by example, e.g., to find images with similar visual content (possibly for licensing or copyright reasons) or a person's identity in an image.

In this section, we present deep learning methods for visual concept classification, person recognition, image similarity search, and person similarity search to support these use cases. The visual quality of the DRA video footage is quite heterogeneous - the collection includes color and gray-scale material, low-resolution videos, and even some screen recordings. These requirements and the GDR specific visual concepts pose challenges for generic video mining solutions, such as the Microsoft Video Indexer¹⁵. The retrieval methods, especially the concept and person recognition models, need to be adapted to the target domain and continuously updated to the archive users' changing search requirements. Therefore, the proposed methods are integrated into a client-server application, called VIVA, that supports all steps from the acquisition of training material, the annotation of individual images, to the training and management of deep learning models. The goal is to enable archivists to carry out the entire training process with a user-friendly interface and to add or update new personalities or visual concepts to the retrieval process when new requirements arise.

To summarize, we make the following contributions:

- We present VIVA, a software tool based on deep learning methods for visual information retrieval in videos. To address potentially changing user requirements, VIVA enables users to build and adapt deep learning models for visual concepts and persons with a user-friendly interface.
- We present an efficient workflow for collecting training data and building classification models for concepts and persons with low manual effort. Based on web crawling and similarity search results, large amounts of training data can be collected quickly. The (re)training process can be started by just one click to make deep learning user-friendly. Furthermore, user feedback on the retrieval results for the entire data collection can be integrated. The workflow is generally repeated several times to continuously improve the concept or person models.
- Due to the time-consuming task of completely labeling all training images in a multi-label scenario, we present a cross-entropy loss function for incomplete multi-label ground-truth data.
- We use VIVA to build deep learning models for 91 GDR specific concepts and 98 personalities from the former GDR, based on internal and external DRA user requests.
- We evaluate VIVA for the four use cases (concept classification, person recognition, image similarity search, and person similarity search) in the context of the DRA using a unique historical collection of about 34,000 hours of GDR television recordings.

VIVA is published under an open source license at <https://github.com/umr-ds/VIVA> and <https://github.com/TIBHannover/VIVA>.

Parts of this section have been published in: Markus Mühling, Nikolaus Korfhage, Kader Pusturen, Joanna Bars, Mario Knapp, Hicham Bellafkir, Markus Vogelbacher, Daniel Schneider,

¹⁵<https://azure.microsoft.com/de-de/services/media-services/video-indexer>

Angelika Hörth, Ralph Ewerth, and Bernd Freisleben. “VIVA: Visual Information Retrieval in Video Archives.” in: *International Journal on Digital Libraries* 23.4 (2022), pp. 319–333.

6.3.2 Related Work

To the best of our knowledge, a comparable video retrieval approach that integrates the training of deep learning models to enable the search for custom concepts and persons in a single platform does not exist. However, there are several tools related to individual components of VIVA.

An important part of the presented workflow is the data acquisition step. Various open source projects exist (e.g., LabelMe [Rus+08]) for labeling image or video datasets. While in some tools it is possible to upload pre-annotated images or apply an available deep learning model to support the labeling process, this is usually cumbersome and requires additional manual effort. In contrast, VIVA provides semi-automatic data acquisition steps so that users can focus on collecting, labeling, and reviewing images and keyframes. Another important step in the pipeline is the training of deep learning models. Existing GUI-based tools (such as Nvidia DIGITS [Yea+15]) require expert knowledge to conduct the training process to build CNN-based models. In VIVA, training deep learning models is simplified as much as possible so that even inexperienced users are able to build, run, and monitor deep neural networks for visual concept classification, person recognition, and image similarity search on GPUs.

Furthermore, video retrieval results generated using these deep learning models are also visualized in VIVA. There are several comparable interactive frame-based video retrieval engines that are similar to VIVA [Lok+19]. Tools such as SOMHunter [Kra+20], VERGE [And+21], Visione [Ama+21], vitrivr [GRS19], VIREO [Ngu+20] combine different automatic content analysis methods to facilitate search with different feature modalities. In particular, these tools support searching for objects, concepts or actions based on pre-defined classes and models built with publicly available data sets, such as ImageNet [Den+09] or OpenImages [Kuz+20]. Recently, a library for deep learning for unsupervised image retrieval called PyRetri [Hu+20] has been introduced. It offers methods for important retrieval steps such as feature extraction, indexing, and evaluation using off-the-shelf deep learning models. However, in contrast to these works, VIVA additionally supports the training of individually defined concepts and persons. This can be a crucial feature when it comes to historical or domain-specific video content. In this context, current content-based retrieval tools might not be sufficient for the search intent of an archive user as described in the considered case studies.

6.3.3 Image and Video Retrieval Approaches

The fundamental problem of content-based image and video retrieval is to overcome the discrepancy between the extracted features and the human interpretation of the (audio-)visual data. In the literature, this discrepancy is also known as the “semantic gap” [Sme+00b].

Nowadays, deep learning approaches, in particular deep convolutional neural networks, are applied to almost all computer vision tasks to reduce the semantic gap and even surpass human performance in many areas such as face or concept recognition [He+15b; Pus+19].

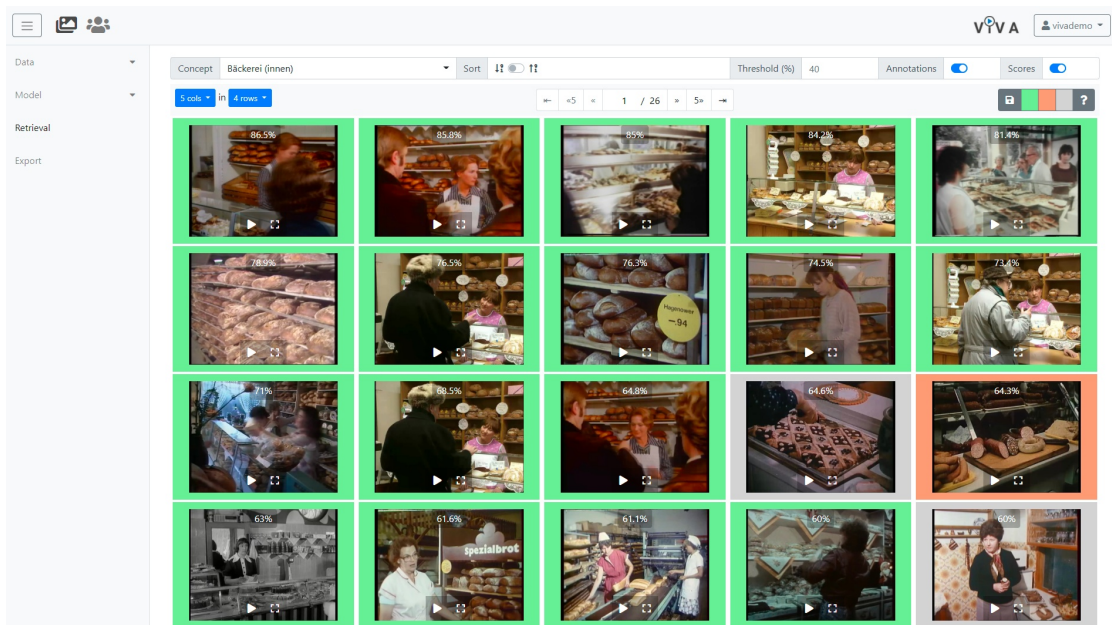


Figure 6.24: Retrieval results and user feedback for the visual concept “bakery”. A user highlights correct results in green, wrong results in orange, and unclear/neutral results in grey.

In the following sections, we present our image and video retrieval approaches for the previously described use cases. The proposed visual concept classification method is described in Section 6.3.3. In Section 6.3.3, the similarity search approach is introduced. Finally, the person identification and person similarity search approach are presented in Section 6.3.3 and 6.3.3, respectively. As a preprocessing step, a shot segmentation approach [Müh+07; Müh+19] is applied to the videos to find representative keyframes used for the subsequent content-based video analysis algorithms. The resulting metadata of the concept and person recognition approaches are written to a database, including tags and the corresponding probabilities. Given this semantic index, search queries for persons and concepts can be processed efficiently and the query results are returned to the user as a retrieval list, ranked according to the probability of the desired content being present.

Visual Concept Classification

Visual concept classification is a challenging task due to the large variability and complexity of the possible concepts. To support a wide range of user requests, the DRA conducted a requirements analysis. The definition of the GDR specific concept lexicon is based on the analysis of past user search queries with a focus on queries that were experienced as difficult and time-consuming to answer manually. Considering the utility or usefulness for (future) search queries, the observability by humans and the feasibility in the sense of automatic detection, a lexicon of 91 GDR specific concepts has been defined, including objects, sites, scenes, events, and activities, such as “Brandenburg Gate”, “Elbe panorama”, “baby”, “bakery (inside)” or “christmas”. The complete list of concepts is presented in Table 6.4.

Our approach to visual concept classification relies on the EfficientNet architecture [TL19] that achieves state-of-the-art visual recognition performance. Currently, automatically designed neural network architectures outperform hand-crafted CNN architectures for content-based image recognition [EMH19; Liu+18]. The EfficientNet architecture is the result of a neural architecture search approach that uses a new compound model scaling method and optimizes recognition quality as well as efficiency. As a tradeoff between quality and efficiency, we use the B3 variant of the EfficientNet architecture as our base model. Since the training of a deep CNN model requires millions of training images, the parameters of the base model are initialized using pre-trained weights from the ImageNet dataset [Den+09] and adapted to the target domain using a fine-tuning strategy.

Most network architectures for image recognition consider only a single concept per image (“single-label”) [Sze+15; He+16]. In contrast, the real world task of visual concept classification is a multi-label problem. However, a complete annotation of the training samples is impractical, because it is quite time-consuming to specify for each concept whether it occurs in an image or not. The manual effort for annotating an image or keyframe increases linearly with the number of concepts. Therefore, we are dealing with incomplete multi-label data.

In the data acquisition phase using a web crawler or our similarity search component, mainly positive examples of a concept are collected. Negative labels are added via review and user feedback. Thus, each image in the training set is labeled for at least one concept. Since in some cases there are very few negative training samples, especially when a deep neural model is built for the first time, positive training samples for a concept are used as weak negative samples for the other concepts.

Hence, the proposed network architecture uses an optimized cross-entropy loss with a new weighting scheme. The weighting scheme consists of two components. First, instead of ignoring training samples in the loss function for unlabeled concepts, they are included as negative examples with a lower weighting factor compared to the actually labeled samples. Second, a modulating factor motivated by the success of focal loss [Lin+17b] in the field of object detection is introduced to emphasize difficult samples during the training process. The overall loss is defined as follows:

$$L = \sum_{k=1}^K l(y_k, p_k), \quad (6.4)$$

with

$$l(y, p) = \begin{cases} -\alpha_{pos}(1-p)^\gamma \log(p) & \text{if } y \text{ is positive} \\ -\alpha_{weak} p^\gamma \log(1-p) & \text{if } y \text{ is weakly negative} \\ -\alpha_{neg} p^\gamma \log(1-p) & \text{if } y \text{ is negative} \end{cases}$$

where K is the number of concepts, p_k is the predicted probability for the k -th concept, y_k is the k -th ground truth label, α_{pos} is the weighting factor for positive labels, α_{weak} for weak negative or undefined labels, α_{neg} for negative labels and γ is the focusing parameter.

To further minimize the manual effort for labeling training samples, a deep neural network model in VIVA is improved iteratively. Starting from an initial set of training samples, a baseline

6 Image Similarity Search in Applications

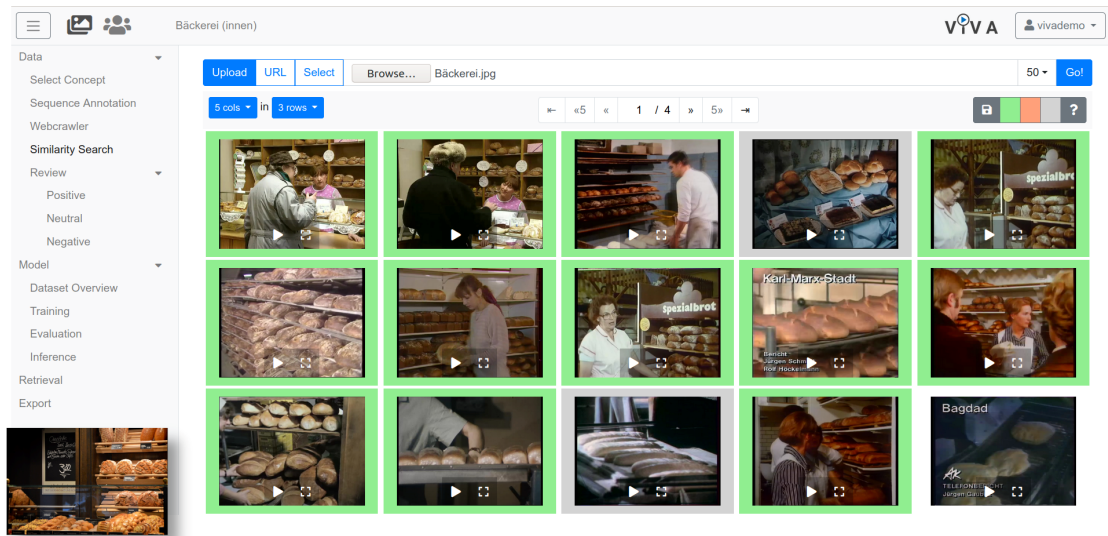


Figure 6.25: VIVA GUI showing similarity search results for a query image of a “bakery”. The query image is downloaded using a web crawler and displayed in the bottom-left corner of the figure.

model is trained. This model is applied to the entire video collection to find keyframes that contain the desired concepts. These results are presented to the user in form of a retrieval list to collect user feedback. The user feedback annotations are added to the training set and thus lead to iterative improvements. Figure 6.24 shows an example of the concept “bakery”.

Image Similarity Search

Query-by-image is a popular strategy to represent a user’s search intention in content-based image and video retrieval scenarios [KMF20]. It is useful in many use cases of the DRA from searching for specific scenes, detecting duplicates, and clearing copyrights to supporting data acquisition for concept recognition.

Since CNN features are high-dimensional float vectors and linearly searching in large databases like the video collection of the DRA is time-consuming, several large-scale similarity search approaches focus on learning compact representations [Wan+15; RCC20; Wan+17] or use product quantization approaches [JDS10; JDJ19; Wan+21]. To perform fast and accurate image similarity search on the entire collection of more than 15.7 million keyframes, a method based on deep hashing is used [KMF21]. Deep hashing methods learn a binary representation of images to facilitate fast distance computations in Hamming space. Furthermore, the distance computation complexity is reduced by multi-index hashing [NPF12].

The training of the deep hashing model consists of two phases that both use ADAM [KB15] as the optimization method because it automatically adapts the learning rate. In the first phase, an EfficientNetB3 model pre-trained on ImageNet is trained on a dataset with a high number of different classes to obtain a fine-grained embedding with high generalizability to a wide range of query images. The dataset used contains all ImageNet classes with more than 1,000 training

images and all classes from the Places2 [Zho+17] dataset, which results in a total number of 5,390 classes. The model is fine-tuned to this dataset using Softmax with cross-entropy loss. After training the weights of the last layer with a learning rate of 0.01 for two epochs, all layers are trained for another 16 epochs with an initial learning rate of 0.0001.

In the second phase, the weights of the classification model from the first phase are used to initialize the deep hashing model. This model includes an additional coding layer prior to the classification layer with a *tanh* activation and 256 outputs to learn the embedding. It is trained on the same dataset as before, however by combining cross-entropy loss on the output and hard triplet loss [SKP15b] on the coding layer. This model is trained for 5 epochs with a learning rate of 0.0001. The parameters for the initial learning rates and the number of epochs resulted from observation of the training process.

Due to the technically challenging historical video collection, data augmentation is used in both phases to improve the robustness of the similarity search approach. For example, brightness modification, mirroring, cropping, zooming, rotation, width/height shifting, and JPEG compression artifacts are used as augmentation methods.

At this stage, the model generates 256-bit codes. However, using codes of this length in a linear search on the entire image corpus is too expensive. For this reason, we implemented a two-stage method, where we first use 64-bit codes for a coarse search and then the 256-bit codes for re-ranking the candidate retrieval list. The first step involves multi-index hashing in Hamming space [NPF12]. To extract the 64 most important bits from the 256-bit codes, we first partition the 256-bit codes into four partitions by applying the Kernighan-Lin algorithm [KL70] to bit correlations. From each of the decorrelated partitions, we take the first 16 bits to compose the 64-bit codes.

