

---

# Evaluierung von Gesichtserkennungsmodellen für historische Gemälde am Beispiel ausgewählter Werke Lucas Cranachs zur optimierten Positionierung von Overlays

Bachelorarbeit zur Erlangung des akademischen Grades  
*Bachelor of Science*  
im Studiengang Medieninformatik  
an der Fakultät für Informatik und Ingenieurwissenschaften  
der Technischen Hochschule Köln

vorgelegt von: Niklas Mehlem  
Matrikel-Nr.: 111 405 18  
Adresse: Brucknerstraße 78  
53721 Siegburg  
niklas.mehlem@smail.th-koeln.de

eingereicht bei: Prof. Christian Noss  
Zweitgutachter\*in: Prof. Dr. Daniel Gaida

Siegburg, 13.06.2025

## **Kurzfassung/ *Abstract***

Eine Kurzfassung (wenn verlangt) in Deutsch und/oder in Englisch (*Abstract*) umfasst auf etwa 1/2 bis 1 Seite die Darstellung der Problemstellung, der angewandten Methode(n) und des wichtigsten Ergebnisses.

Wie man ein gelungenes Abstract verfasst, erfahren Sie in den Seminaren oder der Beratung des Schreibzentrums der Kompetenzwerkstatt<sup>1</sup>.

Schlagwörter/Schlüsselwörter: evtl. Angabe von 3 bis 10 Schlagwörtern.

---

<sup>1</sup><https://www.th-koeln.de/schreibzentrum>

# Inhaltsverzeichnis

<b>Tabellenverzeichnis</b>	<b>V</b>
<b>Abbildungsverzeichnis</b>	<b>VI</b>
<b>Glossar</b>	<b>VII</b>
<b>Abkürzungsverzeichnis</b>	<b>VIII</b>
<b>Symbolverzeichnis</b>	<b>IX</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Recherche</b>	<b>2</b>
2.1 Haar-Cascade . . . . .	2
2.2 Caffe . . . . .	3
2.3 MediaPipe . . . . .	4
2.4 Dlib HOG . . . . .	5
2.5 Dlib HOG Landmark . . . . .	6
2.6 Dlib CNN . . . . .	7
2.7 MTCNN . . . . .	8
2.8 Yunet . . . . .	9
2.9 RetinaFace (CPU) . . . . .	9
<b>3 Tests</b>	<b>11</b>
3.1 Caffe . . . . .	11
3.2 Dlib CNN . . . . .	12
3.3 MTCNN . . . . .	12
3.4 Yunet . . . . .	13
3.5 RetinaFace (CPU) . . . . .	13
<b>4 Implementierung</b>	<b>15</b>
<b>5 Fazit</b>	<b>18</b>

<b>6</b>	<b>Formaler Aufbau</b>	<b>19</b>
6.1	Reihenfolge . . . . .	19
6.2	Deckblatt . . . . .	19
6.3	Inhaltsverzeichnis . . . . .	20
6.4	Abbildungsverzeichnis und Tabellenverzeichnis . . . . .	20
6.5	Abkürzungsverzeichnis . . . . .	20
6.6	Literaturverzeichnis . . . . .	20
6.7	Rechtschreibung, Grammatik . . . . .	21
6.8	Umfang der Arbeit . . . . .	21
<b>7</b>	<b>Gestaltung: Textsatz mit L<sup>A</sup>T<sub>E</sub>X</b>	<b>22</b>
7.1	Unter der Haube . . . . .	23
7.2	Überschriften . . . . .	23
7.3	Absätze . . . . .	24
7.4	Silbentrennung . . . . .	24
7.5	Aufzählungen . . . . .	24
7.6	Abbildungen . . . . .	25
7.6.1	Bilder . . . . .	25
7.6.2	Vektorgrafiken . . . . .	26
7.7	Tabellen . . . . .	27
7.8	Abbildungs- und Tabellenverzeichnis . . . . .	27
7.9	Formeln . . . . .	27
7.10	Quellcode, Pseudocode . . . . .	28
7.11	Weitere Verzeichnisse . . . . .	29
7.11.1	Glossar erstellen . . . . .	30
7.11.2	Abkürzungsverzeichnis erstellen . . . . .	30
7.11.3	Symbolverzeichnis erstellen . . . . .	30
7.12	Verweise . . . . .	31
7.13	Besondere Abstände und Zeichen . . . . .	31
7.14	Wahl der Grundschriftart . . . . .	32
7.15	Metadaten für den pdf-Betrachter . . . . .	32
7.16	Wechsel zwischen ein- und doppelseitigem Layout . . . . .	33
7.17	Kompilieren . . . . .	33
7.18	Diese Vorlage . . . . .	34
<b>8</b>	<b>Zitate und Literaturangaben</b>	<b>36</b>
8.1	Zitieren . . . . .	36
8.1.1	Quellenverweise . . . . .	36
8.1.2	Derselbe und Ebenda . . . . .	37
8.1.3	Zitate aus zweiter Hand . . . . .	37
8.1.4	Abbildungen und Tabellen zitieren . . . . .	37

8.2	Literaturverzeichnis . . . . .	38
8.2.1	Inhalt und Anordnung der Literaturangaben . . . . .	38
8.2.2	Bibliographische Angaben im Literaturverzeichnis . . . . .	39
	<b>Literatur</b>	<b>40</b>
	<b>Anhang</b>	<b>43</b>

## Tabellenverzeichnis

7.1	Eine einfache Tabelle . . . . .	27
-----	---------------------------------	----

## Abbildungsverzeichnis

7.1	Vielleicht handelt es sich hierbei um Kunst? . . . . .	25
7.2	Eine schöne Grafik, die im Quellcode erzeugt wird! . . . . .	26
7.3	Beispiel für ein listing . . . . .	29

## Glossar

**Hund** Behaartes, vierbeiniges Säugetier. Bester Freund des Menschen. 22



## Abkürzungsverzeichnis

**Caffe** Convolutional Architecture for Fast Feature Embedding. 3, 4

**CNN** Convolutional Neural Network. 6–8

**HOG** Histogram of Oriented Gradients. 4–6

**MMOD** Max-Margin Object Detection. 6

**MTCNN** Multi-task Cascaded Convolutional Networks. 7, 8

**NMS** Non-Maximum Suppression. 7

**O-Net** Output Network. 8

**P-Net** Proposal Network. 7, 8

**R-Net** Refine Network. 7

**SSD** Single Shot MultiBox Detector. 3, 4, 8, 9

**SVM** Support Vector Machine. 5, 22

## Symbolverzeichnis

$\vec{F}$  Kraft, vektorielle Größe. 23

# 1 Einleitung

Gesichtserkennungsmodelle werden hauptsächlich mit Trainingsdaten bestehend aus Gesichtern von Realen Menschen entwickelt, und sind darauf ausgelegt Gesichter von Realen Menschen zu erkennen. Als für das Cranach Digital Archive neue Wasserzeichen entwickelt wurden, kam die Idee auf ob man diese Automatisiert auf den verschiedenen Bildern der Werke platzieren könnte, während Gesichter aus Ästhetischen Gründen frei gelassen werden sollten. Hieraus entstand die Motivation dieser Arbeit. Das Testen und Vergleichen von bestehenden Gesichtserkennungsmodellen. Herauszufinden ob sich diese für den Einsatz auf historischen Gemälden eignen und falls ja, welche sich von ihnen am besten eignen. Hierfür wurden die Modelle von Haar-Cascade, Caffe, MediaPipe, Dlib HOG, Dlib Landmark, Dlib CNN, MTCNN, Yunet und RetinaFace verglichen. Aus ausgewählten geeigneten Modellen soll dann ein Python-Modul entwickelt werden, welches Bereiche von Gesichtern ausgibt. So soll die Möglichkeit gegeben werden zu überprüfen ob sich ein Overlay auf einem Gesicht befindet.

Im Recherche Kapitel wird kurz auf die getesteten Modellen eingegangen und ihre grobe Funktionsweise erläutert um sie ihre Ergebnisse in einen Kontext zu packen. Als erster Schritt der Testreihe werden in den Stichprobentests die Verschiedenen Modelle auf ihre Eignung getestet. Es soll geschaut werden ob das getestete Modell überhaupt in der Lage ist mit historischen Gemälden zu arbeiten. Falls nicht werden sie bereits hier verworfen um in späteren Tests Zeit und Aufwand zu sparen. Danach Folgen Tests mit Bildern verschiedener Auflösungen. So soll geschaut werden ob die Auflösung eines Bildes Auswirkungen auf die Ergebnisse eines Modells hat und welche dies sind. Auch soll so bestimmt werden welche Auflösung die Bilder der folgenden Tests haben sollen, um bestmögliche Ergebnisse zu ermöglichen. Im vorletzten Test, sollen die confidence Grenzwerte der Modelle optimiert werden. So soll sichergestellt werden dass die bestmöglichen Ergebnisse der Modelle im finalen Test verglichen werden um eine möglichst sichere Entscheidung zutreffen welche Modelle für das implementierte Modul verwendet werden sollen. Der Finale Test vergleicht die False-Negative, False-Positives und Anzahl an Gesichtern die nur vom jeweiligen Modell erkannt wurden. So soll festgestellt werden welche Modelle die sichersten sind, und ebenfalls welche diese ergänzen könnten um auf möglichste jeder Art von Bild alle Gesichter zu erkennen.

## 2 Recherche

Nicht alle Gesichtserkennungsmodelle sind in der Lage überhaupt zuverlässig Gesichtern in alten Gemälden zu erkennen. Bevor also zu viel Zeitaufwand entsteht alle möglichen Modelle intensiv zu testen und mit einander zu vergleichen werden ihnen Stichprobenartig jeweils 4 Bilder vorgelegt um einzuschätzen ob diese sich für weitere Test eignen.

Recherche dient um Grob Idee zu geben und damit man weiß womit man eigentlich Arbeitet. Es soll nicht ins Detail gegangen werden da diese den Rahmen der Arbeit sprengen würde.

ERKLÄRE WIE DIE STICHPROBEN TESTS FUNKTIONIEREN

BILD Maße Normal und "Hochauflösendend"

### 2.1 Haar-Cascade

Der **Haar-Cascade-Detektor** basiert auf dem von Viola und Jones entwickelten Verfahren der schnellen Objekterkennung mittels einer kaskadierten Boosting-Architektur (Viola & Jones, 2001). Dabei werden Haar-ähnliche Merkmale erfasst, die als Differenz der Pixel-Summen benachbarter rechteckiger Regionen berechnet werden. Mithilfe eines Integralbilds lassen sich diese Merkmale dann in konstanter Zeit evaluieren. Anschließend erfolgt die Merkmalsauswahl durch den AdaBoost-Algorithmus, welcher eine Vielzahl schwacher Klassifikatoren zu einem starken Klassifikator kombiniert. Die Klassifikatoren sind in mehreren Stufen kaskadiert, wobei jede Stufe unpassende Bildbereiche frühzeitig verwirft um schneller ein Ergebnis zu erzielen. In der ursprünglichen Implementierung umfasste die Kaskade 38 Stufen, beginnend mit nur einem Merkmal in der ersten Stufe und steigend bis zu 6000 Features in späteren Stufen. Die Implementierung in dieser Arbeit erfolgt über OpenCV, welche mit der Klasse **CascadeClassifier** vortrainierte XML-Modelle lädt. Trainiert wurde das verwendete **haarcascade\_frontalface\_default.xml** Modell mit dem **Caltech Frontal Face Dataset** von 1999 (Howse, 2019, 101–102).

Haar-Cascade, genauer **haarcascade\_frontalface\_default.xml**, eignet sich in keiner Weise. Es scheitert sowohl darin zuverlässig das Gesicht in einem Portrait

zu erkennen, als auch darin mehrere Gesichter in einem Gruppen Bild zu erkennen. Zudem kommt es oft zu false-positives. Teilweise werden dadurch mehr false-positives als tatsächliche Gesichter markiert. Haar Cascade hat nicht direkt einen 'confidence' Wert den man bearbeiten kann aber ein äquivalent, jedoch ist selbst mit verschiedenen Werten Haar Cascade nicht in der Lage zuverlässige Ergebnisse zu liefern und scheidet damit bereits hier für weitere Versuche aus. Anzumerken ist ebenfalls das Haar-Cascade bei gleichen Motiv mit höherer Auflösung noch schlechter abschneidet, in dem es zu mehr false-positives kommt.

## 2.2 Caffe

**Convolutional Architecture for Fast Feature Embedding** (Caffe) ist ein Deep-Learning-Framework das zur Entwicklung und Ausführung von Modellen dient (Jia et al., 2014). Verwendet wird in dieser Arbeit `res10_300x300_ssd_iter_140000.caffemodel` zusammen mit `deploy.prototxt` zur Definierung der Parameter der Netzwerkarchitektur. Wie der Name des Modells hergibt basiert es auf dem **Single Shot MultiBox Detector** (SSD) Framework mit einem **ResNet10**-Backbone. SSDs bestehen aus einem SSD-Head und einem Backbone (Esri Developer, o.D.). Dabei ist es oft so das ein Netzwerk wie ResNet genommen wird und die finale Klassifikations-Ebene entfernt wird um daraus das Backbone zu erstellen. SSD nutzt mehrere Feature-Maps um auf diesen Default-Boxes zu platzieren, welche auf Merkmale untersucht werden (Liu et al., 2016). Die 300x300 im Namen des Modells geben an, dass das Modell mit Bildern mit 300 x 300 Pixelgröße trainiert wurde. Damit das Modell größere Eingaben verarbeiten kann wird die `blobFromImage` von OpenCV genutzt um das Bild für die Gesichtserkennung auf 300 x 300 zu skalieren (Serengil, 2020). Genaue Trainingsdaten für das Modell wurden leider nicht Dokumentiert.

