
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@mail.th-koeln.de

eingereicht bei: Prof. Christian Noss
Zweitgutachter: Prof. Dr. Daniel Gaida

Siegburg, 12.06.2025

Inhaltsverzeichnis

Tabellenverzeichnis	II
Abbildungsverzeichnis	III
Glossar	IV
Abkürzungsverzeichnis	VI
1 Einleitung	1
2 Recherche	3
3 Tests	7
3.1 Stichprobentests	7
3.2 Größentests	18
3.3 Confidencetests	18
3.4 Vergleich	22
4 Implementierung	25
5 Fazit	28
Literatur	30

Tabellenverzeichnis

3.1	Ergebnisse des Größentests (Erkannte Gesichter [A = Alle erkannt] False-Positives)	18
3.2	Ergebnisse der Confidencetests (Minimal Gesichtserkennung [inkl. Nebengesichter] Maximal False-Positiv)	19
3.3	Ergebnisse der finalen Modelltests (False-Negatives [In negativen Zahlen dargestellt] False-Positives Gesichter die nur von diesem Modell erkannt wurden)	22

Abbildungsverzeichnis

3.1	Stichprobenergebnis des Haar-Cascade-Modells bei einem Zwei-Personen-Porträt (Gemälde aus dem Cranach Digital Archive (CDA), Cranach Digital Archive, o.D. e)	8
3.2	Stichprobenergebnis des Caffe-Modells bei einem Drei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. d)	10
3.3	Stichprobenergebnis des MediaPipe-Modells bei einem Einzelporträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. a)	11
3.4	Stichprobenergebnis des Dlib-HOG-Modells bei einem Zwei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. e) . .	12
3.5	Stichprobenergebnis des Dlib-HOG-Landmark-Modells bei einem Zwei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. e)	13
3.6	Stichprobenergebnis des Dlib-CNN-Modells bei einem Zwei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. e) . .	14
3.7	Stichprobenergebnis des MTCNN-Modells bei einem Drei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. d) . .	15
3.8	Stichprobenergebnis des Yunet-Modells bei einem Drei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. d)	16
3.9	Stichprobenergebnis des RetinaFace-Modells bei einem Drei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. d) . .	17
3.10	Confidenceergebnis des Caffe-Modells bei dem ein erkanntes Gesicht einen geringeren Wert als ein False-Positives hat (Gemälde aus dem CDA, Cranach Digital Archive, o.D. b)	20
3.11	Confidenceergebnis des MTCNN-Modells mit False-Positives (Gemälde aus dem CDA, Cranach Digital Archive, o.D. c)	21
3.12	MTCNN und RetinaFace haben jeweils Gesichter, die nur von ihnen markiert wurden (Gemälde aus dem CDA, Cranach Digital Archive, o.D. f)	24
4.1	GUI von <code>cranach_detector()</code> (Eigenes Bild)	26

Glossar

Anker In der Objekterkennung vordefinierte Referenzrechtecke unterschiedlicher Größen und Seitenverhältnisse, die verwendet werden, um mögliche Positionen und Größen von Objekten in einem Bild vorab zu definieren. Detektoren vergleichen diese Anker (engl. *anchors* oder *anchor boxes*) mit tatsächlichen Objekten, um deren Lage und Größe genauer zu bestimmen. 6

Backbone Grundlegendes neuronales Netzwerk-Modul in einem tiefen Lernmodell, das für die Extraktion von Merkmalen aus Eingabedaten zuständig ist. Im Kontext der Bildverarbeitung bezeichnet der Begriff oft ein vortrainiertes Convolutional Neural Network, das als Basis für weitere Aufgaben wie Klassifikation oder Objekterkennung dient. 3, 6, 13

Bildpyramide Mehrstufige Darstellung eines Bildes in verschiedenen Auflösungen. Höhere Ebenen der Pyramide enthalten kleinere, herunterskalierte Versionen des Originalbildes. Wird verwendet, um Objekterkennung effizient auf unterschiedlichen Skalen durchzuführen. 4, 5

Boosting Ein Verfahren des überwachenden Lernens, bei dem mehrere schwache Lernalgorithmen sequenziell trainiert und zu einem starken Klassifikator kombiniert werden. Jeder neue Klassifikator konzentriert sich auf die Fehler der vorherigen, um die Gesamtgenauigkeit zu erhöhen. 3, 5

Bounding-Box Rechteckiger Bereich in einem Bild, der ein Objekt umschließt, um dessen Lage und Größe zu bestimmen. 5, 6, 9

Confidence Alternativ bezeichnet als *Score*. Numerischer Wert, der die Wahrscheinlichkeit angibt, mit der ein Modell eine bestimmte Vorhersage für korrekt hält. Beschreibt in der Gesichtserkennung typischerweise die Sicherheit, mit der ein erkannter Bereich als Gesicht klassifiziert wird. 7–9, 12–14, 18, 19, 21–23, 25, 26, 28

Detektor In der Bildverarbeitung ein System oder Algorithmus, das bzw. der bestimmte Objekte in einem Bild lokalisiert. 3, 5, 6

Feature-Vektor Ein numerischer Vektor, der die Merkmale (engl. *features*) eines Objekts oder eines Bildausschnitts in strukturierter Form repräsentiert. Er dient als Eingabe für maschinelle Lernverfahren und Klassifikatoren. 4

Klassifikator Ein Algorithmus oder Modell, das Eingabedaten bestimmten Klassen zuordnet, um in der Bildverarbeitung beispielsweise Muster, Objekte oder Merkmale auf deren Vorhandensein zu überprüfen. 3, 4

Abkürzungsverzeichnis

Caffe Convolutional Architecture for Fast Feature Embedding. 3, 9, 19

CDA Cranach Digital Archive. 1, 7, 8, 10–18, 20, 21, 24, III

CNN Convolutional Neural Network. 5, 6, 11, 20, 22, 23, 27, 28

HOG Histogram of Oriented Gradients. 4, 5, 9–11, 18

MMOD Max-Margin Object Detection. 5

MTCNN Multi-task Cascaded Convolutional Networks. 5, 11, 21, 23, 27, 28

NMS Non-Maximum Suppression. 5

O-Net Output Network. 6

P-Net Proposal Network. 5

R-Net Refine Network. 5

SSD Single Shot MultiBox Detector. 3, 4, 6

SVM Support Vector Machine. 4

1 Einleitung

Im Rahmen der Masterarbeit von Pagelsdorf, 2024 wurden neue Wasserzeichen für das CDA entwickelt. Infolgedessen wurde der Ansatz entwickelt, diese automatisiert auf den verschiedenen Bildern der Werke des Archivs zu platzieren. Unter Zuhilfenahme von Gesichtserkennungsmodellen sollten die Gesichter aus ästhetischen Gründen ausgespart werden. Das Problem hierbei ist, dass Gesichtserkennungsmodelle hauptsächlich mit Trainingsdaten bestehend aus Gesichtern realer Menschen entwickelt werden und darauf ausgelegt sind, Gesichter realer Personen zu erkennen. Ziel dieser Arbeit ist das Testen und Vergleichen bestehender Gesichtserkennungsmodelle. Es soll festgestellt werden, ob sich diese für den Einsatz auf historischen Gemälden eignen. Sollten sich mehrere Modelle eignen, wird überprüft, welche für den bestimmten Anwendungskontext priorisiert werden. Hierfür werden die Modelle *Haar-Cascade*, *Caffe*, *MediaPipe*, *Dlib HOG*, *Dlib Landmark*, *Dlib CNN*, *MTCNN*, *Yunet* und *RetinaFace* verglichen. Aus den am besten geeigneten Modellen soll anschließend ein Python-Modul entwickelt werden, das die Gesichtsbereiche eines Bildes ausgibt. Mithilfe dieser Daten soll die Möglichkeit gegeben werden, zu überprüfen, ob sich ein Wasserzeichen potenziell auf einem Gesicht befindet, um so einen Automatisierungsprozess für dessen Platzierung zu ermöglichen.