The multi-index hashing approach is integrated into ElasticSearch¹⁶. The time for obtaining an answer for a single query image in ElasticSearch in a corpus of roughly 10 million images is about 256 milliseconds on an Intel Core i7-4771 processor with 3.5 GHz clock speed. While there are approaches to similarity search that build on inverted indexes [Ama+17], the integration in ElasticSearch allows us to use its benefits, such as processing multi-modal queries (text and image), load balancing, and scalability.

Figure 6.25 shows the VIVA GUI with similarity search results for a query image of a “bakery”.

Person Recognition

A requirements analysis based on user queries has been conducted to define a lexicon of GDR personalities for the task of person identification. Considering different areas of interest such as politics, sports or entertainment, the final lexicon includes 98 persons, such as “Angela Merkel” or “Erich Honecker”.

The task of face recognition is typically tackled by learning discriminative face representations [SKP15b] using large scale face datasets [Cao+18a; Guo+16]. Recent methods introduce new loss functions that enhance the discriminative power for large scale face recognition [Den+19a; Wan+18a].

¹⁶<https://www.elastic.co>

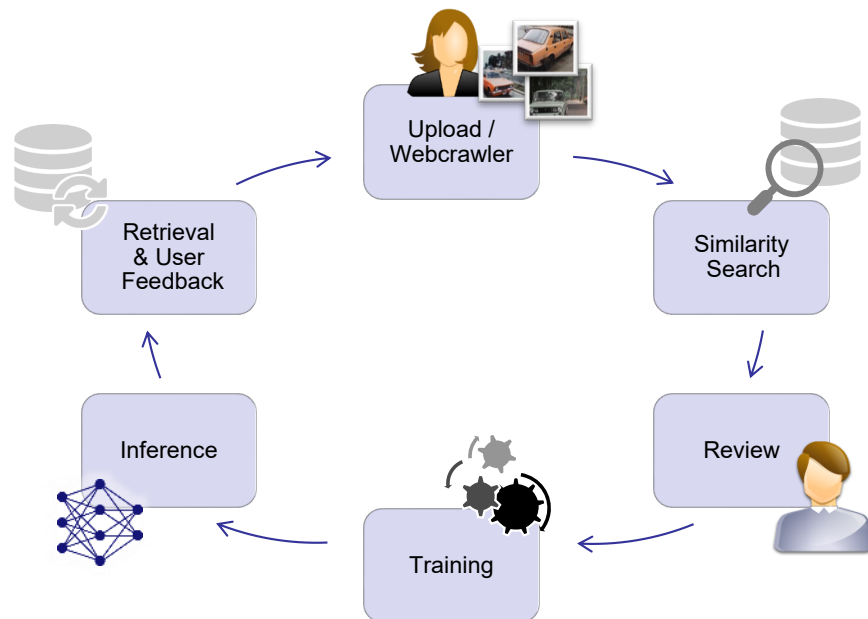


Figure 6.26: VIVA workflow.

Based on the predefined lexicon, our person identification approach includes several steps. The face processing steps consist of the following components: face detection, face alignment, feature extraction, and face recognition. To initially detect faces in the images, the RetinaFace [Den+19b] detector is used. A subsequent frontal alignment of the detected faces using the shape predictor of Dlib [Kin09] ensures that detected faces are normalized in their pose. To reduce annotation overhead and eliminate unwanted (background) persons in images, detected faces in the training material are grouped using agglomerative clustering. For each person to be trained, the annotation effort is limited to 100 clusters each. Training face images of the lexicon personalities are represented by feature vectors that are extracted using FaceNet [SKP15b]. The used FaceNet implementation¹⁷ is trained on a dataset (VGGFace2 [Cao+18a]) with over 9,000 identities and 3.3 million images and achieves a high accuracy of 99.65% on the established LFW (Labeled Faces in the Wild) benchmark. For the final identification of the personalities, a classifier is trained using Support Vector Machines (SVM), for which a subsampling strategy is applied in order to keep training faces for each class balanced.

Person Similarity Search

Similarity search for persons allows users to display similar persons from the database based on a query image that portrays a unique person to be searched. The fundamental difference to concept-based image search is that similarities are evaluated exclusively between detected faces in images. In this way, unknown persons in photographs, for example, can be identified by searching them in the entire DRA collection. The context or existing annotations of the found images or video shots can provide important information about the person's identity. To

¹⁷<https://github.com/davidsandberg/facenet>

preprocess the dataset to be indexed and subsequently the query images, face detection, face alignment, and feature extraction steps described in Section 6.3.3 are performed. Based on the extracted feature representations, an index using the FAISS library [JDJ19] is built. The index is based on product quantization [JDS10] and allows efficient comparisons between query vectors and stored vectors based on cosine similarity and returns nearest neighbors.

6.3.4 VIVA

The approaches presented in the previous section are integrated into a client-server application, called VIVA, that provides all steps from collecting training samples, through the annotation of individual images or keyframes, to training the classification models as well as visualizing the results and collecting user feedback. In a preprocessing step, a shot segmentation approach and a keyframe extractor are applied to the target video collection, in our case the GDR television recordings from the DRA. Once the preprocessing is done and the database of VIVA is initialized with these videos, the user can build concept or person models as well as search and index the video collection.

The goal is to enable non-expert users to carry out the entire training process. For this purpose, the application offers an intuitive graphical user interface (GUI) to learn models for new personalities or concepts. Efficient workflows are provided in the GUI to train and continuously improve concept and person recognition models with as little manual effort as possible. We present the GUI, the implemented workflows, and the software architecture of VIVA below.

Since the concept and person recognition workflows involve similar but not identical steps, VIVA is separated into two applications. Users can simply switch between the concept and person application via intuitive buttons in the left upper corner of the main window. In both cases, the workflow is divided into three main stages: the data collection and annotation stage, the training and evaluation stage, and the retrieval and user feedback stage. Furthermore, VIVA allows the export of the inference results. The detailed steps of the three stages of the workflow are shown in Figure 6.26.

The individual steps can be accessed via a menu structure on the left side of the application window. Users can hide the menu structure to display more keyframes or images, which is especially helpful in the data acquisition phase. The different steps of the workflow are described in more detail in the following subsections.

Data Collection and Annotation

In the data collection and annotation stage, new training images or keyframes are acquired semi-automatically for each concept. For this purpose, a new class (concept or person) has to be defined or an existing class in the overview table has to be selected. The selected concept is displayed at the top of the application window. The overview table also shows the number of positive, negative, and neutral annotations per concept as well as a description of the concept. Besides positive and negative labels, difficult or indistinct samples can be labeled as neutral. Classes can be added, removed or renamed at the overview page.

6 Image Similarity Search in Applications



Figure 6.27: VIVA GUI for starting and monitoring the training process.

A central element for labeling images or keyframes and displaying search results is the grid view component. The size and number of images per page in the grid can be optimized to the screen resolution and user preferences by adjusting the number of rows and columns. There are buttons to navigate between the pages. Within the grid, images can be efficiently labeled as positive, negative, or neutral. Additionally, there are buttons to label all images on a page at the same time with the same label. Keyboard shortcuts exist to improve the efficiency. Furthermore, individual images can be displayed in full screen. In the case of keyframes, the corresponding video shot can also be played.

VIVA provides several options to acquire training samples semi-automatically for the selected concept. A web crawler allows users to collect training images from the WWW using search engines like Google and Bing. Users only have to enter a search term. The resulting images of the web search are displayed in the grid view and can be labeled for further usage. Only labeled images are downloaded from the web and stored in the database. Using the web crawler, hundreds of training samples are acquired quickly. However, to fit the model to the target domain, training samples from the same distribution are required. Therefore, the positive training samples from the web search are important because they can be used as similarity search queries to quickly find new specific training samples from the target domain. Figure 6.25 shows an example image from the WWW and the corresponding domain specific similarity search results. Again, the search results are labeled directly in the grid component. Besides the selection of query images from the already labeled positive samples, query images can be also submitted by URL or uploaded from an image file.

Furthermore, a sequence import allows the upload of labeled video sequences from external tools. In our case, the imported sequences are search results from the DRA search engine (FESAD interface) based on human generated metadata. FESAD is the common television archive database management system of the public German broadcasting association ARD.

At the end of the data acquisition phase, the review page gives an overview of the labeled positive, negative, and neutral training samples and offers the possibility to correct them.

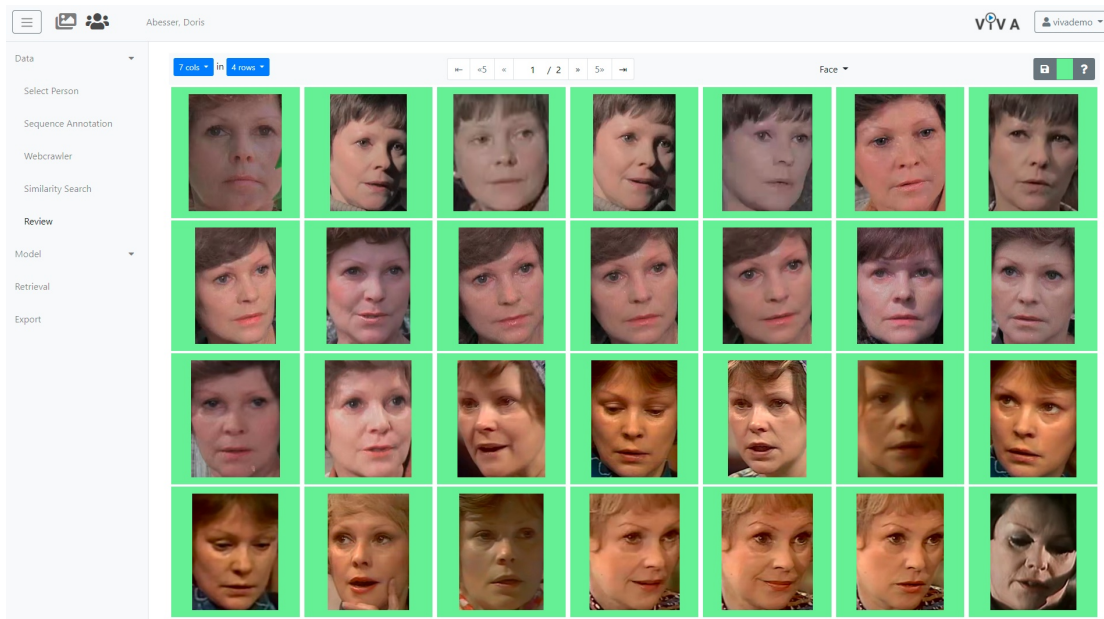


Figure 6.28: User review of the labelled faces of the person “Doris Abesser”; positive labels are marked in green color.

Training and Evaluation

While the data collection and annotation stage is related to a specific concept, the training and evaluation stage is related to all concepts. Deep neural models are trained, evaluated, and inferred based on all concepts. The training and evaluation stage is separated into four steps represented by corresponding menu items and pages. The first step shows a dataset overview. A minimum of 100 positive training samples is required for using a concept in the training process. This threshold only provides an initial indication of the number of training samples required. Of course, the number depends on the complexity of the underlying concept and the diversity of the collected training data. If necessary, the training data can be extended from training iteration to training iteration.

The overview page visualizes the number of positive and negative training samples per concept in a bar chart. The concepts can be ordered by the number of positive samples, and the concepts not meeting the requirements are highlighted. The user gets a quick overview of concepts that require further annotation effort. The training data should cover the possible variations of the underlying concept. The more data is collected and the greater the data diversity, the lower the risk of overfitting and the better the generalization capabilities of the model can be estimated.

In the second step, the training process is started. The training page allows non-expert users in the field of deep learning to build models with minimal configuration effort (see Figure 6.27). At the top of the page, available GPU resources are listed. After selecting the GPU resources, a user only needs to press the start button. The images for the training and validation set are automatically sampled from the labeled instances, as described in Section 6.3.5. Learning

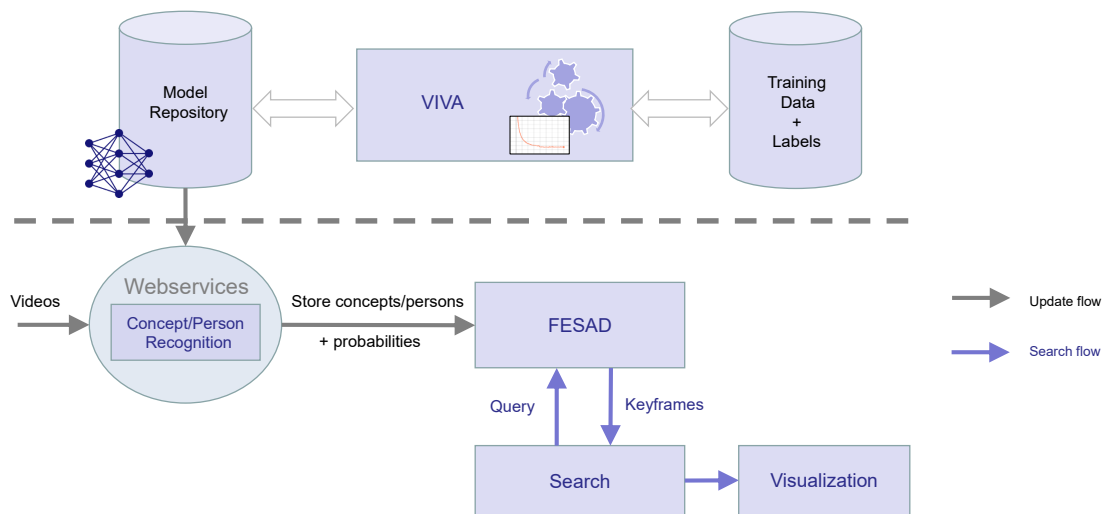


Figure 6.29: VIVA Software Architecture and Integration into FESAD.

curves visualizing the loss on the training set and the performance in terms of mean average precision on the validation set are used for monitoring the learning progress. Further available monitoring tools are a progress bar, a learning rate chart, a window showing the log content as well as GPU and CPU utilization charts.

Once the training is successfully completed, detailed results on the validation set can be inspected and compared to the previous model at the evaluation page. If the user is satisfied with the quality of the results, the model can be applied to the entire video collection at the inference page by just selecting the GPU resources and pressing the start button. Progress, CPU, and GPU utilization are again displayed for monitoring purposes.

Retrieval and User Feedback

Finally, the retrieval and user feedback stage is used to show the retrieval results and collect user feedback to improve the accuracy of the model in the next training phase. Figure 6.24, for example, shows the retrieval results for the visual concept “bakery” together with the user feedback annotations.

Concepts and Persons

While the person and concept recognition components follow the same main stages for automatic footage indexing, there are some differences. A key difference for persons is that more fine-grained annotations of faces are made in the images. Therefore, in the data acquisition steps, face detection must be performed in the background before users can annotate relevant (positive) faces (see Figure 6.28). Similarly, for the training and inference steps, face detection and/or feature extraction are performed before the model is trained or can be used for classification. Further variations are described in Section 6.3.3.

Software Architecture

VIVA is realized as a client-server application using the Python web framework Django¹⁸. Figure 6.29 shows the software architecture as well as the integration of VIVA into FESAD. The service-oriented architecture includes a database containing training data and labels, a deep neural model repository, and web services for providing visual concept classification and person recognition. The final neural models are automatically exported, optimized, and deployed as web services using TensorFlow Serving¹⁹. The web services for concept classification and person recognition can be used to integrate VIVA functionality into FESAD.

Further components (not shown in the figure) are a backup service and web services for similarity search and person similarity search.

Individual Docker²⁰ containers are used to deploy the different components of VIVA. This containerization strategy improves the portability of the application and allows the distribution of the software components to different resources. VIVA can also be deployed in Kubernetes²¹ environments.

Since there are typically multiple users involved, VIVA provides, besides data acquisition, annotation, training, and retrieval capabilities, support for user and role management. Different users can be configured, and the role management allows the assignment of rights for collecting and annotating training images or keyframes, for training and inference, as well as for the administration of users.

6.3.5 Experimental Results

In this section, the VIVA tool is evaluated based on the use cases of the content-based video retrieval approaches using the video collection of historical GDR television recordings.