Caffe besteht den Stichproben Test. In 1er, 2er und 3er Portraits schafft es jeweils alle 'Hauptgesichter' zu erkennen. Zusätzlich ist es dabei auch sehr sicher ohne ein einziges false-positiv bei allen Stichproben Tests. Allerdings scheint es dafür mit kleineren Gesichtern, oder Gesichtern die nicht Hauptfokus des Portraits sind Probleme zu haben. So wurde im Gruppen Bild nicht ein Gesicht markiert, auch kein false-positiv. Reduziert man den confidence Wert auf 0.2 so ist Caffe auch in der Lage alle Gesichter im 3er Portrait zu erkennen, weiter ohne false-positives. Das Gruppenbild hingegen bleibt der Schwachpunkt des Modells. Um Gesichter im Gruppenbild zu erkennen muss der confidence Wert so weit reduziert werden, dass es zu so vielen false-positives kommt das die Ergebnisse unbrauchbar werden. Verwendet man Bilder mit höherer Auflösung so werden gleiche Ergebnisse in den Stichproben Tests erzielt. Lediglich beim Gruppenbild kommt es zu mehr False-Positives. Einen großen Unterschied macht dies allerdings nicht, da das Caffe Modell schon zuvor Schwierigkeiten mit diesem Bild

hatte. Verwendet man Bilder mit geringer Auflösung so schafft es Caffe weiter alle Gesichter zu erkennen ohne False-Positives, wenn auch im 3er Portrait ein Nebengesicht nur halb getroffen wurde. Überraschender weiße wurden im Gruppenbild mit geringer Auflösung tatsächlich Bereiche maskiert. Auch wenn dabei einige Gesichter korrekt markiert wurden, waren die markierten Bereiche viel zu groß, als das sie sinnvoll wären. Überraschenderweise als der Stichproben-Test mit Bildern mittlerer Auflösung (600 x 1130 Pixel) wiederholt wurde wurde im 3er Portrait das Nebengesicht nicht erkannt, obwohl es bei größeren und kleineren Bild Größen zuvor erkannt wurde. Diese inkonsistent wird es schwerer machen eine geeignete Bildgröße für die Tests zu finden, oder es könnte ein Grund werden warum Caffe am Ende doch verworfen wird.

## 2.3 MediaPipe

**MediaPipe** ist ein open-source Framework von Google, das zur Erstellung von on-device machine learning pipelines entwickelt wurde (Google AI Edge, 2024, 2025b). MediaPipe ist in der Lage mehr als nur Gesichtserkennung zu machen, in dieser Arbeit wird sich der Einfachheit nur auf MediaPipe Face Detection fokussiert, welches ein Teil von MediaPipe Solutions ist. Als Ergebnis liefert MediaPipe sowohl die Position des Gesichts als auch die von Gesichtsmerkmalen wie Augen, Nasenspitze und Mund (Google AI Edge, 2025a). MediaPipe Face Detection nutzt **BlazeFace** als Modell in seiner Pipeline, welches auf 128 x 128 Pixel trainiert wurde. (Google AI Edge, 2025a). BlazeFace verwendet ein SSD Framework das auf schnellere Performanz modifiziert wurde, so dass eine Geschwindigkeit von ca. 200-1000 FPS erreicht werden kann. Es wurde explizit entwickelt um mit Mobile-GPUs zu Arbeiten und effizienter mit GPU Ressourcen umzugehen (Bazarevsky et al., 2019).

MediaPipe schafft es beim 1er Portrait das Gesicht sicher zu erkennen ohne false-positives. Dies ist aber auch der einzige Stichproben Test den MediaPipe besteht. Bei allen anderen Tests wird von MediaPipe nichts markiert, weder Gesichter noch false-positives. Das lässt vermuten, dass MediaPipe nur in der Lage ist große Gesichter zu erkennen. Da dieses Modell sehr ineffektiv für den gewünschten Kontext scheint wird es hier aussortiert und wird nicht weiter getestet. In Stichproben Tests mit höher auflösenden Bildern erzielt MediaPipe Identische Ergebnisse.

## 2.4 Dlib HOG

**Dlib** ist ein open source C++-Toolkit, welches unter anderem Gesichtserkennung mit vorgefertigten Methoden ermöglicht (dlib, 2022). Die hier verwendete `get_frontal_face_detector` nutzt den **Histogram of Oriented Gradients** (HOG). HOG ist ein Feature-Deskriptor, welcher Bilder zu Feature-Vektoren verarbeitet indem er lokale Gradienten berechnet und aus diesen Histogramme erstellt und diese blockweise normalisiert (Dalal & Triggs, 2005). Dlib kombiniert HOG mit einem Sliding-Window-Verfahren über eine Bildpyramide und einer **Support Vector Machine** (SVM) (dlib, o.D. d, o.D. e). Beim Sliding-Window-Verfahren fährt man mit einer festen Fenstergröße über das Bild und analysiert jeden so erzeugten Bildausschnitt (Esri Developer, o.D.). Die SVM dient als Klassifikator um die von HOG erzeugten Feature-Vektoren aus den verschiedenen Bildausschnitten zu bewerten um festzustellen ob diese Gesichter enthalten (Dalal & Triggs, 2005).

Das HOG basierte Gesichtsdetektionsmodell von dlib zeigt in den Stichproben Tests, dass es gut darin ist kleine Gesichter zu erkennen aber Probleme mit größeren hat. So hat es sehr große Probleme das Gesicht beim 1er Portrait zu erkennen, ohne dass man den confidence Wert ins unbrauchbare reduziert. Je nach confidence Wert kommt es auch noch zu kleinen Mengen (ca. 0 bis 3) von false-positives. Doch wurden die Gesichter im 2er, 3er und auch größten Teils im Gruppenbild gut erkannt. Das Modell weist zwar offensichtliche Schwäche auf, allerdings sind die Ergebnisse gut genug das es weiter getestet wird. Es könnte sich als nützliche alternative für Bilder herausstellen bei denen andere Modelle Probleme haben die Gesichter zu erkennen. Verwendet man hochauflösende Bilder in den Stichproben Tests so schafft es HOG mit gleicher confidence das Gesicht im 1er Portrait zu erkennen, allerdings erhöht sich die Anzahl an false-positives. Die Ergebnisse erinnern somit an jene mit geringerem confidence wert bei Bildern mit geringere Auflösung. Es scheint so als müsste der confidence Wert von HOG in abhängigkeit der Bildauflösung eingestellt werden. Sollte dies der Fall sein könnte es das Modell deutlich unattraktiver durch den erhöhten Aufwand in der Verwendung machen. Bei einem weiteren Test mit Bildern geringer Auflösung ergibt sich beim 1er Portrait weiter nur ein false-positiv, dafür keine False-Positives bei den 2er und 3er Portraits. Jedoch gibt es beim Gruppen Bild eine deutliche Verschlechterung aus 6 erkannten Gesichtern und 3 False-Positives wurden 1 erkanntes Gesicht und 2 False-Positives. Zugegeben durch die geringe Auflösung sind die Gesichter des Gruppen Bilds relativ schwer erkennbar. Daran wird gezeigt das Portraits und Gruppenbilder unterschiedlich behandelt werden sollten und sich fürs erste weiter auf Portraits fokussiert werden sollte. Testet man Dlib HOG mit Bildern mittlerer Auflösung, so werden in allen Bildern, bis auf dem Gruppenbild, alle Gesichter erkannt. Dabei kam es in jedem Bild auch zu 1-2 False-Positives.

## 2.5 Dlib HOG Landmark

Verwendet wird in dieser Arbeit das `shape_predictor_68_face_landmarks.dat` Modell von Davis King (King, 2024). Das Modell basiert auf der Methode aus dem Paper "One Millisecond Face Alignment with an Ensemble of Regression Trees" von Kazemi und Sullivan, 2014 (dlib, o.D. c). Darin wird gezeigt wie Gesicht-Landmarks innerhalb von Millisekunden mittels Regressionsbäumen und Boosting geschätzt werden können. Wie im Namen abzulesen ist markiert das Modell 68 Landmarks in einer vordefinierten Bounding-Box (dlib, o.D. b). Landmarks sind dabei unter anderem Punkte im Gesicht wie Augen, Nase und Lippen (dlib, o.D. c). Das Modell nimmt Bounding-Boxes und markiert in diesen die Landmarks. Es braucht zusätzlich die Hilfe eines Detektors der die benötigten Bounding-Boxes zur Verfügung stellen kann. Im Fall von `shape_predictor_68_face_landmarks.dat` wurde dieses darauf ausgelegt zusammen mit Dlib HOG verwendet zu werden, weswegen in dieser Arbeit Dlib HOG für die Erstellung der Bounding-Boxes verwendet wird (King, 2024). Das Modell wurde mithilfe des iBUG 300-W Datensatzes trainiert (dlib, o.D. c; King, 2024).

Vor der Recherche wurde davon ausgegangen das das Landmark Modell seine Eingaben überprüft, so das genauere Ergebnisse im Vergleich zum einfachen Dlib HOG. Dies scheint nicht der Fall zu sein, allerdings können die markierten Landmarks analysiert werden um so von Dlib HOG markierte false-positives auszusortieren. Mit diesem Ansatz wird Dlib HOG Landmark in den Stichproben-Tests untersucht ob es tatsächlich genauer sein kann. Da in dieser Arbeit Dlib HOG verwendet wurde um die Bounding-Boxes für das Landmark Modell zu bestimmen sind die Ergebnisse beider Modelle identisch. Das Landmark Modell hat dabei keine Probleme die Landmarks der identifizierten Gesichter korrekt zu platzieren. Die markierten Gesichter wurden nun noch einmal überprüft ob false-positives auszusortieren für genauere Ergebnisse im Vergleich zu Dlib HOG ohne Landmark. Allerdings wurde dabei meist die korrekten Gesichter aussortiert statt false-positives. Es wurde auch false-positives aussortiert, was jedoch irrelevant ist wenn das Modell Gesichter nur schlecht erkennt. Dlib HOG Landmark wird daher nicht weiter getestet, da es im Vergleich zu Dlib HOG keinen Mehrwert liefert. Hochauflösende Bilder liefern dabei auch keine besseren Ergebnisse, sondern in den meisten Fällen eher mehr false-positives. Im 3er Portrait wurde zwar 1 Gesicht mehr erkannt, was jedoch immer noch deutlich Schlechter als einfaches Dlib HOG ist, welches alle Gesichter im 3er Portrait erkannt hat.



## 2.6 Dlib CNN

`mmod_human_face_detector.dat` ist ein Modell von Dlib, welches **Convolutional Neural Network** (CNN) verwendet. Dlib selbst beschreibt das Modell als genauer im Vergleich zu HOG, informiert aber auch das die höhere Genauigkeit auf Kosten höherer benötigter Rechenleistung und damit verbundener Latenz kommt (dlib, o.D. a). Wie aus dem Namen des Modells zu entnehmen wurde es mithilfe von **Max-Margin Object Detection** (MMOD) trainiert. MMOD ist ein Trainingsverfahren bei dem alle möglichen Sub-Fenster in einem Bild berücksichtigt werden, anders als beim Sliding-Window-Verfahren wo nur über Stichproben von möglichen Fensteroptionen iteriert wird (King, 2015). Ein CNN erkennt Gesichter, indem es das Eingabebild schrittweise mit kleinen Matrizen, genannt Filtern oder Kernels, faltet (GeeksforGeeks, 2025). Dabei gleiten die Filter über das Bild und erzeugen durch die Faltung für jeden Filter eine eigene Feature-Map. Anschließend wird Pooling angewendet, das jeweils benachbarte Bereiche zusammenfasst und so die Auflösung sowie den Rechenaufwand reduziert (GeeksforGeeks, 2025). Durch die Beibehaltung der zweidimensionalen Struktur kann ein CNN räumliche Zusammenhänge im Bild ausnutzen und so Merkmale wie Kanten zuverlässig erkennen.

Dlib CNN, genauer `mmod_human_face_detector.dat`, ist das spürbar langsamste Modell von allen bisher getesteten. In der Dokumentation wurde bereits über eine hohe Latenz berichtet, allerdings ist diese deutlich spürbarer als erwartet (dlib, o.D. a). Bereits bei Bildern der Größe 998 x 1314 Pixel muss für ca. 20 Sekunden auf ein Ergebnis gewartet werden, während die anderen Modelle unter einer Sekunde bleiben. Das Ergebnis der Stichproben Tests ist dafür Fehlerlos. Bei 1er, 2er und 3er Portraits wurden alle Gesichter erkannt ohne false-positives. Nur beim Gruppenbild wurde kein Gesicht oder false-positiv erkannt. Mit einem so zuverlässigen Ergebnis wird Dlib CNN weiter getestet und verglichen. Am Ende muss sich zeigen, dass die deutlich längere Bearbeitungszeit sich auch in deutlich besserer Zuverlässigkeit widerspiegelt um sie zu rechtfertigen. Verwendet man hochauflösende Bilder, hier 1200 x 1593 Pixel, so bleiben die Ergebnisse gleich. Lediglich die Wartezeit erhöht sich deutlich. Aus ca. 20 Sekunden Bearbeitungszeit wurden ca. 40. Es handelt sich hierbei zwar nur um Stichproben, allerdings könnte dies der große Nachteil von CNN sein wenn bei bereits 20% größeren sich die Bearbeitungszeit verdoppelt wenn bereits die Ursprüngliche Bearbeitungszeit deutlich über dem Durchschnitt der anderen getesteten Modelle liegt. Verwendet man Bilder mit geringer Auflösung ca. 400 x 531 Pixel so arbeitet das Modell mit vergleichbarer Geschwindigkeit verglichen mit den anderen Modellen. Allerdings war es dann nur noch in der Lage Gesichter im 1er Portrait zu erkennen, auf allen anderen Bildern wurde nichts mehr markiert. Bei Stichprobentests mit Bildern mittlerer Auflösung, werden wieder alle Gesichter im 1er, 2er und 3er Portrait gefunden bis auf das Nebengesicht im 3er Portrait. Anders als bei den meisten

anderen Modellen die getestet wurden, scheint Dlib CNN mit größeren Bildern besser zu arbeiten. Nachteil bleibt das dadurch die Bearbeitungszeit weiter hoch bleibt um von Dlib CNNs Genauigkeit zu nutzen.