Im Recherche-Kapitel wird kurz auf die getesteten Modelle eingegangen und ihre grobe Funktionsweise erläutert, um ihre Ergebnisse in einen Kontext einordnen zu können. Als erster Schritt der Testreihe werden in Stichprobentests die verschiedenen Modelle auf ihre grundsätzliche Eignung geprüft. Es soll festgestellt werden, ob ein getestetes Modell überhaupt in der Lage ist, mit historischen Gemälden zu arbeiten. Modelle, die diese Voraussetzung nicht erfüllen, werden bereits an dieser Stelle ausgeschlossen, um in späteren Tests Zeit und Aufwand zu sparen. Im Anschluss daran werden Tests mit Bildern unterschiedlicher Auflösungen durchgeführt. Ziel ist es, zu untersuchen, ob die Bildauflösung einen Einfluss auf die Ergebnisse der Modelle hat und welche Auswirkungen dabei zu beobachten sind. Zudem soll auf dieser Basis festgelegt werden, mit welcher Auflösung die Bilder in den folgenden Tests verwendet werden sollten, um möglichst gute Ergebnisse zu erzielen. Darauf folgend werden die Confidence-Grenzwerte der Modelle im nächsten Test optimiert. Dies soll sicherstellen, dass in der finalen Auswertung die jeweils besten Ergebnisse der Modelle miteinander verglichen werden können, um eine fundierte Entscheidung darüber zu treffen, welche Modelle für das zu entwickelnde Python-Modul verwendet werden sollten. Der finale Test

vergleicht die False-Negatives, False-Positives sowie die Anzahl der Gesichter, die jeweils nur von einem bestimmten Modell erkannt wurden. Das Ziel dieses Tests besteht darin, festzustellen, welche Modelle die zuverlässigsten Ergebnisse liefern und welche sich potenziell ergänzen lassen, um in möglichst allen Bildtypen eine vollständige Gesichtserkennung zu ermöglichen.

2 Recherche

Ziel dieses Kapitel ist es, die Funktionsweise der verschiedenen Gesichtserkennungsmodelle grob zu erläutern, um ihre Ergebnisse besser einordnen zu können. Dabei wird bewusst auf eine vertiefte technische Beschreibung verzichtet, da dies den Rahmen dieser Arbeit überschreiten würde. Des Weiteren haben erste Stichprobentests gezeigt, dass die Modelle bei Bildern historischer Werke messbar andere Ergebnisse liefern als bei realen Fotos.

- **Haar-Cascade:** Der **Haar-Cascade-Detektor** basiert auf dem von Viola und Jones entwickelten Verfahren zur schnellen Objekterkennung mittels einer kaskadierten Boosting-Architektur (Viola & Jones, 2001). Haar-ähnliche Merkmale werden ermittelt, indem die Differenz der Pixel-Summen benachbarter rechteckiger Regionen berechnet wird. Mithilfe eines Integralbilds lassen sich diese Merkmale in konstanter Zeit auswerten. Anschließend erfolgt die Merkmalsauswahl durch den AdaBoost-Algorithmus, der eine Vielzahl schwächer Klassifikatoren zu einem starken Klassifikator kombiniert. Die Klassifikatoren sind in mehreren Stufen kaskadiert, wobei jede Stufe unpassende Bildbereiche frühzeitig verwirft, um die Erkennungszeit zu verkürzen. In der ursprünglichen Implementierung umfasste die Kaskade 38 Stufen, beginnend mit nur einem Merkmal in der ersten Stufe und steigend auf bis zu 6000 Merkmale in späteren Stufen. In dieser Arbeit wird OpenCV verwendet, um mit der Klasse `CascadeClassifier` das vorgeführte XML-Modell zu laden. Das verwendete Modell `haarcascade_frontalface_default.xml` wurde mit dem **Caltech Frontal Face Dataset** von 1999 trainiert (Howse, 2019, 101–102).
- **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). In dieser Arbeit wird das Modell `res10_300x300_ssd_iter_140000.caffemodel` zusammen mit `deploy.prototxt` verwendet, um die Parameter der Netzwerkarchitektur zu definieren. Wie der Modellname bereits andeutet, basiert es auf dem Single Shot MultiBox Detector (SSD)-Framework mit einem **ResNet10**-Backbone. SSD-Modelle bestehen aus einem SSD-Head und einem Backbone (Esri Developer, o.D.). Häufig wird ein Netzwerk wie ResNet verwendet, wobei die finale Klassifikationsebene entfernt wird, um daraus das Backbone zu erzeugen. SSD nutzt mehrere Feature-Maps,

auf denen sogenannte Default-Boxes platziert werden, welche auf Merkmale untersucht werden (Liu et al., 2016). Die Angabe „300x300“ im Modellnamen zeigt an, dass das Modell mit Bildern der Größe 300×300 Pixel trainiert wurde. Damit das Modell auch größere Eingaben verarbeiten kann, wird die Funktion `blobFromImage` von OpenCV genutzt, um das Bild für die Gesichtserkennung auf 300×300 zu skalieren (Serengil, 2020). Die Trainingsdaten des Modells sind nicht öffentlich dokumentiert

- **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 bietet neben der Gesichtserkennung auch weitere Funktionen wie Hand- und Körpererkennung. In dieser Arbeit wird ausschließlich das Teilmodul *MediaPipe Face Detection* betrachtet, 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 128×128 Pixel trainiert wurde (Google AI Edge, 2025a). *BlazeFace* verwendet ein SSD-Framework, das für eine schnellere Performanz modifiziert wurde, sodass eine Geschwindigkeit von 200 bis zu 1000 FPS erreicht werden kann. Es wurde explizit für den Einsatz auf mobilen Endgeräten optimiert und geht daher effizienter mit GPU-Ressourcen um (Bazarevsky et al., 2019).
- **Dlib HOG:** Dlib ist ein Open-Source-C++-Toolkit, das unter anderem Gesichtserkennung mit vorgefertigten Methoden ermöglicht (dlib, 2022). Die hier verwendete Methode `get_frontal_face_detector` nutzt den Histogram of Oriented Gradients (HOG). HOG ist ein Feature-Deskriptor, der Bilder in Feature-Vektoren überführt, indem er lokale Gradienten berechnet, daraus Histogramme erstellt und diese blockweise normalisiert (Dalal & Triggs, 2005). Dlib kombiniert HOG mit einem Sliding-Window-Verfahren über eine Bildpyramide sowie einer Support Vector Machine (SVM) (dlib, o.D. d, o.D. e). Beim Sliding-Window-Verfahren wird mit einer festen Fenstergröße über das Bild iteriert und jeder so erzeugte Bildausschnitt analysiert (Esri Developer, o.D.). Die SVM dient als Klassifikator, um die vom HOG erzeugten Feature-Vektoren aus den verschiedenen Bildausschnitten zu bewerten und festzustellen, ob diese Gesichter enthalten (Dalal & Triggs, 2005).
- **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 Punkte im Gesicht wie Augen, Nase und Lippen (dlib, o.D. c). Das Modell verwendet 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).

- **Dlib CNN:** **mmod_human_face_detector.dat** ist ein Modell von Dlib, das ein Convolutional Neural Network (CNN) verwendet. Dlib selbst beschreibt das Modell als genauer im Vergleich zu HOG, weist aber auch darauf hin, dass die höhere Genauigkeit auf Kosten eines höheren Rechenaufwands und damit verbundener Latenz erreicht wird (dlib, o.D. a). Wie aus dem Namen des Modells ersichtlich, 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 – im Gegensatz zum Sliding-Window-Verfahren, das nur über Stichproben möglicher Fensteroptionen iteriert (King, 2015). Ein CNN erkennt Gesichter, indem es das Eingabebild schrittweise mit kleinen Matrizen, sogenannten Filtern oder Kernels, faltet (O’Shea & Nash, 2015). Dabei gleiten die Filter über das Bild und erzeugen durch die Faltung für jeden Filter eine eigene Feature-Map. Anschließend wird Pooling angewendet, welches benachbarte Bereiche zusammenfasst und so die Auflösung sowie den Rechenaufwand reduziert (GeeksforGeeks, 2025; O’Shea & Nash, 2015). Durch die Beibehaltung der zweidimensionalen Struktur kann ein CNN räumliche Zusammenhänge im Bild ausnutzen und so Merkmale wie Kanten zuverlässig erkennen.
- **MTCNN: Multi-task Cascaded Convolutional Networks (MTCNN)** kombinieren Gesichtserkennung und Gesichtsausrichtung in einem Modell (Zhang et al., 2016). Dafür werden drei 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, weniger tiefes CNN, das mithilfe einer Bildpyramide die ersten möglichen Bounding-Boxes markiert. Anschließend wird Non-Maximum Suppression (NMS) verwendet, um überlappende Bounding-Boxes zusammenzufügen. Im zweiten Schritt werden weitere falsche Bounding-Boxes, die keine Gesichter enthalten, vom sogenannten Refine Network (R-Net) verworfen. Dieses ist ein komplexeres CNN im Vergleich zum P-Net (Zhang et al., 2016). Zuletzt wird das stärkste CNN eingesetzt, nachdem nur noch

wenige Bounding-Boxes verbleiben. Das Output Network (O-Net) markiert fünf Landmark-Punkte und generiert so das Endergebnis (Zhang et al., 2016).