The results are evaluated using the average precision (AP) score that is the most commonly used quality measure for retrieval results. The AP score is calculated from a list of ranked documents as follows:

$$AP(\rho) = \frac{1}{|R \cap \rho^N|} \sum_{k=1}^N \frac{|R \cap \rho^k|}{k} \psi(i_k), \quad (6.5)$$

$$\text{with } \psi(i_k) = \begin{cases} 1 & \text{if } i_k \in R \\ 0 & \text{otherwise} \end{cases}$$

where N is the length of the ranked document list, $\rho^k = \{i_1, i_2, \dots, i_k\}$ is the ranked document list up to rank k , R is the set of relevant documents, $|R \cap \rho^k|$ is the number of relevant documents in the top- k of ρ and $\psi(i_k)$ is the relevance function. Generally speaking, AP is the

¹⁸<https://www.djangoproject.com/>

¹⁹<https://github.com/tensorflow/serving>

²⁰<https://www.docker.com/>

²¹<https://kubernetes.io/>

Concept	Approach			Concept	Approach		
	A	B	C		A	B	C
Advertising column	89.26	92.25	92.15	Meissen Cathedral (Meissen)	93.02	93.65	94.25
Autumn	59.53	73.25	76.19	Mini skirt/dress	33.98	38.49	42.80
Baby	46.31	73.81	77.70	Monument statue	28.97	65.72	65.72
Baby carriage	81.22	81.68	81.15	Mop top hairstyle	56.29	70.99	69.03
Bakery (inside)	45.73	45.23	44.64	Newscaster	63.51	74.38	71.05
Band (Rock Pop)	84.16	93.20	90.56	Newspaper magazine	54.05	68.35	81.10
Barbecue	57.49	55.03	57.53	Night (outside)	59.17	58.25	67.60
Bathroom	83.00	89.36	90.44	Nude person	42.67	62.91	65.64
Bird's eye view	42.39	61.86	65.78	Outside broadcast van	65.92	90.82	93.61
Brandenburg Gate (Berlin)	99.38	99.26	99.78	Old person	72.91	83.46	80.76
Butcher shop (inside)	36.02	37.87	36.15	Olympic rings (logo)	79.76	92.48	89.88
Car license plate	87.72	92.59	94.19	Orchestra	66.59	84.67	87.34
Cartoon drawing	79.66	83.44	82.50	Palace of Culture (Warsaw)	77.42	76.83	76.66
Cat (pet)	95.28	93.62	95.58	Pharmacy (inside)	74.39	78.49	82.42
Cemetery	72.66	75.27	73.40	Picnic	58.28	77.94	79.65
Child	69.04	70.04	72.46	Pittiplatsch	85.53	91.37	91.09
Choir	41.00	41.21	48.77	Press conference	75.01	87.14	89.21
Christmas	63.90	74.83	71.69	Rain	61.21	70.99	68.68
City theater (Weimar)	95.14	98.25	98.23	Red City Hall (Berlin)	81.02	88.31	87.84
City-Hochhaus (Leipzig)	97.09	98.52	98.77	Red Tower (Halle)	88.35	87.68	87.13
Classroom	66.91	73.17	78.44	Sandman	78.23	91.64	91.67
Close-up	76.09	80.20	77.16	Sanssouci (Potsdam)	89.95	93.42	95.20
Conference podium	55.14	66.50	69.98	Schwerin Castle	87.39	89.72	89.25
Control room	78.44	78.60	83.25	Sewing machine	84.69	86.99	86.72
Dog (pet)	74.25	84.24	85.61	Silhouette	70.35	81.88	84.32
Dove of peace	77.92	89.67	92.28	Space	76.21	81.62	82.60
Elbe panorama (Dresden)	95.43	98.13	99.26	Spire	47.37	49.31	54.55
Erfurt Cathedral (Erfurt)	76.97	90.26	92.86	Spring	41.44	38.91	48.62
Flag ceremony	78.39	86.52	86.12	St. Thomas church (Leipzig)	95.22	96.64	93.52
Flash	75.68	83.56	84.48	Street sign	45.98	64.93	68.28
Frau Elster	78.73	86.91	89.36	Subway	24.76	45.65	62.85
Gewandhaus (Leipzig)	88.55	88.75	88.04	Sunrise/sunset	85.24	82.64	87.24
Hairdresser (inside)	65.37	67.09	68.18	Teepott & lighthouse (Rostock)	97.09	97.95	98.11
Hammer and sickle	66.99	82.72	82.64	Telephone	89.55	94.66	94.60
Herr Fuchs	83.12	84.21	87.88	Television camera	31.47	74.90	75.80
Hiking (in nature)	48.58	51.01	55.23	Television set	65.49	84.82	79.18
Jeans	31.52	53.79	60.36	Television tower (Berlin)	82.81	85.56	86.27
Jugendweihe	93.71	97.37	95.66	Long shot	52.89	56.77	62.90
Karl Marx Monument	83.99	86.16	83.88	Town hall tower (Gera)	93.44	95.33	95.76
Kremlin (Moscow)	93.00	95.18	95.80	Town sign	76.87	81.85	84.42
Lange Straße (Rostock)	92.38	92.09	91.17	Tractor	85.71	78.47	83.29
Library (indoor)	84.27	88.63	90.04	Traditional costumes	55.13	59.55	58.26
Living room	42.46	39.21	40.73	Underwater shot	63.74	82.84	79.62
Magdeburg Cathedral	95.34	96.35	98.37	Unitower (Jena)	94.80	96.04	96.08
Map	67.65	79.02	79.09	Wheelchair user	71.25	85.66	88.30
Market	69.00	55.39	67.05	mAP	71.11	78.31	79.88

Table 6.4: Concept classification results for the approaches A, B and C in terms of AP.

average of the precisions at each relevant document. To evaluate the overall performance, the mean AP score is calculated by taking the mean value of the AP scores from different queries.

Experimental results for visual concept classification are described in Section 6.3.5. Section 6.3.5 presents the person recognition results. Our similarity search results for images and persons are presented in Sections 6.3.5 and 6.3.5, respectively.

For the use cases of image similarity search and person search, we used 40 images downloaded from the WWW as our query images in each case to estimate the retrieval quality. Each retrieval result is evaluated up to the first 100 retrieved keyframes. While the evaluation of the similarity search approaches is based on the entire video corpus of more than 15.7 million keyframes and 2.8 million faces, the approaches for visual concept classification and person recognition use a randomly selected validation set of labeled instances, as described in the corresponding subsections.

Visual Concept Classification

Altogether, the VIVA tool was used to acquire training and validation data for the previously introduced 91 GDR specific concepts. Several experiments were performed to evaluate the models trained using the VIVA tool. Since the training of a deep neural network from scratch is much more challenging in terms of monitoring the training process and requires much more resources, both concerning data acquisition and computational runtime, a transfer learning strategy is used. Therefore, the models were initialized with pre-trained weights from ImageNet and a two-stage approach was applied for fine-tuning the model to the target domain. In the first stage, only the last layer of the network was trained with a learning rate of 0.001, while the remaining layers with pre-trained weights were frozen. This prevents that the random initialization of the last layer harms the learned feature representations of the previous layers. In the second stage, more layers (the number of layers is empirically set to the top 20 layers) were involved in the training process to improve the adaptation starting with a learning rate of 0.0001. To train the models, we used the ADAM optimizer [KB15], because it automatically adapts the learning rate during the training process.

A total of 113,104 positive and 35,500 negative concept annotations are currently in the database of VIVA. The validation set was randomly sampled from the labeled instances by selecting 30 positive samples for each concept and 20% of the concept's negative annotations. The remaining samples were used for training. Altogether, the training set consists of 110,358 positive and 28,370 negative labels and the validation set contains 2,746 positive and 7,130 negative labels.

Three different approaches were evaluated:

- Approach A uses only samples with at least one positive label. The weights of the loss function are $\alpha_p = 0.8$ and $\alpha_w = 0.1$.
- Approach B uses all training samples with the following weighting scheme: $\alpha_p = 0.8$, $\alpha_n = 0.1$ and $\alpha_w = 0.1$. The actually labeled negative samples are considered but weighted as low as the weak negative samples.

Top 10		Bottom 10	
Joachim Gauck	100.0	Bärbel Wachholz	90.0
Nina Hagen	100.0	Tamara Danz	89.8
Andreas Holm	100.0	Margot Ebert	88.7
Zsuzsa Koncz	100.0	Uwe Ampler	87.8
Jan-Josef Liefers	100.0	Britt Kersten	85.2
Thomas Lück	100.0	Rosemarie Ambe	83.8
Angela Merkel	100.0	Täve Schur	81.1
John Peet	100.0	Herbert Köfer	76.5
Matthias Sammer	100.0	Erich Honecker	75.2
Hilmar Thate	100.0	Kurt Masur	64.4

Table 6.5: Person recognition results in terms of AP.

- In approach C, positive and negative samples are weighted equally. Diminishing the impact of weak negative samples leads to the following weighting scheme: $\alpha_p = 0.8$, $\alpha_n = 0.8$ and $\alpha_w = 0.2$.

The results of the different approaches investigating the impact of negatively labeled training samples are presented in Table 6.4. The negatively labeled training samples from the review and user feedback components are mainly hard negative examples and lead to a clear performance improvement. Using only the weakly negative samples in approach A achieves a performance of 71.11% mean AP. Approach C assigns a high weighting factor for both positive as well as negative labels and a low weighting factor for weakly negative samples. It outperforms approach B in terms of mean AP and leads to the best performance of 79.88% mean AP.

Person Recognition

To train the recognition model for 98 GDR personalities, around 14,400 person images were labelled. Applying a subsampling strategy that utilized up to 100 images per person, a linear SVM using a RBF kernel was trained. To measure the performance of the person recognition model, a stratified k-fold cross-validation was performed during training. Using k=5 folds, a mean average precision of 95.10% was obtained across all classes. In Table 6.5, AP results for the top 10 and bottom 10 persons are presented. While most persons are classified with a high accuracy, lower AP results occur for persons with less training data and less variance regarding training samples and poses. Furthermore, misclassifications are often due to differences in image quality and face occlusions.

Image Similarity Search

To verify the performance of the image similarity search component in the context of VIVA, we selected query images from the WWW for 40 concepts of our concept lexicon. We computed

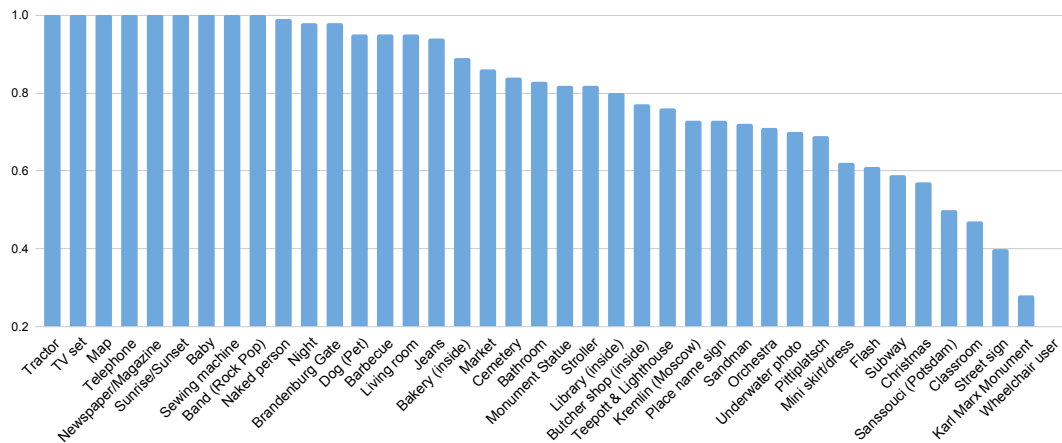


Figure 6.30: AP@100 for 40 query images corresponding to 40 concepts.

the query results based on the entire video collection of more than 15.7 million keyframes. The ranked retrieval lists are evaluated up to rank 100 using the AP score (AP@100). The results are shown in Figure 6.30.

Altogether, a mean AP of 81.12% is achieved. In some cases, the interpretation whether two images are similar is subjective and context specific. In particular, simply presenting an image as a query is often insufficient to express a user’s intention [KMF20], due to ambiguities in the query image. For example, if we are searching for an underwater shot using a query image showing an underwater scene with fishes and a diver, a retrieved image containing a diver in a boat is incorrect regarding the search intention but also semantically similar.

Despite these ambiguities, our approach for semantic similarity search yields high-quality search results. Furthermore, the similarity search approach is quite effective for semi-automatic data acquisition. An example result for a query image showing a “bakery” is presented in Figure 6.25.

Person Similarity Search

Similar to visual concepts, the performance of our person similarity search approach was evaluated for the top 100 results of randomly selected queries for 40 persons. However, we ensured that query faces met the criteria of distinctly showing the person of interest with a reasonable quality (no blur or occlusion). In our experimental evaluation, a mean AP@100 of 81.50% was obtained. The performance results for each person query are shown in Figure 6.31.

Discussion

The presented content-based video retrieval approaches are used to complement human annotations and to support users in finding relevant video shots. Our visual concept and person

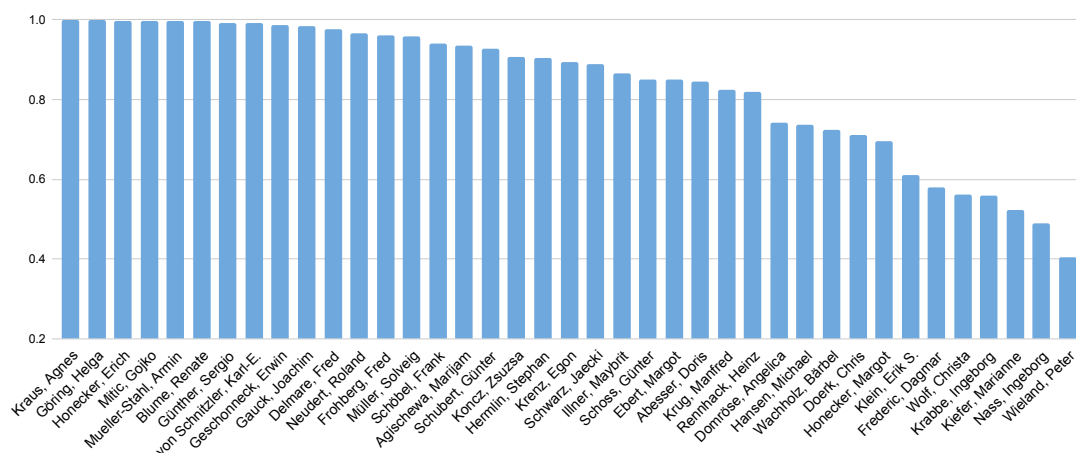


Figure 6.31: AP@100 for 40 query images corresponding to 40 persons.

models are applied to the entire video collection of historical GDR television recordings of the DRA containing more than 15.7 million keyframes and more than 2.8 million faces. The resulting indices are integrated into the FESAD search environment of the DRA and provide a valuable supplement to the human-generated metadata. Concept, person, and similarity search queries as well as queries based on human-generated metadata can be combined to answer a wide range of user queries. Altogether, the very good quality of the presented concept, person, and similarity search results is highly beneficial for archivists as well as for users of the archive.

6.3.6 Summary

We presented VIVA, a software tool based on deep learning models for visual information retrieval in videos. To address potentially changing user requirements, VIVA enables non-expert users in the field of deep learning to semi-automatically acquire training data with minimal manual effort, to perform the training process with a user-friendly interface, to visualize the retrieval results, and to collect user feedback. Considering the four use cases of visual concept classification, person recognition, similarity search for images, and similarity search for persons, we evaluated VIVA using a unique historical collection of about 34,000 hours of GDR television recordings. We used the VIVA tool to build high-quality deep learning models for 91 GDR specific concepts and 98 personalities from the former GDR. Our experimental results demonstrated the benefits for both archivists and archive users.

To provide efficient data acquisition and training workflows in VIVA, the video retrieval system operates on images and keyframes. This leads to limitations with respect to concepts representing actions. However, the GDR specific concept lexicon contains only very few of such concepts.

7

Conclusion

7.1 Summary

Image similarity search is important in many applications. The main focus of this thesis was on how image segmentation can help to improve image similarity search. First, contributions were presented in the areas of detection and segmentation, as well as in the area of image similarity search. Methods from these areas provide the foundation for a novel approach to segmentation-based image similarity search, which was presented in this thesis for the first time. It was shown that segmentation-based similarity search not only opens up new search possibilities, but that the use of features from image regions can also improve the retrieval quality. Finally, applications of image similarity search in different systems were discussed.

In the area of detection and segmentation, the following contributions were made:

- A new deep learning-based method was presented to significantly accelerate the creation of an etymological dictionary by automating the detection, alignment, and recognition of textual stamps on digitized index cards.
- A new approach was presented to identify and segment macrophage cells in complex, crowded fluorescence microscopy images; it improves segmentation performance by also considering nucleus features.

In the area of image similarity search, a number of innovations were presented that affect both the efficiency of the search and the quality of the retrieval results:

- A multi-task learning approach was presented that integrates image classification and similarity search into a single architecture.
- A method was presented for identifying new textual stamps in a corpus of digitized index cards using similarity search; it enhances the efficiency of processing philological index cards.
- A novel intentional image similarity search method was presented; it includes a plugin mechanism to support specialized models tailored to specific search tasks, using hybrid features and an analysis technique to find relevant regions to improve the quality of the retrieved results.
- ElasticHash was presented, a new efficient approach for large-scale semantic image similarity search in Elasticsearch, using deep hashing and a two-stage search approach.