## 2.7 MTCNN

**Multi-task Cascaded Convolutional Networks** (MTCNN) kombiniert Gesichtserkennung und Gesichtsausrichtung in einem Modell (Zhang et al., 2016). Dafür werden 3 CNNs kaskadiert um sowohl effizient als auch effektiv zu Arbeiten. Das erste CNN wird **Proposal Network** (P-Net) genannt (Zhang et al., 2016). Das P-Net ist ein schnelles oberflächliches CNN, das mithilfe einer Bildpyramide die ersten möglichen Bounding-Boxes markiert. Anschließend wird **Non-Maximum Suppression** (NMS) verwendet um überlappende Bounding-Boxes zusammen zufügen. Im zweiten Schritt werden weiter falsche Bounding-Boxes vom sogenannten **Refine Network** (R-Net) verworfen die keine Gesichter enthalten, welches ein komplexeres CNN im Vergleich zum P-Net ist (Zhang et al., 2016). Nun da es nur noch eine kleine Anzahl von Bounding-Boxes gibt, wird das stärkste CNN, das **Output Network** (O-Net), eingesetzt welches fünf Landmark Punkte markiert und damit das Endergebnis generiert (Zhang et al., 2016).

MTCNN hat den Stichproben Test für 1er, 2er und 3er Portraits fehlerlos bestanden. Beim Gruppen Bild wurden bloß ein paar Gesichter erkannt, dabei sollten nur Ergebnisse mit  $> 0.5$  confidence markiert werden. Es gab des weiteren nicht einen false-positiv. Somit wird MTCNN sicher weiter getestet. Verwendet man Bilder mit höherer Auflösung so zeigt sich im Stichproben-Test das die confidence erkannter Gesichter im schnitt gleich bleibt, allerdings kommt es bei manchen Bildern zu 1 bis 3 false-positives. Im Gruppen Bild hingegen wurden ohne weitere false-positives bei höherer Auflösung mehr Gesichter erkannt, wenn auch nur 6 von 13 insgesamt. Die false-positives bei höherer Auflösung haben sehr hohe confidence Werte (ca. 0.85) was das bestimmen eines besserer geeigneten confidence Wertes erschwert. In den kommenden Tests muss sich zeigen ob MTCNN weiter verlässlich bleibt, oder ob sich doch false-positives unvermeidlich dazu mischen. Werden Bilder mit geringer Auflösung getestet so erkennt MTCNN weiter alle Gesichter in 1er, 2er und 3er Portraits korrekt markiert noch dazu mit 1.0 confidence. Weiter gibt es keine False-Positives, und im Gruppen Bild werden keine Bereiche markiert. In Stichprobentests mit mittlerer Auflösung bleiben die Ergebnisse größtenteils Identisch, nur wurden im Gruppenbild 4 Gesichter erkannt mit ca. 0.85 confidence.

## 2.8 Yunet

Yunet ist ein leichtgewichtiger Detektor und CNN der speziell für den Einsatz auf ressourcenbeschränkten Geräten wie Smartphones entwickelt wurde (Wu et al., 2023). Anders als Detektoren wie SSD, nutzt Yunet einen ankerfreien Ansatz (Wu et al., 2023). Dadurch dass ohne Anker oder Default-Boxes gearbeitet wird, benötigt Yunet weniger Rechenleistung für seine Arbeit. Allerdings haben ankerfreie Ansätze Probleme mit kleineren Gesichtern oder wenn es in einem Bild unterschiedlich große Gesichter gibt (Wang et al., 2018).

Das Yunet Modell, in diesem fall `face_detection_yunet_2023mar.onnx`, ist ohne weitere Einstellungen sehr streng. Bei den Stichproben Test kam es zu keinen false-positives, allerdings wurde in manchen Bildern nicht alle Gesichter markiert. Nachdem der confidence Wert auf 0.8 eingestellt wurde, wurde Problemlos alle Gesichter des 1er, 2er und 3er Portraits erkannt. Aus dem Gruppen Bild wurden nur wenige Gesichter markiert. Ob 0.8 der optimale Wert ist wird sich zeigen wenn Yunet intensiver getestet wird. Aufgrund seiner Genauigkeit wird Yunet weiter getestet. Nutzt man Bilder mit höherer Auflösung in den Stichproben Tests so verbessert sich der confidence Wert der gefundenen Gesichter. Beim Gruppenbild wurde ein zuvor erkanntes Gesicht nicht erkannt, ein vorher nicht erkanntes Gesicht dann wiederum schon. Weiter gab es bei keinem der Stichproben Tests mit höherer Auflösung ein false-positiv, womit Yunet ein sicherer Kandidat bleibt für die kommenden Tests. Im Stichprobentest mit Bildern geringer Auflösung findet Yunet weiter alle Geischter in allen Bildern bis auf im Gruppenbild wo es keines mehr erkennt. Werden Bildern mit mittlerer Auflösung verwendet so bleiben die Ergebnisse Identisch zu denen aus den Test geringer Auflösung.

## 2.9 RetinaFace (CPU)

RetinaFace ist ein einstufiger, pixelweiser Gesichtsdetektor, welcher je nach Variante ResNet oder MobileNet als Backbone verwendet (Deng et al., 2019). Einstufig bedeutet, dass Klassifikation und Lokalisierung in einem einzigen Durchlauf gemeinsam ausgeführt werden, ähnlich wie beim SSD. Pixelweise bedeutet hier, dass für jede Gitterzelle der Feature-Map eine Vorhersage getroffen wird. Pro Zelle gibt RetinaFace einen Face-Score, eine Bounding-Box, fünf Landmark-Koordinaten sowie zusätzliche Informationen für ein 3D-Mesh des Gesichts aus (Deng et al., 2019). Um Gesichter verschiedener Größen zuverlässig zu erkennen, nutzt RetinaFace Feature-Pyramiden, bei denen Feature-Maps unterschiedlicher Auflösungen kombiniert werden. Trainiert wurde das Modell auf dem **WIDER FACE** Datensatz (Deng et al., 2019). Zusätzlich

zu klassischen Trainingsansätzen wurden auf allen erkennbaren Gesichtern des Datensatzes fünf Landmark-Punkte händisch annotiert, um die Lokalisierungsgenauigkeit zu erhöhen. Ein zusätzlicher Zweig des Modells erzeugt außerdem ein 3D-Gesichts-Mesh, welches für selbst überwachtetes Lernen verwendet wird (Deng et al., 2019).

Verwendet wird in dieser Arbeit RetinaFace mit ResNet-50 Backbone und CPU Einsatz. Grund dafür ist, dass RetinaFace mit GPU Einsatz lediglich schneller und nicht genauer sein soll. So bleiben die Werte von RetinaFace später vergleichbarer. Die Option das mit einer geeigneten GPU schnellere Ergebnisse erzielt werden können, kann später bei der Auswertung noch berücksichtigt werden. Was die Ergebnisse des Stichproben Tests angeht hat RetinaFace das beste Ergebnisse aller Modelle bisher. Alle Gesichter ohne false-positives im 1er, 2er und 3er Portrait erkannt. Noch dazu wurden 10 der 13 Gesichter im Gruppenbild erkannt, wo die besten Ergebnisse bisher ca. 6 Gesichter waren. Jedoch sei anzumerken das es 2 false-positives gab bei denen die Gesichter von Pferden markiert wurden. Ob RetinaFace diese Quote beibehalten kann wird sich in den folgenden Tests ergeben. RetinaFace bleibt bei den Stichproben Tests mit höherer auflösenden Bildern konstant. Confidence Werte verbessern sich leicht oder verschlechtern sich um wenige 0.05 Punkte. Weiter gibt es auch keine neuen false-positives. Lediglich beim Gruppenbild wurde ein Gesicht weniger erkannt, dieses hatte aber auch schon im Test zuvor den geringsten confidence Wert von allen erkannten Gesichtern. Im Test mit Bildern mit geringer Auflösung schneidet RetinaFace weiter am besten ab. Es gab keine False-Positives und es wurden alle Gesichter im 1er, 2er und 3er Portrait erkannt. Die Anzahl erkannter Gesichter sankt zwar auf 8 von 13 im Gruppenbild, ist aber bei weitem das beste Ergebnis mit dieser Auflösung wo andere Modelle meist nichts mehr erkannt haben. Im Stichprobentest mit Bildern mittlerer Auflösung werden weiter alle Gesichter im 1er, 2er und 3er Portrait fehlerfrei erkannt. Im Gruppen Bild werden 9 von 13 Gesichtern erkannt, sowie die zwei false-positivs der Pferdegesichter.

## 3 Tests

Alle Größen wurden nochmal mit allen relevanten Kandidaten getestet. Man hat sich für Größe L entschieden, da trotz False-Positives alle Gesichter hier erkannt werden, was relevanter ist. HOG ist ausgeschieden da es im Vergleich deutlich mehr False-Positives hat und es bereits genug Modelle mit akzeptablen Werten gibt, so dass das Modell als Kandidat nicht mehr relevant ist.

Bevor alle verbleibenden Modelle intensiv getestet werden können müssen alle confidence Werte bzw score thresholds angepasst werden, das jedes Modell bestmöglich repräsentiert werden kann. Dafür wird der threshold auf minimalen Wert gesetzt und die Ausgaben beobachtet und notiert. Am interessantesten sind Werte wie kleinster confidence Wert für ein erkanntes Gesicht und höchster confidence Wert für ein False-Positiv. Sollten sich diese überschneiden, so wird mithilfe des Kontext der anderen Ergebnisse bestimmt welcher confidence Wert sich für das jeweilige Modell am besten eignet zur Gesichtserkennung für historische Gemälde.

Erklären das Confidence Wert unterschiedlich ist pro Modell und nicht der höchste Wert automatisch der Beste ist

Geschwindigkeit wird nicht getestet weil nicht relevant für Kontext

### 3.1 Caffe

Die Ergebnisse von Caffe waren bei -1 confidence Grenzwert viel zu unkenntlich, also wurde der Test wiederholt. Im zweiten Anlauf wurden alle markieren mit einem confidence Wert  $> 0.1$  akzeptiert, da alle Ergebnisse unter diesem Wert aufgrund der riesigen mengen an False-Positiv unbrauchbar wären. Minimaler Confidence Wert für erkannte Gesichter bei 1er Portraits war 0.11 während der maximale False-Positiv Wert 0.21 ist. Der Wert wurde bei einem Bild gemessen alle anderen erreichten einen Wert von 1.00 oder ca. 0.80. Da nur ein Bruchteil des Cranach Archivs aufgrund der limitierten Zeit getestet werden kann wird der Wert nicht als Ausreißer sondern als gleichwertig betrachtet, da es im restlichen Archiv bestimmt ähnliche Bilder zum 0.11 Kandidaten gibt. Ergebnisse des Tests mit 2er Portraits waren deutlich weniger konstant als im letzten Test. Geringster Wert für ein erkanntes Gesicht war 0.14, 0.13 wenn Nebengesichter mitgezählt werden. Allerdings gab es ein großes False-Positiv

sowohl vom Wert als auch von der Fläche her mit 0.4. Der Test mit 3er Portraits verlief sehr schlecht für Caffe. Der geringste erkannte Wert ist 0.1, es ist aber auch sehr wahrscheinlich das der eigentliche Wert noch geringer ist, da in manchen Bilder nicht alle Gesichter erkannt wurden. Auch gab es eine sehr große Menge an False-Positives, höchster Wert in diesem Test 0.28. Mit minimaler Gesichtsklitterung Confidence von 0.1 und maximalem False-Positiv Confidence von 0.4 ist das bestimmen eines optimalen Grenzwertes sehr schwierig. Nach mehreren weiteren Test wurde sich entschieden den Grenzwert auf 0.14 zu setzen. Leider werden damit einige Ergebnisse von erkannten Gesichtern verworfen, allerdings gibt es bis 0.13 noch große mengen an false-positivs, was die gesamt Ergebnisse brauchbarer macht.

## 3.2 Dlib CNN

Dlib CNNs Tests mit 1er Portraits haben selbst bei einem Grenzwert von -1 keine False-Positives gehabt. Der Minimale Wert für erkannte Gesichter lag bei 1.03. Im Test mit 2er Portraits gab es weiter keine Flase-Positives, allerdings wurde auch einige Neben- so wie Hauptgesichter nicht erkannt. 0.70 war der kleinste gemessene Wert für ein Gesicht. Weiter werden im Test mit 3er Portraits keine False-Positiv, aber auch Teils manche Gesichter nicht erkannt. Der geringst gemessene Wert für ein erkanntes Gesicht war in diesem Test 0.97 womit 0.7 der gesamt niedrigste Wert für Dlib CNN ist. Da bisher keine False-Positives erkannt und Dlib CNN sehr streng scheint, wird der verwendete Grenzwert für Dlib CNN 0.6 sein.

## 3.3 MTCNN

MTCNN ist sehr sicher durch den Test mit 1er Portraits gekommen. Keine False-Positives und und ein minimaler confidence Wert bei der Gesichtserkennung von 0.99. Generell sind alle Werte konstant zwischen 0.99 und 1.00. Im Test mit 2er Portraits wurden weiter keine Flase-Positives erkannt, allerdings ebenso einige Gesichter nicht. Der geringst gemessene Wert bei der Gesichtserkennung beträgt 0.82. Die Ergebnisse des Tests mit 3er Portraits erschweren das bestimmen eines endgültigen Grenzwertes für MTCNN. Der höchst gemessene Wert für Flase-Positives beträgt 0.95, generell haben wie zuvor erwähnt alle False-Positives von MTCNN sehr hohe confidence Werte. Die Entscheidung wo der Grenzwert gesetzt werden soll fällt sehr schwer. Die beiden Möglichkeiten sind 0.82 für den minimalen Wert eines erkannten Gesichts, oder 0.91 um den größten Teil der False-Positives auszusortieren. Der Grund warum 0.82 überhaupt in Erwägung gezogen kann liegt daran das die makierten False-Positives bisher immer kleine Bereiche gewesen sind, die ignoriert werden könnten. Der Grenzwert wird

vorläufig auf 0.82 gesetzt, je nach Ergebnissen wird das Modell mit 0.91 Grenzwert getestet und entschieden welcher Wert verwendet werden soll.

