

Braces
die Zahnspangen-Simulation

Projektarbeit im Fach Informatik

vorgelegt
von

Nellie Reichmann

Angefertigt am

Lehrstuhl für Mustererkennung (Informatik 5)
Department Informatik
Friedrich-Alexander-Universität Erlangen-Nürnberg.

Betreuer: Vincent Christlein

Abgabe der Arbeit: 23.04.2024

Inhaltsverzeichnis

1	Einleitung	2
2	Segmentation Task	2
2.1	Image Segmentation	2
2.2	U-Net Architektur	2
2.3	Datensatz	3
2.4	Transformations	4
2.5	Training	4
2.6	Auswertung	5
3	Platzierung der Zahnpfange	5
3.1	Post-Processing	5
3.2	Segmentierung der individuellen Zähne	6
3.3	Positionierung der Brackets und des Drahtes	7
3.4	Auswertung	7
4	Anwendung	7
4.1	Das Hauptsystem	7
4.2	API-Endpunkte	7
4.3	Auswertung	8
5	Fazit	8
6	Future Work	8
A	Anhang	10
A.1	Transformations	10
A.2	Experimente der Hyperparameter	11
A.3	Beispielbilder	14
	Literaturverzeichnis	16

1 Einleitung

Die Entscheidung für eine Behandlung von Kieferfehlstellungen bringt für viele Patientinnen Unsicherheiten mit sich, insbesondere hinsichtlich der ästhetischen Veränderungen beim Sprechen, Lachen und Zähnezeigen. Da die Behandlungsoptionen variieren, sowohl in ihrer Invasivität als auch Sichtbarkeit, steht oft die Frage im Raum, wie man mit einer Zahnpange aussehen wird. Derzeitige Apps zur Visualisierung bieten hier keine zufriedenstellenden Lösungen. Sie ermöglichen meist nur das Aufsetzen eines generischen Zahnpangen-Stickers auf das Bild der Nutzerin ohne individuelle Anpassungen.

In diesem Projekt wurde die App „Braces“ entwickelt, die mittels eines Deep Learning Modells namens U-Net die Zähne in einem Bild erkennt und eine individuell angepasste Zahnpange platziert. So bietet die Anwendung eine Vorstellung vom Aussehen mit Zahnpange und adressiert die ästhetischen Sorgen der Patientinnen. Diese Arbeit beschreibt die Entwicklung des U-Nets, die Gestaltung der Algorithmen-Pipeline zur Zahnsegmentierung und die Designentscheidungen zur Realisierung der Anwendung.

2 Segmentation Task

2.1 Image Segmentation

Dieses Projekt fokussiert sich auf die binäre Segmentierung von Zähnen. Aus komplexen visuellen Bilddaten werden Segmentation-Masks erstellt, die jedem Pixel eine Wahrscheinlichkeit zuweisen, ob er zur Klasse „Zähne“ gehört. Unter den verfügbaren Methoden sind Deep Learning (DL)-Modelle, speziell für anspruchsvolle Bild-Muster, die effektivsten. Eine herausragende Architektur, die sich besonders für Image Segmentation eignet, ist das U-Net.

2.2 U-Net Architektur

Das U-Net, entwickelt von Olaf Ronneberger und Kollegen [Ron⁺15], präsentiert sich als Convolutional Neural Network (CNN) mit einer Encoder-Decoder-Struktur (dt: Kontraktions- und Expansionspfad), die durch Skip-Connections verbunden werden. Es entsteht eine U-förmige Architektur, die dem Netzwerk seinen Namen verleiht [Bild 1].

Der Kontraktionspfad besteht aus mehreren Blöcken, die ähnlich eines klassischen CNN aufgebaut sind: Convolutions, nicht-lineare ReLU-Activations und Max-Pooling Layers. Diese Struktur erfasst Kontextinformationen und Feature-Maps auf verschiedenen Ab-

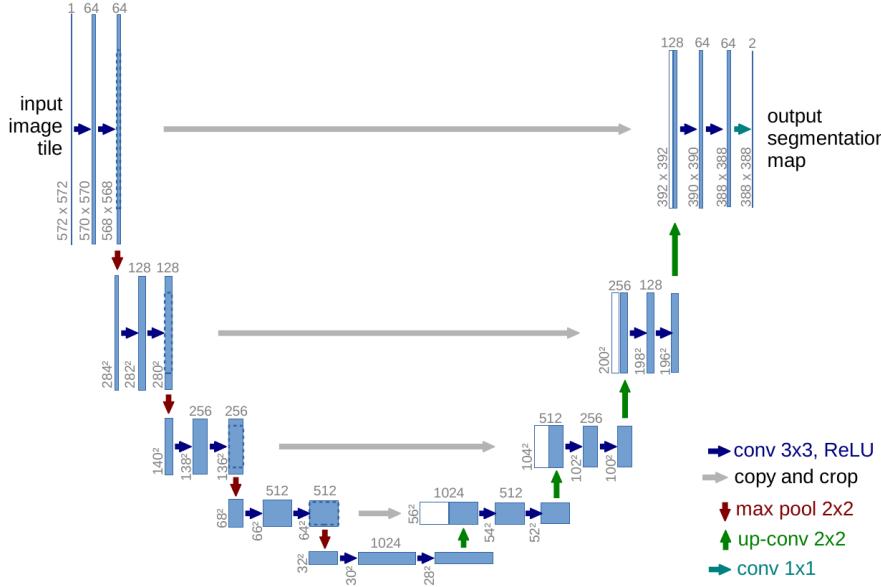


Bild 1: U-Net Architektur: Encoder (links) und Decoder (rechts), verbunden durch Skip-Connections, aus [Ron⁺15].

