

---

# Bachelorarbeit

Human-in-the-loop  
Image-Segmentation

---

Verfasser: Marcel H██████████ und Roger N██████████  
Referent: Prof. Dr. Marco L██████████  
Koreferent: Dr. Shao J██████████  
Abgabe: 26.08.2022

Bachelorstudiengang: Systemtechnik: Vertiefung Ingenieurinformatik  
Schule: Ostschweizer Fachhochschule  
Departement: Technik  
Institut: INF Institut für Ingenieurinformatik



## Zusammenfassung

Die Image-Segmentierung ist eines der Schlüsselprobleme auf dem Gebiet der Computer Vision. Durch das Aufteilen eines Bildes in Segmente (Teilbereiche) kann der Weg zum vollständigen Verständnis des Bildinhalts geebnet werden. Die Bedeutung dieses Problems wird durch die Tatsache unterstrichen, dass eine wachsende Zahl von Anwendungen auf die Ableitung von Wissen aus Bildern angewiesen sind. Einige dieser Anwendungen umfassen selbstfahrende Fahrzeuge, Augmented Reality oder Gesichtserkennung. Der momentane Stand der Technik ist die Nutzung von künstlichen neuronalen Netzwerken für die Image-Segmentierung. Für das Training dieser Netzwerke werden jedoch grosse Datenmengen benötigt, die nicht immer für jede Problemstellung zur Verfügung stehen. In solchen Fällen kann keine genügend hohe Genauigkeit mit dem Netzwerk erreicht werden und die Segmentierung muss manuell erledigt werden.

Bei dem Human-in-the-loop Ansatz wird die Verwendung von künstlichen neuronalen Netzwerken mit einer manuellen Nachbearbeitung verbunden. Dabei soll der Mensch die Segmentierung des neuronalen Netzwerks überprüfen und gegebenenfalls anpassen. Durch die verbesserte Segmentierung generiert der Benutzer weitere Daten. Diese können benutzt werden, um das neuronale Netzwerk weiter zu trainieren. Eine Verbesserung des Netzwerks erzielt genauere Vorschläge, welche die Bearbeitung eines Segmentierungsauftrages beschleunigen. Somit lernt das System durch die Benutzung automatisch weiter und kann sich dynamisch der Anwendung anpassen. Der Ansatz ist in der folgenden Abbildung 1 dargestellt.

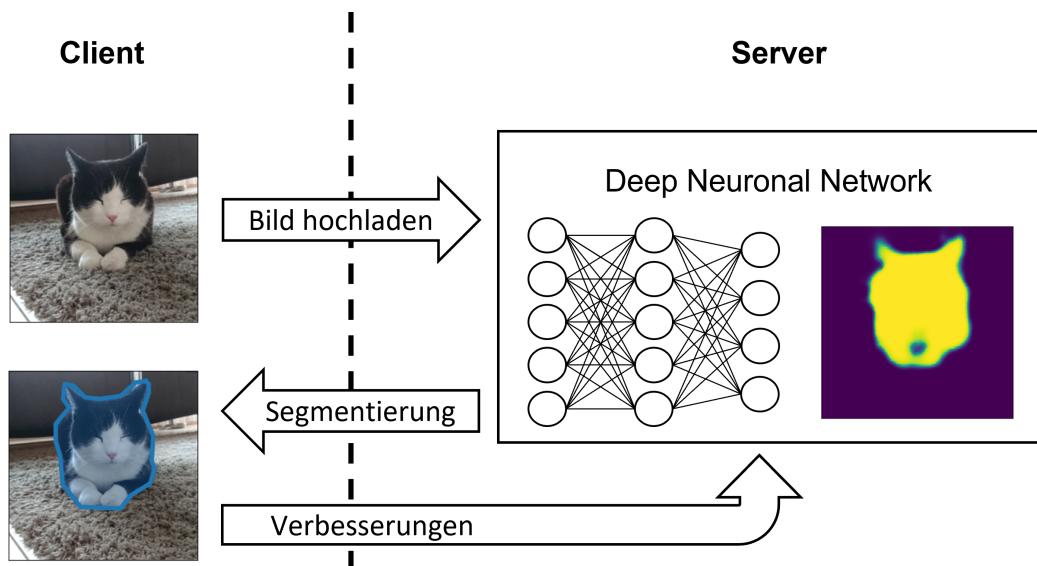


Abbildung 1: Katzenbild-Segmentierung mit dem Human-in-the-loop Ansatz.

Das Resultat dieser Bachelorarbeit ist ein System zur AI-unterstützten Bildsegmentierung, welches sich leicht in eine bestehende Infrastruktur integrieren lässt. Dabei wurde das neuronale Netzwerk austauschbar gemacht. Dadurch ist man nicht nur auf einen Anwendungsfall beschränkt. Diese Generalisierung erlaubt unterschiedliche Anwendungsfälle, wie zum Beispiel eine Segmentierung von Katzenbildern oder Gebäudeplänen. Das System besteht aus einem Frontend-, einem Backend- und einer Image-Segmentation-Komponente.