- **Yunet:** **Yunet** ist ein leichtgewichtiger Detektor und CNN, das speziell für den Einsatz auf ressourcenbeschränkten Geräten wie Smartphones entwickelt wurde (Wu et al., 2023). Anders als andere Detektoren wie SSD nutzt Yunet einen ankerfreien Ansatz (Wu et al., 2023). Indem auf Anker oder Default-Boxes verzichtet wird, benötigt Yunet weniger Rechenleistung. Allerdings haben ankerfreie Ansätze Probleme mit Bildern, die kleine oder Gesichtern unterschiedlicher Größen zeigen (Wang et al., 2018).
- **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 heißt 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 selbstüberwachtes Lernen verwendet wird (Deng et al., 2019).

3 Tests

Viele Gesichtserkennungsmodelle wurden mit Datensätzen trainiert, die ausschließlich Gesichter von realen Personen enthalten und für die Erkennung dieser optimiert. Modelle, die zuverlässig reale Gesichter detektieren, weisen bei der Anwendung auf Bilder historischer Gemälde häufig eine deutlich geringere Erkennungsgenauigkeit auf. Ziel der folgenden Tests ist es, die Eignung und Leistungsfähigkeit verschiedener Gesichtserkennungsmodelle für die Anwendung auf historischen Gemälden zu evaluieren. Begonnen wird mit den Stichprobentests, in welchen ungeeignete Modelle aussortiert werden sollen. Es folgen die Größentests, die dazu dienen, die Ergebnisse bei verschiedenen Auflösungen zu vergleichen. So soll sichergestellt werden, dass weitere Tests nur Bilder mit einer für die jeweiligen Modelle geeigneten Auflösung verwenden. Im Anschluss werden die Confidence-Grenzwerte der Modelle optimiert. Dafür werden die Confidence-Werte für erkannte Gesichter und False-Positives verglichen, um den bestmöglichen Grenzwert zu ermitteln. Im letzten Test werden alle Modelle direkt miteinander verglichen. Hierbei werden die False-Negatives, False-Positives sowie die Gesichter gezählt, die nur von einem Modell erkannt wurden.

3.1 Stichprobentests

Um Zeit und Aufwand zu sparen, wird in Stichprobentests die Eignung der verschiedenen Gesichtserkennungsmodelle überprüft. Ungeeignete Modelle werden am Ende dieser Stichprobentests ausgeschlossen. Hauptkriterium für die Eignung eines Modells ist die Fähigkeit, Gesichter auf historischen Gemälden zuverlässig zu erkennen. Als sekundäres Kriterium sollte die Anzahl an False Positives in einem vertretbaren Verhältnis stehen. Ein Modell, das zwar alle Gesichter erkennt, jedoch auch große Teile des restlichen Bildes fälschlicherweise als Gesichter markiert, erlaubt keine präzise Aussage darüber, ob sich ein Overlay auf einem Gesicht befindet. Weitgehend vernachlässigt wird der Aspekt der Rechenzeit. Im Kontext des CDA gilt es als wichtiger, Gesichter möglichst zuverlässig zu erkennen, als dies möglichst schnell zu tun. Hierfür wurden vier Werke ausgewählt: ein Einzelporträt, ein Zwei-Personen-Porträt, ein Drei-Personen-Porträt sowie ein Gruppenbild. Jedes Modell wird auf jedes der vier Werke angewendet, wodurch sich direkt ein erster Eindruck ergibt, wie gut sich die Modelle im Vergleich eignen. Durch die verschiedenen Bildtypen ist es zudem möglich, dass

sich bereits Eignungen für bestimmte Bildkategorien abzeichnen. Schließlich dienen die Tests auch dazu, erste Confidence-Grenzwerte für die Modelle zu definieren.

- **Haar-Cascade:** Haar-Cascade, genauer `haarcascade_frontalface_default.xml`, eignet sich in keiner Weise. Es scheitert sowohl daran, zuverlässig das Gesicht in einem Einzelporträt zu erkennen, als auch daran, mehrere Gesichter in einem Gruppenbild zu identifizieren. Zudem kommt es häufig zu False-Positives – teilweise werden mehr falsche als tatsächliche Gesichter markiert (siehe dazu Abbildung 3.1). Haar-Cascade besitzt keinen direkt einstellbaren Confidence-Grenzwert, jedoch ein Äquivalent. Selbst bei Variation dieses Wertes liefert das Modell keine zuverlässigen Ergebnisse und scheidet somit bereits an dieser Stelle für weitere Versuche aus. Auffällig ist zudem, dass Haar-Cascade bei identischen Motiven mit höherer Auflösung noch schlechter abschneidet, da vermehrt False-Positives entstehen.



Abbildung 3.1: Stichprobenergebnis des Haar-Cascade-Modells bei einem Zweipersonen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. e)

- **Caffe:** Das Caffe-Modell besteht den Stichprobentest. In Einzel-, Zwei- und Drei-Personen-Porträts werden jeweils alle Hauptgesichter erkannt, ohne False-Positive (siehe dazu Abbildung 3.2). Allerdings zeigt das Modell Schwächen bei kleineren Gesichtern oder solchen, die nicht im Fokus des Porträts stehen. Im Gruppenbild wurde kein einziges Gesicht erkannt – auch keine False-Positives. Um Gesichter auf dem Gruppenbild zu erkennen, müsste der Confidence-Grenzwert so stark reduziert werden, dass die resultierenden False-Positives die Ergebnisse unbrauchbar machen würden. Bei höherer Auflösung ändern sich die Resultate kaum – nur im Gruppenbild treten mehr False-Positives auf, was die Problematik unterstreicht. Bei geringerer Auflösung erkennt Caffe weiterhin alle Gesichter korrekt und dies ohne False-Positives. Im Drei-Personen-Porträt wurde ein Nebengesicht nur halb erkannt. Überraschenderweise wurden im Gruppenbild mit geringer Auflösung tatsächlich Bereiche markiert. Dabei wurden einige Gesichter korrekt erkannt, jedoch waren die Bounding-Boxes zu groß, um sinnvoll nutzbar zu sein. Wiederholte Tests mit mittlerer Auflösung (600×1130 Pixel) zeigten, dass ein Nebengesicht im Drei-Personen-Porträt nicht erkannt wurde, obwohl dies bei höheren und niedrigeren Auflösungen gelang. Diese Inkonsistenz erschwert die Auswahl einer geeigneten Bildauflösung und könnte ein Grund sein, Caffe letztlich auszuschließen.
- **MediaPipe:** MediaPipe erkennt das Gesicht im Einzelporträt zuverlässig und ohne False-Positives (siehe dazu Abbildung 3.3). Dies ist jedoch der einzige bestandene Stichprobentest. In allen anderen Tests werden weder Gesichter noch False-Positives markiert. Das deutet darauf hin, dass MediaPipe nur große Gesichter erkennt. Da das Modell im gewünschten Kontext kaum brauchbar ist, wird es nicht weiter getestet. Auch bei höher aufgelösten Bildern bleiben die Ergebnisse identisch.
- **Dlib HOG:** Das HOG-basierte Modell von dlib zeigt gute Leistungen bei der Erkennung kleiner Gesichter, hat jedoch Probleme mit größeren. Das Gesicht im Einzelporträt wird nur erkannt, wenn der Confidence-Grenzwert stark reduziert wird – was wiederum zu 0–3 False-Positives führt. Die Gesichter in Zwei- und Drei-Personen-Porträts sowie die meisten Gesichter im Gruppenbild werden hingegen gut erkannt (siehe dazu Abbildung 3.4). Trotz offensichtlicher Schwächen sind die Ergebnisse solide genug für weiterführende Tests. Bei hochauflösenden Bildern erkennt das Modell auch das Einzelporträt, allerdings steigt die Zahl der False-Positives, ähnlich wie bei geringem Confidence-Grenzwert und niedriger Auflösung. Dies deutet darauf hin, dass der Confidence-Grenzwert abhängig von der Bildauflösung angepasst werden muss, was den Einsatzaufwand erhöht. Bei geringer Auflösung gibt es im Einzelporträt ein False-Positive, im Zwei- und Drei-Personen-Porträt jedoch keine. Im Gruppenbild verschlechtern sich die Ergebnisse deutlich: Statt 6 Gesichtern mit 3 False-Positives werden nur