straktionsebenen und reduziert die räumliche Information. Nach dem Scheitelpunkt des U-Nets kehrt sich die Funktionsweise um: Convolutionen werden durch Up-Convolutionen und Pooling-Layers durch Upsampling-Layers ersetzt. Skip-Connections von den Blöcken des Kontraktionspfades übertragen verlorene räumliche Informationen zurück zum Decoder, was zusammen mit den hochauflösten Feature-Maps eine präzise Lokalisation der Features ermöglicht.

Die Implementierung erfolgte in PyTorch und Torchvision und orientierte sich eng am ursprünglichen U-Net Aufbau, wobei ein Padding zur Vermeidung von Randpixel-Verlust und Batch Normalizations vor den ReLU-Layern für stabilere Trainingsbedingungen hinzugefügt wurden.

2.3 Datensatz

Für das Training des U-Nets wurde ein Datensatz aus Selfie-Bildern menschlicher Gesichter zusammengestellt, basierend auf dem unter der CC BY-SA 4.0 Lizenz veröffentlichten EasyPortrait Datensatzes von Alexander Kapitanov et al [Kap⁺23]. Dieser umfasst 20.000 hochauflösende, belabelte Bilder verschiedener Individuen, inklusive der relevanten Klasse

„Zähne“. Die Nutzung bereits vorhandener Labels machte ein manuelles und zeitaufwändiges Labeln unnötig. Zum Zusammenstellen der passenden Trainingsdaten lag der Fokus auf:

- der Auswahl von Bildern mit dem Label „Zähne“, da diese pixelmäßig ohnehin unterrepräsentiert sind,
- der Reduktion von Bildern mit herausgestreckter Zunge,
- der Erhöhung der Anzahl von Bildern, auf denen Personen deutlich grinsen.

Der resultierende Datensatz besteht aus 200 Trainingsbildern und 50 Validierungsbildern. Die Daten wurden im PyTorch-typischen channel-first Format und normalisiert auf Basis der Durchschnitts-Channel-Werte und Standardabweichungen des gesamten EasyPortrait Datensatzes verwendet.

2.4 Transformations

Um Overfitting entgegenzuwirken, wurden Transformations aus der albumentations library eingesetzt. Diese eignet sich besonders gut für Segmentation Tasks, da Transformations gleichzeitig auf Bild und Mask angewendet werden und so die Konsistenz zwischen transformiertem Bild und transformierter Mask auch bei randomisierten Transformations sicherstellt ist. Augmentation-Techniken wie Cutmix, Cutout, Mixup wurden nicht verwendet, da der kleine Zahnbereich dadurch möglicherweise aus den Bildern ausgeschnitten worden wäre. Eine Auflistung und Beispielbilder der verwendeten Transformations sind im Anhang A.1 zu finden.

2.5 Training

Trotz der Verfügbarkeit von mehreren Labels im Datensatz wurde das U-Net ausschließlich auf die Zahn-Labels trainiert. Dabei kam die Binary Cross Entropy Loss (`BCEWithLogitsLoss` von PyTorch) zum Einsatz, eine gängige Loss-Funktion für unbalancierte binäre Segmentation Tasks wie dieser. Die Intersect over Union (IoU) Metrik mit einem sensiblen Threshold von 0,1 diente zur besseren Nachvollziehbarkeit des Trainingsverlaufs, unterstützt durch umfangreiches Logging mit dem Tool wandb¹.

Als Optimizer wurde AdamW [Los+19] mit einem Weight Decay von 0.001 verwendet, da dieser im Vergleich zu seinem Vorgänger Adam das Training bei gleicher Learning Rate beschleunigte. Ein Cosine Scheduler mit einem Warmup von 20% brachte die Learning Rate

¹<https://wandb.ai>

auf 0,01, bevor sie langsam wieder verringert wurde. In Experimenten erwiesen sich dieser Hyperparameterkombinationen als besonders effektiv (vlg. Grafiken 4, 5 und 6 im Anhang A.2).

In den endgültigen Versuchsdurchläufen wurde das Model über 120 Epochen trainiert, wobei in den letzten 20 Epochen eine Art Early-Stopping angewendet wurde: Der Model-Zustand wurde nur bei Verbesserung der Metriken abgespeichert. Dies ermöglicht das am besten performende Model auszuwählen.