## Frontend

Das Frontend ist mit dem React-Framework als Webapplikation implementiert. Es lässt sich dadurch einfach in bestehende Webseiten integrieren und diese um eine interaktive AI-unterstützte Segmentierung von Bildern erweitern. Für die Kommunikation mit dem Backend wird eine REST-Schnittstelle verwendet. Das User-Interface ist schlicht gehalten, um einen unkomplizierten Arbeitsablauf sicherzustellen. Der Benutzer lädt zunächst das zu segmentierende Bild in die Webapplikation. Diese stellt im Hintergrund eine Segmentierungsanfrage an das Backend. Das Backend erarbeitet die Segmentierung und gibt als Resultat eine Liste von Polygonen in Form eines COCO-JSONs zurück. Diese werden im User-Interface über das Bild gelegt. An dieser Stelle kann der Benutzer den Segmentierungsvorschlag verwerfen, annehmen oder verbessern. Als letzter Schritt kann die Segmentation als COCO-JSON exportiert werden. Dieser Stand der Segmentierung wird gleichzeitig als Korrektur an das Backend geschickt.

## Backend

Das Backend nimmt Segmentierungsanfragen über eine REST-Schnittstelle entgegen. Um die Schnittstelle für das Frontend zu implementieren wird das Framework FastAPI genutzt. Die Anfragen werden an die Image-Segmentierung weitergeleitet und die Segmentierung wird als Resultat anschliessend an das Frontend geschickt. Die vom Benutzer korrigierte Segmentierung wird abschliessend im Backend abgelegt und später verwendet, um die Image-Segmentierung zu verbessern. Bei einer Verbesserungsrunde werden verschiedene Metriken gespeichert. Diese werden über ein Dashboard dargestellt. Über dieses kann die Entwicklung des neuronalen Netzwerks innerhalb der Image-Segmentierung beobachtet werden.

## Image-Segmentierung

Das Image-Segmentierung-Modul befasst sich mit der eigentlichen Segmentierung der Bilder. Dabei ist es möglich das neuronale Netzwerk auszutauschen, um verschiedenste Anwendungsbereiche zu ermöglichen. Die Schnittstelle zum Backend ist durch ein Python-Interface realisiert. Erhält die Image-Segmentierung einen Auftrag vom Backend wird als erster Schritt ein Bild in den Arbeitsspeicher geladen und auf die Inputgrösse des neuronalen Netzwerkes skaliert. Das skalierte Bild wird dem neuronalen Netzwerk übergeben und dieses erstellt für jeden Pixel eine Wahrscheinlichkeit, ob er der jeweiligen Kategorie zugehört. Durch das so entstehende Binärbild werden Polygone generiert und anschliessend auf die Orginalgrösse des Bildes skaliert. Schlussendlich wird mit den gefundenen Polygonen ein COCO-JSON-Dokument erstellt, welches an das Backend weitergegeben wird.

Programmcode auf:

<https://github.com/Ba-MH-RN/Human-in-the-loop-Image-Segmentation>

## Abstract

Image segmentation is one of the key problems in the field of computer vision. Splitting an image into segments (sub-regions) can pave the way to a complete understanding of the image content. The importance of this problem is underlined by the fact, that a growing number of applications depends on knowledge derived from images. Some of these applications include self-driving vehicles, augmented reality or facial recognition. The current state of the art is the use of artificial neuronal networks for image segmentation. However, training these networks requires large amounts of data, which are not available for every problem. In such cases, sufficiently high accuracy cannot be achieved with the network and the segmentation has to be done manually. The human-in-the-loop approach combines the use of artificial neuronal networks with manual editing. A human is checking the segmentation of the neuronal network and adjust it if necessary. Through the improved segmentation the user generates further data. These can be used to further train the neuronal network. An improvement of the network achieves more accurate suggestions, which accelerates the workflow. Thus, the system learns with the usage and can dynamically adapt to the use case. The approach is shown in the following figure 2 .