### 3.4 Yunet

Yunet hat bei seinem 1er Portrait Test keine False-Positives gehabt und ein Gesicht übersehen, ansonsten betrug der minimale Wert für ein erkanntes Gesicht 389.12. Der Test mit 2er Portraits verlief mit einem ähnlichen Ergebnis und einem minimalen Wert von 233.79. Der letzte Confidence Test von Yunet mit 3er Portraits ergab weiter keine False-Positives und auch den gesamt minimal Wert für erkannte Gesichter von 154.46. Da Yunet wie Dlib CNN eher streng zu sein scheint, wird der Grenzwert für Yunet auf 125 gesetzt um weitere mögliche erkannte Gesichter zuzulassen.

### 3.5 RetinaFace (CPU)

RetinaFace hat als minimalen confidence Wert 0.74 für Gesichter im Test mit 1er Portraits, ohne Fehler. Die Ergebnisse für 2er Portraits sind ein minimal Wert von 0.71 und keine False-Positives, es wurden aber auch manche Gesichter nicht erkannt. Ergebnis des Test mit 3er Portraits ist auch gleich der gesamt minimal Wert von RetinaFace 0.68 confidence für Gesichter und 0.52 für Nebengesichter. Da in keinem Test False-Positives markiert wurden wird der absolut geringst gemessen Wert 0.52 als Grenzwert verwendet.

Keines der Modelle hat es geschafft jedes Gesicht aus den Tests Fehlerfrei zu erkennen. Somit werden im letzten Test alle fünf verbliebenen Modelle gleichzeitig getestet. Dabei werden die Anzahl von False-Negatives, False-Positives und Anzahl von Gesichtern die nur von diesem Modell erkannt wurden gezählt. Ziel ist es eine kleine Auswahl von Modellen herauszuarbeiten die genutzt werden können um möglichst viele Gesichter auf historischen Gemälden zu erkennen.

Die Entscheidung Welches oder Welche Modelle sich zur weiter Entwicklung eignen ist keine leichte, da es kein fehlerfreies Modell gibt das einfach ausgewählt werden könnte. Auch ist es so das in manchen Bildern es vorkommen kann das mehrere Gesichter von unterschiedlichen einzelnen Modellen erkannt werden. Es gab bereits die Idee je nach Bild confidence und oder Modell zu wählen, jedoch bleibt dann weiter das Problem das kein einzelnes Modell alle Gesichter eines Bildes erkennt. Die Idee ist es nun je nach Bild Modelle dazu und abzuschalten um so mehr Gesichter zu erkennen, oder False-Positives auszublenden wenn ein redundantes Modell gerade nicht benötigt wird. Mit diesem Anwendungszweck im Sinn fällt die finale Wahl

auf: RetinaFace (CPU), MTCNN und CNN. RetinaFace ist das Modell welches die wenigstens False-Negatives von allen Modellen hatte, zusätzlich hat es auch keine False-Positives gehabt womit es als eine gute Basis dient. MTCNN wurde gewählt das es zusammen mit CNN eines der Modelle ist welches Gesichter markiert hat, die von keinem anderen Modell markiert wurden. Dabei wird der Confidence-Grenzwert von 0.82 beibehalten da viele der besonders kleinen Gesichter nur von MTCNN erkannt werden und unter den alternativen Wert von 0.91 fallen. Da MTCNN abgeschaltet werden kann falls die False-Positives stören ist es dennoch eine gute Ergänzung zur Modell Auswahl. CNN wird aus dem gleichen Grund wie MTCNN dazu geholt. Es hat zwar keine False-Positives die zu Probleme werden könnten, sollte aber nur auf bedarf dazugeschaltet werden da es, wie in vorherigen Tests bereits angemerkt, die Bearbeitungsdauer deutlich erhöht. Die Ergebnisse des Modells enthalten bis auf das gelegentliche Gesicht was nur davon erkannt wird oft viele False-Negatives, weswegen es nur situational benutzt werden sollte.



## 4 Implementierung

Ursprünglich war die Idee ein Eigenständiges Programm zu entwickeln, welches Selbstständig Wasserzeichen auf Bildern platziert. Dabei sollten die Gesichter freigelassen werden um die Ästhetik des Bildes nicht zu stören. Aufgrund der limitierten Zeit wird dieser Ansatz reduziert, was ebenfalls ermöglicht ihn flexibler zu gestalten. Indem statt eines eigenständigen Programms ein Modul erstellt wird, ermöglicht man mehr Anwendungszwecke über das einfache Platzieren von Wasserzeichen hinaus.

Bisher wurden für das Laden der Bilder in dieser Arbeit CV2 genutzt, genauer die Funktion `cv2.imread()`, welche Bilder im BGR-Format lädt. Bei der Implementierung ist allerdings aufgefallen das diese Funktion Probleme mit Umlauten hat, was Problematisch ist wenn Dateipfade geladen werden die diese Enthalten. Also wurde auf die Funktion `Image.open().convert("RGB")` von PIL gewechselt. Interessanter Weise waren die Ergebnisse von RetinaFace mit RGB-Input, obwohl angegeben wurde das BGR als Input verwendet werden sollte (lounging-lizard & nttstar, 2025). Die confidence Werte waren minimal anders ca. 0.02 und es kam zu weniger weniger False-Negatives, weshalb RGB nun als Input für RetinaFace verwendet wird.

Hauptfunktion des Moduls ist `cranach_detector()`, welche gleichzeitig mit RetinaFace, MTCNN und Dlib CNN Bereiche mit Gesichtern auf Bildern markiert und diese in einer Liste zurück gibt. Das Modul liefert Funktionen mit, die einen überprüfen lassen ob sich eine Position oder Fläche auf einem Gesicht befindet, sollten diese allerdings nicht ausreichen so kann man die Ausgegebene Liste für eigene Funktionen verwenden.

Um möglichst viele Verschiedene Anwendungszwecke abzudecken akzeptiert die Funktion `cranach_detector()` verschiedene Arten von eingaben:




- None: Lässt Nutzer\*innen einen Ordner in einem GUI Manuell wählen durch den Iteriert werden soll.
- File: Es kann ein Pfad zu einer einzelnen Bild Datei hinterlegt werden welche bearbeitet werden soll.
- Path: Wird stattdessen der Pfad eines Ordners übergeben, so werden alle Bilder dieses Ordners zu einer Liste verarbeitet durch die dann iteriert wird.

- List: Wenn die interne Funktion unzureichend ist für den Anwendungskontext oder man einfach bereits eine List mit Bilderpfaden hat so kann man diese an das Paket übergeben um diese direkt zu verwenden.

Weiter können auch alle Modelle einzeln dazu oder abgeschaltet werden. Sowohl beim Funktionsaufruf, als auch während der Laufzeit. RetinaFace ist dabei per Default aktiviert und alle anderen Modelle deaktiviert. Da RetinaFace bereits gute Ergebnisse liefert reicht es meist alleine aus, so wird Rechenzeit gespart und False-Positives minimiert.

Auch wenn die confidence Grenzwerte optimiert wurden, kann es dennoch sein dass bestimmte Bilder andere Werte benötigen. Aus diesem Grund wurde die Möglichkeit hinzugefügt die confidence Werte während der Laufzeit zu konfigurieren um für jedes Bild und jedes Modell einzeln Grenzwerte zu konfigurieren falls nötig.

Die von den Modellen erkannten Bereiche werden als Dict in einer Liste gespeichert mit den Attributen x, y, w (width), h (height), model, confidence und image\_name. Diese wird von der Funktion `cranach_detector()` nach durchlaufen aller übergebenen Bilder zurückgegeben, dabei wird die Liste auch weiter intern in einer Variable gespeichert um sie mit anderen internen Funktionen wie `position_isIntersecting()` und `area_isIntersecting()` direkt zu nutzen. Sollte dies nicht ausreichen, kann auf die ausgegebene Liste zurückgegriffen werden um diese entweder in eigenen Python-Funktionen zu verwenden, zum Beispiel als JSON zu formatieren und so zu exportieren. Das Format in dem RetinaFace seine Ergebnisse ausgibt unterscheidet sich leicht von MTCNN und Dlib CNN. RetinaFace gibt als Ergebnis die Eckpunkte eines Rechtecks, während die beiden anderen Modelle einen Startpunkt inklusive Breite und Höhe geben. Um die Arbeit mit den gemeinsamen Ergebnissen der Modellen zu erleichtern werden die Ergebnisse von RetinaFace in das Format von MTCNN und Dlib CNN gebracht, auch wenn das Format von RetinaFace leichter zu verarbeiten ist. Der Grund warum Funktionen in diesem Modul dennoch für Attribute eingaben im Format x, y, breite und höhe erwarten kommt daher, das es im Frontend eher mit Werten im Format x, y, breite und höhe gearbeitet wird. Zudem sind nicht alle Overlays Rechteckig, so dass Eckpunkte nur schwer bestimmt werden könnten.

Die Farben der markierten Gesichter wurden für das Package angepasst so dass sie von Farbenblinden noch gut unterschieden werden können. Als Basis für die Auswahl der Farben diente das Paper von Petroff, 2021. Gewählt wurden (87, 144, 252)  Blau für RetinaFace, (248, 156, 32)  Orange für MTCNN und (228, 37, 54)  Rot für Dlib CNN. Blau ist die Farbe die sich am besten eignet um von anderen unterschieden werden zu können. Sie wurde bewusst für RetinaFace gewählt, da das Modell als Basis dient und so am häufigsten mit den anderen Farben in Kontakt kommt. Allerdings können die gewählten Farben auf manchen Bildern etwas

schlechter erkannt werden, weshalb die Option hinzugefügt wurde die Markierungen mit dunkleren Farben angezeigt zu bekommen.

Sollten mehrere Modelle das selbe Gesicht markieren so wird nur der kleinste markierte Bereich für das Gesicht behalten. Dafür wird überprüft ob sich zwei markierte Bereiche überschneiden und Fläche mit der Fläche des kleineren Bereiches verglichen. Wenn der überschneidende Bereich 75% oder mehr der Fläche des kleineren Bereiches ausmacht wird der größere Bereich verworfen. So werden möglichst genaue Ergebnisse behalten, und vermieden das Markierung von Gesichtern die nah beieinander liegen verworfen werden.

Um zu überprüfen ob ein eine Position oder ein Bereich auf einem Gesicht liegt wurden die Funktionen `position_isIntersecting()` und `area_isIntersecting()` implementiert. Als Argumente erwarten beide Funktionen ein Tupel: (x,y) im Fall von `position_isIntersecting()` und (x,y, width, heigth) bei `area_isIntersecting()`, zusammen mit dem Dateinamen eines Bildes, welches zuvor schon einmal von `cranach_detector()` bearbeitet wurde, damit intern die markierten Bereiche zur Verfügung stehen. Die Funktionen geben True zurück, sollte sich die Position oder der Bereich auf einem Gesicht befinden. Optional kann auch eine Margin angegeben werden, falls ein gewisser Abstand zu Gesichter eingehalten werden soll. Die Funktion gibt dann ebenfalls True zurück sollte sich ein Gesicht innerhalb der Margin um den Bereich oder Punkt befinden.

## 5 Fazit

Das Ziel der Arbeit, Gesichtserkennungsmodelle zu finden, welche für historische Gemälde geeignet sind konnte größtenteils erfüllt werden. Das Modell RetinaFace alleine ist in der Lage beinahe alle Gesichter auf getesteten 1er, 2er und 3er Portraits zu finden. Zusammen mit MTCNN und Dlib CNN konnte ein Python-Modul entwickelt werden, welches in der Lage ist alle Gesichtern auf den getesteten Portraits zu finden. Mit der Möglichkeit Modelle je nach Bild an oder abzuschalten, sowie ihre confidence Grenzwerte anzupassen, ist das Modul in der Lage flexible auf verschiedene Werke angewendet zu werden um bestmögliche Ergebnisse zu Erzielen. Des weiteren konnten Funktionen implementiert werden die Nutzer\*innen ermöglichen zu Prüfen ob sich ein Overlay auf einem Gesicht befindet. Mit der Möglichkeit eigene Funktionen zu implementieren in dem die ausgegebene Liste direkt in Python genutzt wird oder sie zum Beispiel zu einer JSON Datei formatiert um sie in anderen Anwendung zu nutzen.

Als mögliche Fortsetzungen zur Arbeit kann das GUI des Moduls überarbeitet um die Nutzerfreundlichkeit zu optimieren. Ein Mögliches Feature welches hinzugefügt werden könnte wäre ein Zähler, welcher Zeigt wie viele Bereiche von je welchem Modell markiert wurden. So kann auf die schneller von Nutzer\*innen erkannt werden ob sich auf einem Bild False-Positives befinden, wenn die Zahl der Bereiche die der tatsächlichen Gesichter auf dem Bild übersteigt. Die dafür notwendigen Funktionen sind auch bereits implementiert, nur aufgrund von Zeitmangel keine grafische Darstellung für sie implementiert. Auch könnte eine Funktion implementiert werden, welche die Liste der markierten Bereiche direkt zu einer JSON Datei formatiert, um sie auch in anderen Programmen nutzen zu können.