2.6 Auswertung

Das U-Net zeigt sich schon mit den Standardparametern als sehr leistungsfähig. Die Validation-Loss sank schnell unter 0,1 und die Test-Predictions sind im Zielbereich der „Zähne“ sehr präzise. Selten treten kleine Artefakte auf, die fälschlicherweise als Zähne identifiziert werden. Die Anpassung der Hyperparameter führte zu einem optimalen Trainingsergebnis. Aus den endgültigen Versuchsdurchläufen konnte ein Model mit Validation-Loss von 0,003 und IoU von 0,56 ausgewählt werden. Eine Reduzierung der Trainingsdauer auf unter 100 Epochen war jedoch nicht möglich.

3 Platzierung der Zahnpangage

3.1 Post-Processing

Nach dem ersten Segmentierungsprozess des U-Nets wird ein Threshold angewandt, der nur Wahrscheinlichkeiten über einem bestimmten Wert als signifikant erachtet. Die resultierende binäre Segmentation-Map wird mittels `cv2.connectedComponentsWithStats` der open-cv library in einzelnen Komponenten zerlegt. Für jede Komponente werden eine Bounding Box, die Fläche, der Centroid und die Mask berechnet.

Für die weitere Verarbeitung erfolgt eine Reduktion der Komponenten der Prediction auf den primären Zahnbereich. Kleine Bereiche unter 1000 Pixeln werden eliminiert, um Artefakte zu entfernen. Bei weiterhin bestehenden multiplen Komponenten wird ein weiterer Threshold von 0.2 angewandt und die Reduktion wiederholt. Sollten immer noch mehrere Komponenten vorhanden sein, wird die flächenmäßig größte ausgewählt.

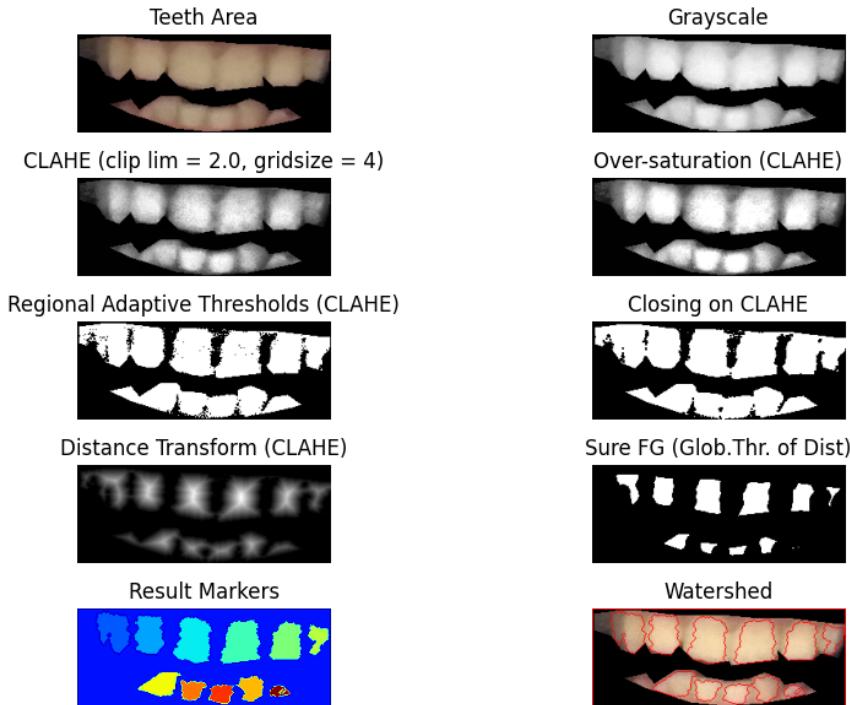


Bild 2: Beispiel der Algorithmen-Pipeline zum Segmentieren der individuellen Zähne.

3.2 Segmentierung der individuellen Zähne

Für die detaillierte Positionierung der Zahnpange ist eine Segmentierung in individuelle Zähne notwendig. Der Zahnbereich wird zuerst ausgeschnitten, in Graustufen umgewandelt und mittels CLAHE kontrastverstärkt. CLAHE liefert im Gegensatz zu Histogram Equalization einen hervorragenden Kontrast ohne einzelne Bereiche des Bildes zu überbelichten. Eine leichte Erhöhung der Helligkeit und ein Blur bereiten das Bild zum adaptiven Thresholding vor, welches eine binäre Einteilung des Bildes in Zähne (weiß) und Zwischen-Bereiche (schwarz) erzeugt. Eine abschließende Closing-Operation entfernt kleineren (schwarzen) Noise.

Der Watershed-Algorithmus wird anschließend genutzt, um einzelne Zähne zu identifizieren, beginnend mit dem Distance Transform zur Lokalisation potentieller Zahnmittelpunkte. Nach einem erneuten Thresholding und einer Erosion werden Startpunkte für den Watershed-Algorithmus festgelegt, der die individuellen Zähne definiert und voneinander abgrenzt.

3.3 Positionierung der Brackets und des Drahtes

Die Zahn-Mittelpunkte werden über die moments der Watershed-Markers berechnet und dort die Brackets platziert.

Um die Brackets mit einem Draht zu verbinden, wird eine Polynom-Regression mittels der sklearn library verwendet, um eine quadratische Funktion an die Bracket-Koordinaten anzupassen. Bei der Sichtbarkeit von oberen und unteren Zahnreihen wird KMeans-Clustering der sklearn library eingesetzt, um die Reihen zu separieren. Durch Skalierung der y-Koordinaten der Brackets um das Zehnfache wird die Clusterbildung entlang der Horizontalen verstärkt.