- Search Anything was presented, a novel approach to image similarity search using point, box, and text prompts, employing self-supervised learning and foundation models for fine-grained region-level indexing and searching.

Finally, three systems with different purposes were presented, in which the application of image similarity search is a core component:

- A distributed system was introduced for analyzing and retrieving content from GDR television archives, incorporating machine learning algorithms for shot detection, concept classification, person recognition, and text recognition.
- A system containing various deep learning methods was presented for automating content labeling in professional media production, enhancing video inspection and retrieval through visual concept classification, face recognition, and innovative visualization.
- VIVA was presented, a software tool allowing non-expert users creating and using deep learning models for efficient video retrieval, including a semi-automatic data collection workflow with web crawling, image similarity search, and user feedback to streamline the collection of training samples.

7.2 Future Work

There is future work in each of the research areas covered in the thesis: detection and segmentation, image similarity search, and, in particular, region-based image similarity search.

7.2.1 Detection and Segmentation

Stamp Detection and Recognition. In the future, we want to integrate the individual components of our approach into an end-to-end system for textual stamp detection, alignment, and recognition. Furthermore, we intend to use an active learning framework to iteratively improve our deep learning models during operation. Finally, we plan to apply our approach for textual stamp recognition to index cards of other philological projects by using transfer learning and to further investigate the generalizability of our approach.

Cell Segmentation. Sharing backbone weights to build a model that can be trained end-to-end for nucleus/cell detection and segmentation could be interesting. For this purpose, separate Mask R-CNN heads for nuclei and cells have to be attached to the architecture for region proposal generation, regression, classification, and segmentation. The integration of a full-image segmentation loss in addition to an instance-based detection and segmentation loss should be investigated. Optimizing the anchor box sampling strategies for nuclei and cells could lead to further performance improvements.

7.2.2 Image Similarity Search

In this thesis, two approaches that tackle different aspects of similarity search systems were presented. The approach to intentional image similarity search works well for small datasets. However, when applied to a large-scale scenario, runtime improvements are necessary for the relevant region and handcrafted feature extraction stages. A combination with deep hashing approaches could improve the query times. Another interesting research area in image similarity search is instance retrieval [Che+22]. Methods in this area could also be beneficial for intentional image similarity search. Furthermore, a user could be interested in searching for multiple regions in a query image, e.g., for two specific persons. For this purpose, intentional image similarity search has to be extended by region specific options. Finally, further combinations of CNNs and handcrafted features should be evaluated to satisfy a user's intentions. The proposed two-stage method for efficient image similarity search could be improved in several ways. Another backbone model could be used to improve retrieval performance. Also, the current implementation is based on a model trained on a large dataset containing many classes; an even more general backbone could be another improvement. The backbone model could be replaced by a foundation model, and the model could be trained in a self-supervised manner based on more general features. For example, models that are more robust against distribution shift such as CLIP could be used. It would also be interesting to compare the approach for efficient similarity search in ElasticSearch with other vector search libraries in terms of energy consumption. This also includes a detailed comparison of the hardware resources used. While libraries such as FAISS may heavily utilize available GPU or CPU resources, the performance of ElasticHash depends primarily on the I/O throughput of the SSD.

7.2.3 Segmentation-based Image Similarity Search

The presented region-based image similarity search approach is a first implementation that is based on two foundation models, which could be replaced by more powerful or more efficient ones in the future. The next step is to unify the architecture and to train a model end-to-end. The training would rely on the SA-1B dataset on the one hand, and on extracted features of a powerful image encoding model on the other hand. These features that are extracted per mask could be used to train the model in a knowledge-distillation setting. Furthermore, the proposed approach currently only works for single-region queries. Thus, we plan to extend it to support multiple region queries and to take into account spatial relationships between objects. Another possibility for improvement is the deep hashing block trained to compress the hash codes for the mask. It is trained with a straightforward unsupervised loss. Here, more investigation and ablation studies could help to find a better network architecture and loss function. In particular, the quantization error in the loss function should be taken into account. An improved deep hashing model could maintain the retrieval performance while decreasing the number of bits needed, which would reduce storage requirements and accelerate query time. Especially for region-based similarity searches, this would lead to a significant improvement, since there can be dozens to hundreds of codes per image. Another way to improve performance is to reduce the number of regions to consider. In its current implementation, the approach cannot prioritize regions for indexing other than by size, which does not always correspond to the

7 Conclusion

importance of a region. Finding a way of integrating information about global attention to help selecting more relevant regions could be an interesting direction of further research.

List of Figures

Chapter 1: Introduction	1
Chapter 2: Fundamentals	9
2.1 Neural network	10
2.2 Sigmoid activation	12
2.3 Tanh activation	13
2.4 ReLU activation	14
2.5 SiLU activation	15
2.6 Gradient descent	20
2.7 Backpropagation	21
2.8 2-D convolution	26
2.9 Convolution filter	27
2.10 Learned filters	27
2.11 Max-pooling	30
2.12 Dropout	33
2.13 Transformer architecture	35
2.14 Vision Transformer	38
2.15 Fast R-CNN	39
2.16 Faster R-CNN	41
2.17 Feature Pyramid Network	42
2.18 Spatial Pyramid Pooling	44
2.19 Segmentation tasks	47
2.20 U-Net	47
2.21 Mask R-CNN	48
2.22 CLIP	50
2.23 Image Similarity Search	52
Chapter 3: Detection and Segmentation	57
3.1 Fotos of boxes with index cards	59
3.2 Index card	60
3.3 Challenging stamps	61
3.4 Textual stamp recognition workflow	63
3.5 Index card variations	64
3.6 Synthetic textual stamps	67
3.7 Semi-automatic stamp recognition workflow	68
3.8 Class distribution	70
3.9 Input image with ground truth segmentation	75
3.10 Feature pyramid fusion of nucleus features	79
3.11 Including nucleus information for cell segmentation	81
3.12 Instance weighting	82
3.13 Cell segmentation errors	85
3.14 Cell segmentation of clustered cells	88
3.15 Segmentation of clustered cells (AP)	89

3.16	Segmentation of clustered cells (Visualization)	89
Chapter 4: Image Similarity Search		91
4.1	Query images	93
4.2	Heatmaps visualized in retrieval results.	98
4.3	Retrieval results face module	99
4.4	Retrieval by semantic concepts	100
4.5	Retrieval by semantic concepts re-ranked	101
4.6	ElasticHash workflow	105
4.7	Correlation between bits	107
4.8	Indices in Elasticsearch	109
4.9	Retrieval results ElasticHash	111
Chapter 5: Segmentation-based Image Similarity Search via Region Prompts		115
5.1	Segmentation-based similarity search	115
5.2	Similarity search with region prompts	119
5.3	Small objects	125
5.4	Improving retrieval results by masking	126
5.5	Example query	127
5.6	Ambiguous box prompt 1/3	130
5.7	Ambiguous box prompt 2/3	131
5.8	Ambiguous box prompt 3/3	132
5.9	Region appearance 1/3	133
5.10	Region appearance 2/3	134
5.11	Region appearance 3/3	135
Chapter 6: Image Similarity Search in Applications		137
6.1	Video retrieval system	141
6.2	Deep convolutional neural network	142
6.3	Content-based similarity search	144
6.4	Text recognition pipeline	145
6.5	Service-oriented architecture for CBVR	146
6.6	Retrieval tool	148
6.7	Workflow	149
6.8	BoVW vs. CNN	150
6.9	Similarity search results for 50 query images from the Internet	152
6.10	Retrieval result for a query of a meal	153
6.11	Similarity search results	155
6.12	OCR retrieval results for 46 text queries	155
6.13	Content based similarity search	162
6.14	Multi-task learning of hashcodes	163
6.15	Face indexing workflow	165
6.16	Service-oriented architecture for content-based video retrieval	167
6.17	Video analysis and retrieval tool	168
6.18	Similarity search results	169
6.19	Face identification results	172

6.20	Face retrieval results	173
6.21	Person retrieval results	174
6.22	Visualization of person retrieval results	174
6.23	Similarity search results in retrieval tool	176
6.24	Retrieval result for “bakery”	180
6.25	VIVA GUI	182
6.26	VIVA workflow	184
6.27	VIVA GUI training	186
6.28	User review	187
6.29	VIVA integration	188
6.30	Similarity search results for 40 concepts	193
6.31	Similarity search results for 40 persons	194
Chapter 7: Conclusion		195

List of Tables

Chapter 1: Introduction	1
Chapter 2: Fundamentals	9
Chapter 3: Detection and Segmentation	57
3.1 Index card dataset	70
3.2 Stamp detection performance	71
3.3 Stamp recognition performance	72
3.4 k -Nearest neighbor performance	72
3.5 Retrieval performance for unknown stamps	73
3.6 Reduced ResNet-50 architecture	80
3.7 Detection and segmentation results for nuclei	85
3.8 Detection results for cells	86
3.9 Segmentation results for cells	87
3.10 Detailed cell segmentation results for IoU threshold of 0.75	87
3.11 Detection results for clustered cells	88
3.12 Segmentation results for clustered cells	88
Chapter 4: Image Similarity Search	91
4.1 Retrieval quality in terms of mean AP	112
4.2 Search latencies for ES and FAISS queries	112
Chapter 5: Segmentation-based Image Similarity Search via Region Prompts	115
5.1 Masking semantic features	124
5.2 Masking feature maps	124
5.3 Context semantic features	125
5.4 Concatenated features	126
5.5 Zero-shot retrieval performance	127
5.6 Retrieval performance binary vs. float	129
5.7 Query time binary vs. float	129
5.8 Context semantic features extended	135
Chapter 6: Image Similarity Search in Applications	137
6.1 Face Recognition Results	154
6.2 Concept detection results on the top 100 shots per concept	170
6.3 Similarity search results on the top 100 shots per concept	171
6.4 Concept scores	190
6.5 Person recognition results	192
Chapter 7: Conclusion	195

Listings

4.1	Mapping of retrieval index	108
4.2	Query for adding an entry to neighbor lookup index	108
4.3	Query for adding a Painless Script	109
4.4	Query for performing two-stage similarity search	110

Bibliography

- [Aba+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. “Tensorflow: A System for Large-scale Machine Learning.” in: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283 (cit. on p. 81).
- [ABB20] Abeer Al-Mohamade, Ouiem Bchir, and Mohamed Maher Ben Ismail. “Multiple Query Content-based Image Retrieval Using Relevance Feature Weight Learning.” in: *Journal of Imaging* 6.1 (2020), p. 2 (cit. on p. 95).
- [Abd17] Waleed Abdulla. *Mask R-CNN for Object Detection and Instance Segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN. 2017 (cit. on p. 81).
- [Abr+17] A Abreu, F-X Frenois, S Valitutti, P Brousset, P Denèfle, B Naegel, and C Wemmert. “Optimal Cut in Minimum Spanning Trees for 3-D Cell Nuclei Segmentation.” in: *10th International Symposium on Image and Signal Processing and Analysis*. IEEE. 2017, pp. 195–199 (cit. on p. 76).
- [Ahm+13] Sheraz Ahmed, Faisal Shafait, Marcus Liwicki, and Andreas Dengel. “A Generic Method for Stamp Segmentation Using Part-based Features.” in: *12th International Conference on Document Analysis and Recognition*. IEEE. 2013, pp. 708–712 (cit. on p. 63).
- [AHP04] T. Ahonen, A. Hadid, and M. Pietikainen. “Face Recognition With Local Binary Patterns.” in: *Proceedings IEEE European Conference on Computer Vision (ECCV)*. 2004, pp. 469–481 (cit. on p. 145).
- [AJ15] Dan Albertson and Boryung Ju. “Design Criteria for Video Digital Libraries: Categories of Important Features Emerging From Users’ Responses.” in: *Online Information Review* 39.2 (2015), pp. 214–228 (cit. on p. 139).
- [Ak+18] Kenan E Ak, Ashraf A Kassim, Joo Hwee Lim, and Jo Yew Tham. “Learning Attribute Representations With Localization for Flexible Fashion Search.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7708–7717 (cit. on pp. 1, 115).
- [Akr+16] Saad Ullah Akram, Juho Kannala, Lauri Eklund, and Janne Heikkilä. “Cell Segmentation Proposal Network for Microscopy Image Analysis.” in: *Deep Learning and Data Labeling for Medical Applications*. Springer, 2016, pp. 21–29 (cit. on p. 77).
- [Al-+10] Yousef Al-Kofahi, Wiem Lassoued, William Lee, and Badrinath Roysam. “Improved Automatic Detection and Segmentation of Cell Nuclei in Histopathology Images.” in: *IEEE Transactions on Biomedical Engineering* 57.4 (2010), pp. 841–852 (cit. on p. 76).
- [Al-+18] Yousef Al-Kofahi, Alla Zaltsman, Robert Graves, Will Marshall, and Mirabela Rusu. “A Deep Learning-based Algorithm for 2-D Cell Segmentation in Microscopy Images.” in: *BMC Bioinformatics* 19.1 (2018), pp. 1–11. ISSN: 14712105. DOI: 10.1186/s12859-018-2375-z (cit. on p. 76).

- [Ama+14] Fernando Amat, William Lemon, Daniel P Mossing, Katie McDole, Yinan Wan, Kristin Branson, Eugene W Myers, and Philipp J Keller. “Fast, Accurate Reconstruction of Cell Lineages From Large-scale Fluorescence Microscopy Data.” in: *Nature Methods* 11.9 (2014), p. 951 (cit. on p. 76).
- [Ama+17] Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Fausto Rabitti. “Searching and Annotating 100M Images With YFCC100M-HNfc6 and MI-file.” in: *Proceedings of the 15th International Workshop on Content-based Multimedia Indexing*. 2017, pp. 1–4 (cit. on p. 183).
- [Ama+18] Giuseppe Amato, Paolo Bolettieri, Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro. “Large-scale Image Retrieval With Elasticsearch.” in: *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2018, pp. 925–928 (cit. on p. 104).
- [Ama+21] Giuseppe Amato, Paolo Bolettieri, Fabio Carrara, Franca Debole, Fabrizio Falchi, Claudio Gennaro, Lucia Vadicamo, and Claudio Vairo. “The VISIONE Video Search System: Exploiting Off-the-shelf Text Search Engines for Large-scale Video Retrieval.” in: *Journal of Imaging* 7.5 (2021), p. 76. doi: 10.3390/jimaging7050076. URL: <https://doi.org/10.3390/jimaging7050076> (cit. on p. 179).
- [And+21] Stelios Andreadis, Anastasia Moutzidou, Konstantinos Gkountakos, Nick Pantelidis, Konstantinos Apostolidis, Damianos Galanopoulos, Ilias Gialampoukidis, Stefanos Vrochidis, Vasileios Mezaris, and Ioannis Kompatsiaris. “VERGE in VBS 2021.” in: *Proceedings of the 27th International Conference on MultiMedia Modeling (MMM 2021)*. vol. 12573. Lecture Notes in Computer Science. Springer, 2021, pp. 398–404. doi: 10.1007/978-3-030-67835-7_35. URL: https://doi.org/10.1007/978-3-030-67835-7_35 (cit. on p. 179).
- [ARP16] Alireza Alaei, Partha Pratim Roy, and Umapada Pal. “Logo and Seal Based Administrative Document Image Retrieval: A Survey.” in: *Computer Science Review* 22 (2016), pp. 47–63 (cit. on p. 62).
- [AZ12] Relja Arandjelović and Andrew Zisserman. “Multiple Queries for Large Scale Specific Object Retrieval.” in: *BMVC 2012 - Electronic Proceedings of the British Machine Vision Conference 2012* (2012), pp. 1–11 (cit. on p. 95).
- [Bae+19] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. “Character Region Awareness for Text Detection.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9365–9374 (cit. on pp. 62, 65).
- [Bal+74] Kurt Baldinger, Jean-Denis Gendron, Georges Straka, Martina Fietz-Beck, Frankwalt Möhren, Sabine Tittel, and Thomas Städtler. *Dictionnaire Étymologique De l’Ancien Français*. 1974 (cit. on p. 62).
- [Bel+22] Hicham Bellafkir, Markus Vogelbacher, Jannis Gottwald, Markus Mühling, Nikolaus Korfhage, Patrick Lampe, Nicolas Frieß, Thomas Nauss, and Bernd Freisleben. “Bat echolocation call detection and species recognition by transformers with self-attention.” in: *Intelligent Systems and Pattern Recognition: Second International Conference, ISPR 2022, Hammamet, Tunisia, March 24–26, 2022, Revised Selected Papers*. Springer. Cham: Springer International Publishing, 2022, pp. 189–203 (cit. on p. 6).