Eine weitere mögliche Fortsetzung der Arbeit wäre ein intensiveres Testen mit Fokus auf Gruppenbildern. Gruppenbilder haben bereits in den Stichproben Tests gezeigt das es deutlich schwieriger ist auf ihnen zuverlässig Gesichter zu finden. Aufgrund des Mangels der Ressource Zeit wurden später Tests an Gruppenbildern verworfen um sich besser auf Portraits fokussieren zu können. Dies geschah unter der Begründung das das Platzieren eines Overlays auf dem Gesicht eines Portraits die Ästhetik stärker negativ beeinflusst als auf einem Gruppenbild. Würde man die Forschung mit Gruppenbildern Fortsetzen wollen, so müssten vermutlich neue Modelle die nicht in dieser Versuchsreihe getestet wurden herangezogen werden, oder in Erwägung gezogen werden ein eigenes Modell für diesen Anwendungszweck zu trainieren.

## 6 Formaler Aufbau

In diesem Kapitel finden Sie grundlegende Hinweise zum formalen Aufbau Ihrer Arbeit.

### 6.1 Reihenfolge

Eine wissenschaftliche Arbeit besteht in der Regel aus den folgenden Teilen:

1. Deckblatt
2. Kurzfassung/Abstract (optional)
3. Inhaltsverzeichnis
4. Abbildungs- und Tabellenverzeichnis (auch am Ende üblich)
5. Abkürzungsverzeichnis (auch am Ende üblich)
6. Einleitung
7. Hauptteil
8. Zusammenfassung/Fazit
9. Literaturverzeichnis
10. Anhänge (optional)
11. Erklärung

### 6.2 Deckblatt

Das Deckblatt beinhaltet: Titel der Arbeit, Art der Arbeit, Verfasser\*in, Matrikelnummer, Abgabetermin, Betreuer\*in sowie Zweitgutachter\*in. Das Deckblatt wird bei Arbeiten, die länger sind als 15 Seiten, bei der Seitenanzahl zwar mitgezählt, jedoch nicht nummeriert.

## 6.3 Inhaltsverzeichnis

Wir empfehlen eine Dezimalgliederung wie in diesem Dokument angelegt. Werden innerhalb eines Kapitels Unterüberschriften verwendet, müssen mindestens zwei vorhanden sein: wo ein 2.1 ist, muss es ein 2.2 geben.

Das Inhaltsverzeichnis enthält immer die Seitenangaben zu den aufgelisteten Gliederungspunkten; es wird dabei aber selbst nicht im Inhaltsverzeichnis aufgelistet. Die Seiten, die das Inhaltsverzeichnis selbst einnimmt, können römisch gezählt werden.

Für eine Abschlussarbeit ist eine Gliederungstiefe von wenigstens drei Ebenen üblich. In der Regel werden nur bis zu vier Ebenen vorne im Inhaltsverzeichnis abgebildet. Hier sollten Sie aber unbedingt die Gepflogenheiten in Ihrem Fach berücksichtigen und ggf. in Erfahrung bringen.

## 6.4 Abbildungsverzeichnis und Tabellenverzeichnis

Abbildungen und Tabellen werden in entsprechenden Verzeichnissen gelistet. In dieser Vorlage erscheinen sie direkt nach dem Inhaltsverzeichnis. Dann können die entsprechenden Seiten römisch gezählt werden. Die Verzeichnisse können jedoch auch am Ende der Arbeit vor oder hinter dem Literaturverzeichnis stehen. Dann werden sie regulär mit Seitenzahlen versehen.

## 6.5 Abkürzungsverzeichnis

Die Zahl der Abkürzungen sollte übersichtlich bleiben. Das Abkürzungsverzeichnis enthält lediglich wichtige fachspezifischen Abkürzungen in alphabetischer Reihenfolge, insbesondere Abkürzungen von Organisationen, Verbänden oder Gesetzen. Gängige Abkürzungen wie „u. a.“, „z. B.“, „etc.“ werden nicht aufgenommen.

Zur technischen Umsetzung mit L<sup>A</sup>T<sub>E</sub>X vergleiche auch Abschnitt 7.18.

## 6.6 Literaturverzeichnis

Das Literaturverzeichnis wird alphabetisch nach Autorennamen geordnet. Es enthält alle im Text zitierten Quellen – und nur diese. Mehrere Schriften einer Person werden nach Erscheinungsjahr geordnet. Schriften derselben Person aus einem Erscheinungsjahr müssen Sie selbst unterscheidbar machen. In den Ingenieurwissenschaften wird

zusätzlich häufig ein Nummern- oder Autorenkürzel dem Namen in eckigen Klammern voran-gestellt. Mehr hierzu und weitere wichtige Regeln des Zitierens lernen Sie in den E-Learning-Kursen des Schreibzentrums<sup>1</sup> kennen.

Zur Verwaltung der verwendeten Literatur eignen sich entsprechende Softwaretools wie Citavi oder Zotero, die mit verschiedenen Textverarbeitungsprogrammen kompatibel sind.

## 6.7 Rechtschreibung, Grammatik

Achten Sie bei der Abgabe Ihrer Arbeit auf ein einwandfreies Deutsch bzw. Englisch. Wenn Fehler die Lesbarkeit beeinträchtigen, kann sich dies durchaus negativ auf die Note auswirken. Nutzen Sie daher unbedingt die Rechtschreibprüfung Ihres Textverarbeitungsprogramms, auch wenn diese nicht alle Fehler erkennt.

Für alle, die sich bei diesem Thema unsicher fühlen, empfehlen wir die E-Learning-Kurse des Schreibzentrums<sup>2</sup>. Wenden Sie sich ggf. auch an die Beauftragte für Studierende mit Beeinträchtigung<sup>3</sup>.

## 6.8 Umfang der Arbeit

Alle Fächer nennen verbindliche Angaben zu Unter- und Obergrenzen, die in der Regel eingehalten werden müssen. Verzeichnisse und Anhänge werden dabei in aller Regel nicht mitgezählt. In Einzelfällen – insbesondere bei empirischen Arbeiten – können abweichende Vereinbarungen mit der Betreuungsperson getroffen werden.

---

<sup>1</sup>[https://ilu.th-koeln.de/goto.php?target=cat\\_52109&client\\_id=thkilu](https://ilu.th-koeln.de/goto.php?target=cat_52109&client_id=thkilu)

<sup>2</sup>[https://ilu.th-koeln.de/goto.php?target=cat\\_52109&client\\_id=thkilu](https://ilu.th-koeln.de/goto.php?target=cat_52109&client_id=thkilu)

<sup>3</sup>[https://www.th-koeln.de/studium/studieren-mit-beeintraechtigung\\_169.php](https://www.th-koeln.de/studium/studieren-mit-beeintraechtigung_169.php)

## 7 Gestaltung: Textsatz mit $\text{\LaTeX}$

Mit  $\text{\LaTeX}$  ist es verhältnismäßig einfach, Dokumente zu erstellen, die professionellen Ansprüchen genügen. Ein entscheidender Vorteil ist, dass der Nutzer fast nur den Inhalt beisteuert, während die korrekte äußere Form dann automatisch erzeugt wird.  $\text{\LaTeX}$  basiert auf  $\text{\TeX}$ , das von Donald Knuth entwickelt wurde **knuth:tex**. Einige weitere Vorteile gegenüber gängiger Textverarbeitung:

**Frei/Plattformunabhängig:** Bei  $\text{\LaTeX}$  handelt es sich um freie Software. Es wird kein proprietärer Editor benötigt, um  $\text{\LaTeX}$ -Dokumente zu schreiben. Tatsächlich können die Dokumente auf *jedem* Rechner mit *jedem* beliebigen Editor bearbeitet werden.

**Reines Textformat:** Der Quelltext – die **tex**-Datei – ist ein reines Textformat. Dadurch eignen sich  $\text{\LaTeX}$ -Dokumente auch hervorragend zur Versionskontrolle mit beispielsweise git. Dies wiederum ermöglicht eine effiziente Zusammenarbeit mehrerer Autor\*innen.

**Aufteilen großer Dokumente:** Der Quelltext großer Dokumente, wie beispielsweise von Projektarbeiten, kann auf mehrere Dateien aufgeteilt werden. So können beispielsweise mehrere Personen an jeweils einem eigenen Kapitel arbeiten. Aufgrund der beiden oberen Punkte wird es auch nicht zu Kompatibilitätsproblemen kommen.

**Trennen von Layout/Inhalt:** Mit  $\text{\LaTeX}$  kann man explizit das Layout für das gesamte Dokument festlegen – oder die verwendete Dokumentklasse kümmert sich implizit darum. Zeitgemäße Textverarbeitung bietet mit Formatvorlagen zwar entsprechende Funktionalitäten; aber durch den programmatischen Ansatz mit  $\text{\LaTeX}$  kann noch genauer Einfluss auf das Layout genommen werden. Anschließend kann die ganze Konzentration auf das Schreiben gelegt werden.

**Professionelles Ergebnis:** Ein mit  $\text{\LaTeX}$  erzeugtes Dokument schaut professioneller aus, als ein entsprechendes, mit Textverarbeitung erzeugtes Dokument. Das gilt vor allem für mathematiklastige Dokumente. Aber auch andere Dokumente können von einem einheitlichen Layout, gleichmäßigem Grauwert des Fließtexts, stimmigeren Seitenumbrüchen und hochwertigen Vektorgraphiken profitieren – um nur mal einige Punkte zu nennen.



**Vielseitig einsetzbar:** Mit L<sup>A</sup>T<sub>E</sub>X können nicht nur „einfache“ Dokumente erzeugt werden. Es existieren unzählige Dokumentklassen, die beispielsweise auch das Erstellen von Präsentationen oder Postern ermöglichen.

In den folgenden Abschnitten 7.1 bis 7.18 wird auf diverse Aspekte eingegangen, die Sie beim Erstellen Ihres Dokuments berücksichtigen sollten.

## 7.1 Unter der Haube

Sie definieren in Ihren TEX-Dokumenten, was Ihre Inhalte sind (Text mit Gliederung, Bilder, Tabellen, Literaturverweise, ...) und wie diese jeweils grob aussehen sollen (z. B. Platzierung von Abbildungen mittels *Gleitumgebungen*, vgl. Abschnitt 7.6).

Beim Erstellen des endgültigen Dokuments wendet L<sup>A</sup>T<sub>E</sub>X „unter der Haube“ eine ganze Menge Regeln an, die festlegen, wie das alles bestmöglich umgesetzt werden kann. Diese Regeln betreffen z. B. den Anteil von Text und Bildern pro Seite, Abstände innerhalb von Zeilen, aber auch Sonderfälle wie das Vermeiden einzelner Zeilen eines Abschnitts alleine auf einer Seite (sog. „Schusterjungen“ oder „Hurenkinder“).

Im Ergebnis kann es also passieren, dass z. B. Ihre Abbildungen „springen“, während Sie an Ihrem Text arbeiten. Das hat im Zweifel alles seine Richtigkeit und kann im Notfall am Ende noch optimiert werden.

In diesem Zusammenhang ist zu vermeiden, in den Gestaltungsprozess einzugreifen, indem z. B. manuell Zeilenumbrüche („`\newline`“ oder „`\\`“) eingefügt werden oder Abstände. Ausnahmen bitte nur in begründeten Fällen wie in dieser Vorlage bei der Gestaltung des Deckblatts.

Weitere Infos dazu, wie Sie mit dieser Vorlage hier weiterarbeiten können, finden Sie in Abschnitt 7.18.

## 7.2 Überschriften

Wir nutzen in dieser Vorlage das Kapitel („`\chapter`“) als höchste Gliederungsebene. Danach kommen Abschnitte („`\section`“) und Unterabschnitte („`\subsection`“). Diese drei Ebenen werden nummeriert und erscheinen im Inhaltsverzeichnis. Falls Sie Ihren Text weiter gliedern wollen, gibt es noch den „`\paragraph`“-Befehl.

Bitte beachten Sie, dass im Text nie zwei Überschriften direkt aufeinander folgen sollten. Nach einer Überschrift kommt immer erst etwas Text (siehe z. B. die Kapitelanfänge hier auf Seite 19 und Seite 22). Für weitere Hinweise vgl. Abschnitt 6.3.

## 7.3 Absätze

Stellen im Text, an denen ein neuer Absatz beginnen soll, können im Quellcode durch „`\par`“ markiert werden. Wie diese Absätze im fertigen Dokument genau aussehen, wird durch den Parameter „`\parskip`“ in der Dokumentenklasse bestimmt – dazu mehr in Abschnitt 7.18. Das ist ein großer Vorteil von L<sup>A</sup>T<sub>E</sub>X: Der Stil kann jederzeit für das gesamte Dokument einfach verändert werden.

Hinweis: Sie erhalten das gleiche Verhalten auch, wenn Sie im Quellcode statt des „`\par`“-Befehls eine leere Zeile stehen lassen. Vielleicht gefällt Ihnen das sogar noch besser.

## 7.4 Silbentrennung

Die automatische Silbentrennung in L<sup>A</sup>T<sub>E</sub>X funktioniert grundsätzlich gut. Es kann aber immer mal kleinere Probleme und erwünschtes Verhalten geben. Wenn Sie die Trennung für ein bestimmtes Wort beeinflussen möchten, können Sie mit dem „`\hyphenation`“-Befehl manuell die erlaubten Trennstellen spezifizieren. So kann man insbesondere erreichen, dass bestimmte Wörter nie getrennt werden, was z. B. für Eigennamen unerwünscht sein könnte.

Zum Beispiel werden Wörter, die einen Bindestrich enthalten, *nur* dort getrennt, das kann dazu führen, dass Zeilen nicht richtig dargestellt werden können, was zu einer Warnung führt (siehe Abschnitt 7.17). In solche Fällen müssten Sie im Quellcode manuell zusätzlich Trennstellen angeben.

## 7.5 Aufzählungen

Nutzen Sie die Umgebungen

- „`\begin{itemize}`“ ... „`\end{itemize}`“
- „`\begin{enumerate}`“ ... „`\end{enumerate}`“
- „`\begin{description}`“ ... „`\end{description}`“
- „`\begin{labeling}`“ ... „`\end{labeling}`“

um schöne Listen zu erstellen. Auch hier gilt, dass das genaue Aussehen im Dokument global eingestellt wird, das können Sie jederzeit verändern, dazu mehr in Abschnitt 7.18.