3.4 Auswertung

Verschiedene traditionelle Segmentierungs-Algorithmen wurden mit unterschiedlichen Parametern getestet, um optimale Voreinstellungen zu identifizieren. Diese sind im Anwendungsbeispiel in Bild 2 veranschaulicht. Die festgelegten Parameter sind nicht universell einsetzbar und geringfügige Abweichungen im Ursprungsbild können zu erheblichen Fehlern führen. Um dem entgegenzuwirken, erlaubt die Anwendung eine individuelle Anpassung der Parameter über das Nutzerinterface, wie in Abschnitt 4.2 beschrieben.

4 Anwendung

4.1 Das Hauptsystem

Zur Nutzbarmachung des Projekts wurde eine Software mit API-Endpunkten und Nutzerinterface entwickelt, die vier Hauptkomponenten umfasst:

- (1) Die TeethPredictor Klasse lädt das trainierte Model und erstellt Vorhersagen auf Bildern.
- (2) Die TeethSegmentor Klasse regelt die Segmentierung der individuellen Zähne basierend auf Standardparametern.
- (3) Die Braces Klasse konstruiert und platziert eine Zahnspannen-Maske auf dem Bild.
- (4) Die Orthodontist Klasse (dt: Kieferorthopäd:in) dient als Systemzugangspunkt.

4.2 API-Endpunkte

Die Server-Client-Kommunikation erfolgt über folgende API-Endpunkte:

- (1) Ein GET-Request liefert ein Interface zur Bildauswahl und -anzeige.

- (2) Ein „apply braces“-Button lädt das Bild hoch, worauf das Hauptsystem eine Vorhersage erstellt, die Zähne segmentiert und die Zahnpange positioniert. Das Ergebnis wird zurück zum Client gesendet.
- (3) Nutzerinnen können die Segmentierungsparameter anpassen, wobei Änderungen erst nach einer kurzen Pause verarbeitet werden (debounce), um den Server nicht zu überlasten.

4.3 Auswertung

Das Nutzerinterface ermöglicht eine einfache Interaktion mit dem System und die automatische sowie individuelle Anpassung der Zahnpangenpositionierung. Die Anwendung ist nur für die lokale Nutzung vorgesehen. Sie speichert die Daten direkt im Dateisystems des Servers und führt nur minimale Sicherheitsüberprüfungen durch. Bei einem Produktiveinsatz sind erweiterte Sicherheitsmaßnahmen unbedingt notwendig.

5 Fazit

Das Projekt demonstriert die Machbarkeit einer DL-gestützten Zahnpangen-Simulation. Ein U-Net wurde implementiert, trainiert und die Hyperparameter optimiert, um Zahnbereiche in Selfie-Bildern zuverlässig zu erkennen und zu lokalisieren. Weiterhin wurde eine Pipeline aus Segmentierungs-Algorithmen definiert und Standardparameter festgelegt, um die Zahnmittelpunkte zu identifizieren, Brackets zu platzieren und die Zahnpange mit Drähten zu vervollständigen. Das System lässt sich über eine grafische Nutzeroberfläche bedienen und feinjustieren.

Die Anwendung funktioniert generell zuverlässig, besonders bei hochauflösten Bildern, in denen der Zahnbereich deutlich erscheint und die einzelnen Zähne klar voneinander abgegrenzt sind. Allerdings ist oft eine Anpassung der Segmentierungs-Parameter nötig, um eine passende Zahnpange zu erzeugen. Probleme treten insbesondere bei Bildern von Personen mit Bart auf, wo helle Bart-Teile fälschlicherweise als Zähne interpretiert werden.

6 Future Work

Für eine generelle Verbesserung der Anwendung sind mehrere Entwicklungsrichtungen möglich:

- (1) Anpassungen des U-Nets auf Randfälle, insbesondere bei Personen mit Bart oder ungewöhnlichen Gesichtsausdrücken, welche die aktuelle Anwendung nicht abdeckt.
- (2) Optimierung der Segmentierungs-Algorithmen für schräge und seitliche Mundpositionen und Anpassung der Parameter an Bildgröße und Auflösung.
- (3) Verbesserung der optischen Darstellung der Zahnsbrücke durch Repositionierung oder Entfernen von fehlerhaft platzierten Brackets und eine realistischere Darstellung der Brackets und Drähte.
- (4) Automatisierung der Zahnsbrücke-Positionierung und Speicherung der Simulation zur Weiterentwicklung und Training eines möglicherweise neuen Modells.
- (5) Entwicklung einer Echtzeit-Simulation für die Anwendung mit einer Handykamera.

Das Projekt bietet eine solide Grundlage für vielfältige Weiterentwicklungen und zeigt, dass die Grenzen dieser Technologie nur durch die Kreativität gesetzt sind.