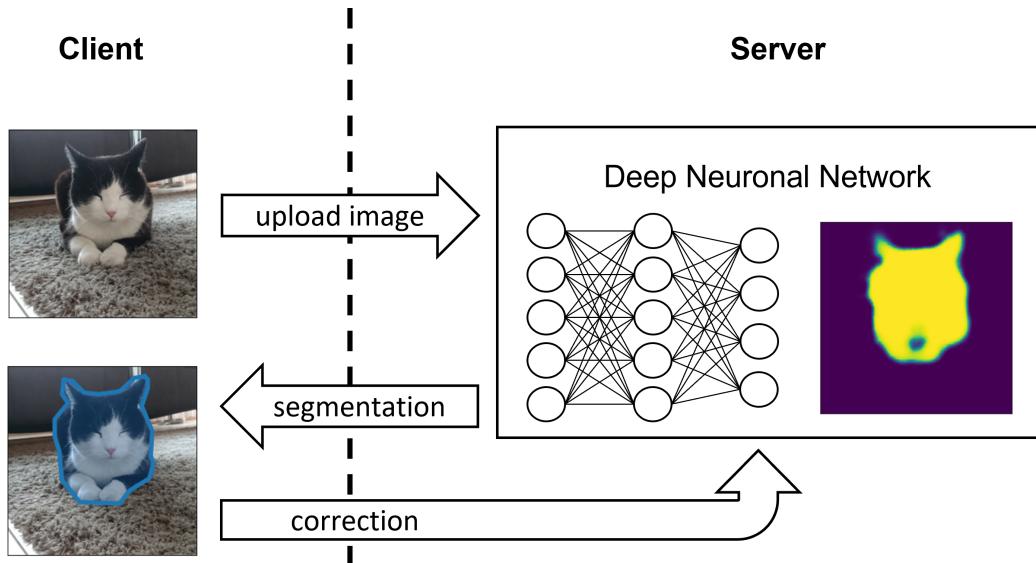


Figure 2: Cat image segmentation using the human-in-the-loop approach.

The result of this bachelor thesis is a system for AI-assisted image segmentation, which can be easily integrated into an existing infrastructure. In doing so, the neuronal network was made interchangeable. Thus, one is not limited to only one use case. This generalization allows different use cases, such as a segmentation of cat images or floor plans. The system consists of a frontend, a backend and an image segmentation component.

## Frontend

The frontend is implemented as a web application using the React framework. It can therefore be easily integrated into existing websites. Thus, extending the website with a interactive AI-supported segmentation of images. A REST interface is used for communication with the backend. The user interface is kept simple to ensure an easy workflow. The user first loads the image into the web application. This makes a segmentation request to the backend in the background. The backend works out the segmentation and returns a list of polygons as the result. These are overlayed on the image in the user interface. At this point, the user can reject, accept or improve the segmentation proposal. As a final step, the segmentation can be exported as a COCO-JSON. This state of the segmentation is simultaneously sent to the backend as a correction.

## Backend

The backend accepts segmentation requests via a REST interface. To implement the interface for the frontend the framework FastAPI is used. The requests are forwarded to the image segmentation and the segmentation is then sent to the frontend as a result. The segmentation corrected by the user is finally stored in the backend and is later used to improve the image segmentation. During an improvement round, various metrics are stored. These are displayed via a dashboard. The development of the neural network can be observed through it.

## Image segmentation

The image segmentation module deals with the actual segmentation of the images. It is possible to exchange the neuronal network to enable a wide range of applications. The interface to the backend is realized by a Python interface. When the image segmentation receives a request from the backend, the first step is to load an image into memory and scale it to the input size of the neuronal network. The scaled image is passed to the neuronal network and the neuronal network generates a probability for each pixel to the respective category. The resulting binary image is used to generate polygons and these are scaled to the original size of the image. Finally, a COCO-JSON document is created with the polygons found and passed to the Backend.

Program at:

<https://github.com/Ba-MH-RN/Human-in-the-loop-Image-Segmentation>

## Danksagung

An dieser Stelle möchten wir uns ganz herzlich bei all jenen bedanken, die uns während unserer Bachelorarbeit unterstützt haben.

Allen voran möchten wir uns bei Herrn Prof. Dr. Marco Lehmann bedanken für die Möglichkeit diese abwechslungsreiche Arbeit durchführen zu dürfen. Seine konstanten Inputs ermöglichen ein rasches Vorankommen. Ebenfalls bedanken wir uns bei Herrn Dr. Shao Jü Woo für seinen Einsatz als Korrektor. Beide trugen mit ihrem enormen Wissen massgeblich zur Erstellung dieser Arbeit bei.

Ein grosser Dank gebührt auch Amira Jäger für die Bereitstellung einer anfänglichen Image-Segmentation für Solarpanels. Dies ermöglichte erste Erfolge und den Aufbau eines Prototypen. Die gewonnenen Erkenntnisse flossen in das endgültige System mit ein.

Ferner bedanken wir uns bei Kemaro AG, welche uns eine anfängliche Problemstellung und Anwendungsfall aufzeigte. Diese legte den Grundstein für die Arbeit. Die zur Verfügung gestellten Gebäudepläne von grossen Industriehallen haben uns einen Einblick in die Problematik der Aufgabe verschafft.

Ausserdem möchten wir uns bei unseren Familien und unserem Umfeld für deren Unterstützung bedanken.