## 7.6 Abbildungen

Wenn jemand Ihre fertige Arbeit in die Hände bekommt, kann es gut sein, dass sie/er zunächst grob durchblättert, dabei kaum Text liest, aber die Abbildungen anschaut. Aus dieser Erfahrung entstammt die „Regel“, dass man die wichtigsten Punkte der Arbeit auf diese Weise verstehen können sollte.

Abbildungen stehen nie alleine, sondern werden durch die Unterschrift (*caption*) beschrieben. Dabei sollte alles enthalten sein, was notwendig ist, um die Abbildung zu verstehen. Nur in Ausnahmefällen muss man in der Unterschrift auf den Text verweisen. Umgekehrt muss auf jede Abbildung mindestens ein Mal im Text verwiesen werden, dazu siehe auch Abschnitt 7.12.

In den folgenden beiden Abschnitten wird zwischen Bildern (in Abschnitt 7.6.1) und Vektorgrafiken (in Abschnitt 7.6.2) unterschieden, da es sich um ganz unterschiedliche Techniken handelt, die jeweils passend genutzt werden sollten.

Denken Sie daran, dass nicht alle Menschen alle Farben gleich gut sehen können. Etwa 10 % der Männer in Deutschland sind beispielsweise von einer Rot-Grün-Schwäche betroffen. Vielleicht wird Ihre Arbeit auch auf einem Schwarz-Weiß-Drucker gedruckt. Daher sollten Sie Abbildungen im besten Fall so gestalten, dass sie auch ohne Farben verständlich sind.

### 7.6.1 Bilder

Bilder können Sie mit „`\includegraphics`“ einbinden. Es reicht (und wird sogar empfohlen!), den Dateinamen ohne Endung und ohne Pfad anzugeben. Beim Kompilieren werden alle Verzeichnisse durchsucht, die im „`\graphicspath`“ angegeben sind. In aller Regel soll ein Bild nicht alleine im Dokument erscheinen, sondern in einer



Abbildung 7.1: Vielleicht handelt es sich hierbei um Kunst?

Umgebung, die die automatische Nummerierung sicherstellt, eine Bildunterschrift hinzufügt und schließlich ermöglicht, dass die Abbildung an einer optimalen Stelle platziert wird (daher auch die Bezeichnung „Gleitumgebung“. In diesem Fall ist das die „`\figure`“-Umgebung.

Für die Umgebung stellen wir ein, wo sie auftauchen darf (dazu siehe auch Abschnitt 7.1). Dabei steht `t` für ganz oben auf der Seite, `b` für ganz unten und `h` für „hier“, was also die Positionierung innerhalb des Texts meint. Falls Sie mal Platz sparen müssen, sind `t` und `b` zu bevorzugen.

Achtung: Viele Inhalte wie Formeln, Code, Diagramme, Visualisierung von Daten, usw. sollten *nicht* als Bild eingefügt werden, sondern in einer passenden Form. Dazu siehe den folgenden Abschnitt über Vektorgrafiken.

### 7.6.2 Vektorgrafiken

Einfache Abbildungen (z. B. Koordinatensysteme, Ablaufdiagramme, usw.) müssen Sie nicht als Bild einfügen. Stattdessen können diese im Quellcode direkt erzeugen können. Dafür bietet sich das mächtige „`tikz`“-Paket an.

Ein Vorteil ist, dass Ihr Dokument so kleiner bleibt. Aber auch, dass die Abbildungen i. d. R. hübscher aussehen. Das gilt insbesondere beim Betrachten am Bildschirm, da sich Vektorgrafiken beliebig skalieren lassen. Das erlaubt es Ihnen sogar, Ihre Daten,

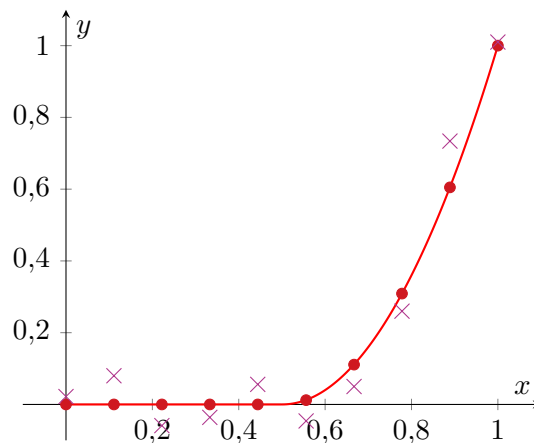


Abbildung 7.2: Eine schöne Grafik, die im Quellcode erzeugt wird!

z. B. aus Experimenten, separat zu halten und entsprechende Abbildungen dynamisch daraus zu generieren. Siehe dazu das Beispiel in Abbildung 7.2.

Sie finden ganz viel Beispiele zu TikZ natürlich im Internet. Außerdem gibt es ein aktuelles Buch **kottwitz:tikz**.

## 7.7 Tabellen

Grundsätzlich werden Tabellen in L<sup>A</sup>T<sub>E</sub>X mit der „`\tabular`“-Umgebung gebaut. Das ist dann nur die Tabelle selbst, ohne Nummerierung und ohne Bildunterschrift. Das Prinzip ist also das gleiche wie bei Abbildungen (s.o.): Erst die Umgebung (hier „`\table`“), darin die Tabelle selbst. Vielleicht sind die Befehle `rowcolor` oder

Überschrift links	Überschrift rechts
1	2222
10	222
100	22

Tabelle 7.1: Eine einfache Tabelle

`multicolumn` irgendwann für Sie nützlich. Es gibt noch viele weitere Pakete, die helfen, noch hübschere Tabellen zu gestalten, beispielhaft seien hier nur `array`, `booktabs` und `tabularx` genannt.

## 7.8 Abbildungs- und Tabellenverzeichnis

Mit L<sup>A</sup>T<sub>E</sub>X lassen sich Abbildungs- und Tabellenverzeichnis automatisch erstellen. Dabei tauchen alle Einträge entsprechend auf, für die Sie *Gleitumgebungen* korrekt angelegt haben (siehe Abschnitt 7.6.1 und 7.7).

Für diese Verzeichnisse wird standardmäßig der Text aus der `caption` übernommen. Dabei kommt es immer wieder vor, dass diese Beschreibung zu lang ist. Dafür kann mit in der `caption` in eckigen Klammern optional eine kürzere Version angeben. Dazu siehe auch Abschnitt 8.1.4.

## 7.9 Formeln

Eine der größten Stärken von L<sup>A</sup>T<sub>E</sub>X ist, dass man viele Möglichkeiten hat, Formeln einfach und schön aufzuschreiben. Das „`amsmath`“-Paket ist in diesem Zusammenhang

besonders beliebt, weil es ganz viele Möglichkeiten bietet. Hier nur ein kleines Beispiel mit der „align“-Umgebung:

$$\sum_{i=1}^n i = 1 + 2 + \dots + (n-1) + n \quad (7.1)$$

$$= (1 + n) + (2 + (n-1)) + \dots \quad (7.2)$$

$$= \frac{1}{2} \cdot (n+1) \quad (7.3)$$

Aber auch einfache Formeln im Text wie  $x \in \mathbb{N}$  sind natürlich möglich. Ein häufiger Fehler dabei ist, dass der „Mathe-Modus“ im Text vergessen wird: Wir sprechen über den  $x$ -Wert und *nicht* den x-Wert.

Ebenso häufig gibt es den Fehler auch andersrum, also dass Text im Mathe-Modus geschrieben wird.:  $a_{falsch} = 42$ , aber  $a_{richtig} = 42$ , vielleicht auch  $a_{\text{richtig}} = 42$ .

Funktionen wie  $\sin$  werden automatisch gut dargestellt, wie in

$$\sin \alpha = \left( \frac{a}{c} \right) \quad (7.4)$$

Die Klammern wurden hier nur eingefügt, um den entsprechenden Mechanismus zu demonstrieren: automatisch wachsende Klammern!

Manchmal wollen Sie einen eigenen „Operator“ benutzen, der optisch gleich aussehen soll. Genau dafür ist der „`\DeclareMathOperator`“-Befehl da, damit kann man fehlende Funktionen wie etwa  $\text{sgn}(x)$  hinzufügen.

Es empfiehlt sich, allen mathematischen Symbole, die Sie in Ihrer Arbeit benutzen, im Quellcode sprechende Namen zu geben, das geht am einfachsten mit dem „`\newcommand`“-Befehl. Dann können Sie jederzeit anpassen, wie Sie den Gewichtsvektor  $\mathbf{w}$  im gesamten Dokument darstellen wollen oder wie die imaginäre Einheit  $i$  mit  $i^2 = -1$  aussehen soll. Das setzt natürlich voraus, dass Sie die Bezeichnungen konsequent nutzen. Dadurch wird aber auch Ihr Quellcode besser lesbar!

## 7.10 Quellcode, Pseudocode

Soll in der Abschlussarbeit ein Ausschnitt vom Quelltext dargestellt werden, so ist die naheliegende Idee, einfach einen Screenshot davon aufzunehmen und via `\includegraphics` als Abbildung einzufügen. Allerdings entpuppt sich diese Idee als schlecht, sobald das fertige Dokument näher herangezoomt wird: Sofort verpixelt der dargestellte Quelltext. L<sup>A</sup>T<sub>E</sub>X bietet hierfür jedoch eine elegantere Alternative:

Das Paket `listings` zum Darstellen von Quelltext – direkt im Quelltext des L<sup>A</sup>T<sub>E</sub>X-Dokuments oder aber direkt aus einer externen Datei ausgelesen.

Mithilfe der Umgebung `lstlisting` lässt sich der Quelltext direkt im L<sup>A</sup>T<sub>E</sub>X-Dokument eingeben. Mit dem Befehl `\lstinputlisting{<Datei>}` lässt sich der Quelltext aus einer externen `<Datei>` auslesen und darstellen. Achtung: Der Pfad zu `<Datei>`, relativ zum L<sup>A</sup>T<sub>E</sub>X-Dokument, muss hierbei angegeben werden.

Außerdem kann das Layout des im fertigen Dokument dargestellten Quelltexts beeinflusst werden. Hierfür existiert eine *key-value-Interface*, über welche mithilfe spezieller *keys* Einfluss auf Dinge wie beispielsweise die zu verwendende Schriftart oder die Hintergrundfarbe genommen wird. Dazu wird der Befehl `\lstdefinestyle{<Stil>}` verwendet. Dabei ist `<Stil>` eine Liste mehrerer Paare der Form `<key>=<value>`, welche jeweils durch ein Komma voneinander getrennt werden. Ein Beispiel ist in der Präambel dieser Vorlage, in der Datei `definitions.tex` zu finden. Für nähere Informationen sei an dieser Stelle auf die Dokumentation des Paktes verwiesen. Ein Beispiel für solch ein mit der Umgebung `lstlisting` erzeugten Quelltext ist in Abbildung 7.3 gegeben.

```
\lstdefinestyle{myLaTeX}{  
  language=TeX,  
  basicstyle=\footnotesize\ttfamily,  
  frame=single,  
  backgroundcolor=\color{gray!10},  
}
```

```
\begin{lstlisting}  
\lstdefinestyle{myLaTeX}{  
  language=TeX,  
  basicstyle=\footnotesize\ttfamily,  
  frame=single,  
  backgroundcolor=\color{gray!10},  
}  
\end{lstlisting}
```

Abbildung 7.3: Beispiel für ein `listing`, welches mithilfe der Umgebung `lstlisting` erstellt worden ist. Links ist das fertige Listing zu sehen, rechts ist der entsprechende Quelltext dargestellt, der zu ebenjener Ausgabe führt. Zufälligerweise handelt es sich um einen Ausschnitt desjenigen Stils, der in dieser Vorlage verwendet wird.

## 7.11 Weitere Verzeichnisse

Mithilfe des Pakets `glossaries` lassen sich weitere Verzeichnisse erzeugen. Ein Glossar sowie ein Abkürzungs- oder Symbolverzeichnis lassen sich direkt erzeugen. Außerdem können auch weitere Verzeichnisse definiert werden. Wer das komplette Potential von

`glossaries` ausschöpfen möchte, benötigt Perl auf dem Rechner sowie das Perl-Skript `makeglossaries`. Allerdings existiert auch eine „eingedampfte“ Variante mit etwas eingeschränkter Funktionalität, welche komplett ohne Perl und externes Skript auskommt. Hierzu sei auf die Dokumentation des Pakets verwiesen.

Durch die Option `toc` beim Laden von `glossaries` erscheinen die zusätzlichen Verzeichnisse auch im Inhaltsverzeichnis. Wird weiterhin das Paket `hyperref` verwendet, so sind die im Text ausgegebenen Einträge dieser Verzeichnisse Links, die direkt in das entsprechende Verzeichnis führen. Die Verzeichnisse selbst können dann durch den Befehl `\printglossaries` an der gewünschten Stelle im Dokument ausgegeben werden. Auch hier wird für weiterführende Informationen wieder auf die Dokumentation des Pakets verwiesen.

### 7.11.1 Glossar erstellen

Ein Glossar kann ohne weitere Vorkehrungen direkt verwendet werden. Ein Eintrag im Glossar kann dann über den Befehl `\newglossaryentry{<Label>}{<Spezifikation>}` definiert werden. Dabei ist `<Spezifikation>` eine *key-value*-Liste. Die wichtigsten *keys* sind `name` und `description`, über welche der Name und die Beschreibung des zu definierenden Eintrags festgelegt werden. Über den Befehl `\gls{<Label>}` kann dann der zuvor definierte Begriff im Text ausgegeben werden. Beispiel gefällig? Der Hund ist der beste Freund des Menschen.