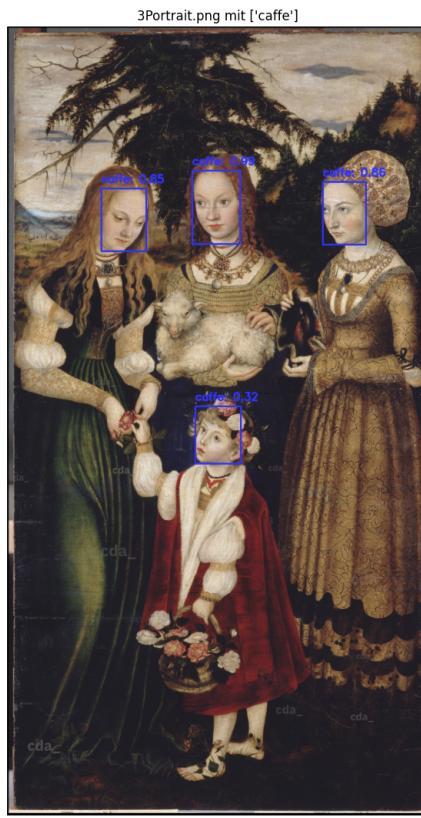


Abbildung 3.2: Stichprobenergebnis des Caffe-Modells bei einem Drei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. d)

noch 1 Gesicht und 2 False-Positives erkannt. Das zeigt, dass Gruppenbilder und Porträts separat betrachtet werden sollten. Bei mittlerer Auflösung erkennt das Modell bei Einzel-, Zwei- und Drei-Personen-Porträts alle Gesichter mit 1–2 False-Positives pro Bild.

- **Dlib HOG Landmark:** Anfangs wurde geplant, das Landmark-Modell ergänzend zur Überprüfung der Ergebnisse von Dlib HOG zu verwenden. Ziel war es, False-Positives auszusortieren und dadurch präzisere Ergebnisse zu erzielen. Diese Annahme bestätigt sich nicht. Zwar platziert das Modell die Landmarks korrekt, jedoch führt das Aussortieren basierend auf den Landmark-Daten häufig zur Entfernung korrekter Gesichter statt False-Positives (siehe dazu Abbildung 3.5). Das Modell liefert somit keinen Vorteil gegenüber Dlib HOG und wird daher nicht weiter getestet. Auch hochauflösende Bilder verbessern die Ergebnisse nicht. Im Gegenteil, es entstehen häufig mehr False-Positives. Im Drei-Personen-

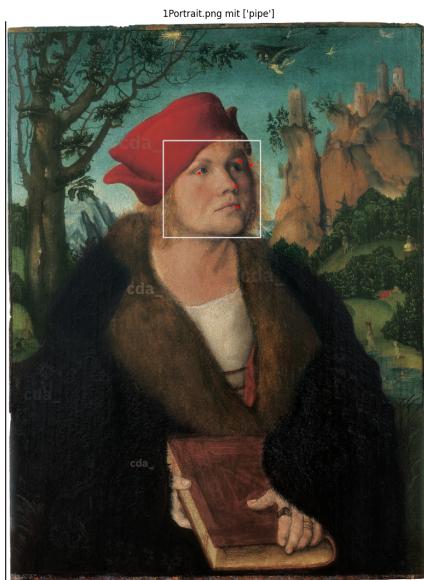


Abbildung 3.3: Stichprobenergebnis des MediaPipe-Modells bei einem Einzelporträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. a)

Porträt wurde zwar ein zusätzliches Gesicht erkannt, die Gesamtleistung liegt dennoch unter der vom einfacherem Dlib HOG.

- **Dlib CNN:** Dlib CNN, konkret `mmod_human_face_detector.dat`, ist das langsamste aller getesteten Modelle. Bereits bei Bildern mit einer Größe von 998×1314 Pixeln liegt die Bearbeitungszeit bei etwa 20 Sekunden, während alle anderen Modelle bei unter Drei Sekunden bleiben. Die Ergebnisse der Stichprobentests auf Porträts sind hingegen fehlerfrei (siehe dazu Abbildung 3.6). Im Gruppenbild wurde hingegen nichts erkannt. Aufgrund dieser hohen Zuverlässigkeit wird Dlib CNN weiter getestet. Ob sich die lange Bearbeitungszeit durch eine entsprechend hohe Genauigkeit rechtfertigt, muss sich noch zeigen. Bei hochauflösenden Bildern (1200×1593 Pixel) bleiben die Ergebnisse gleich, allerdings verdoppelt sich die Bearbeitungszeit auf ca. 40 Sekunden. Bei geringer Auflösung (400×531 Pixel) arbeitet das Modell deutlich schneller, erkennt jedoch nur das Gesicht im Einzelporträt. Bei mittlerer Auflösung werden wieder alle Gesichter erkannt, mit Ausnahme des Nebengesichts im Drei-Personen-Porträt. Anders als viele andere Modelle lässt Dlib CNN vermuten, mit höheren Auflösungen besser zu funktionieren, wobei es zu einer erheblich höheren Rechenzeit kommt.
- **MTCNN:** MTCNN besteht die Stichprobentests für Einzel-, Zwei- und Drei-Personen-Porträts fehlerfrei (siehe dazu Abbildung 3.7). Im Gruppenbild werden



Abbildung 3.4: Stichprobenergebnis des Dlib-HOG-Modells bei einem Zwei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. e)

einige Gesichter erkannt, allerdings nur solche mit $\text{Confidence} > 0.5$. Es gibt keine False-Positives. Somit wird MTCNN weiter getestet. Bei höherer Auflösung bleibt der durchschnittliche Confidence-Wert gleich, es treten jedoch in einzelnen Fällen 1–3 False-Positives auf. Im Gruppenbild werden mehr Gesichter erkannt (6 von 13), sowie keine neuen False-Positives. Einige dieser False-Positives haben hohe Confidence-Werte (0.85), was die Auswahl eines optimalen Schwellwerts erschwert. Bei geringer Auflösung erkennt MTCNN weiterhin zuverlässig alle Gesichter in Einzel-, Zwei- und Drei-Personen-Porträts mit einer Confidence von 1.0 und ohne False-Positives. Im Gruppenbild wird nichts markiert. Bei mittlerer Auflösung bleibt das Verhalten stabil – im Gruppenbild werden vier Gesichter mit ca. 0.85 Confidence erkannt.

- **Yunet:** Das Yunet-Modell (`face_detection_yunet_2023mar.onnx`) ist ohne Konfiguration sehr strikt. Es erkennt Gesichter mit hoher Präzision, ist aber auch sehr selektiv. Nach Einstellung des Confidence-Grenzwertes auf 0.8 werden



Abbildung 3.5: Stichprobenergebnis des Dlib-HOG-Landmark-Modells bei einem Zwei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. e)

alle Gesichter in Einzel-, Zwei- und Drei-Personen-Porträts erkannt (siehe dazu Abbildung 3.8), jedoch nur wenige im Gruppenbild. Ob 0.8 optimal ist, wird sich in weiteren Tests zeigen. Bei höherer Auflösung verbessert sich die Confidence der erkannten Gesichter leicht. Im Gruppenbild verändert sich die Erkennung – ein vorher erkanntes Gesicht fehlt, ein anderes wird zusätzlich erkannt. Es gibt weiterhin keine False-Positives. Bei geringer Auflösung erkennt Yunet alle Gesichter bis auf die im Gruppenbild, wo kein Gesicht markiert wird. Auch bei mittlerer Auflösung bleiben die Ergebnisse identisch zu jenen bei niedriger Auflösung.