[REDACTED] im August 2022

## Abkürzungsverzeichnis

Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
API	Application Programming Interface
BTN	Button
Cat	Category
CB	Callback
COCO	Common Objects in Context
CORS	Cross Origin Resource Sharing
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DB	Database
Dict	Dictionary
DOM	Document Object Model
FastFCN	Fast Fully Convolutional Network
FK	Foreign Key
FN	False Negative
FP	False Positive
FSM	Finite State Machine
HRNet	High Resolution Network
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identification
Init	Initialization
Int	Integer
IoU	Intersection over Union
JPU	Joint Pyramide Unit
JS	JavaScript
JSON	JavaScript Object Notation
JSX	Javascript XML
LSF	Labelstudio Frontend
mIoU	Mean Intersection over Union
ORM	Object Relational Mapping
PK	Primary Key
REST	Representational State Transfer
ROI	Return on Investment
SQL	Structured Query Language
Str	String
TN	True Negative
TP	True Positive
TS	Timestamp
URL	Uniform Resource Locator
Val	Validation
varchar	Variable Character Field
VC	Version Control
XML	Extensible Markup Language

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
<b>2. Übersicht</b>	<b>2</b>
2.1. Systemaufbau . . . . .	2
2.2. Verwendete Technologien . . . . .	4
2.3. Abgrenzungen . . . . .	4
<b>3. Human-in-the-loop</b>	<b>5</b>
3.1. Verbesserungsprozess . . . . .	6
3.2. Model-Version-Control . . . . .	7
<b>4. Frontend</b>	<b>8</b>
4.1. Umsetzung . . . . .	8
4.1.1. Informationsfluss . . . . .	10
4.1.2. Finite State Machine . . . . .	11
4.2. Bedienoberfläche . . . . .	13
4.2.1. Kontrollelemente . . . . .	13
4.2.2. Informationsbereich . . . . .	14
4.2.3. Verfügbare Labels . . . . .	15
4.2.4. Arbeitsfläche . . . . .	15
4.2.5. Übersicht Annotierungen . . . . .	15
4.3. Integrationsbeispiel . . . . .	15
4.4. Verwendete Technologien . . . . .	16
4.4.1. COCO-JSON . . . . .	16
4.4.2. React . . . . .	18
4.4.3. Label-Studio-Frontend . . . . .	18
4.4.4. Axios . . . . .	18
4.4.5. CSS-Module . . . . .	18
<b>5. Backend</b>	<b>19</b>
5.1. REST-API . . . . .	20
5.1.1. FastAPI . . . . .	20
5.1.2. Endpunkte . . . . .	20
5.2. Middleware . . . . .	21
5.3. Segmentierungsauftrag . . . . .	21
5.4. Datenablage . . . . .	24
5.5. Evaluierung der Image-Segmentierung . . . . .	26
5.5.1. Metriken . . . . .	26
5.5.2. Dashboard . . . . .	27
5.6. Softwaretests . . . . .	28
<b>6. Image-Segmentierung</b>	<b>29</b>
6.1. Ablauf der Segmentierung . . . . .	30

---

6.2. Künstliche neuronale Netzwerke für die Image-Segmentierung . . . . .	31
6.2.1. Dataset und Data-Augmentation . . . . .	31
6.2.2. Modellarchitekturen . . . . .	32
6.2.3. Verlustfunktionen . . . . .	34
6.2.4. Optimierungsprozess . . . . .	34
6.3. Modell zur Segmentierung von Katzenbildern . . . . .	35
6.4. Modell zur Segmentierung von Gebäudeplänen . . . . .	36
6.5. Nachtrainieren . . . . .	38
6.6. Model-Version-Control . . . . .	38
<b>7. Fazit</b>	<b>39</b>
<b>8. Ausblick</b>	<b>40</b>
<b>A. Verzeichnisse</b>	<b>41</b>
<b>B. Aufgabenstellung und Zieldefinition</b>	<b>45</b>
<b>C. Eidesstattliche Erklärung</b>	<b>46</b>
<b>D. Testbericht Backend und Image-Segmentierung</b>	<b>47</b>
<b>E. Schemas der REST-Schnittstelle</b>	<b>48</b>
<b>F. Aufbau des COCO-JSON Formates</b>	<b>52</b>
<b>G. Image-Segmentation Katzen</b>	<b>54</b>
<b>H. Image-Segmentation Floorplans</b>	<b>64</b>
<b>I. Fachmodul</b>	<b>99</b>

## 1. Einleitung

Diese Arbeit behandelt das Entwickeln eines Systemes für die semi-automatische Segmentierung von Bildern. Dabei wird ein künstliches neuronales Netzwerk mit anschliessender Korrektur durch einen Experten kombiniert. Die Korrekturen werden in einem nächsten Schritt als Trainingsdaten für die Verbesserung des neuronalen Netzwerks genutzt. Dieses Konzept entspricht dem Human-in-the-loop Ansatz. Die Arbeit baut dabei auf den Ergebnissen und Erkenntnissen auf, welche im dazugehörigen Fachmodul erarbeitet wurden. Der Bericht zum Fachmodul ist im Anhang I beigefügt.