- [Bel+23] Hicham Bellafkir, Markus Vogelbacher, Daniel Schneider, Markus Mühling, Nikolaus Korfhage, and Bernd Freisleben. “Edge-based Bird Species Recognition via Active Learning.” in: *International Conference on Networked Systems*. Springer, 2023, pp. 17–34 (cit. on p. 6).
- [Ben+09] Yoshua Bengio et al. *Learning Deep Architectures for AI*. 2009 (cit. on p. 25).
- [BGC17] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep Learning*. vol. 1. MIT press Cambridge, MA, USA, 2017 (cit. on p. 16).
- [Bia+12] Jingwen Bian, Zheng Jun Zha, Hanwang Zhang, Qi Tian, and Tat Seng Chua. “Visual Query Attributes Suggestion.” in: *MM 2012 - Proceedings of the 20th ACM International Conference on Multimedia*. 2012, pp. 869–872 (cit. on p. 95).
- [BK97] Peter N Belhumeur and David J Kriegman. “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.7 (1997), pp. 711–720 (cit. on p. 145).
- [BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “Segnet: A Deep Convolutional Encoder-decoder Architecture for Image Segmentation.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495 (cit. on p. 76).
- [BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer Normalization.” in: *arXiv Preprint arXiv:1607.06450* (2016) (cit. on p. 37).
- [Bla+16] Gustavo Blanco, Marcos VN Bedo, Mirela T Cazzolato, Lucio FD Santos, Ana Elisa Serafim Jorge, Caetano Traina, Paulo M Azevedo-Marques, and Agma JM Traina. “A Label-scaled Similarity Measure for Content-based Image Retrieval.” in: *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2016, pp. 20–25 (cit. on pp. 97, 161).
- [BM18] Serge Beucher and Fernand Meyer. “The Morphological Approach to Segmentation: The Watershed Transformation.” in: *Mathematical Morphology in Image Processing*. CRC Press, 2018, pp. 433–481 (cit. on p. 46).
- [Bot12] Léon Bottou. “Stochastic Gradient Descent Tricks.” in: *Neural Networks: Tricks of the Trade: Second Edition*. Springer, 2012, pp. 421–436 (cit. on p. 34).
- [BPW75] Kurt Baldinger, Inge Popelar, and Heidelberger Akademie der Wissenschaften. *Dictionnaire Onomasiologique De l’Ancien Occitan: DAO*. Niemeyer, 1975 (cit. on p. 62).
- [Bra00] G. Bradski. “The OpenCV Library.” in: *Dr. Dobb’s Journal of Software Tools* (2000) (cit. on pp. 64, 78).
- [Bre+13] Thomas M. Breuel, Adnan Ul-Hasan, Mayce Ali Al-Azawi, and Faisal Shafait. “High-performance OCR for Printed English and Fraktur Using LSTM Networks.” in: *Proceedings International Conference on Document Analysis and Recognition*. 2013, pp. 683–687 (cit. on pp. 143, 147, 160).
- [Bro+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. “Language Models Are Few-shot Learners.” in: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1877–1901 (cit. on p. 118).

- [Cai+15] Junjie Cai, Zheng Jun Zha, Meng Wang, Shiliang Zhang, and Qi Tian. “An Attribute-assisted Reranking Model for Web Image Search.” in: *IEEE Transactions on Image Processing* 24.1 (2015), pp. 261–272. ISSN: 10577149 (cit. on p. 94).
- [Cai+19a] Juan C. Caicedo, Allen Goodman, Kyle W. Karhohs, Beth A. Cimini, Jeanelle Ackerman, Marzieh Haghighi, Cher Keng Heng, Tim Becker, Minh Doan, Claire McQuin, Mohammad Rohban, Shantanu Singh, and Anne E. Carpenter. “Nucleus Segmentation Across Imaging Experiments: The 2018 Data Science Bowl.” in: *Nature Methods* 16.12 (2019), pp. 1247–1253. ISSN: 15487105. DOI: 10.1038/s41592-019-0612-7. URL: <http://dx.doi.org/10.1038/s41592-019-0612-7> (cit. on p. 76).
- [Cai+19b] Juan C. Caicedo, Jonathan Roth, Allen Goodman, Tim Becker, Kyle W. Karhohs, Matthieu Broisin, Csaba Molnar, Claire McQuin, Shantanu Singh, Fabian J. Theis, and Anne E. Carpenter. “Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images.” in: *Cytometry Part A* 95.9 (2019), pp. 952–965. ISSN: 15524930. DOI: 10.1002/cyto.a.23863 (cit. on p. 76).
- [Can86] John Canny. “A Computational Approach to Edge Detection.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (1986), pp. 679–698 (cit. on p. 46).
- [Cao+17a] Yuan Cao, Heng Qi, Wenrui Zhou, Jien Kato, Keqiu Li, Xiulong Liu, and Jie Gui. “Binary Hashing for Approximate Nearest Neighbor Search on Big Data: A Survey.” in: *IEEE Access* 6 (2017), pp. 2039–2054 (cit. on p. 104).
- [Cao+17b] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. “Hashnet: Deep Learning to Hash by Continuation.” in: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5608–5617 (cit. on pp. 54, 104).
- [Cao+18a] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. “Vg-face2: A Dataset for Recognising Faces Across Pose and Age.” in: *Proceedings of the 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE. 2018, pp. 67–74 (cit. on pp. 183, 184).
- [Cao+18b] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. “Deep Cauchy Hashing for Hamming Space Retrieval.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1229–1237 (cit. on pp. 54, 104).
- [Car+20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. “End-to-end Object Detection With Transformers.” in: *European Conference on Computer Vision*. Springer. 2020, pp. 213–229 (cit. on p. 45).
- [CAS20] Bingyi Cao, Andre Araujo, and Jack Sim. “Unifying Deep Local and Global Features for Image Search.” in: *Computer Vision—Eccv 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX* 16. Springer. 2020, pp. 726–743 (cit. on pp. 51, 117).
- [Cha+14] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Return of the Devil in the Details: Delving Deep Into Convolutional Nets.” in: *Proceedings of the British Machine Vision Conference*. 2014, pp. 1–11 (cit. on pp. 144, 161).

- [Cha+19] Sukalpa Chanda, Prashant Kumar Prasad, Anders Hast, Anders Brun, Lasse Martensson, and Umapada Pal. “Finding Logo and Seal in Historical Document Images - An Object Detection Based Approach.” in: *Asian Conference on Pattern Recognition*. Springer. 2019, pp. 821–834 (cit. on p. 63).
- [Che+15] Xu Chen, Yanqiao Zhu, Fuhai Li, Ze-Yi Zheng, Eric C Chang, Jinwen Ma, and Stephen TC Wong. “Accurate Segmentation of Touching Cells in Multi-channel Microscopy Images With Geodesic Distance Based Clustering.” in: *Neurocomputing* 149 (2015), pp. 39–47 (cit. on p. 75).
- [Che+22] Wei Chen, Yu Liu, Weiping Wang, Erwin M Bakker, Theodoros Georgiou, Paul Fieguth, Li Liu, and Michael S Lew. “Deep Learning for Instance Retrieval: A Survey.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022) (cit. on pp. 2, 51, 117, 197).
- [Chr+95] Michael Christel, Takeo Kanade, M Mauldin, Raj Reddy, Marvin Sirbu, Scott M. Stevens, and Howard D. Wactlar. “Informedia Digital Video Library.” in: *Communications of the ACM* 38.4 (1995), pp. 57–58 (cit. on p. 139).
- [Chu+09] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. “NUS-WIDE: A Real-world Web Image Database From National University of Singapore.” in: *Proceedings of the ACM International Conference on Image and Video Retrieval*. 2009, pp. 1–9 (cit. on pp. 104, 151).
- [CR+09] Jierong Cheng, Jagath C Rajapakse, et al. “Segmentation of Clustered Nuclei With Shape Markers and Marking Function.” in: *IEEE Trans. On Biomedical Engineering* 56.3 (2009), pp. 741–748 (cit. on p. 76).
- [CUF18] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. “COCO-Stuff: Thing and Stuff Classes in Context.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1209–1218 (cit. on p. 122).
- [CV95] Corinna Cortes and Vladimir Vapnik. “Support-vector Networks.” in: *Machine Learning* 20 (1995), pp. 273–297 (cit. on p. 39).
- [CZ03] Xiangrong Chen and Hong-Jiang Zhang. “Photo Time-stamp Detection and Recognition.” in: *Seventh International Conference on Document Analysis and Recognition*. IEEE. 2003, pp. 319–322 (cit. on p. 63).
- [Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A Large-scale Hierarchical Image Database.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 248–255 (cit. on pp. 80, 97, 143, 160–162, 179, 181).
- [Den+14] Cheng Deng, Rongrong Ji, Dacheng Tao, Xinbo Gao, and Xuelong Li. “Weakly Supervised Multi-graph Learning for Robust Image Reranking.” in: *IEEE Transactions on Multimedia* 16.3 (2014), pp. 785–795. ISSN: 15209210 (cit. on p. 94).
- [Den+19a] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. “ArcFace: Additive Angular Margin Loss for Deep Face Recognition.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4690–4699 (cit. on p. 183).

- [Den+19b] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. “Retinaface: Single-stage Dense Face Localisation in the Wild.” in: *ArXiv Preprint arXiv:1905.00641* (2019) (cit. on p. 184).
- [DHS11] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.” in: *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159. ISSN: 15324435. arXiv: arXiv:1103.4296v1. URL: <http://jmlr.org/papers/v12/duchi11a.html> (cit. on p. 24).
- [Dim+14] Sotiris Dimopoulos, Christian E Mayer, Fabian Rudolf, and Joerg Stelling. “Accurate Cell Segmentation in Microscopy Images Using Membrane Patterns.” in: *Bioinformatics* 30.18 (2014), pp. 2644–2651 (cit. on p. 76).
- [DMS15] Soumyadeep Dey, Jayanta Mukherjee, and Shamik Sural. “Stamp and Logo Detection From Document Images by Finding Outliers.” in: *2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*. IEEE. 2015, pp. 1–4 (cit. on p. 63).
- [Dos+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. “An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale.” in: *arXiv Preprint arXiv:2010.11929* (2020) (cit. on pp. 37, 38).
- [Drö+18] Johannes Dröner, Nikolaus Korfhage, Sebastian Egli, Markus Mühlhling, Boris Thies, Jörg Bendix, Bernd Freisleben, and Bernhard Seeger. “Fast Cloud Segmentation Using Convolutional Neural Networks.” in: *Remote Sensing* 10.11 (11 Nov. 2018), p. 1782. ISSN: 20724292. DOI: 10.3390/rs10111782 (cit. on pp. 6, 48).
- [EF04] Ralph Ewerth and Bernd Freisleben. “Video Cut Detection Without Thresholds.” in: *Proceedings of the 11th International Workshop on Signals, Systems and Image Processing (IWSSIP '04)*. Poznan, Poland, 2004, pp. 227–230 (cit. on pp. 142, 159).
- [EF09] Ralph Ewerth and Bernd Freisleben. “Unsupervised Detection of Gradual Video Shot Changes With Motion-based False Alarm Removal.” in: *International Conference on Advanced Concepts for Intelligent Vision Systems*. 2009, pp. 253–264 (cit. on pp. 142, 159).
- [EMF07] Ralph Ewerth, Markus Mühlhling, and Bernd Freisleben. “Self-supervised Learning of Face Appearances in TV Casts and Movies.” in: *International Journal of Semantic Computing* 1.2 (2007), pp. 185–204 (cit. on pp. 145, 166).
- [EMF11] Ralph Ewerth, Markus Mühlhling, and Bernd Freisleben. “Robust Video Content Analysis via Transductive Learning.” in: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.3 (2011), pp. 1–26 (cit. on p. 151).
- [EMH19] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. “Neural Architecture Search: A Survey.” in: *The Journal of Machine Learning Research* 20.1 (2019), pp. 1–21 (cit. on p. 181).
- [Eri+15] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. “Deep Hashing for Compact Binary Codes Learning.” in: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2475–2483 (cit. on pp. 104, 118).

- [EUD18] Stefan Elfving, Eiji Uchibe, and Kenji Doya. “Sigmoid-weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning.” in: *Neural Networks* 107 (2018), pp. 3–11 (cit. on p. 15).
- [Eve+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. “The Pascal Visual Object Classes VOC Challenge.” in: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338 (cit. on pp. 84, 127).
- [Ewe+07] Ralph Ewerth, Martin Schwalb, Paul Tessmann, and Bernd Freisleben. “Segmenting Moving Objects in MPEG Videos in the Presence of Camera Motion.” in: *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference On*. IEEE, 2007, pp. 819–824 (cit. on p. 142).
- [Ewe+12] Ralph Ewerth, Khalid Ballafkir, Markus Mühling, Dominik Seiler, and Bernd Freisleben. “Long-term Incremental Web-supervised Learning of Visual Concepts via Random Savannas.” in: *IEEE Transactions on Multimedia* 14.4 (2012), pp. 1008–1020 (cit. on p. 151).
- [FIY19] Ryosuke Furuta, Naoto Inoue, and Toshihiko Yamasaki. “Efficient and Interactive Spatial-semantic Image Retrieval.” in: *Multimedia Tools and Applications* 78 (2019), pp. 18713–18733 (cit. on p. 118).
- [FM13] Paweł Forczmański and Andrzej Markiewicz. “Low-level Image Features for Stamps Detection and Classification.” in: *8th International Conference on Computer Recognition Systems (CORES 2013)*. Springer, 2013, pp. 383–392 (cit. on p. 63).
- [FSL15] Sachin Sudhakar Farfade, Mohammad J Saberian, and Li-Jia Li. “Multi-view Face Detection Using Deep Convolutional Neural Networks.” in: *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. ACM, 2015, pp. 643–650 (cit. on p. 164).
- [Fuk80] Kunihiko Fukushima. “Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position.” in: *Biological Cybernetics* 36.4 (1980), pp. 193–202 (cit. on p. 25).
- [GB10] Xavier Glorot and Yoshua Bengio. “Understanding the Difficulty of Training Deep Feedforward Neural Networks.” in: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256 (cit. on pp. 12, 16, 34).
- [Ge+21] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. “Yolox: Exceeding Yolo Series in 2021.” in: *arXiv Preprint arXiv:2107.08430* (2021) (cit. on p. 43).
- [GE04] Julinda Gllavata and Ralph Ewerth. “Text Detection in Images Based on Unsupervised Classification of High-frequency Wavelet Coefficients.” in: *Proceedings of 17th International Conference on Pattern Recognition (ICPR '04)*. IEEE, 2004, pp. 425–428 (cit. on p. 146).
- [Gen+10] Claudio Gennaro, Giuseppe Amato, Paolo Bolettieri, and Pasquale Savino. “An Approach to Content-based Image Retrieval Based on the Lucene Search Engine Library.” in: *Proceedings 14th Eur. Conference on Research and Advanced Technology for Digital Libraries*. 2010, pp. 55–66 (cit. on p. 105).

Bibliography

- [GIM+99] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. “Similarity Search in High Dimensions via Hashing.” in: *VLDB*. vol. 99. 6. 1999, pp. 518–529 (cit. on p. 53).
- [Gir+14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587 (cit. on pp. 38, 39).
- [Gir15] Ross Girshick. “Fast R-CNN.” in: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1440–1448 (cit. on pp. 38, 39).
- [GMH13] A Graves, A Mohamed, and G Hinton. “Speech Recognition With Deep Recurrent Neural Networks.” in: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013)*. 2013, pp. 6645–6649 (cit. on pp. 143, 160).
- [Gon+13] Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, and Sergey Ioffe. “Deep Convolutional Ranking for Multilabel Image Annotation.” in: *arXiv Preprint arXiv:1312.4894* (2013) (cit. on pp. 143, 161).
- [Goo+13a] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. “Maxout Networks.” in: *International Conference on Machine Learning*. PMLR. 2013, pp. 1319–1327 (cit. on p. 32).
- [Goo+13b] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. “Multi-digit Number Recognition From Street View Imagery Using Deep Convolutional Neural Networks.” in: (Dec. 2013), pp. 1–13. arXiv: 1312.6082. URL: <http://arxiv.org/abs/1312.6082v4> (cit. on p. 14).
- [GQ07] Jian Guan and Guoping Qiu. “Learning User Intention in Relevance Feedback Using Optimization.” in: *Proceedings of the ACM International Multimedia Conference and Exhibition*. 2007, pp. 41–50 (cit. on p. 95).
- [GRS19] Ralph Gasser, Luca Rossetto, and Heiko Schuldt. “Multimodal Multimedia Retrieval With VitriVr.” in: *Proceedings of the International Conference on Multimedia Retrieval (ICMR 2019)*. ACM, 2019, pp. 391–394. doi: 10.1145/3323873.3326921. URL: <https://doi.org/10.1145/3323873.3326921> (cit. on p. 179).
- [Gud+08] Prabhakar R Gudla, K Nandy, J Collins, KJ Meaburn, T Misteli, and SJ Lockett. “A High-throughput System for Segmenting Nuclei Using Multiscale Techniques.” in: *Cytometry Part A* 73.5 (2008), pp. 451–466 (cit. on p. 76).
- [Guo+16] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. “MS-Celeb-1M: A Dataset and Benchmark for Large-scale Face Recognition.” in: *Proceedings of 14th European Conference on Computer Vision*. Lecture Notes in Computer Science. Springer, 2016, pp. 87–102 (cit. on pp. 164, 183).
- [Guo+20] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. “Accelerating Large-scale Inference With Anisotropic Vector Quantization.” in: *International Conference on Machine Learning*. PMLR. 2020, pp. 3887–3896 (cit. on p. 118).
- [HBS13] Christian Hentschel, Ina Blümel, and Harald Sack. “Automatic Annotation of Scientific Video Material Based on Visual Concept Detection.” in: *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*. ACM. 2013, p. 16 (cit. on p. 139).