- **RetinaFace (CPU):** Verwendet wird RetinaFace mit ResNet-50-Backbone und CPU-Nutzung, da GPU lediglich die Geschwindigkeit, nicht aber die Genauigkeit erhöht. RetinaFace erzielt in den Stichprobentests die besten Ergebnisse aller



Abbildung 3.6: Stichprobenergebnis des Dlib-CNN-Modells bei einem Zwei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. e)

Modelle: Alle Gesichter in Einzel-, Zwei- und Drei-Personen-Porträts werden ohne False-Positives erkannt (siehe dazu Abbildung 3.9). Im Gruppenbild werden 10 von 13 Gesichtern identifiziert. Es treten zwei False-Positives auf, bei denen Pferdegesichter markiert wurden. Bei höherer Auflösung bleiben die Ergebnisse stabil, die Confidence-Werte ändern sich nur minimal. Im Gruppenbild wurde ein schwach erkanntes Gesicht nicht erneut erkannt. Bei geringer Auflösung erkennt RetinaFace weiterhin alle Gesichter in Einzel-, Zwei- und Drei-Personen-Porträts korrekt. Die Anzahl erkannter Gesichter im Gruppenbild sinkt auf 8 von 13, bleibt damit jedoch konkurrenzlos hoch. Auch bei mittlerer Auflösung bleiben die Erkennungsergebnisse stabil und präzise.

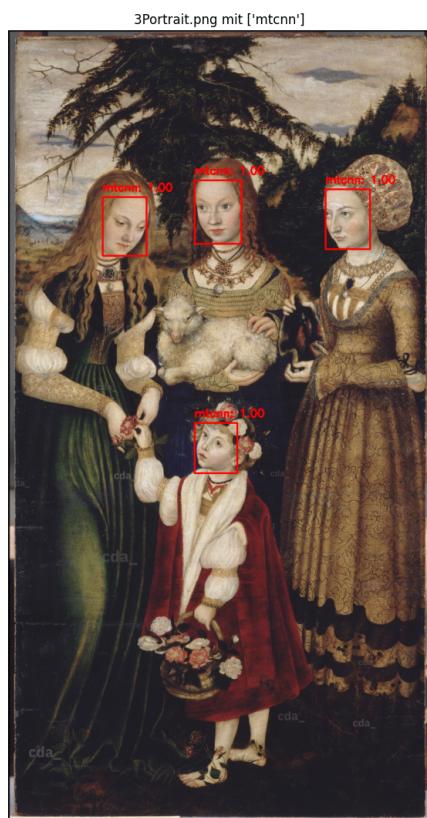


Abbildung 3.7: Stichprobenergebnis des MTCNN-Modells bei einem Drei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. d)

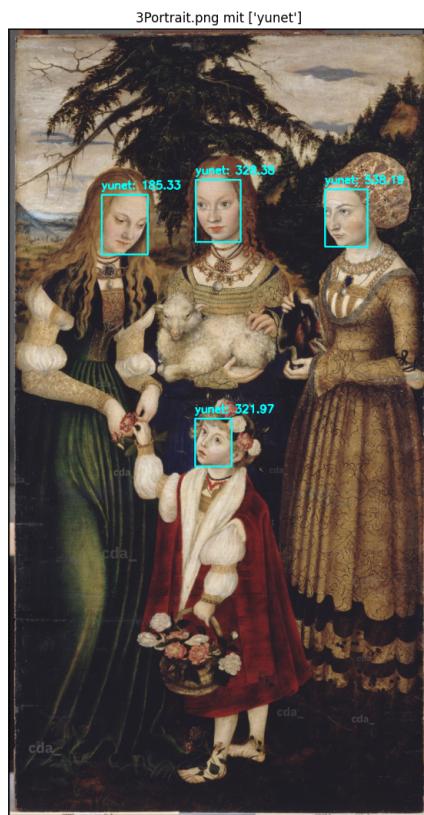


Abbildung 3.8: Stichprobenergebnis des Yunet-Modells bei einem Drei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. d)

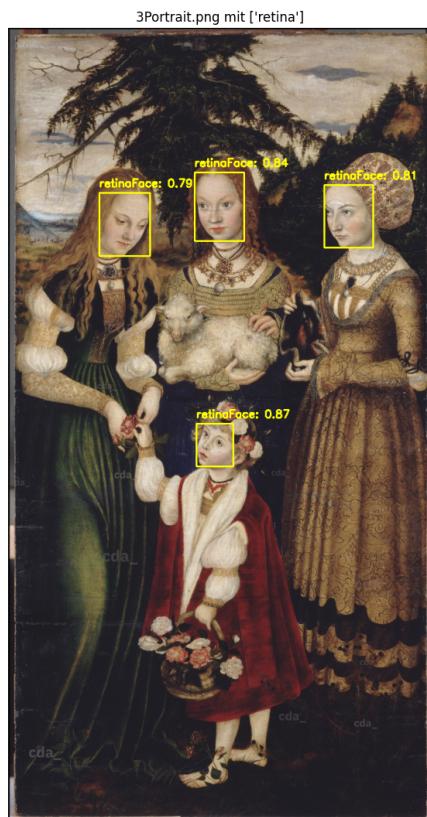


Abbildung 3.9: Stichprobenergebnis des RetinaFace-Modells bei einem Drei-Personen-Porträt (Gemälde aus dem CDA, Cranach Digital Archive, o.D. d)

3.2 Größentests

Das CDA speichert Bilder von Werken in drei Größen: Small, Medium und Large. Da jedes Werk sein eigenes Format hat, haben nicht alle Bilder die exakt gleiche Auflösung. Small-Bilder haben 400 px Breite, Medium-Bilder haben 600 px Breite, Large-Bilder haben 1200 px Breite. Während der Stichprobentests wurde bereits ein wenig mit Bildern verschiedener Auflösungen experimentiert, die Ergebnisse waren jedoch nicht sehr übersichtlich oder eindeutig, sodass nochmal alle verbliebenen Modelle mit allen drei Größen getestet werden.

Portrait	Caffe	Dlib HOG	Dlib CNN	MTCNN	Yumet	RetinaFace	Gesamt
1er S	A 0	0 1	A 0	A 0	A 0	A 0	-1 1
2er S	A 0	A 0	0 0	A 0	A 0	A 0	-2 0
3er S	3 1	A 0	0 0	A 0	A 0	A 0	-4 1
1er M	A 0	A 1	A 0	A 0	A 0	A 0	A 1
2er M	A 0	A 2	A 0	A 0	A 0	A 0	A 2
3er M	3 0	A 1	3 0	A 0	A 0	A 0	-2 1
1er L	A 0	A 1	A 0	A 0	A 0	A 0	A 1
2er L	A 0	A 7	A 0	A 0	A 0	A 0	A 7
3er L	A 0	A 5	A 0	A 2	A 0	A 0	A 7

Tabelle 3.1: Ergebnisse des Größentests (Erkannte Gesichter [A = Alle erkannt] | False-Positives)

Als Ergebnis wird festgehalten, dass Large-Bilder sich am besten eignen. Diese haben den Nachteil, dass sie die meisten False-Positives generieren. Dennoch werden auf ihnen die meisten Gesichter erkannt (siehe dazu Tabelle 3.1), was in diesem Anwendungskontext das relevantere Kriterium ist. Des Weiteren wird davon ausgegangen, dass einige dieser False-Positives darauf zurückzuführen sind, dass die Confidence-Grenzwerte noch nicht optimiert wurden. Dlib HOG wird verworfen, da es während der Tests im Vergleich deutlich mehr False-Positives gezeigt hat und es bereits genug Modelle mit akzeptablen Werten gibt, sodass das Modell als Kandidat nicht mehr relevant ist.

3.3 Confidencetests

Zur bestmöglichen Repräsentation der Modelle werden alle Confidence-Grenzwerte bzw. Score-Threshholds der Modelle angepasst. Zunächst wird kein Grenzwert gesetzt und die Ausgaben beobachtet. Dabei wird explizit auf die Werte des kleinsten Confidence-Wertes für ein erkanntes Gesicht und des höchsten Confidence-Wertes für ein False-

Positiv geachtet. Sollten sich diese überschneiden, so wird mithilfe des Kontextes der anderen Ergebnisse bestimmt, welcher Confidence-Grenzwert sich für das jeweilige Modell am besten zur Gesichtserkennung auf historischen Gemälden eignet. Hier sei noch einmal angemerkt, dass jedes Modell seine eigene Implementierung für Confidence-Werte hat. Ein höherer Wert bedeutet jedoch nicht zwangsläufig, dass ein Modell besser ist – er gibt lediglich an, wie sicher sich das Modell ist, ein Gesicht erkannt zu haben.