Ein konkretes Fallbeispiel und damit die Idee für diese Bachelorarbeit wurde von der Kemaro AG zur Verfügung gestellt. Bei diesem Fallbeispiel hätte ein Nutzer mehrere Gebiete in einem Gebäudeplan eingezeichnet, welche zur Planung der Reinigungsroute eines autonomen Reinigungsroboter benutzt werden. Bei dieser Aufgabe sollte der Nutzer von einem künstlichen neuronalen Netzwerk unterstützt werden.

Dieses Fallbeispiel stellt im Kern eine Image-Segmentierung dar. Die Image-Segmentierung ist eines der Schlüsselprobleme auf dem Gebiet der Computer-Vision. Durch das Aufteilen eines Bildes in Segmente (Teilbereiche) können Erkenntnisse zum Bildinhalt gewonnen werden. Darüber hinaus können die Segmente weiterverwendet werden, um eine detaillierte Aussagen zu ermöglichen. Die Bedeutung dieses Problems wird durch die Tatsache unterstrichen, dass neben dem genannten Fallbeispiel eine wachsende Zahl von Anwendungen auf die Ableitung von Wissen aus Bildern angewiesen ist. Einige weitere Anwendungen umfassen selbstfahrende Fahrzeuge, Augmented-Reality oder Gesichtserkennung. Um diese Probleme zu lösen, werden immer häufiger neuronale Netzwerke eingesetzt, da diese heutzutage die besten Ergebnisse in diesem Bereich erzielen. Ein grosses Hindernis beim Erstellen dieser Netzwerke ist das Sammeln von genügend Trainingsdaten. Falls zu wenige Trainingsdaten vorliegen, kann keine genügend hohe Genauigkeit mit dem Netzwerk erreicht werden und die Segmentation muss manuell erledigt werden.

Der Human-in-the-loop Ansatz setzt auf die Verwendung von neuronalen Netzwerken mit einer manuellen Nachbearbeitung. Dabei soll der Nutzer die Segmentierung des neuronalen Netzwerks überprüfen und gegebenenfalls anpassen. Durch die verbesserte Segmentierung generiert der Nutzer weitere Daten. Diese können genutzt werden, um das neuronale Netzwerk weiter zu trainieren. Eine Verbesserung des Netzwerks erzielt genauere Vorschläge, welche die Bearbeitung eines Segmentierungsauftrages beschleunigt. Somit lernt das System durch die Benutzung automatisch weiter und kann sich dynamisch der Anwendung anpassen.

Das Ziel dieser Arbeit ist ein System zur Image-Segmentierung, welches den Human-in-the-loop Ansatz implementiert. Dabei soll das System möglichst einfach in eine bestehende Infrastruktur integriert werden können. Dies umfasst auch das Auswechseln des neuronalen Netzwerks.

## 2. Übersicht

Das entstandene System wird durch diesen Bericht dokumentiert. Dazu werden die einzelnen Bestandteile des Systems im Detail thematisiert. Zu Beginn wird das Human-in-the-loop Konzept vorgestellt. Die nächsten drei Kapitel behandeln die einzelnen Hauptkomponenten des Systems. Beim Frontend Kapitel steht die Umsetzung der Bedienoberfläche und deren Nutzung im Fokus. Dieses wird gefolgt von dem Kapitel über das Backend. Hier werden die Schnittstellen und der Aufbau erläutert. Abschliessend wird die Image-Segmentierung in Theorie und anhand der Segmentierung von Katzenbildern und Gebäudeplänen thematisiert. Der Bericht wird mit einem Fazit und einem Ausblick abgeschlossen. Ebenfalls werden mögliche Richtungen für eine Weiterentwicklung und Verbesserungen beschrieben. Im Anhang befinden sich Verzeichnisse, Aufgabenstellung, Testbericht zum Backend und Image-Segmentierung, Machine-Learning Modelle und das Fachmodul.

### 2.1. Systemaufbau

Das System ist darauf ausgelegt, sich einfach in eine bestehende Infrastruktur integrieren zu lassen. In folgender Abbildung 3 sind die einzelnen Komponenten des Systems dargestellt und an welchem Ort sich diese befinden.