- [HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality Reduction by Learning an Invariant Mapping.” in: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. vol. 2. IEEE. 2006, pp. 1735–1742 (cit. on p. 19).
- [He+15a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition.” in: *arXiv Preprint arXiv:1512.03385* (2015) (cit. on p. 37).
- [He+15b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving Deep Into Rectifiers: Surpassing Human-level Performance on Imagenet Classification.” in: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1026–1034 (cit. on pp. 15, 34, 179).
- [He+15c] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916 (cit. on pp. 43, 44).
- [He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778 (cit. on pp. 79, 143, 161, 162, 181).
- [He+17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask R-CNN.” in: *IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 2980–2988 (cit. on pp. 46, 48, 76–78).
- [Hel+11] Christian Held, Jens Wenzel, Rike Webel, Manfred Marschall, Roland Lang, Ralf Palmisano, and Thomas Wittenberg. “Using Multimodal Information for the Segmentation of Fluorescent Micrographs With Application to Virology and Microbiology.” in: *IEEE International Conference on Engineering in Medicine and Biology*. IEEE. 2011, pp. 6487–6490 (cit. on p. 76).
- [HGC10] Wei Huang, Yan Gao, and Kap Luk Chan. “A Review of Region-based Image Retrieval.” in: *Journal of Signal Processing Systems* 59 (2010), pp. 143–161 (cit. on p. 118).
- [HKL12] Alan Hanjalic, Christoph Kofler, and Martha Larson. “Intent and Its Discontents.” in: 2012 (cit. on p. 94).
- [HMS17] Ryota Hinami, Yusuke Matsui, and Shin’ichi Satoh. “Region-based Image Retrieval Revisited.” in: *Proceedings of the 25th ACM International Conference on Multimedia*. 2017, pp. 528–536 (cit. on p. 118).
- [Hoi+04] Derek Hoiem, Rahul Sukthankar, Henry Schneiderman, and Larry Huston. “Object-based Image Retrieval Using the Statistical Structure of Images.” in: *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*. vol. 2. IEEE. 2004, pp. II–II (cit. on p. 118).
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-term Memory.” in: *Neural Computation* 9.8 (1997), pp. 1735–1780 (cit. on pp. 16, 34).

- [Hu+20] Benyi Hu, Ren-Jie Song, Xiu-Shen Wei, Yazhou Yao, Xian-Sheng Hua, and Yuehu Liu. “PyRetri: A PyTorch-based Library for Unsupervised Image Retrieval by Deep Convolutional Neural Networks.” in: *Proceedings of the 28th ACM International Conference on Multimedia*. ACM, 2020, pp. 4461–4464. doi: 10.1145/3394171.3414537. URL: <https://doi.org/10.1145/3394171.3414537> (cit. on p. 179).
- [Hua+17] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. “Speed/Accuracy Trade-offs for Modern Convolutional Object Detectors.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7310–7311 (cit. on p. 38).
- [Hud+16] Marco A. Hudelist, Claudiu Cobârzan, Christian Beecks, Rob van de Werken, Sabrina Kletz, Wolfgang Hürst, and Klaus Schoeffmann. “Collaborative Video Search Combining Video Retrieval With Human-based Visual Inspection.” in: *22nd International Conference on Multimedia Modelling, Miami, FL, USA, 2016*. Springer International Publishing, 2016, pp. 400–405 (cit. on p. 161).
- [Ilh+21] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. *OpenCLIP*. version 0.1. July 2021. doi: 10.5281/zenodo.5143773. URL: <https://doi.org/10.5281/zenodo.5143773> (cit. on p. 121).
- [IS15] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” in: *International Conference on Machine Learning*. pmlr. 2015, pp. 448–456 (cit. on pp. 33, 80).
- [JB16] R. Jarrar and M. Belkhatir. “On the Coupled Use of Signal and Semantic Concepts to Bridge the Semantic and User Intention Gaps for Visual Content Retrieval.” in: *International Journal of Multimedia Information Retrieval* 5.3 (2016), pp. 165–172. issn: 2192662X (cit. on p. 95).
- [JCQ23] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *YOLO by Ultralytics*. version 8.0.0. Jan. 2023. URL: <https://github.com/ultralytics/ultralytics> (cit. on pp. 15, 38, 120).
- [JD]19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. “Billion-scale Similarity Search With GPUs.” in: *IEEE Transactions on Big Data* (2019) (cit. on pp. 54, 69, 104, 112, 119, 122, 128, 182, 185).
- [JDS10] Herve Jegou, Matthijs Douze, and Cordelia Schmid. “Product Quantization for Nearest Neighbor Search.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.1 (2010), pp. 117–128 (cit. on pp. 54, 104, 118, 182, 185).
- [Jia+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. “Caffe: Convolutional Architecture for Fast Feature Embedding.” in: *Proceedings ACM International Conference on Multimedia*. 2014, pp. 675–678 (cit. on pp. 143, 161).
- [Jin+04] Feng Jing, Mingjing Li, Hong-Jiang Zhang, and Bo Zhang. “An Efficient and Effective Region-based Image Retrieval Framework.” in: *IEEE Transactions on Image Processing* 13.5 (2004), pp. 699–709 (cit. on p. 118).

- [JL10] Vidit Jain and Erik G Learned-Miller. “Fdadb: A Benchmark for Face Detection in Unconstrained Settings.” in: *UMass Amherst Technical Report* (2010) (cit. on p. 164).
- [JL17] Huaizu Jiang and Erik Learned-Miller. “Face Detection With the Faster R-CNN.” in: *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*. IEEE. 2017, pp. 650–657 (cit. on p. 164).
- [Joh19] Jeremiah W Johnson. “Automatic Nucleus Segmentation With Mask-RCNN.” in: *Science and Information Conference*. Springer. 2019, pp. 399–407 (cit. on p. 77).
- [Jun+17] Anna Lena Jung, Christina Elena Herkt, Christine Schulz, Kathrin Bolte, Kerstin Seidel, Nicoletta Scheller, Alexandra Sittka-Stark, Wilhelm Bertrams, and Bernd Schmeck. “Legionella Pneumophila Infection Activates Bystander Cells Differentially by Bacterial and Host Cell Vesicles.” in: *Scientific Reports* 7.1 (2017), p. 6301 (cit. on p. 77).
- [JVZ14] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. “Deep Features for Text Spotting.” in: *European Conference on Computer Vision*. Springer. 2014, pp. 512–528 (cit. on pp. 59, 62).
- [KB15] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” in: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. 2015 (cit. on pp. 20, 24, 182, 191).
- [KH11] Alex Krizhevsky and Ge Hinton. “Using Very Deep Autoencoders for Content-based Image Retrieval.” in: *Proceedings Europ. Symposium on Artificial Neural Networks*. 2011, pp. 1–7 (cit. on pp. 144, 161).
- [Kin09] Davis E King. “Dlib-Ml: A Machine Learning Toolkit.” in: *The Journal of Machine Learning Research* 10 (2009), pp. 1755–1758 (cit. on p. 184).
- [Kir+19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. “Panoptic Segmentation.” in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9404–9413 (cit. on pp. 46, 116).
- [Kir+23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. “Segment Anything.” in: *arXiv Preprint arXiv:2304.02643* (2023) (cit. on pp. 49, 117, 121).
- [KL70] B. W. Kernighan and S. Lin. “An Efficient Heuristic Procedure for Partitioning Graphs.” in: *The Bell System Technical Journal* 49.2 (1970), pp. 291–307 (cit. on pp. 106, 183).
- [Kla+15] Brendan F Klare, Ben Klein, Emma Taborsky, Austin Blanton, Jordan Cheney, Kristen Allen, Patrick Grother, Alan Mah, Mark Burge, and Anil K Jain. “Pushing the Frontiers of Unconstrained Face Detection and Recognition: IARPA Janus Benchmark A.” in: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015, pp. 1931–1939 (cit. on p. 164).
- [KLH16] Christoph Kofler, Martha Larson, and Alan Hanjalic. “User Intent in Multimedia Search: A Survey of the State of the Art and Future Challenges.” in: *ACM Computing Surveys* 49.2 (2016), pp. 1–37. ISSN: 0360-0300 (cit. on p. 94).

- [KMF20] Nikolaus Korfhage, Markus Mühling, and Bernd Freisleben. “Intentional Image Similarity Search.” in: *Artificial Neural Networks in Pattern Recognition: 9th IAPR TC3 Workshop, ANNPR 2020, Winterthur, Switzerland, September 2–4, 2020, Proceedings 9*. vol. 12294 LNAI. Springer. Springer Science and Business Media Deutschland GmbH, 2020, pp. 23–35. doi: 10.1007/978-3-030-58309-5_2 (cit. on pp. 5, 94, 117, 182, 193).
- [KMF21] Nikolaus Korfhage, Markus Mühling, and Bernd Freisleben. “ElasticHash: Semantic Image Similarity Search by Deep Hashing With Elasticsearch.” in: *Computer Analysis of Images and Patterns: 19th International Conference, CAIP 2021, Virtual Event, September 28–30, 2021, Proceedings, Part II 19*. Springer. 2021, pp. 14–23 (cit. on pp. 5, 7, 52, 103, 182).
- [KMF24] Nikolaus Korfhage, Markus Mühling, and Bernd Freisleben. “Search Anything: Segmentation-based Similarity Search via Masked Region Prompts.” in: *Submitted; Under Review*. 2024 (cit. on pp. 5, 116).
- [KNJ14] Vijay Kumar, Anoop M Namboodiri, and CV Jawahar. “Face Recognition in Videos by Label Propagation.” in: *22nd International Conference on Pattern Recognition (ICPR)*. IEEE. 2014, pp. 303–308 (cit. on p. 172).
- [Kor+20] Nikolaus Korfhage, Markus Mühling, Stephan Ringshandl, Anke Becker, Bernd Schmeck, and Bernd Freisleben. “Detection and Segmentation of Morphologically Complex Eukaryotic Cells in Fluorescence Microscopy Images via Feature Pyramid Fusion.” in: *PLOS Computational Biology* 16.9 (9 Sept. 2020), e1008179. issn: 15537358. doi: 10.1371/JOURNAL.PCBI.1008179 (cit. on pp. 5, 7, 76).
- [Kor+24] Nikolaus Korfhage, Hicham Bellafkir, Markus Mühling, Markus Vogelbacher, Elton Prifti, and Bernd Freisleben. “Deep Learning for Textual Stamp Recognition on Index Cards of the Lessico Etimologico Italiano.” in: *Submitted; Under Review*. 2024 (cit. on pp. viii, 5, 7, 62).
- [KPK03] Sungyoung Kim, Soyoun Park, and Minhwan Kim. “Central Object Extraction for Object-based Image Retrieval.” in: *Second International Conference on Image and Video Retrieval, Urbana-Champaign, IL, USA, July 24–25, 2003 Proceedings 2*. Springer. 2003, pp. 39–49 (cit. on p. 118).
- [Kra+13] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. “3d Object Representations for Fine-grained Categorization.” in: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2013, pp. 554–561 (cit. on p. 123).
- [Kra+20] Miroslav Kratochvíl, Frantisek Mejzlík, Patrik Veselý, Tomáš Soucek, and Jakub Lokoc. “SOMHunter: Lightweight Video Search System with SOM-guided Relevance Feedback.” in: *Proceedings of the 28th International Conference on Multimedia (MM)*. ACM, 2020, pp. 4481–4484. doi: 10.1145/3394171.3414542. url: <https://doi.org/10.1145/3394171.3414542> (cit. on p. 179).
- [KRS20] Muhammad Kashif, Gulistan Raja, and Furqan Shaukat. “An Efficient Content-based Image Retrieval System for the Diagnosis of Lung Diseases.” in: *Journal of Digital Imaging* 33.4 (2020), pp. 971–987 (cit. on pp. 1, 115, 116).

- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification With Deep Convolutional Neural Networks.” in: *Advances in Neural Information Processing Systems* (2012), pp. 1–9 (cit. on pp. 13, 27, 32, 104, 143, 160).
- [Kum+09] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. “Attribute and Simile Classifiers for Face Verification.” in: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 365–372 (cit. on p. 171).
- [Kuz+20] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Mallocci, Alexander Kolesnikov, et al. “The Open Images Dataset V4.” in: *International Journal of Computer Vision* 128.7 (2020), pp. 1–26 (cit. on pp. 51, 111, 179).
- [LE22] Timo Lüddecke and Alexander Ecker. “Image Segmentation Using Text and Image Prompts.” in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7086–7096 (cit. on p. 117).
- [Lea14] Gary B. Huang Erik Learned-Miller. *Labeled Faces in the Wild: Updates and New Reporting Procedures*. tech. rep. UM-CS-2014-003. University of Massachusetts, Amherst, 2014 (cit. on p. 164).
- [LeC+12] Yann a. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus Robert Müller. “Efficient Backprop.” in: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7700 LECTU (2012), pp. 9–48. doi: 10.1007/978-3-642-35289-8-3 (cit. on p. 12).
- [LeC+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based Learning Applied to Document Recognition.” in: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 25).
- [Lev66] Vladimir I Levenshtein. “Binary Codes Capable of Correcting Deletions, Insertions, and Reversals.” in: *Soviet Physics Doklady*. vol. 10. 8. 1966, pp. 707–710 (cit. on p. 146).
- [LH16] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent With Warm Restarts.” in: *International Conference on Learning Representations*. 2016 (cit. on p. 31).
- [LH18] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization.” in: *International Conference on Learning Representations*. 2018 (cit. on p. 121).
- [Li+20] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. “Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection.” in: *Advances in Neural Information Processing Systems* 33 (2020), pp. 21002–21012 (cit. on p. 44).
- [Lia+16] Ru-Ze Liang, Lihui Shi, Haoxiang Wang, Jiandong Meng, Jim Jing-Yan Wang, Qingquan Sun, and Yi Gu. “Optimizing Top Precision Performance Measure of Content-based Image Retrieval by Learning Similarity Function.” in: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 2954–2958 (cit. on p. 97).

Bibliography

- [Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. “Microsoft COCO: Common Objects in Context.” in: *European Conference on Computer Vision*. Springer. 2014, pp. 740–755 (cit. on pp. 65, 71, 84, 104, 122, 125, 126).
- [Lin+15] Kevin Lin, Huei-fang Yang, Jen-hao Hsiao, and Chu-song Chen. “Deep Learning of Binary Hash Codes for Fast Image Retrieval.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 27–35 (cit. on pp. 118, 144, 145, 161, 162).
- [Lin+17a] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. “Feature Pyramid Networks for Object Detection.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2117–2125 (cit. on pp. 41, 62, 65, 77, 78, 119).
- [Lin+17b] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. “Focal Loss for Dense Object Detection.” in: *Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017)*, pp. 2980–2988 (cit. on pp. 45, 181).
- [Lin+23] Fengyin Lin, Mingkang Li, Da Li, Timothy Hospedales, Yi-Zhe Song, and Yong-gang Qi. “Zero-shot Everything Sketch-based Image Retrieval, and in Explainable Style.” in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 23349–23358 (cit. on p. 118).
- [Liu+16a] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. “Deep Supervised Hashing for Fast Image Retrieval.” in: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2064–2072 (cit. on p. 104).
- [Liu+16b] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. “SSD: Single Shot Multibox Detector.” in: *European Conference on Computer Vision*. Springer. 2016, pp. 21–37 (cit. on pp. 42, 164).
- [Liu+18] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. “Progressive Neural Architecture Search.” in: *European Conference on Computer Vision*. 2018 (cit. on pp. 83, 181).
- [Liu+23] Bing Liu, Yixin Jia, Luyang Liu, Yuanyuan Dang, and Shinan Song. “Skip DETR: End-to-end Skip Connection Model for Small Object Detection in Forestry Pest Dataset.” in: *Frontiers in Plant Science* 14 (2023) (cit. on p. 45).
- [Lok+19] Jakub Lokoc, Klaus Schoeffmann, Werner Bailer, Luca Rossetto, and Cathal Gurrin. “Interactive Video Retrieval in the Age of Deep Learning.” in: *Proceedings of the International Conference on Multimedia Retrieval (ICMR)*. ACM, 2019, pp. 2–4. doi: 10.1145/3323873.3326588. URL: <https://doi.org/10.1145/3323873.3326588> (cit. on p. 179).
- [Low99] David G. Lowe. “Object Recognition From Local Scale-invariant Features.” in: *Proceedings of the 7th IEEE International Conference on Computer Vision*. vol. 2. 8. 1999, pp. 1150–1157 (cit. on pp. 51, 97, 143, 151, 160).
- [LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440 (cit. on p. 76).