Portrait	Caffe	Dlib CNN	MTCNN	Yunet	RetinaFace
1er	0.11 0.21	1.03 0	0.99 0	389.12 0	0.74 0
2er	0.14 (0.13) 0.40	0.70 0	0.82 0	233.79 0	0.71 0
3er	0.10 0.28	0.97 0	0.91 (0.83) 0.95	154.46 0	0.68 (0.52) 0
Maximal	0.10 0.4	0.70 0	0.82 0.95	154.46 0	0.68 (0.52) 0

Tabelle 3.2: Ergebnisse der Confidencetests (Minimal Gesichtserkennung [inkl. Nebengesichter] | Maximal False-Positiv)

- **Caffe:** Die Ergebnisse von Caffe waren bei einem Confidence-Grenzwert von -1 viel zu unkenntlich, also wurde der Test wiederholt. Im zweiten Anlauf wurden alle Markierungen mit einem Confidence-Wert $\geq 0,1$ akzeptiert, da alle Ergebnisse unter diesem Wert aufgrund der riesigen Mengen an False-Positives unbrauchbar wären. Der minimale Confidence-Wert für erkannte Gesichter beim Einzelporträt ist 0,11, während der maximale False-Positive-Wert 0,21 ist. Dieser Wert wurde bei einem der Testbilder gemessen (siehe dazu Abbildung 3.10), alle anderen erreichten einen Wert von 1 oder ca. 0,8. 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 vermutlich ähnliche Bilder gibt. Ergebnisse des Tests mit Zwei-Personen-Porträts waren deutlich weniger konstant als die des letzten Tests. Der geringste Wert für ein erkanntes Gesicht war 0,14 bzw. 0,13 wenn Nebengesichter mitgezählt werden. Allerdings gab es mit 0,4 ein großes False-Positive, sowohl vom Wert als auch von der Fläche her. Der Test mit Drei-Personen-Porträts verlief sehr schlecht für Caffe. Der geringste erkannte Wert ist 0,1. Es ist aber auch sehr wahrscheinlich, dass der eigentliche Wert noch geringer ist, da in manchen Bildern nicht alle Gesichter erkannt wurden. Auch gab es eine sehr große Menge an False-Positives mit einem höchst Wert von 0,28. Mit minimaler Gesichtserkennung bei Confidence von 0,1 und maximalem False-Positive-Confidence von 0,4 ist das Bestimmen eines optimalen Grenzwertes sehr schwierig. Nach mehreren weiteren Tests wird 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-Positives, was

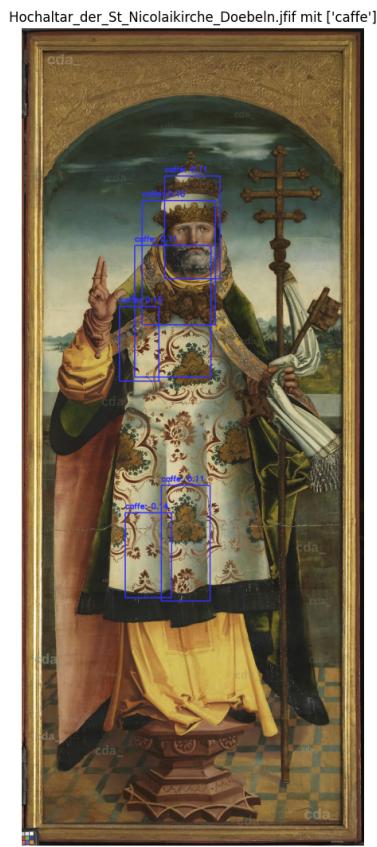


Abbildung 3.10: Confidenceergebnis des Caffe-Modells bei dem ein erkanntes Gesicht einen geringeren Wert als ein False-Positives hat (Gemälde aus dem CDA, Cranach Digital Archive, o.D. b)

die Gesamtergebnisse brauchbarer macht.

- **Dlib CNN:** Dlib CNNs Tests mit Einzelporträts haben selbst bei einem Grenzwert von -1 keine False-Positives gezeigt. Der minimale Wert für erkannte Gesichter liegt bei 1,03. Im Test mit Zwei-Personen-Porträts gab es weiterhin keine False-Positives, allerdings wurden auch einige Neben- sowie Hauptgesichter nicht erkannt. 0,7 ist der kleinste gemessene Wert für ein Gesicht. Weiter werden im Test mit Drei-Personen-Porträts keine False-Positives erkannt, aber auch teilweise 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 wurden und Dlib CNN sehr streng scheint, ist der verwendete Grenzwert für Dlib CNN 0,6.

- **MTCNN:** MTCNN hat keine False-Positives und einen minimalen Confidence-Wert von 0,99 bei der Gesichtserkennung. Generell sind alle Werte konstant zwischen 0,99 und 1. Im Test mit Zwei-Personen-Porträts wurden weiterhin keine False-Positives erkannt, allerdings ebenso einige Gesichter nicht. Der geringste gemessene Wert bei der Gesichtserkennung beträgt 0,82. Die Ergebnisse des Tests mit Drei-Personen-Porträts erschweren das Bestimmen eines endgültigen Grenzwertes für MTCNN. Der höchste Wert für False-Positives beträgt 0,95, generell haben alle False-Positives von MTCNN sehr hohe Confidence-Werte (siehe dazu Abbildung 3.11). 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 wird, ist, dass die markierten False-Positives Bereiche kleine sind und ignoriert werden könnten. Der Grenzwert wird vorläufig auf 0,82 gesetzt. Abhängig von den Ergebnissen wird das Modell mit 0,91 Grenzwert getestet und entschieden, welcher Wert verwendet wird.

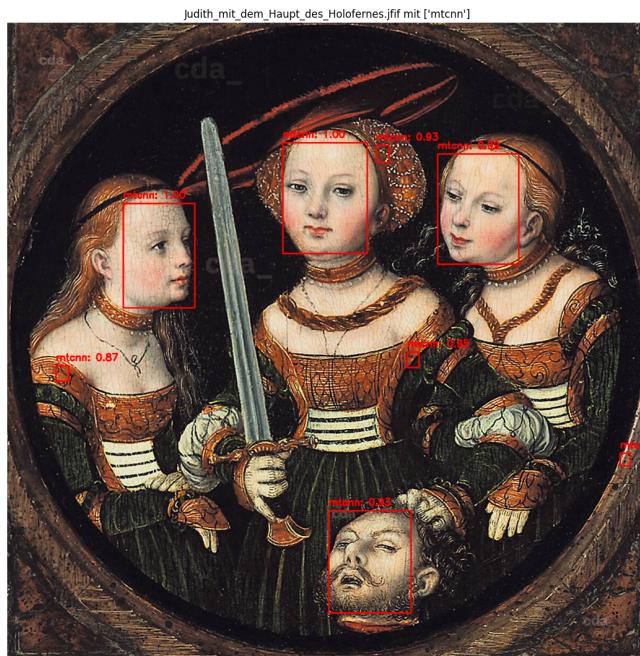


Abbildung 3.11: Confidenceergebnis des MTCNN-Modells mit False-Positives (Gemälde aus dem CDA, Cranach Digital Archive, o.D. c)

- **Yunet:** Yunet hat bei seinem Einzelporträt-Test keine False-Positives gehabt und ein Gesicht übersehen, ansonsten betrug der minimale Wert für ein erkanntes

Gesicht 389,12. Der Test mit Zwei-Personen-Porträts verlief mit einem ähnlichen Ergebnis und einem minimalen Wert von 233,79. Der letzte Confidence-Test von Yunet mit Drei-Personen-Porträts ergab weiterhin keine False-Positives und auch den gesamt minimalen 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.

- **RetinaFace (CPU):** RetinaFace hat als minimalen Confidence-Wert 0,74 für Gesichter im Test mit Einzelporträts, ohne Fehler. Die Ergebnisse für Zwei-Personen-Porträts zeigen einen minimalen Wert von 0,71 und keine False-Positives, es wurden aber auch manche Gesichter nicht erkannt. Das Ergebnis des Tests mit Drei-Personen-Porträts entspricht dem gesamt minimalen 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 gemessene Wert 0,52 als Grenzwert verwendet.