### 7.11.2 Abkürzungsverzeichnis erstellen

Um ein Abkürzungsverzeichnis verwenden zu können, muss `glossaries` mit der Option `acronym` geladen werden. Eine Abkürzung kann dann in der Präambel über den Befehl `\newacronym{<Label>}{<Abkürzung>}{<Ausgeschrieben>}` definiert werden. Über den Befehl `\gls{<Label>}` kann dann die zuvor definierte Abkürzung im Text ausgegeben werden. Dabei stellt L<sup>A</sup>T<sub>E</sub>X dann automatisch sicher, dass die Abkürzung bei der ersten Erwähnung im Text ausgeschrieben wird. Bei allen späteren Erwähnungen wird dann nur noch die Abkürzung ausgegeben. Beispiel gefällig? Das ist eine SVM. Und dort ist gleich noch eine SVM.

### 7.11.3 Symbolverzeichnis erstellen

Um ein Symbolverzeichnis verwenden zu können, muss `glossaries` mit der Option `symbols` geladen werden. Ein Symbol kann dann in der Präambel über den Befehl `\newglossaryentry{<Label>}{<Spezifikation>}` definiert werden. Der Befehl



ist also genau derselbe wie beim Glossar. Zusätzlich muss für *Spezifikation* noch `type=symbols` angegeben werden. Über den Befehl `\gls{<Label>}` kann dann das zuvor definierte Symbol im Text ausgegeben werden. Beispiel gefällig? Die Kraft  $\vec{F}$  ist gemäß der folgenden Gleichung definiert:

$$\vec{F} = m \cdot \vec{a}$$

## 7.12 Verweise

Setzen Sie in Ihrem Quellcode Marken mit dem „`\label`“-Befehl. Aus der Platzierung geht hervor, auf welche Nummerierung sich die Marke bezieht, also etwa Gliederungsebene (siehe Abschnitt 7.2), Tabelle (siehe Abschnitt 7.7), Abbildung (siehe Abschnitt 7.6) oder Gleichung (siehe Abschnitt 7.9). Alle genannten werden nämlich separat nummeriert, das kann am Anfang etwas gewöhnungsbedürftig sein.

Auf die markierten Stellen können Sie dann mit dem „`\ref`“-Befehl verweisen, wobei der eben nur die passende Nummer liefert. Die passende Bezeichnung, z. B. „Abbildung“, müssten Sie dann selbst ergänzen. Daher haben wir hier das Paket `cleveref` eingebunden, das uns den zuletzt genannten Schritt automatisiert.

Auf jede Abbildung und jede Tabelle muss im Text verwiesen werden, es dürfen keine nummerierten Umgebungen einfach „in der Luft hängen“. Im Gegensatz dazu müssen Abschnitte und Gleichungen nicht alle explizit referenziert werden. Sie können aber Ihren Leser\*innen helfen, wenn Sie über sinnvolle Verweise nachdenken.

## 7.13 Besondere Abstände und Zeichen

An Leerzeichen kann grundsätzlich ein Zeilenumbruch (oder sogar Seitenumbruch) erfolgen. In manchen Fällen möchte man das vermeiden, u. a., weil Zeilen nicht mit Zahlen beginnen sollten. Ein typisches Beispiel ist „Lange Straße 123“. Hier benötigt man hinter „Straße“ ein sog. geschütztes Leerzeichen, das mit einer Tilde erzeugt wird.

Genauso wie Gleitumgebungen optimal verteilt werden (vgl. Abschnitt 7.1), werden auch horizontale Abstände zwischen Wörtern und Sätzen in L<sup>A</sup>T<sub>E</sub>X in jeder Zeile dynamisch angepasst. Dabei werden alle Punkte als Satzende interpretiert. Bei Abkürzungen wie „z. B.“ sieht das nicht schön aus, der Leerzeichen-Abstand ist zu groß. Hier muss in der Mitte manuell ein halbes Leerzeichen erzeugt werden mit „\,“. Wenn Ihnen das Tippen solcher Konstrukte zu umständlich erscheint, können Sie

sich eigene Kommandos wie „\zb“ definieren. Zum Definieren eigener Befehle vgl. Abschnitt 7.9.

Auch bei waagerechten Strichen gibt es, wie bei Leerzeichen, unterschiedliche Längen. Für Gedankenstriche – solche hier – oder wenn „bis“ gemeint ist (wie in 14:00–16:00), reicht der einfache Bindestrich (Minuszeichen) nicht aus, das passende Teichen wird in L<sup>A</sup>T<sub>E</sub>X einfach durch ein Doppel-Minus erzeugt.

Problematisch im Quellcode sind alle Zeichen, die in L<sup>A</sup>T<sub>E</sub>X eine Funktion haben: Prozentzeichen %, kaufmännisches Und &, Unterstrich \_, geschweifte Klammern { ... } sind typische Beispiele. Diese müssen im Quelltext mit einem *Backslash* eingegeben werden, sonst erhält man Fehlermeldungen.

## 7.14 Wahl der Grundschriftart

Standardmäßig verwendet L<sup>A</sup>T<sub>E</sub>X für den Fließtext eine serifenbehaftete Schriftart. Für gedruckte Arbeiten sind serifenbehaftete Schriften vorteilhaft, weil die Serifen die Grundlinie betonen und somit das Auge beim Rücksprung am Zeilenende zum Beginn der nächsten Zeile unterstützt. Außerdem führen die unterschiedlichen Strichstärken zu eindeutigeren Wortbildern und unterstützen somit den Leseprozess.

Wird solch ein Dokument jedoch an einem alten Monitor mit geringer Auflösung betrachtet, so kann es sein, dass die feinen Serifen nicht mehr vernünftig dargestellt werden. In solch einem Fall kann es vorteilhaft sein, eine serifenlose Schrift zu verwenden. Auch kann es sein, dass serifenlose Schriften aus Gründen der Barrierefreiheit bevorzugt werden.

In diesem Fall kann mit dem Befehl `\renewcommand{\familydefault}{\sfdefault}` eine serifenlose Schrift als Grundschriftart festgelegt werden. Wer `lualatex` zum Kompilieren sowie das Paket `fontspec` verwendet, kann außerdem auf alle verfügbaren Schriften zugreifen. Ist auf dem Rechner die Schriftart Arial vorhanden, so kann mit dem Befehl `\setsansfont{Arial}` die Schriftart Arial als serifenlose Schrift festgelegt werden. Am Ende der Datei `definitions.tex` sind die beiden besagten Zeilen zu finden und müssen bei Bedarf nur auskommentiert werden.

## 7.15 Metadaten für den pdf-Betrachter

Manche pdf-Betrachter können zusätzliche Metadaten, wie Name des Autors, Titel des Dokuments (auch abweichend vom Namen der Datei) oder Schlüsselbegriffe anzeigen. Mit dem Paket `hyperref` lassen sich diese Metadaten mit dem Befehl

`\hypersetup{⟨Einsellungen⟩}` konfigurieren. Dabei ist *⟨Einsellungen⟩* eine *key-value*-Liste. Die wesentlichsten *keys* sind `pdfauthor`, `pdftitle` und `pdfkeywords`. Die Bedeutung dieser *keys* ist selbsterklärend.

Außerdem werden Links im Dokument (bei Verwendung des Pakets `hyperref`) in manchen pdf-Betrachtern als farbige Kästchen hervorgehoben. Diese farbigen Kästchen erscheinen natürlich nicht im gedruckten Dokument. Sie dienen lediglich als Hilfe, dass man nicht „auf gut Glück“ mit dem Cursor über das Dokument fahren muss, bis man den Link gefunden hat. Wenn die farbigen Kästchen stören, so können diese in `\hypersetup` mit `hidelinks` deaktiviert werden.

## 7.16 Wechsel zwischen ein- und doppelseitigem Layout

Diese Vorlage ist für ein einseitiges Layout optimiert. Dabei sind die linken und rechten Ränder jeweils gleich groß auf allen Seiten, neue Kapitel beginnen unmittelbar auf der nächsten Seite. Wird das fertige pdf-Dokument am Computer betrachtet, sieht das genau richtig aus. Soll das Dokument hingegen doppelseitig ausgedruckt werden, so kann das Layout noch etwas angepasst werden: Typischerweise sind die inneren Ränder dann etwas schmaler als die äußeren Ränder. Und neue Kapitel beginnen jeweils auf einer neuen, rechten Seite – was zu einzelnen Vakantseiten zwischen den Kapiteln führen kann. Für doppelseitig ausgedruckte Dokumente sieht das dann besser aus.

Um das doppelseitige Layout zu aktivieren, genügt es bereits, die Auskommentierung der Option `twoside` im optionalen Argument von `\documentclass` zu entfernen.

## 7.17 Kompilieren

Das Erstellen (Kompilieren) von großen Dokumenten mit L<sup>A</sup>T<sub>E</sub>X kann verhältnismäßig lange dauern. Da man i. d. R. nur an wenigen Stellen gleichzeitig arbeitet, kann es daher sinnvoll sein, übrige Teile auszukommentieren. Das geht besonders leicht, wenn man Text in getrennte Dateien auslagert und mit dem „`\input`“- und/oder „`\include`“-Befehl einbindet. So bleibt auch das Hauptdokument übersichtlich.

Grundsätzlich sollte das Ziel sein, dass Ihr Dokument ohne Warnungen kompiliert. Am besten kümmert man sich regelmäßig darum, entsprechende Probleme zu beheben.

Eine typische Warnung ist „*Reference ... undefined*“. Vielleicht verschwindet sie beim nochmaligen Erstellen, denn erst dann sind ggf. neue Positionen bekannt. Wenn

diese Warnung bleibt, muss das Problem unbedingt behoben werden, sonst haben Sie irgendwo Fragezeichen im Text stehen.

Warnungen, die sich auf zu volle Boxen beziehen, sind teilweise schwieriger zu verstehen und / oder zu beheben. Im **draft**-Modus (vgl. Abschnitt 7.18) werden die zugehörigen Stellen genau markiert, das kann eine große Hilfe sein. Gegen zu lange Zeilen hilft teilweise, Trennstellen zu markieren (vgl. Abschnitt 7.4). Sonst muss ggf. ein Satz minimal umformuliert werden.

Der **draft**-Modus hat drüber hinaus den Vorteil, dass das Kompilieren schneller geht (s. o.), dafür werden für Abbildungen nur Platzhalter eingefügt.

## 7.18 Diese Vorlage

In dieser Vorlage wird KOMA-Script verwendet, eine „Sammlung von Klassen und Paketen für L<sup>A</sup>T<sub>E</sub>X“<sup>1</sup>, die insbesondere das Erstellen von deutschen Texten mit den entsprechenden üblichen typographischen Standards unterstützt.

Dokumente mit L<sup>A</sup>T<sub>E</sub>X zu erstellen ist ganz ähnlich wie Programmieren. Ein Beispiel: Überall dort, wo ein Absatz entstehen soll, haben wir in unserem „Quellcode“ den Befehl „**\par**“ benutzt. Was dieser Befehl genau tut, wird durch dessen Implementierung festgelegt. Und diese ergibt sich hier sozusagen aus dem Parameter **parskip** der Dokumentklasse.

Andere Einstellungen, die direkt in der Dokumentklasse erfolgen können, betreffen z. B. die Schriftgröße, die Bindungskorrektur (**BCOR**) und die Größe von Überschriften (**headings**). Im Prinzip kann man auch die Größe der Ränder mit dem **DIV**-Parameter beeinflussen, davon wird aber abgeraten. Die Ränder werden automatisch so eingestellt, dass Zeilen eine Länge haben, die gut zu lesen ist.

Hier haben wir außerdem die Option **twoside** gewählt, für beidseitigen Druck. Daher sind die Ränder außen auf geraden und ungeraden Seiten unterschiedlich. Falls Sie Ihr Dokument am Ende einseitig drucken wollen, stellen Sie das bitte um.

Nach der Festlegung der Dokumentklasse haben wir in der sog. Präambel einige Pakete eingebunden. Zum Beispiel das **scrlayer-scrpage**-Paket, mit dem wir das Aussehen der Fuß- und Kopfzeile definieren können. Diese Vorlage wurde so eingerichtet, dass in den Kopfzeilen einer Doppelseite oben links immer die aktuelle Kapitel-Überschrift und oben rechts die aktuelle Abschnitt-Überschrift angezeigt wird (siehe **definitions.tex**).

---

<sup>1</sup><https://komascript.de>

In der vorliegenden Vorlage werden einige Pakete eingebunden. Im Folgenden wird die Funktion der wichtigsten davon kurz erläutert.

**fontspec** Erlaubt die freie Wahl der Schriftart (funktioniert aber nur bei Kompilation mit **lualatex**)

**babel** Erlaubt das Umstellen der Standard-Sprache auf Deutsch

**selnolig** Sorgt für automatisch korrekt gesetzte Ligaturen (funktioniert aber nur bei Verwendung von **fontspec** und somit auch **lualatex**, übernimmt automatisch die Spracheinstellung von **babel**)

**microtype** Optimiert das Aussehen des Textes (Satzspiegel)

**csquotes** Sorgt für automatisch korrekt gesetzte Anführungszeichen (übernimmt automatisch die Spracheinstellung von **babel**)

**tikz und pgfplots** Damit können Abbildungen direkt im Quellcode erzeugt werden, vgl. Abschnitt 7.6.2

**hyperref** Anklickbare Links im PDF

**biblatex** Verbesserte Quellenangaben und -verzeichnis

**amsmath und amssymb** Große Erweiterung der Möglichkeiten, mathematische Inhalte darzustellen

**listings** Zur Darstellung von Quellcode, vgl. Abschnitt 7.10

**cleveref** Vereinfacht das Einfügen von Verweisen (siehe Abschnitt 7.12)

**glossaries** Komfortables Erstellen eines Abkürzungsverzeichnis

## 8 Zitate und Literaturangaben

Fremdes Gedankengut muss immer kenntlich gemacht werden. Vor allem muss es überprüfbar und auffindbar sein. Hierzu dient die Technik des Zitierens und Belegens.

Verschiedene Fachrichtungen und Studiengänge folgen spezifischen Zitierkonventionen. Wichtige Hinweise zu gängigen Zitationssystem und -stilen finden Sie in den E-Learning-Kursen des Schreibzentrums<sup>1</sup>.