A Anhang

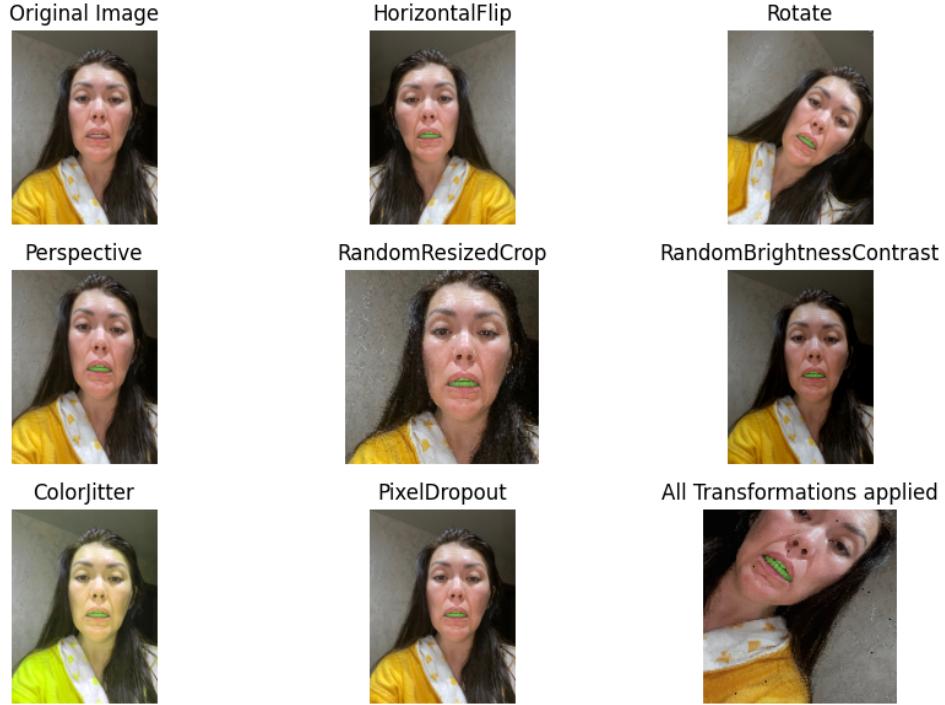


Bild 3: Angewandte Transformations der albumentations library, zugehörige transformierte Mask als overlay in grün dargestellt.

A.1 Transformations

Verwendete Transformations zur Vervielfältigung der Trainingsdaten:

- zufällige horizontale Spiegelung (**HorizontalFlip**)
- zufällige Rotation bis zu einem Winkel von 45 Grad (**Rotate**)
- zufällige Perspektivenverschiebung (**Perspective**)
- zufälliges Zuschneiden und Skalieren auf 256x256 px (**RandomResizedCrop**)
- zufällige Veränderung der Helligkeit und des Kontrastes (**RandomBrightnessContrast**)
- zufällige Farbverschiebung (**ColorJitter**)
- zufälliger Pixelausfall (**PixelDropout**)