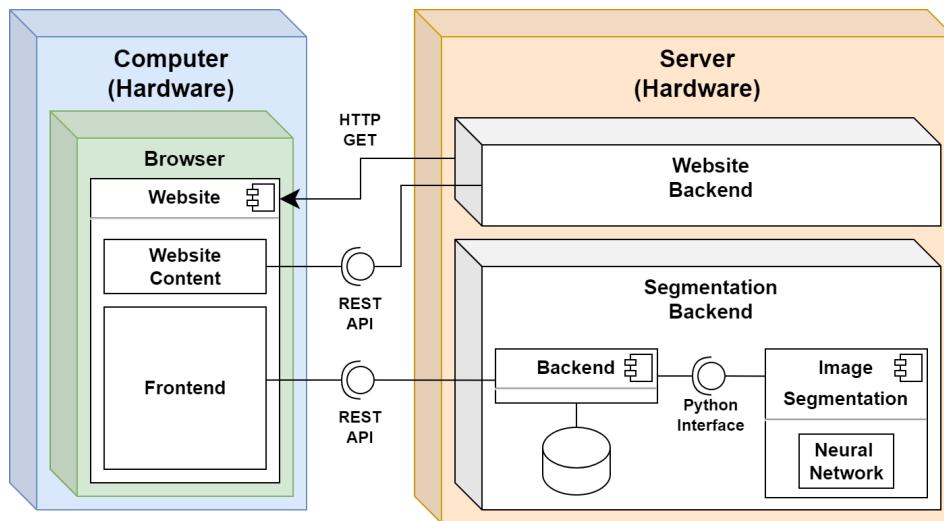


Abbildung 3: Deploymentdiagramm des Systems in bestehender Infrastruktur.

Das Segmentation-Backend existiert abseits vom Website-Backend auf dem Server. In ihm ist das eigentliche Backend und die Image-Segmentierung untergebracht. Das Backend stellt über eine REST-API ein Interface für das Frontend zur Verfügung. Ebenfalls kümmert sich das Backend um die Verwaltung der Trainingsdaten und der nachtrainierten Modelle. Zu diesem Zweck wird eine relationale Datenbank genutzt, wobei der Nutzer keinen Zugriff auf die Trainingsdaten hat. Das Frontend wird als React-Komponente in die Webseite eingebunden. Als Teil der Webseite liegt das Frontend daher auf dem Website-Backend und wird vom Browser angefordert und geladen. Beim Einbinden des Frontends wird ihm die URL des Segmentation-Backends mitgegeben. Durch diese URL kann auf die REST-API zugegriffen und somit das Backend genutzt werden. Die

Daten zur Bearbeitung eines Segmentierungsauftrages liegen dabei lokal im Frontend. Zu diesen Daten gehören das Bild und die Segmentierung. Diese werden, solange wie der Auftrag dauert, im Speicher des Browsers gehalten. Sobald der Auftrag beendet wird, werden auch die Daten verworfen.

Die drei Hauptkomponenten Frontend, Backend und Image-Segmentation sind klar durch Schnittstellen abgegrenzt. Zwischen dem Frontend und Backend kommt dazu eine REST-API zum Einsatz, während zwischen Backend und Image-Segmentation die Schnittstelle über ein Python-Interface implementiert wird. Die drei Hauptkomponenten und deren Kommunikation zwischen-einander wird in der folgenden Abbildung 4 dargestellt.