- [Luo+23] Xiao Luo, Haixin Wang, Daqing Wu, Chong Chen, Minghua Deng, Jianqiang Huang, and Xian-Sheng Hua. “A Survey on Deep Hashing Methods.” in: *ACM Transactions on Knowledge Discovery From Data* 17.1 (2023), pp. 1–50 (cit. on pp. 118, 119).
- [LWL11] Lingqiao Liu, Lei Wang, and Xinwang Liu. “In Defense of Soft-assignment Coding.” in: *Proceedings of the 13th IEEE International Conference on Computer Vision (ICCV’11)*. 2011, pp. 2486–2493 (cit. on p. 151).
- [Mai+17] Long Mai, Hailin Jin, Zhe Lin, Chen Fang, Jonathan Brandt, and Feng Liu. “Spatial-semantic Image Search by Visual Feature Synthesis.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4718–4727 (cit. on p. 118).
- [Mas+16] Iacopo Masi, Anh Tuan Tran, Tal Hassner, Jatuporn Toy Leksut, and Gérard Medioni. “Do We Really Need to Collect Millions of Faces for Effective Face Recognition?” in: *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11–14, 2016, Proceedings, Part v 14*. Springer. 2016, pp. 579–596 (cit. on p. 164).
- [Mat+04] J. Matas, O. Chum, M. Urban, and T. Pajdla. “Robust Wide-baseline Stereo From Maximally Stable Extremal Regions.” in: *Image and Vision Computing* 22.10 (2004), pp. 761–767 (cit. on p. 146).
- [Mat+18] Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin’ichi Satoh. “A Survey of Product Quantization.” in: *ITE Transactions on Media Technology and Applications* 6.1 (2018), pp. 2–10 (cit. on p. 54).
- [MB11] Barbora Micenková and Joost van Beusekom. “Stamp Detection in Color Document Images.” in: *2011 International Conference on Document Analysis and Recognition*. IEEE. 2011, pp. 1125–1129 (cit. on p. 63).
- [MBS12] Barbora Micenková, Joost van Beusekom, and Faisal Shafait. “Stamp Verification for Automated Document Authentication.” in: *Computational Forensics*. 2012, pp. 117–129 (cit. on p. 63).
- [MEF11] Markus Mühling, Ralph Ewerth, and Bernd Freisleben. “On the Spatial Extents of SIFT Descriptors for Visual Concept Detection.” in: *Proceedings of the 8th International Conference on Computer Vision Systems (ICVS ’11)*. Springer, 2011, pp. 71–80 (cit. on p. 151).
- [MEF15] Markus Mühling, Ralph Ewerth, and Bernd Freisleben. “Improving Cross-domain Concept Detection via Object-based Features.” in: *Proceedings of International Conference on Computer Analysis of Images and Patterns (CAIP ’15)*. 2015 (cit. on p. 151).
- [MG02] Gary Marchionini and Gary Geisler. “The Open Video Digital Library.” in: *D-Lib Magazine* 8.12 (2002), pp. 1082–9873 (cit. on p. 139).
- [MKA16] Mohamed Meddeb, Hichem Karray, and Adel M. Alimi. “Content-based Arabic Speech Similarity Search and Emotion Detection.” in: *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics*. ed. by Aboul Ella Hassanien, Khaled Shaalan, Tarek Gaber, Ahmad Taher Azar, and M. F. Tolba. Springer International Publishing, 2016, pp. 530–539 (cit. on p. 161).

- [Mu+19] Cun Matthew Mu, Jun Raymond Zhao, Guang Yang, Binwei Yang, and Zheng John Yan. “Fast and Exact Nearest Neighbor Search in Hamming Space on Full-text Search Engines.” in: *International Conference on Similarity Search and Applications*. Springer, 2019, pp. 49–56 (cit. on pp. 104, 108, 111).
- [Müh+07] Markus Mühling, Ralph Ewerth, Thilo Stadelmann, Christian Zöfel, Bing Shi, and Bernd Freisleben. “University of Marburg at TRECVID 2007: Shot Boundary Detection and High Level Feature Extraction.” in: *Trecvid. 2007* (cit. on p. 180).
- [Müh+11] Markus Mühling, Ralph Ewerth, Bing Shi, and Bernd Freisleben. “Multi-class Object Detection With Hough Forests Using Local Histograms of Visual Words.” in: *Proceedings of 14th International Conference on Computer Analysis of Images and Patterns (CAIP '11)*. Springer, 2011, pp. 386–393 (cit. on p. 151).
- [Müh+17] Markus Mühling, Nikolaus Korfhage, Eric Müller, Christian Otto, Matthias Springstein, Thomas Langelage, Uli Veith, Ralph Ewerth, and Bernd Freisleben. “Deep Learning for Content-based Video Retrieval in Film and Television Production.” in: *Multimedia Tools and Applications* 76 (21 Nov. 2017), pp. 22169–22194. ISSN: 15737721. DOI: 10.1007/s11042-017-4962-9 (cit. on pp. 5, 159).
- [Müh+19] Markus Mühling, Manja Meister, Nikolaus Korfhage, Jörg Wehling, Angelika Hörth, Ralph Ewerth, and Bernd Freisleben. “Content-based Video Retrieval in Historical Collections of the German Broadcasting Archive.” in: *International Journal on Digital Libraries* 20 (2019), pp. 167–183 (cit. on pp. 5, 115, 140, 180).
- [Müh+20] Markus Mühling, Jakob Franz, Nikolaus Korfhage, and Bernd Freisleben. “Bird Species Recognition via Neural Architecture Search.” in: ed. by L. Cappellato, C. Eickhoff, N. Ferro, and A. Névóol. CEUR-WS.org, 2020. URL: https://ceur-ws.org/Vol-2696/paper_188.pdf (cit. on p. 6).
- [Müh+22] Markus Mühling, Nikolaus Korfhage, Kader Pustu-Iren, Joanna Bars, Mario Knapp, Hicham Bellafkir, Markus Vogelbacher, Daniel Schneider, Angelika Hörth, Ralph Ewerth, and Bernd Freisleben. “VIVA: Visual Information Retrieval in Video Archives.” in: *International Journal on Digital Libraries* 23.4 (2022), pp. 319–333 (cit. on pp. 5, 7, 115, 178).
- [Mül87] Bodo Müller. *Diccionario Del Español Medieval*. 1987 (cit. on p. 62).
- [Nes83] Yurii Nesterov. “A Method for Unconstrained Convex Minimization Problem With the Rate of Convergence $O(1/K^2)$.” in: *Doklady an SSSR*. vol. 269. 3. 1983, pp. 543–547 (cit. on pp. 24, 163).
- [Ngu+20] Phuong Anh Nguyen, Jiaxin Wu, Chong-Wah Ngo, Danny Francis, and Benoit Huet. “VIREO@ Video Browser Showdown 2020.” in: *Proceedings of the 26th International Conference on MultiMedia Modeling (MMM)*. vol. 11962. Lecture Notes in Computer Science. Springer, 2020, pp. 772–777. DOI: 10.1007/978-3-030-37734-2_68. URL: https://doi.org/10.1007/978-3-030-37734-2_68 (cit. on p. 179).
- [NH10] Vinod Nair and Geoffrey E Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines.” in: *Proceedings of the 27th International Conference on Machine Learning* 3 (2010), pp. 807–814 (cit. on p. 13).

- [NHH15] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. “Learning Deconvolution Network for Semantic Segmentation.” in: *IEEE International Conference on Computer Vision*. 2015, pp. 1520–1528 (cit. on p. 76).
- [NM12] L. Neumann and J. Matas. “Real-time Scene Text Localization and Recognition.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 1–8 (cit. on p. 146).
- [NPF12] Mohammad Norouzi, Ali Punjani, and David J Fleet. “Fast Search in Hamming Space With Multi-index Hashing.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2012, pp. 3108–3115 (cit. on pp. 55, 97, 104, 106, 182, 183).
- [OPM02] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. “Multiresolution Gray-scale and Rotation Invariant Texture Classification With Local Binary Patterns.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), pp. 971–987 (cit. on p. 164).
- [OT01] Aude Oliva and Antonio Torralba. “Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope.” in: *International Journal of Computer Vision* 42.3 (2001), pp. 145–175 (cit. on p. 97).
- [Ots79] Nobuyuki Otsu. “A Threshold Selection Method from Gray-level Histograms.” in: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66 (cit. on p. 46).
- [OWJ16] Charles Otto, Dayong Wang, and Anil K Jain. “Clustering Millions of Faces by Identity.” in: *arXiv Preprint arXiv:1604.00989* (2016) (cit. on p. 175).
- [OWS13] Enrique G Ortiz, Alan Wright, and Mubarak Shah. “Face Recognition in Movie Trailers via Mean Sequence Sparse Representation-based Classification.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3531–3538 (cit. on pp. 159, 164, 171, 172, 175).
- [Par+12] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. “Cats and Dogs.” in: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 3498–3505 (cit. on p. 123).
- [PG17] Luca Piras and Giorgio Giacinto. “Information Fusion in Content Based Image Retrieval: A Comprehensive Overview.” in: *Information Fusion* 37 (2017), pp. 50–60. ISSN: 15662535 (cit. on p. 95).
- [PP93] Nikhil R Pal and Sankar K Pal. “A Review on Image Segmentation Techniques.” in: *Pattern Recognition* 26.9 (1993), pp. 1277–1294 (cit. on p. 46).
- [Pri22] Elton Prifti. “Un Resoconto, Con Particolare Attenzione Alla Dialettologia.” in: *Lessicografia Storica Dialettale E Regionale. Atti Del XIV Convegno ASLI (Associazione Per La Storia Della Lingua Italiana) (Milano, 5-7 Novembre 2020)* 12 (2022). ed. by Michele A. Cortelazzo, Silvia Morgana, and Massimo Prada, pp. 293–314 (cit. on p. 58).
- [PS79] E. Prifti and W. Schweickard. *LEI: Lessico Etimologico Italiano, Founded by M. Pfister*. Wiesbaden: Reichert-Verlag, 1979 (cit. on p. 58).

- [Pus+19] Kader Pustu-Iren, Markus Mühling, Nikolaus Korfhage, Joanna Bars, Sabrina Bernhöft, Angelika Hörth, Bernd Freisleben, and Ralph Ewerth. “Investigating Correlations of Inter-coder Agreement and Machine Annotation Performance for Historical Video Data.” in: *Digital Libraries for Open Knowledge: 23rd International Conference on Theory and Practice of Digital Libraries, TPDL 2019, Oslo, Norway, September 9-12, 2019, Proceedings* 23. vol. 11799 LNCS. Springer. Springer Verlag, 2019, pp. 107–114. doi: 10.1007/978-3-030-30760-8_9 (cit. on pp. 6, 179).
- [PVZ15] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. “Deep Face Recognition.” in: *British Machine Vision Conference*. 2015, pp. 1–6 (cit. on pp. 163, 164).
- [Qay+17] Adnan Qayyum, Syed Muhammad Anwar, Muhammad Awais, and Muhammad Majid. “Medical Image Retrieval Using Deep Convolutional Neural Network.” in: *Neurocomputing* 266 (2017), pp. 8–20 (cit. on pp. 1, 115).
- [Rad+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. “Learning Transferable Visual Models From Natural Language Supervision.” in: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763 (cit. on pp. 49, 50, 116, 117, 121).
- [RCC20] Josiane Rodrigues, Marco Cristo, and Juan G Colonna. “Deep Hashing for Multi-label Image Retrieval: A Survey.” in: *Artificial Intelligence Review* 53.7 (2020), pp. 5261–5307 (cit. on p. 182).
- [Red+16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You Only Look Once: Unified, Real-time Object Detection.” in: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788 (cit. on pp. 38, 42).
- [Ren+15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-time Object Detection With Region Proposal Networks.” in: *Advances in Neural Information Processing Systems*. 2015, pp. 91–99 (cit. on pp. 38, 65, 77, 78, 164).
- [RF17] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7263–7271 (cit. on p. 43).
- [RF18] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement.” in: *arXiv Preprint arXiv:1804.02767* (2018) (cit. on pp. 38, 43).
- [RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” in: *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer. 2015, pp. 234–241 (cit. on pp. 37, 46, 76, 81, 83).
- [RHW88] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-propagating Errors.” in: *Neurocomputing: Foundations of Research, James A. Anderson and Edward Rosenfeld (Eds.)*. MIT Press, Cambridge, MA, USA 696-699 (1988) (cit. on p. 19).
- [RPL09] Partha Pratim Roy, Umapada Pal, and Josep Lladós. “Seal Detection and Recognition: An Approach for Document Indexing.” in: *10th International Conference on Document Analysis and Recognition*. IEEE. 2009, pp. 101–105 (cit. on p. 62).

- [RSS15] Jason A Reuter, Damek V Spacek, and Michael P Snyder. “High-throughput Sequencing Technologies.” in: *Molecular Cell* 58.4 (2015), pp. 586–597 (cit. on p. 75).
- [Rus+08] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. “LabelMe: A Database and Web-based Tool for Image Annotation.” in: *International Journal of Computer Vision* 77.1-3 (2008), pp. 157–173 (cit. on p. 179).
- [Rus+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. “Imagenet Large Scale Visual Recognition Challenge.” in: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252 (cit. on pp. 15, 65, 67).
- [Sai+23] Aneeshan Sain, Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Subhadeep Koley, Tao Xiang, and Yi-Zhe Song. “Clip for All Things Zero-shot Sketch-based Image Retrieval, Fine-grained or Not.” in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 2765–2775 (cit. on p. 118).
- [SBY16] Baoguang Shi, Xiang Bai, and Cong Yao. “An End-to-end Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.11 (2016), pp. 2298–2304 (cit. on p. 59).
- [Sch+12] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. “Fiji: An Open-source Platform for Biological-image Analysis.” in: *Nature Methods* 9.7 (2012), pp. 676–682 (cit. on p. 78).
- [Sch+18] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. “Cell Detection With Star-convex Polygons.” in: *Medical Image Computing and Computer Assisted Intervention (MICCAI)* (2018) (cit. on pp. 76, 83).
- [Sch+21] Daniel Schneider, Nikolaus Korfhage, Markus Mühling, Peter Lüttig, and Bernd Freisleben. “Automatic Transcription of Organ Tablature Music Notation With Deep Neural Networks.” in: *Transactions of the International Society for Music Information Retrieval* 4 (1 Feb. 2021), pp. 14–28. issn: 25143298. doi: 10.5334/tismir.77 (cit. on p. 6).
- [Set99] James Albert Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. vol. 3. Cambridge University Press, 1999 (cit. on p. 76).
- [SH09] Ruslan Salakhutdinov and Geoffrey Hinton. “Semantic Hashing.” in: *International Journal of Approximate Reasoning* 50.7 (2009), pp. 969–978 (cit. on pp. 54, 144, 161).
- [She+18] Fumin Shen, Yan Xu, Li Liu, Yang Yang, Zi Huang, and Heng Tao Shen. “Unsupervised Deep Hashing With Similarity-adaptive and Discrete Optimization.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.12 (2018), pp. 3034–3044 (cit. on p. 118).
- [Sib73] Robin Sibson. “SLINK: An Optimally Efficient Algorithm for the Single-link Cluster Method.” in: *The Computer Journal* 16.1 (1973), pp. 30–34 (cit. on p. 66).