A.2 Experimente der Hyperparameter

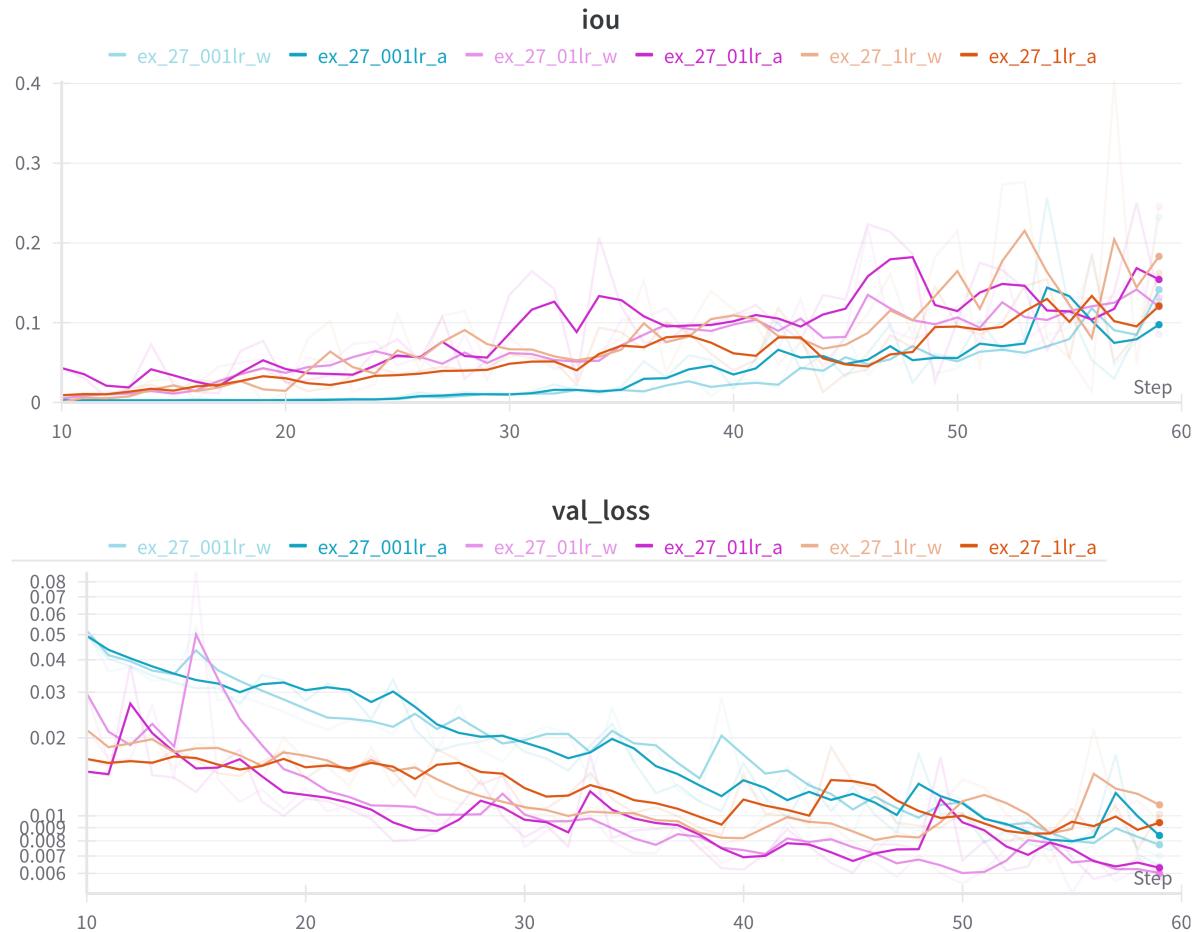


Bild 4: IoU und Validation-Loss im Experiment zum Vergleich verschiedener Learning Rates (blau = 0,001; lila = 0,01; orange = 0,1) zwischen den Optimizern Adam (dunkel) und AdamW (hell). Erstellt mit wandb.

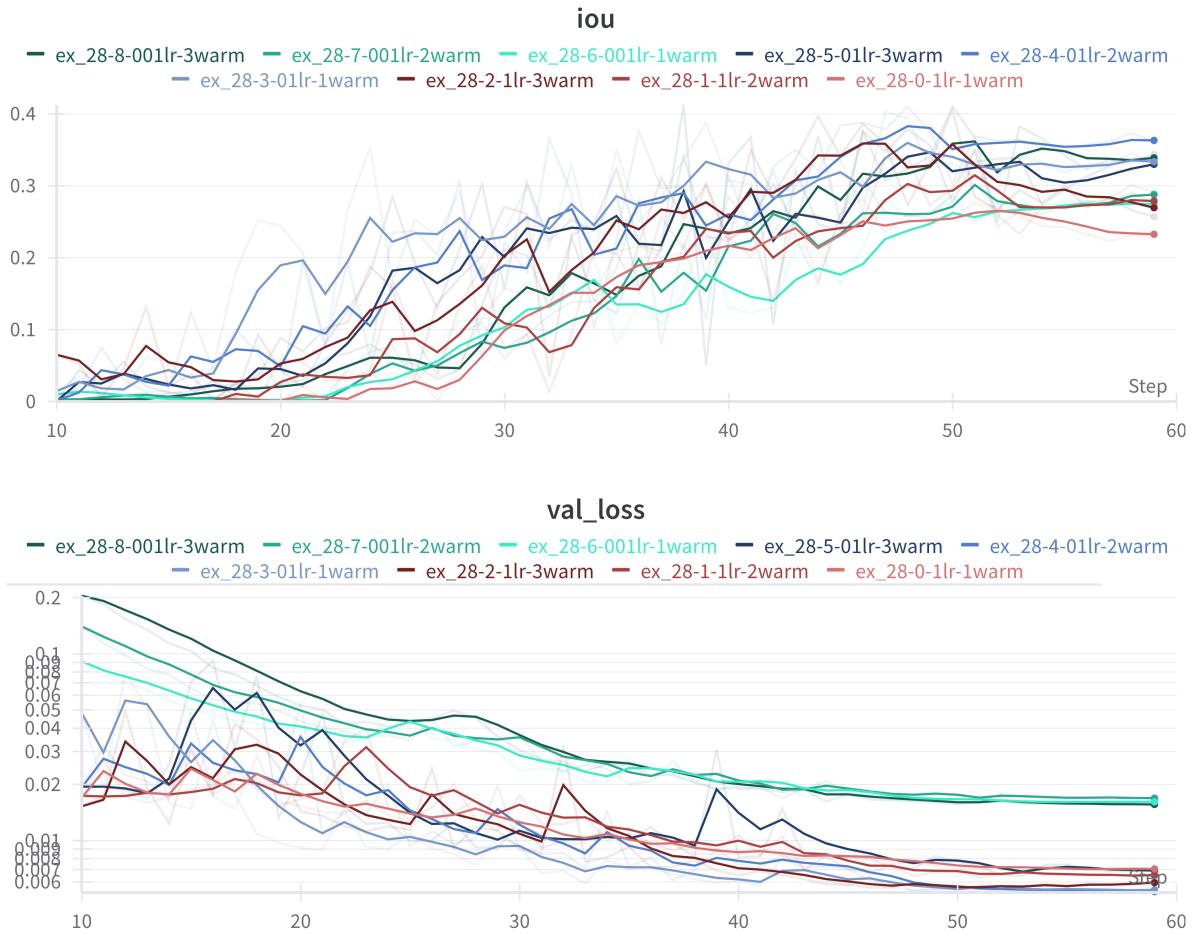


Bild 5: IoU und Validation-Loss im Experiment zum Vergleich verschieden langer Warmups (dunkel = 30%; mittel = 20%; hell = 10%) und Learning Rates (grün = 0,001; blau = 0,01; rot = 0,1). Erstellt mit wandb.