3.4 Vergleich

Keines der Modelle hat es geschafft, jedes Gesicht aus den bisherigen Tests fehlerfrei zu erkennen. Somit werden im letzten Test alle fünf verbliebenen Modelle gleichzeitig getestet. Dabei werden für jedes Modell die Anzahl von False-Negatives, False-Positives und die Anzahl von Gesichtern, die nur von diesem Modell erkannt wurden, gezählt (siehe dazu Tabelle 3.3). Ziel ist es, ein Modell oder eine kleine Auswahl von Modellen herauszuarbeiten, das genutzt werden kann, um so viele Gesichter wie möglich auf historischen Gemälden zu erkennen.

Portrait	Caffe	Dlib CNN	MTCNN	Yunet	RetinaFace
1er	-1 4 0	0 0 0	0 0 0	-1 0 0	0 0 0
2er	-1 2 0	-2 0 0	-2 0 0	-2 0 0	-2 0 0
3er	-11 3 0	-10 0 1	-4 6 7	-4 0 0	-2 0 2
Insgesammt	-13 9 0	-12 0 1	-6 6 7	-7 0 0	-4 0 2

Tabelle 3.3: Ergebnisse der finalen Modelltests (False-Negatives [In negativen Zahlen dargestellt] | False-Positives | Gesichter die nur von diesem Modell erkannt wurden)

Die Entscheidung, welches oder welche Modelle sich zur Weiterentwicklung eignen, ist keine einfache, da es kein fehlerfreies Modell gibt, das ohne Weiteres ausgewählt werden könnte. Auch kommt es in manchen Bildern vor, dass mehrere Gesichter jeweils von unterschiedlichen Modellen erkannt werden (siehe dazu Abbildung 3.12).

Der ursprüngliche Ansatz war, je nach Bild den Confidence-Grenzwert und das Modell zu wählen. Hierbei bleibt jedoch weiterhin das Problem bestehen, dass kein einzelnes Modell alle Gesichter eines Bildes erkennt. Der neue Ansatz ist nun, je nach Bild mehrere Modelle zu- und abzuschalten, um auf diese Weise mehr Gesichter zu erkennen und False-Positives zu vermeiden. Unter Berücksichtigung des angestrebten Anwendungszwecks fällt die finale Wahl auf: RetinaFace (CPU), MTCNN und CNN. RetinaFace ist das Modell mit den wenigsten False-Negatives. Zusätzlich hat es auch keine False-Positives generiert, womit es als eine gute Basis dient. MTCNN wurde gewählt, da 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 Modellauswahl. CNN wird aus dem gleichen Grund wie MTCNN hinzugezogen. Es hat zwar keine False-Positives, die zu Problemen werden könnten, sollte jedoch nur bei Bedarf eingesetzt werden, da es die Rechenzeit deutlich erhöht. Weiter enthalten die Ergebnisse des Modells viele False-Negatives.

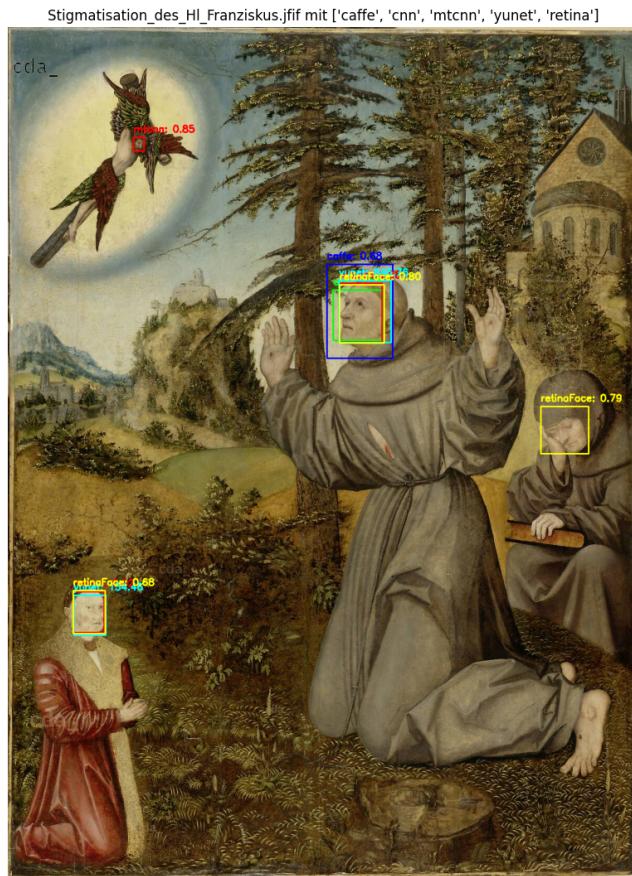


Abbildung 3.12: MTCNN und RetinaFace haben jeweils Gesichter, die nur von ihnen markiert wurden (Gemälde aus dem CDA, Cranach Digital Archive, o.D. f)

4 Implementierung

Ursprünglich war der Ansatz, ein eigenständiges Programm zu entwickeln, welches selbstständig Wasserzeichen auf Bildern platziert. Dabei sollten die Gesichter ausgespart werden, um die Ästhetik des Bildes nicht zu stören. Aufgrund der limitierten Zeit wird dieser Ansatz auf das Markieren von Gesichtsbereichen reduziert. Diese Reduktion erlaubt eine flexiblere Implementierung, indem statt eines eigenständigen Programms ein Modul erstellt wird. Damit ergeben sich mehr Anwendungsmöglichkeiten, die über das einfache Platzieren von Wasserzeichen hinaus gehen.

Bisher wurde 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, dass diese Funktion Probleme mit Umlauten hat. Aus diesem Grund wurde auf die Funktion `Image.open().convert("RGB")` von PIL gewechselt. Bemerkenswert ist, dass RetinaFace mit RGB-Input weniger False-Negatives generiert. Dieses Ergebnis ist insofern unerwartet, da BGR als Input-Format angegeben wurde (lounging-lizard & nttstar, 2025). Die Abweichung der Confidence-Werte war mit etwa 0,02 gering, weshalb RGB nun als Eingabeformat 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 zurückgibt. Das Modul liefert Funktionen mit, um zu überprüfen, 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 manuell über eine grafische Benutzeroberfläche auswählen, dessen Inhalte dann iteriert werden.
- File: Es kann ein Pfad zu einer einzelnen Bilddatei hinterlegt werden, welche bearbeitet werden soll.
- Path: Wird stattdessen der Pfad eines Ordners übergeben, so werden alle Bilder dieses Ordners iteriert.

- List: Falls die interne Funktion für den jeweiligen Anwendungskontext nicht ausreicht oder bereits eine Liste mit Bildpfaden vorliegt, kann diese direkt übergeben werden.

Weiter können auch alle Modelle einzeln zu- oder abgeschaltet werden – sowohl beim Funktionsaufruf als auch während der Laufzeit (siehe dazu Abbildung 4.1). RetinaFace ist dabei standardmäßig aktiviert und alle anderen Modelle deaktiviert. Da RetinaFace bereits gute Ergebnisse liefert, reicht es in den meisten Fällen aus. Dies spart Rechenzeit und verringert die Anzahl von False-Positives.

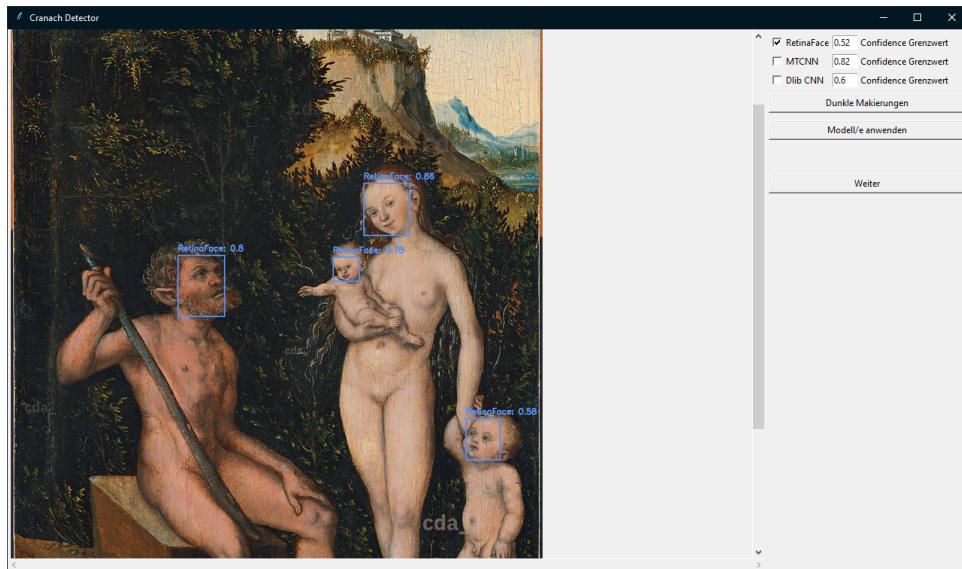


Abbildung 4.1: GUI von `cranach_detector()` (Eigenes Bild)

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 festzulegen, sofern dies erforderlich ist.