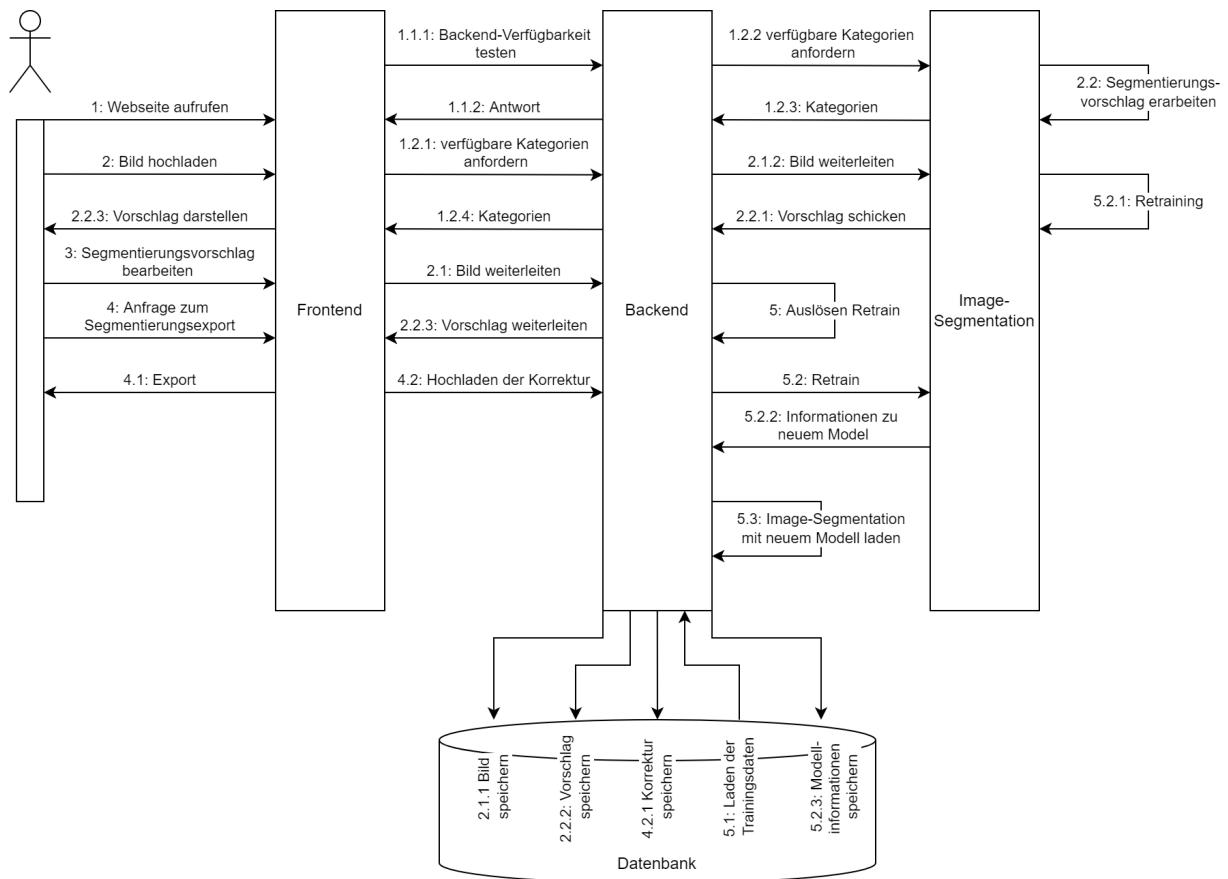


Abbildung 4: Kommunikationdiagramm von Frontend, Backend und Image-Segmentation.

Dabei dienen die Pfade mit 1.x.x der Initialisierung des Systems. Der eigentliche Segmentierungsvorgang ist mit den Pafen 2.x.x abgebildet. Den Abschluss eines Segmentierungsauftrages ist über die Pafen 4.x.x dargestellt. Ist ein Auftrag abgeschlossen wird vom Backend überprüft, ob ein Nachtrainieren ausgelöst werden soll. Dieser Fall ist mit den Pafen 5.x.x dargestellt.

Das Frontend übernimmt die Rolle eines Segmentierungswerkzeuges. Es ermöglicht dies über interaktive Polygone, welche über das Bild gelegt werden. Diese Polygone können manipuliert und abschliessend heruntergeladen werden.

Um einen Benutzer bei dieser Aufgabe zu unterstützen, wird das Bild an das Backend geschickt. Das Backend übernimmt dabei die Rolle der Verwaltung. Neben dem Weiterleiten des Bildes

an die Image-Segmentation, wird das Bild gespeichert, um es später für Trainingszwecke zu benutzen. Für das Training werden auch die Verbesserungen, welche im Frontend stattfinden, benötigt. Diese Daten werden genutzt, um das neuronale Netzwerk nach zu trainieren und somit die Image-Segmentation zu verbessern.

Die Image-Segmentation nutzt ein künstliches neuronales Netzwerk, um Segmentierungsvorschläge zu erarbeiten. Aus diesen wird in einem zweiten Schritt eine Liste von Polygonen erstellt. Diese werden an das Backend zurückgegeben, welches wiederum diese an das Frontend weiterleitet. Hier werden die Vorschläge dem Nutzer grafisch präsentiert.

## 2.2. Verwendete Technologien

Für das Fachmodul (siehe Anhang I) wurden mehrere Technologien für das Frontend, Backend und Image-Segmentation ausgetestet. Die Entscheidungsfindung und -begründung war ein Teil des Fachmoduls und ist dort dokumentiert. Zur Übersicht werden hier die wichtigsten Kerntechnologien und deren Einsatzgebiet kurz erwähnt.

Für das Backend kommt das Framework FastAPI zum Einsatz. Dieses ermöglicht eine REST-API mittels Python zu implementieren. Durch den Einsatz von Python im Backend können Bibliotheken und Frameworks für das Machine-Learning einfach eingebunden werden.

Das Frontend ist als React-Komponente umgesetzt. Als solche kann es von React-Applikationen sowie über HTML-Webseiten eingebunden werden. Für die Darstellung und Bearbeitung der Segmentation wird Label-Studio-Frontend genutzt. Dies ist eine Bibliothek, um eine Datenannotation in Webapplikationen zu realisieren.

## 2.3. Abgrenzungen

Das Resultat dieser Bachelorarbeit ist als ein Konzept zu betrachten. Dementsprechend sind einige Gebiete, welche für eine Applikation in einer Produktionsumgebung essentiell wären, ausgegrenzt worden. Die Sicherheit ist ein Aspekt, welcher nicht behandelt wurde. Mechanismen gegen Angriffe auf das System oder gegen einen Missbrauch wurden nicht integriert. Die Skalierbarkeit des Systems steht nicht im Fokus. Es richtet sich an Nutzer, welche einzelne Bilder segmentieren möchten. Es ist nicht darauf ausgelegt ganze Datensätze zu segmentieren. Die Implementierung eines vollumfänglichen Data- oder Model-Versioning-Systems für die Daten- oder Modellverwaltung wurde auf das Minimum reduziert.