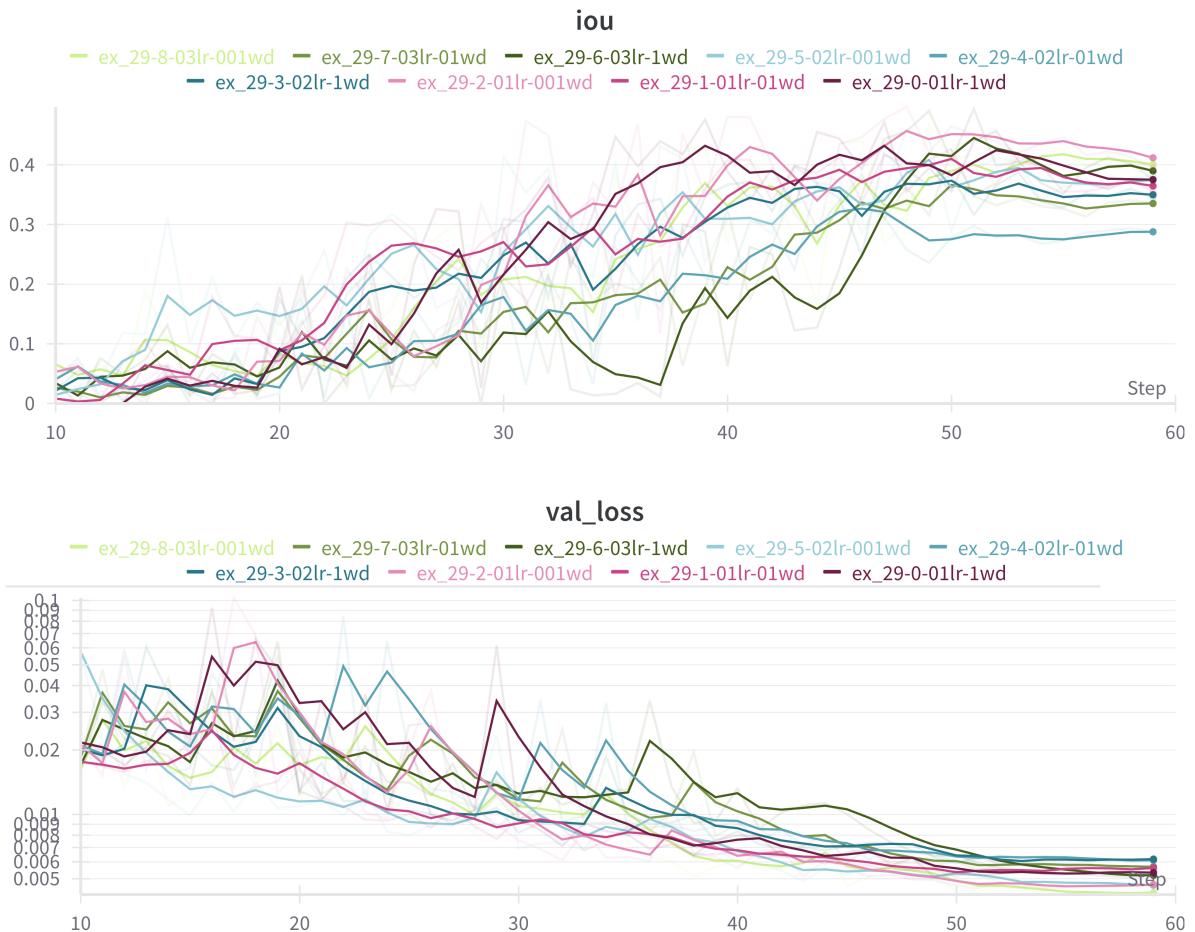


Bild 6: IoU und Validation-Loss im Experiment zum Vergleich verschiedener Learning Rates (grün = 0,03; blau = 0,02; lila = 0,01) und Weight Decays (hell = 0,001; mittel = 0,01; dunkel = 0,1). Erstellt mit wandb.