Die von den Modellen erkannten Bereiche werden jeweils 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 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. Diese kann dann beispielsweise in eige-

nen Python-Funktionen weiterverarbeitet werden oder für externe Anwendungen in das JSON-Format überführt werden. Das Format, in dem RetinaFace seine Ergebnisse ausgibt, unterscheidet sich leicht von MTCNN und Dlib CNN. RetinaFace gibt als Ergebnis die Eckpunkte eines Rechtecks zurück, während die beiden anderen Modelle einen Startpunkt inklusive Breite und Höhe angeben. Um die Arbeit mit den gemeinsamen Ergebnissen der Modelle zu erleichtern, werden die Ergebnisse von RetinaFace in das Format von MTCNN und Dlib CNN überführt, auch wenn das Format von RetinaFace leichter weiterzuverarbeiten ist. Der Grund, warum Funktionen in diesem Modul dennoch Eingaben im Format $x, y, \text{Breite}, \text{Höhe}$ erwarten, liegt darin, dass im Frontend eher mit Werten in diesem Format gearbeitet wird.

Die Farben der markierten Gesichter wurden für das Package angepasst, um auch für Menschen mit Farbsehschwäche unterscheidbar zu bleiben. 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 bestimmten Bildtypen eine reduzierte Erkennbarkeit aufweisen. Aus diesem Grund wurde die Option hinzugefügt, Markierungen in dunkleren Farben anzeigen zu lassen.

Sollten mehrere Modelle dasselbe Gesicht markieren, wird nur der Bereich mit der geringsten Fläche als gültige Gesichtserkennung beibehalten. Dafür wird überprüft, ob sich zwei markierte Bereiche überschneiden, und die Flächen miteinander verglichen. Wenn der überschneidende Bereich 75% oder mehr der Fläche des kleineren Bereichs ausmacht, wird der größere Bereich verworfen. So werden möglichst genaue Ergebnisse behalten und vermieden, dass Markierungen von Gesichtern, die nah beieinander liegen, verworfen werden.

Um zu überprüfen, ob 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 $(x, y, width, height)$ 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 ein Margin angegeben werden, falls ein gewisser Abstand zu Gesichtern eingehalten werden soll. Befindet sich ein Gesicht innerhalb des Margin um den Bereich oder Punkt, gibt die Funktion `True` zurück.

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 Einzel-, Zwei- und Drei-Personen-Porträts zu finden. Zusammen mit MTCNN und Dlib CNN konnte ein Python-Modul entwickelt werden, welches in der Lage ist, alle Gesichter auf den getesteten Porträts zu markieren. Dieses bietet die Möglichkeit, Modelle je nach Bild an- oder abzuschalten sowie ihre Confidence-Grenzwerte anzupassen. Damit ist das Modul in der Lage, flexibel auf verschiedene Werke angewendet zu werden, um bestmögliche Ergebnisse zu erzielen. Des Weiteren konnten Funktionen implementiert werden, die prüfen, ob sich ein Overlay auf einem Gesicht befindet. Alternativ kann die ausgegebene Liste von Gesichtsbereichen verwendet werden, um eigene Funktionen zu implementieren. Sollen die Werte außerhalb von Python verwendet werden, kann die Liste zu einer JSON-Datei formatiert werden, um sie in anderen Anwendungen zu nutzen.

Als mögliche Weiterführung der Arbeit kann die GUI des Moduls überarbeitet werden, um die Nutzerfreundlichkeit zu optimieren. Ein mögliches Feature, das hinzugefügt werden könnte, wäre ein Zähler, welcher zeigt, wie viele Bereiche vom jeweiligen Modell markiert wurden. So kann schneller von Nutzer*innen erkannt werden, ob sich auf einem Bild False-Positives befinden, wenn die Zahl der Bereiche die tatsächliche Anzahl der Gesichter auf dem Bild übersteigt. Die dafür notwendigen Funktionen sind bereits implementiert, jedoch wurde aus Zeitmangel noch keine grafische Darstellung dafür umgesetzt. 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 Fortführung der Arbeit wäre ein intensiveres Testen mit Fokus auf Gruppenbilder. Die Tests an Gruppenbildern haben bereits gezeigt, dass es deutlich schwieriger ist, auf ihnen zuverlässig Gesichter zu finden. Aufgrund des beschränkten zeitlichen Rahmens wurden spätere Tests an Gruppenbildern verworfen, um den Fokus auf Porträts setzen zu können. Dies geschah mit der Überlegung, dass das Platzieren eines Overlays auf dem Gesicht eines Porträts die Ästhetik des Bildes stärker beeinflusst. Würde man die Forschung mit Gruppenbildern fortsetzen wollen, so müssten weitere Modelle, die nicht in dieser Versuchsreihe getestet wurden,

herangezogen werden. Alternativ könnte in Erwägung gezogen werden, ein eigenes Modell für diesen Anwendungszweck zu trainieren oder zu entwickeln.

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>
- Cranach Digital Archive. (o.D. a). *Bildnis des Johannes Cuspinian* [Malerei auf Fichtenholz]. Verfügbar 11. Juni 2025 unter https://lucascranach.org/de/CH_SORW_1925-1b/
- Cranach Digital Archive. (o.D. b). *Hochaltar der St. Nicolaikirche Döbeln [innerer beweglicher Flügel, links]: Segnender Papst* [Malerei auf Holz]. Verfügbar 11. Juni 2025 unter https://lucascranach.org/de/DE_ND_ND001b/
- Cranach Digital Archive. (o.D. c). *Judith mit dem Haupt des Holofernes* [Malerei auf Holz]. Verfügbar 11. Juni 2025 unter https://lucascranach.org/de/DE_SRU_GR1-691/
- Cranach Digital Archive. (o.D. d). *Katharinentalar: Hl. Dorothea, Hl. Agnes, Hl. Kunigunde [linker Seitenflügel, Innenseite]* [Malerei auf Lindenholz]. Verfügbar 11. Juni 2025 unter https://lucascranach.org/de/DE_SKD_GG1906BB/
- Cranach Digital Archive. (o.D. e). *Katharinentalar: Hl. Genoveva und Hl. Apollonia [linker Seitenflügel, Außenseite]* [Malerei auf Lindenholz]. Verfügbar 11. Juni 2025 unter https://lucascranach.org/de/UK_NGL_6511-1/
- Cranach Digital Archive. (o.D. f). *Stigmatisation des Hl. Franziskus* [Malerei auf Lindenholz (Tilia sp)]. Verfügbar 11. Juni 2025 unter https://lucascranach.org/de/DE_GNMN_Gm1352/
- 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
- dlib. (o.D. b). *Dlib C++ Library* [Dlib Python API Documentation]. Verfügbar 2. Mai 2025 unter 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
- 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
- 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
- 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/
- Google AI Edge. (2024). *MediaPipe Framework* [Developer Documentation für MediaPipe]. Verfügbar 26. April 2025 unter 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
- 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
- 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>
- O’Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *CoRR, abs/1511.08458*. <http://arxiv.org/abs/1511.08458>
- Pagelsdorf, B. (2024). *Analyse und Konzeption von Maßnahmen zum Schutz von Bildern im Web am Beispiel der offenen Forschungsdatenbank Cranach Digital Archive mit über 20000 Abbildungen* [Masterarbeit]. Technische Hochschule Köln [Unveröffentlicht].
- 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>

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.

Siegburg 12.06.2025
Ort, Datum

N. Möller
Unterschrift