### 8.1 Zitieren

Wörtlich übernommene Textpassagen werden durch Anführungszeichen unten und oben („...“) kenntlich gemacht.

Wenn Ihr Zitat bereits ein Zitat enthält, müssen Sie die „doppelten Anführungszeichen“ im Text durch ‚einfache Anführungszeichen‘ ersetzen. Dazu vergleiche auch Abschnitt 7.13.

#### 8.1.1 Quellenverweise

Für alle Zitate muss ein Quellenverweis erstellt werden. Der Quellenverweis ist eine im Zitationsstil festgelegte Kurznotation, die auf die vollständige Literaturangabe im Literaturverzeichnis verweist. Quellenverweise können *entweder* im laufenden Text (anglo-amerikanische Zitierweise bzw. Harvard-Stil) *oder* über eine Fußnote am unteren Ende der Seite *oder* in einer Endnote am Ende des gesamten Textes erfolgen. Hier gelten unterschiedliche fachliche Konventionen, die unbedingt beachtet werden müssen.

Entscheidet man sich für Kurzverweise im Textfluss oder für Endnoten, kann der Fußnotenbereich für Kommentare und für Verweise auf Stellen im eigenen Text genutzt werden.

*Beachten Sie die Positionen von Hochzahl und Satzzeichen:*

---

<sup>1</sup>[https://ilu.th-koeln.de/goto.php?target=cat\\_52109&client\\_id=thkilu](https://ilu.th-koeln.de/goto.php?target=cat_52109&client_id=thkilu)

Wenn das wörtliche Zitat selbst mit einem Punkt endet, steht dieser vor dem beendenden Anführungszeichen. Die Hochzahl folgt dann direkt danach ohne Leerzeichen. Ein Punkt für den eigenen Satz entfällt dann.

Wenn das wörtliche Zitat nicht mit einem Punkt endet gibt es zwei Fälle: Steht das Zitat mitten im eigenen Satz, folgt die Hochzahl direkt nach den Anführungszeichen. Steht das Zitat am Ende des eigenen Satzes, notiert man zuerst die Anführungszeichen, dann den eigenen Satzpunkt und erst dann die Hochzahl.

Damit die Quellenverweise auch bei mehreren gleichlautenden Kurztiteln oder Jahreszahlen eines Autors eindeutig bleiben, erhalten die Jahreszahlen einen zusätzlichen kleinen Buchstaben.

### 8.1.2 Derselbe und Ebenda

Textverarbeitungsprogramme machen das Verweisen über *derselbe* und *ebenda* heute überflüssig, da man nicht mehr gezwungen ist, die gleichen Angaben wieder und wieder abzutippen. Sollten Sie sich (zum Beispiel auf Anraten Ihres Prüfenden) dennoch für dieses Verfahren entscheiden, empfehlen wir dringend, *ebenda* und *derselbe* etc. *erst in der redaktionellen Endphase* einzufügen, weil die Bezüge erst dann klar sind.

### 8.1.3 Zitate aus zweiter Hand

Bei Zitaten aus zweiter Hand, sog. Sekundärzitaten, übernimmt man ein Zitat eines Autors oder einer Autorin, ohne sich in der Originalquelle über den Sinnzusammenhang informiert zu haben. Zitate aus zweiter Hand sind nur zulässig, wenn die Originalquelle nicht beschaffbar ist. Bei allgemein zugänglicher wissenschaftlicher Literatur sind Zitate aus zweiter Hand unbedingt zu vermeiden. Ist es nicht möglich, ein Zitat mit dem Originaltext zu vergleichen, dann notiert man zitiert nach (es folgt die Quelle, der man das Zitat entnommen hat) oder zitiert in. Dies kommt zum Beispiel vor, wenn man wissenschaftsgeschichtliche Sachverhalte aus Lehrbüchern zitiert.

### 8.1.4 Abbildungen und Tabellen zitieren

Hier gelten die gleichen Richtlinien wie für Textzitate, d. h. auch hier gibt es getreue und abgeänderte Übernahmen. Beim originalgetreuen Zitat können Sie eine Abbildung z. B. in PowerPoint oder einem Zeichen-/Vektorprogramm genau nachbilden. Beim Scannen ist das Copyright zu berücksichtigen; es ist nur dann erlaubt, wenn der Autor bzw. der Verlag die – zumeist kostenpflichtige – Erlaubnis dazu erteilt hat. Der Quellenverweis ist in diesem Fall wie beim wörtlichen Zitat zu gestalten. Bei eigenen

Veränderungen, muss dem Quellenverweis ein Zusatz angehängt werden (z. B. *mit geringfügigen Veränderungen, mit eigenen Berechnungen*, usw.).

Weil Grafiken häufig übernommen werden, empfiehlt es sich, eigene Grafiken oder Bebilderungen auch als solche zu kennzeichnen.

In Tabellen- und Abbildungsunterschriften wird die Quelle immer direkt unter der Abbildung angegeben und nicht in einer Fuß- oder Endnote. Ins Abbildungs- und Tabellenverzeichnis gehören die Quellenangaben allerdings nicht. Dazu siehe Abschnitt 7.8.

## 8.2 Literaturverzeichnis

Nachfolgend wird die Gestaltung des obligatorischen Literaturverzeichnisses erläutert.

### 8.2.1 Inhalt und Anordnung der Literaturangaben

Im Literaturverzeichnis werden alle verwendeten Schriften in alphabetischer Reihenfolge nach Autorennamen gelistet. Das Literaturverzeichnis muss alle im Text zitierten Quellen beinhalten, aber auch keine darüber hinaus gehenden. Sind Werke nicht nur in gedruckter Form, sondern auch elektronisch publiziert, sollte die Literaturangabe der Druckfassung folgen. Es sei denn, das digitale Dokument hat eine feste Dokumentennummer (DOI). Wurde im Textteil in den Quellenverweisen statt des Titels ein Buchstabenkürzel verwendet, so ist diese Angabe auch im Literaturverzeichnis kenntlich zu machen.

Mehrere Werke, die ein Autor innerhalb eines Jahres veröffentlicht hat, müssen differenziert werden. Für die Kurztitel im Text sind die Jahreszahlen daher durch angehängte Kleinbuchstaben zu unterscheiden. Die Jahreszahl in der Quellenangabe bleibt jedoch unverändert ohne angehängte Buchstaben. Die Reihenfolge von a, b, c etc. richtet sich nach der Reihenfolge der Quellenverweise. Ob Vornamen abgekürzt oder ausgeschrieben werden, hängt von den Konventionen Ihres Faches ab. Bleiben Sie jedoch einheitlich.



### 8.2.2 Bibliographische Angaben im Literaturverzeichnis

Die Reihenfolge in der Notation der Literaturangabe hängt vom Dokumententyp und von fachlichen Konventionen ab. Bei Monografien sieht sie also anders aus als bei Zeitschriftenaufsätzen, in der Chemie wieder anders als in den Geisteswissenschaften oder der Mathematik. Auch für die *Interpunktion* gibt es keine einheitlichen Vorgaben.

Die Daten einer Literaturangabe entnehmen Sie bei Büchern dem so genannten Titelblatt. Dies ist nicht der Buchdeckel, sondern ein bedrucktes Blatt am Buchbeginn mit den wichtigsten werk- und buchidentifizierenden Angaben. Zunächst werden Verfasser oder Herausgeber zusammen mit dem Titel genannt, darauf folgen Erscheinungs- und Druckvermerke wie Verlag, Ort und Erscheinungsjahr. Es gibt zwar eine DIN-Regel für die Titelbeschriftung, die aber sehr unterschiedlich ausgelegt wird.

In den E-Learning-Kursen des Schreibzentrums<sup>2</sup> finden Sie weitere Informationen zu den bibliographischen Angaben für verschiedene Publikationstypen. Die hier folgende Literaturliste ist nur ein Beispiel für ein mögliches Format.

---

<sup>2</sup>[https://ilu.th-koeln.de/goto.php?target=cat\\_52109&client\\_id=thkilu](https://ilu.th-koeln.de/goto.php?target=cat_52109&client_id=thkilu)

## Literatur

- Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., & Grundmann, M. (2019). BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs. *CoRR*, *abs/1907.05047*. <http://arxiv.org/abs/1907.05047>
- Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1, 886–893 vol. 1. <https://doi.org/10.1109/CVPR.2005.177>
- Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., & Zafeiriou, S. (2019). RetinaFace: Single-stage Dense Face Localisation in the Wild. *CoRR*, *abs/1905.00641*. <http://arxiv.org/abs/1905.00641>
- dlib. (2022). *Dlib C++ Library* [Developer Documentation für dlib]. Verfügbar 28. April 2025 unter <https://dlib.net>
- dlib. (o.D. a). *cnn-face-detector* [Developer Documentation für cnn-face-detector.py von dlib]. Verfügbar 3. Mai 2025 unter [dlib.net/cnn\\_face\\_detector.py.html](https://dlib.net/cnn_face_detector.py.html)
- dlib. (o.D. b). *Dlib C++ Library* [Dlib Python API Documentation]. Verfügbar 2. Mai 2025 unter [dlib.sourceforge.net/python/index.html](https://dlib.sourceforge.net/python/index.html)
- dlib. (o.D. c). *face-landmark-detection.py* [Developer Documentation für face-landmark-detection.py von dlib]. Verfügbar 2. Mai 2025 unter [https://dlib.net/face\\_landmark\\_detection.py.html](https://dlib.net/face_landmark_detection.py.html)
- dlib. (o.D. d). *get-frontal-face-detector* [Developer Documentation für get-frontal-face-detector von dlib]. Verfügbar 28. April 2025 unter [https://dlib.net/face\\_detector.py.html](https://dlib.net/face_detector.py.html)
- dlib. (o.D. e). *scan-fhog-pyramid* [Developer Documentation für scan-fhog-pyramid von dlib]. Verfügbar 28. April 2025 unter [https://dlib.net/train\\_object\\_detector.py.html](https://dlib.net/train_object_detector.py.html)
- Esri Developer. (o.D.). *How single-shot detector (SSD) works?* [ArcGIS API for Python Documentation]. Verfügbar 26. April 2025 unter <https://developers.arcgis.com/python/latest/guide/how-ssd-works/>
- GeeksforGeeks. (2025). *Introduction to Convolution Neural Network* [Erklärung von CNNs]. Verfügbar 3. Mai 2025 unter [www.geeksforgeeks.org/introduction-convolution-neural-network/](https://www.geeksforgeeks.org/introduction-convolution-neural-network/)
- Google AI Edge. (2024). *MediaPipe Framework* [Developer Documentation für MediaPipe]. Verfügbar 26. April 2025 unter [ai.google.dev/edge/mediapipe/framework](https://ai.google.dev/edge/mediapipe/framework)

- Google AI Edge. (2025a). *Face detection guide* [Developer Documentation für MediaPipe]. Verfügbar 26. April 2025 unter [https://ai.google.dev/edge/mediapipe/solutions/vision/face\\_detector](https://ai.google.dev/edge/mediapipe/solutions/vision/face_detector)
- Google AI Edge. (2025b). *MediaPipe Solutions guide* [Developer Documentation für MediaPipe]. Verfügbar 26. April 2025 unter <https://ai.google.dev/edge/mediapipe/solutions/guide.md>
- Howse, J. (2019). *OpenCV 4 for Secret Agents: Use OpenCV 4 in secret projects to classify cats, reveal the unseen, and react to rogue drivers, 2nd Edition*. Packt Publishing. <https://books.google.de/books?id=b1qWDwAAQBAJ>
- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *CoRR*, *abs/1408.5093*. <http://arxiv.org/abs/1408.5093>
- Kazemi, V., & Sullivan, J. (2014). One Millisecond Face Alignment with an Ensemble of Regression Trees. <https://doi.org/10.13140/2.1.1212.2243>
- King, D. E. (2009). Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, *10*, 1755–1758.
- King, D. E. (2015). Max-Margin Object Detection. *CoRR*, *abs/1502.00046*. <http://arxiv.org/abs/1502.00046>
- King, D. E. (2024). *dlib-models* [GitHub Page von dlib-models]. Verfügbar 2. Mai 2025 unter [github.com/davisking/dlib-models](https://github.com/davisking/dlib-models)
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In B. Leibe, J. Matas, N. Sebe & M. Welling (Hrsg.), *Computer Vision – ECCV 2016* (S. 21–37). Springer International Publishing.
- lounging-lizard & nttstar. (2025). *Does FacialAnalysis.get expect BGR or RGB?* [GitHub Issue zu FacialAnalysis.get]. Verfügbar 27. Mai 2025 unter <https://github.com/deepinsight/insightface/issues/2741>
- Petroff, M. A. (2021). Accessible Color Cycles for Data Visualization. *CoRR*, *abs/2107.02270*. <https://arxiv.org/abs/2107.02270>
- Serengil, S. I. (2020). *Deep Face Detection with OpenCV in Python* [Online; accessed 2025-04-26]. Verfügbar 26. April 2025 unter <https://sefiks.com/2020/08/25/deep-face-detection-with-opencv-in-python/>
- Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, I–I. <https://doi.org/10.1109/CVPR.2001.990517>
- Wang, C., Luo, Z., Lian, S., & Li, S. (2018). Anchor Free Network for Multi-Scale Face Detection, 1554–1559. <https://doi.org/10.1109/ICPR.2018.8545814>
- Wu, W., Peng, H., & Yu, S. (2023). YuNet: A Tiny Millisecond-level Face Detector. *Machine Intelligence Research*, *20*, 656–665. <https://doi.org/10.1007/s11633-023-1423-y>

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. *CoRR*, *abs/1604.02878*. <http://arxiv.org/abs/1604.02878>

## Anhang

## Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Anmerkung: In einigen Studiengängen findet sich die Erklärung unmittelbar hinter dem Deckblatt der Arbeit.

---

Ort, Datum

---

Unterschrift