A.3 Beispielbilder

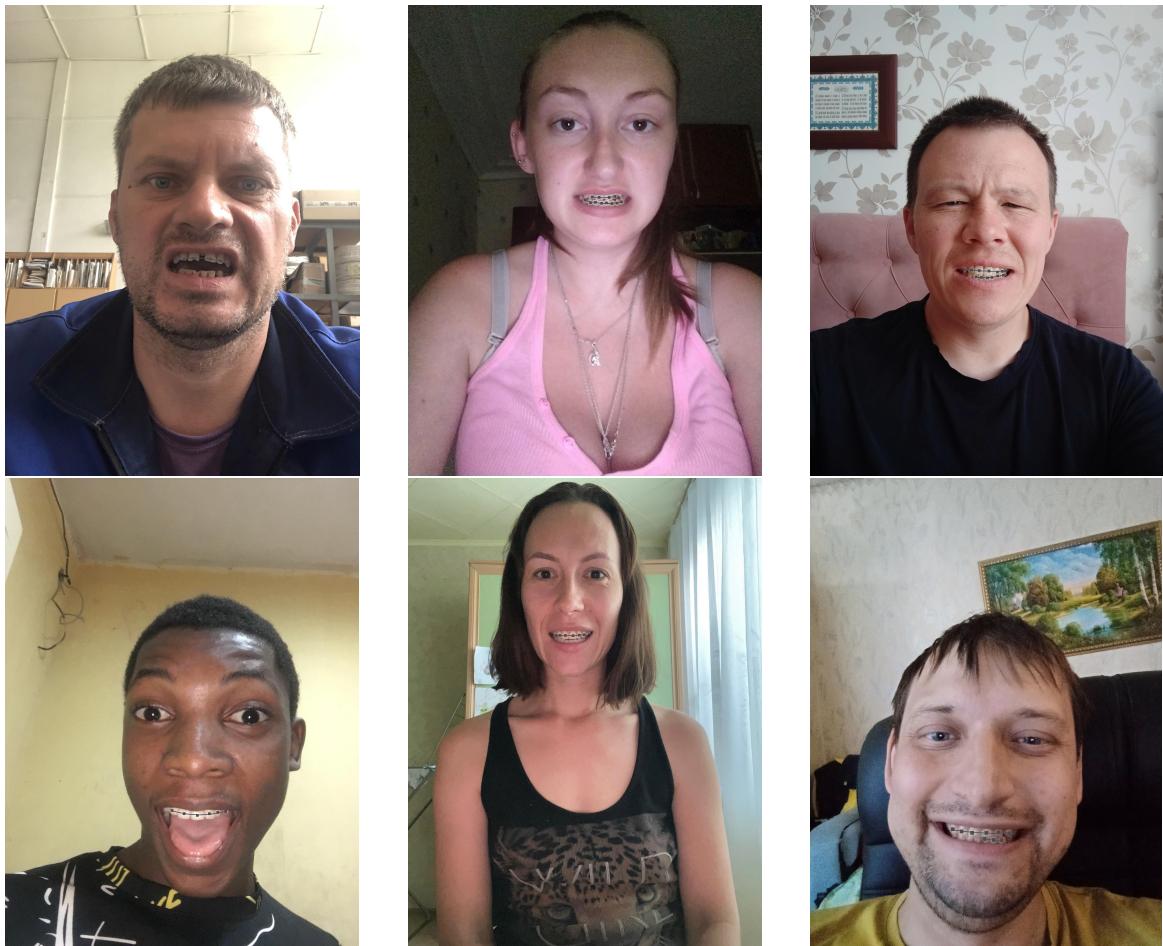


Bild 7: Beispielbilder mit positionierten Zahnsplangen.

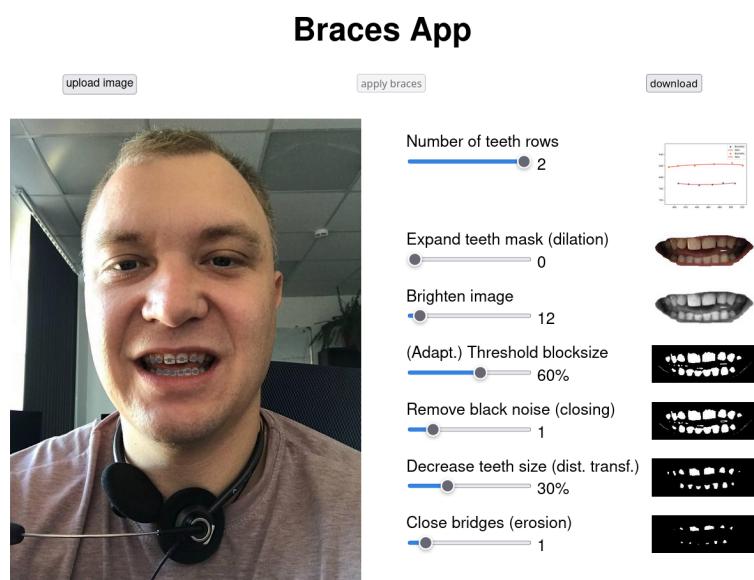


Bild 8: Nutzerinterface mit angepasster Zahnsanke an Beispielbild.

Literatur

- [Kap⁺23] A. Kapitanov, K. Kvanchiani, and K. Sofia. EasyPortrait - face parsing and portrait segmentation dataset. *arXiv preprint arXiv:2304.13509*, 2023 (cited on p. 3).
- [Los⁺19] I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019. arXiv: [1711.05101 \[cs.LG\]](https://arxiv.org/abs/1711.05101) (cited on p. 4).
- [Ron⁺15] O. Ronneberger, P. Fischer, and T. Brox. U-net: convolutional networks for biomedical image segmentation, 2015. arXiv: [1505.04597 \[cs.CV\]](https://arxiv.org/abs/1505.04597) (cited on pp. 2, 3).