Bibliography

- [Sil+22] Wilson Silva, Tiago Gonçalves, Kirsi Härmä, Erich Schröder, Verena Carola Obmann, María Cecilia Barroso, Alexander Poellinger, Mauricio Reyes, and Jaime S Cardoso. “Computer-aided Diagnosis Through Medical Image Retrieval in Radiology.” in: *Scientific Reports* 12.1 (2022), p. 20732 (cit. on pp. 1, 115).
- [SKP15a] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “FaceNet: A Unified Embedding for Face Recognition and Clustering.” in: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. vol. 07-12-June. 2015, pp. 815–823 (cit. on p. 98).
- [SKP15b] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A Unified Embedding for Face Recognition and Clustering.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 815–823 (cit. on pp. 106, 143, 163, 164, 183, 184).
- [Sme+00a] A W Smeulders, M Worring, S Santini, A Gupta, and R Jain. “Content-based Image Retrieval at the End of the Early Years.” in: *IEEE Trans. On Pattern Analysis and Machine Intelligence* 22.12 (2000), pp. 1349–1380 (cit. on p. 2).
- [Sme+00b] Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. “Content-based Image Retrieval at the End of the Early Years.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.12 (2000), pp. 1349–1380 (cit. on pp. 92, 142, 160, 179).
- [Smi07] Ray Smith. “An Overview of the Tesseract OCR Engine.” in: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. vol. 2. IEEE. 2007, pp. 629–633 (cit. on pp. 66, 71).
- [Sol+17] José Alonso Solís-Lemus, Brian Stramer, Greg Slabaugh, and Constantino Carlos Reyes-Aldasoro. “Segmentation and Shape Analysis of Macrophages Using Anglegram Analysis.” in: *Journal of Imaging* 4.1 (2017), p. 2 (cit. on p. 76).
- [Som+19] N. Sompawong, J. Mopan, P. Pooprasert, W. Himakhun, K. Suwannarurk, J. Ngamvirojcharoen, T. Vachiramon, and C. Tantibundhit. “Automated Pap Smear Cervical Cancer Screening Using Deep Learning.” in: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (2019), pp. 7044–7048. ISSN: 1557170X. DOI: 10.1109/EMBC.2019.8856369 (cit. on p. 77).
- [Son08] Min Song. *Handbook of Research on Text and Web Mining Technologies*. IGI Global, 2008 (cit. on p. 174).
- [SP14] Harald Sack and Margret Plank. “AV-Portal: The German National Library of Science and Technology’s Semantic Video Portal.” in: *ERCIM News* 96 (2014) (cit. on p. 139).
- [Spr+21] Matthias Springstein, Stefanie Schneider, Javad Rahnema, Eyke Hüllermeier, Hubertus Kohle, and Ralph Ewerth. “iART: A Search Engine for Art-historical Images to Support Research in the Humanities.” in: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 2801–2803 (cit. on pp. 115, 118, 137).

- [Sri+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks From Overfitting.” in: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958 (cit. on p. 32).
- [Ste+96] Wolf-Dieter Stempel, Helmut Stimm, Claudia Kraus, Renate Peter, and Monika Tausend. *Dictionnaire De l’Occitan Médiéval: DOM*. Niemeyer, 1996 (cit. on p. 62).
- [Str+24] Finja Strehmann, Markus Vogelbacher, Clara Guckenbiehl, Yvonne Ramona Schumm, Juan Masello, Petra Quillfeldt, Nikolaus Korfhage, Hicham Bellafkir, Markus Mühling, Bernd Freisleben, Nina Farwig, Dana Schabo, and Sascha Rösner. “Intrinsic Factors Influence Physiological Stress in a Forest Bird Community: Adults and Females Have Higher H/L Ratios Than Juveniles and Males.” in: *Submitted; Under Review*. 2024 (cit. on p. 6).
- [Su+18] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. “Greedy Hash: Towards Fast Optimization for Accurate Hash Coding in CNN.” in: *Advances in Neural Information Processing Systems* 31 (2018) (cit. on pp. 54, 121).
- [Sun+15] Yi Sun, Ding Liang, Xiaogang Wang, and Xiaoou Tang. “Deepid3: Face Recognition With Very Deep Neural Networks.” in: *arXiv Preprint arXiv:1502.00873* (2015) (cit. on p. 164).
- [Sut+13] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. “On the Importance of Initialization and Momentum in Deep Learning.” in: *30th International Conference on Machine Learning*. vol. 28. 2013, pp. 1139–1147 (cit. on pp. 24, 163).
- [SZ14] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-scale Image Recognition.” in: *arXiv Preprint arXiv:1409.1556* (2014) (cit. on p. 164).
- [Sze+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. “Going Deeper With Convolutions.” in: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9 (cit. on pp. 27, 143, 160, 181).
- [Tai+14] Yaniv Taigman, Ming Yang, Marc’Aurelio Ranzato, and Lior Wolf. “DeepFace: Closing the Gap to Human-level Performance in Face Verification.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 1–8 (cit. on pp. 143, 160).
- [TC23] Juan Terven and Diana Cordova-Esparza. “A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond.” in: *arXiv Preprint arXiv:2304.00501* (2023) (cit. on p. 43).
- [Tia+19] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. “FCOS: Fully Convolutional One-stage Object Detection.” in: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9627–9636 (cit. on p. 43).
- [TL19] Mingxing Tan and Quoc Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” in: *Proceedings of the International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114 (cit. on pp. 15, 62, 67, 105, 181).

- [Tur+22] Osman Tursun, Simon Denman, Sridha Sridharan, Ethan Goan, and Clinton Fookes. “An Efficient Framework for Zero-shot Sketch-based Image Retrieval.” in: *Pattern Recognition* 126 (2022), p. 108528 (cit. on p. 118).
- [Ued95] Katsuhiko Ueda. “Extraction of Signature and Seal Imprint From Bankchecks by Using Color Information.” in: *3rd International Conference on Document Analysis and Recognition*. vol. 2. IEEE. 1995, pp. 665–668 (cit. on p. 63).
- [Uij+13] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. “Selective Search for Object Recognition.” in: *International Journal of Computer Vision* 104 (2013), pp. 154–171 (cit. on p. 39).
- [Usa+16] Mojca Mattiazzi Usaj, Erin B Styles, Adrian J Verster, Helena Friesen, Charles Boone, and Brenda J Andrews. “High-content Screening for Quantitative Cell Biology.” in: *Trends in Cell Biology* 26.8 (2016), pp. 598–611 (cit. on p. 75).
- [VAK19] Aarno Oskar Vuola, Saad Ullah Akram, and Juho Kannala. “Mask-RCNN and U-Net Ensembled for Nuclei Segmentation.” in: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE. 2019, pp. 208–212 (cit. on p. 77).
- [Van+16] David A Van Valen, Takamasa Kudo, Keara M Lane, Derek N Macklin, Nicolas T Quach, Mialy M DeFelice, Inbal Maayan, Yu Tanouchi, Euan A Ashley, and Markus W Covert. “Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-cell Imaging Experiments.” in: *PLoS Computational Biology* 12.11 (2016) (cit. on p. 77).
- [Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need.” in: *Advances in Neural Information Processing Systems* 30 (2017) (cit. on pp. 16, 34, 35).
- [VBK17] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. “Conditional Similarity Networks.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 830–838 (cit. on p. 117).
- [VCM23] Sagar Vaze, Nicolas Carion, and Ishan Misra. “GeneCIS: A Benchmark for General Conditional Image Similarity.” in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 6862–6872 (cit. on p. 117).
- [VFE15] Yuber Velazco-Paredes, Roxana Flores-Quispe, and Raquel E Patino Escarcina. “Region-based Image Retrieval Using Color and Texture Features on Irregular Regions of Interest.” in: *IEEE Colombian Conference on Communication and Computing (IEEE COLCOM 2015)*. IEEE. 2015, pp. 1–6 (cit. on p. 118).
- [VJ01] P. Viola and M. Jones. “Rapid Object Detection Using a Boosted Cascade of Simple Features.” in: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. vol. 1. 2001, pp. 511–518 (cit. on pp. 146, 163).
- [Vog+24] Markus Vogelbacher, Finja Strehmann, Hicham Bellafkir, Markus Mühling, Nikolaus Korfhage, Daniel Schneider, Sascha Rösner, Dana G Schabo, Nina Farwig, and Bernd Freisleben. “Identifying and Counting Avian Blood Cells in Whole Slide Images via Deep Learning.” in: *Birds* 5.1 (2024), pp. 48–66 (cit. on p. 6).

- [Wan+13a] J. Wan, S. Tang, Y. Zhang, L. Huang, and J. Li. “Data Driven Multi-index Hashing.” in: *2013 IEEE International Conference on Image Processing*. 2013, pp. 2670–2673 (cit. on p. 108).
- [Wan+13b] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. “Regularization of Neural Networks Using Dropconnect.” in: *International Conference on Machine Learning*. PMLR. 2013, pp. 1058–1066 (cit. on p. 32).
- [Wan+14] Ji Wan, Dayong Wang, Steven C H Hoi, and Pengcheng Wu. “Deep Learning for Content-based Image Retrieval: A Comprehensive Study.” in: *Proceedings of the ACM International Conference on Multimedia (MM)*. 2014, pp. 157–166 (cit. on pp. 94, 144, 161).
- [Wan+15] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. “Learning to Hash for Indexing Big Data—A Survey.” in: *Proceedings of the IEEE* 104.1 (2015), pp. 34–57 (cit. on pp. 53, 104, 182).
- [Wan+16] Jingbin Wang, Lihui Shi, Haoxiang Wang, Jiandong Meng, Jim Jing-Yan Wang, Qingquan Sun, and Yi Gu. “Optimizing Top Precision Performance Measure of Content-based Image Retrieval by Learning Similarity Function.” in: *23rd International Conference on Pattern Recognition (ICPR)*. 2016 (cit. on p. 161).
- [Wan+17] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. “A Survey on Learning to Hash.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2017), pp. 769–790 (cit. on p. 182).
- [Wan+18a] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. “CosFace: Large Margin Cosine Loss for Deep Face Recognition.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 5265–5274 (cit. on p. 183).
- [Wan+18b] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. “A Survey on Learning to Hash.” in: *IEEE Trans. On Pattern Analysis and Machine Intelligence* 40.4 (2018), pp. 769–790 (cit. on p. 104).
- [Wan+20] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. “CSPNet: A New Backbone That Can Enhance Learning Capability of CNN.” in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 390–391 (cit. on p. 43).
- [Wan+21] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. “Milvus: A Purpose-built Vector Data Management System.” in: *Proceedings of the International Conference on Management of Data*. 2021, pp. 2614–2627 (cit. on p. 182).
- [WBB11] Kai Wang, Boris Babenko, and Serge Belongie. “End-to-end Scene Text Recognition.” in: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 1457–1464 (cit. on p. 59).
- [WBL23] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. “YOLOv7: Trainable Bag-of-freebies Sets New State-of-the-art for Real-time Object Detectors.” in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 7464–7475 (cit. on p. 38).

Bibliography

- [Wei+14] Yunchao Wei, Wei Xia, Junshi Huang, Bingbing Ni, Jian Dong, Yao Zhao, and Shuicheng Yan. “CNN: Single-label to Multi-label.” in: *arXiv Preprint arXiv* (2014), pp. 1–14. arXiv: 1406.5726 (cit. on p. 160).
- [Wen+11] Jens Wenzel, Christian Held, Ralf Palmisano, Stefan Teufel, Jean-Pierre David, Thomas Wittenberg, and Roland Lang. “Measurement of TLR-induced Macrophage Spreading by Automated Image Analysis: Differential Role of Myd88 and MAPK in Early and Late Responses.” in: *Frontiers in Physiology* 2 (2011), p. 71 (cit. on p. 76).
- [WKC12] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. “Semi-supervised Hashing for Large-scale Search.” in: *IEEE Trans. On Pattern Analysis and Machine Intelligence* 34.12 (2012), pp. 2393–2406 (cit. on p. 104).
- [Wu+19] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. *Detectron2*. <https://github.com/facebookresearch/detectron2>. 2019 (cit. on pp. 62, 65).
- [Xie+17] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. “Aggregated Residual Transformations for Deep Neural Networks.” in: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society. 2017, pp. 5987–5995 (cit. on p. 65).
- [Yan+16] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. “WIDER FACE: A Face Detection Benchmark.” in: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 123, 164).
- [Yan+19] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. “Distillhash: Unsupervised Deep Hashing by Distilling Data Pairs.” in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2946–2955 (cit. on pp. 54, 118).
- [Yea+15] Luke Yeager, Julie Bernauer, Allison Gray, and Michael Houston. “Digits: The Deep Learning GPU Training System.” in: *ICML 2015 AutoML Workshop*. 2015 (cit. on p. 179).
- [Yel+18] Sasi Kiran Yelamarthi, Shiva Krishna Reddy, Ashish Mishra, and Anurag Mittal. “A Zero-shot Framework for Sketch Based Image Retrieval.” in: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 300–317 (cit. on p. 118).
- [Yi+14] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. “Learning Face Representation From Scratch.” in: *arXiv Preprint arXiv:1411.7923* (2014) (cit. on p. 164).
- [Yia93] Peter N. Yianilos. “Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces.” in: *Annual ACM-SIAM Symposium on Discrete Algorithms*. 1993, pp. 311–321 (cit. on pp. 145, 162).
- [YK15] Fisher Yu and Vladlen Koltun. “Multi-scale Context Aggregation by Dilated Convolutions.” in: *arXiv Preprint arXiv:1511.07122* (2015) (cit. on p. 30).

- [You+17] Junaid Younas, Muhammad Zeshan Afzal, Muhammad Imran Malik, Faisal Shafait, Paul Lukowicz, and Sheraz Ahmed. “D-Star: A Generic Method for Stamp Segmentation From Document Images.” in: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. vol. 1. IEEE. 2017, pp. 248–253 (cit. on pp. 59, 63).
- [Zaa+20] Wala Zaaboub, Lotfi Tlig, Mounir Sayadi, and Basel Solaiman. “Neural Network-based System for Automatic Passport Stamp Classification.” in: *Information Technology and Control* 49.4 (2020), pp. 583–607 (cit. on p. 62).
- [Zei+10] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. “Deconvolutional Networks.” in: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 2528–2535 (cit. on p. 47).
- [ZF14] Matthew D Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks.” in: *European Conference on Computer Vision*. Springer. 2014, pp. 818–833 (cit. on p. 164).
- [ZH03] Xiang Sean Zhou and Thomas S. Huang. “Relevance Feedback in Image Retrieval: A Comprehensive Review.” in: *Multimedia Systems* 8.6 (2003), pp. 536–544. issn: 09424962 (cit. on p. 95).
- [ZH05] Hui Zou and Trevor Hastie. “Regularization and Variable Selection via the Elastic Net.” in: *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 67.2 (2005), pp. 301–320. doi: 10.1111/j.1467-9868.2005.00503.x (cit. on p. 32).
- [Zha+13] Hanwang Zhang, Zheng Jun Zha, Yang Yang, Shuicheng Yan, Yue Gao, and Tat Seng Chua. “Attribute-augmented Semantic Hierarchy: Towards Bridging Semantic Gap and Intention Gap in Image Retrieval.” in: *Proceedings of the 21st ACM International Conference on Multimedia*. 2013, pp. 33–42 (cit. on p. 95).
- [Zha+16a] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Senior Member, Yu Qiao, and Senior Member. “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks.” in: *IEEE Signal Processing Letters* 23.10 (2016), pp. 1499–1503. issn: 10709908. eprint: 1604.02878 (cit. on p. 98).
- [Zha+16b] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. “Multi-oriented Text Detection With Fully Convolutional Networks.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4159–4167 (cit. on pp. 59, 62).
- [Zha+23] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. “Fast Segment Anything.” in: *arXiv Preprint arXiv:2306.12156* (2023) (cit. on pp. 49, 116, 117, 120, 128).
- [Zhe+20] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. “Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression.” in: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 34. 07. 2020, pp. 12993–13000 (cit. on p. 44).

- [Zho+14] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. “Learning Deep Features for Scene Recognition Using Places Database.” in: *Advances in Neural Information Processing Systems* 27 (2014), pp. 487–495. ISSN: 10495258 (cit. on pp. 143, 144, 160–162).
- [Zho+16] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. “Learning Deep Features for Discriminative Localization.” in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2921–2929 (cit. on p. 97).
- [Zho+17] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. “Places: A 10 Million Image Database for Scene Recognition.” in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6 (2017), pp. 1452–1464 (cit. on pp. 97, 105, 183).
- [Zhu+16] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. “Deep Hashing Network for Efficient Similarity Retrieval.” in: *AAAI Conference on Artificial Intelligence*. vol. 30. 2016 (cit. on p. 104).
- [ZL16] Barret Zoph and Quoc Le. “Neural Architecture Search With Reinforcement Learning.” in: *International Conference on Learning Representations*. 2016 (cit. on p. 97).