### 3. Human-in-the-loop

Das Human-in-the-loop Konzept versucht die menschliche Intelligenz mit der künstlichen Intelligenz zu verbinden. Dabei wird mindestens einer der folgenden Punkte verfolgt [1, Seite 4]:

- Erhöhung der Genauigkeit eines Modells.
- Schnelleres Erreichen der Zielgenauigkeit eines Modells.
- Kombination der menschlichen und künstlichen Intelligenz, um die Genauigkeit zu maximieren.
- Unterstützung des Menschen beim Lösen einer Aufgabe mit künstlicher Intelligenz, um die Effizienz zu steigern.

Der Anwendungsbereich dieses Konzepts ist vielseitig. Eines der grössten Einsatzgebiete findet sich bei den Suchmaschinen. Diese verwenden gestellte Suchanfragen, um das System zu verbessern. Dabei werden bei einer Suchanfrage dem Menschen mehrere Links zur Verfügung gestellt. Falls dieser nicht den ersten Link auswählt, wird diese Information genutzt, um das System auf dieser Suchanfrage zu verbessern. Die Anwendung des Konzepts im Computer-Vision Bereich nimmt in letzter Zeit an Popularität zu, wie zum Beispiel für die Image-Segmentierung. Dabei wird es beim autonomen Fahren, Heimgeräten und Mobiltelefonen verwendet. Eine Umsetzung des Konzepts für die Image-Segmentierung ist in der folgenden Abbildung 5 zu sehen [1, Seite 17].

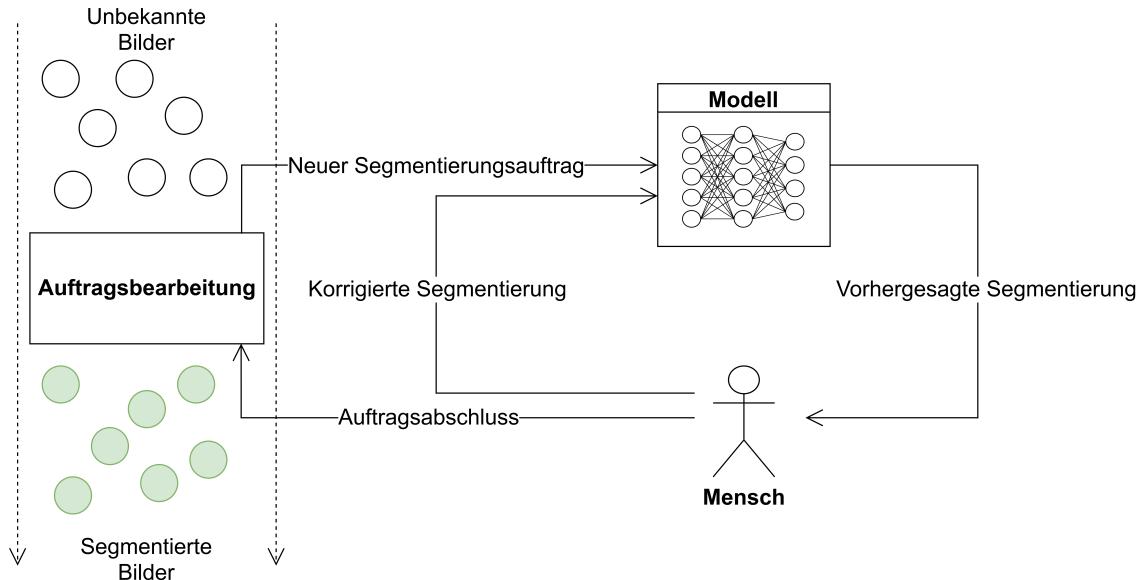


Abbildung 5: Human-in-the-loop Konzept für die Image-Segmentierung. In Anlehnung an [1, Seite 57].

Beim Einsatz des Konzepts für die Image-Segmentierung wird bei jedem Bild, welches segmentiert werden soll, in Zusammenarbeit von Mensch und künstlicher Intelligenz gearbeitet. Dies basiert auf der Tatsache, dass die unbekannten Bilder grundverschieden sind. Wird ein neues Bild dem

System übergeben, versucht das Modell, welches die künstliche Intelligenz beherbergt, eine Segmentation zu erstellen. Diese wird in einem weiteren Schritt von einem Menschen überprüft und gegebenenfalls korrigiert. Dieser schliesst den Segmentierungsauftrag ab, wobei das Endergebnis auch an das Modell weitergeleitet wird und dieses sich mit den gewonnenen Informationen verbessert. Im folgenden Abschnitt wird dieser Prozess beschrieben und abschliessend wird auf den Umgang mit den entstandenen Modellen eingegangen.

### 3.1. Verbesserungsprozess

Das Modell soll sich mit der Benutzung des Systems weiterentwickeln, dies wird durch die Rückmeldung des Menschen ermöglicht. In der folgenden Abbildung 6 ist dies für eine Segmentierungsaufgabe dargestellt.

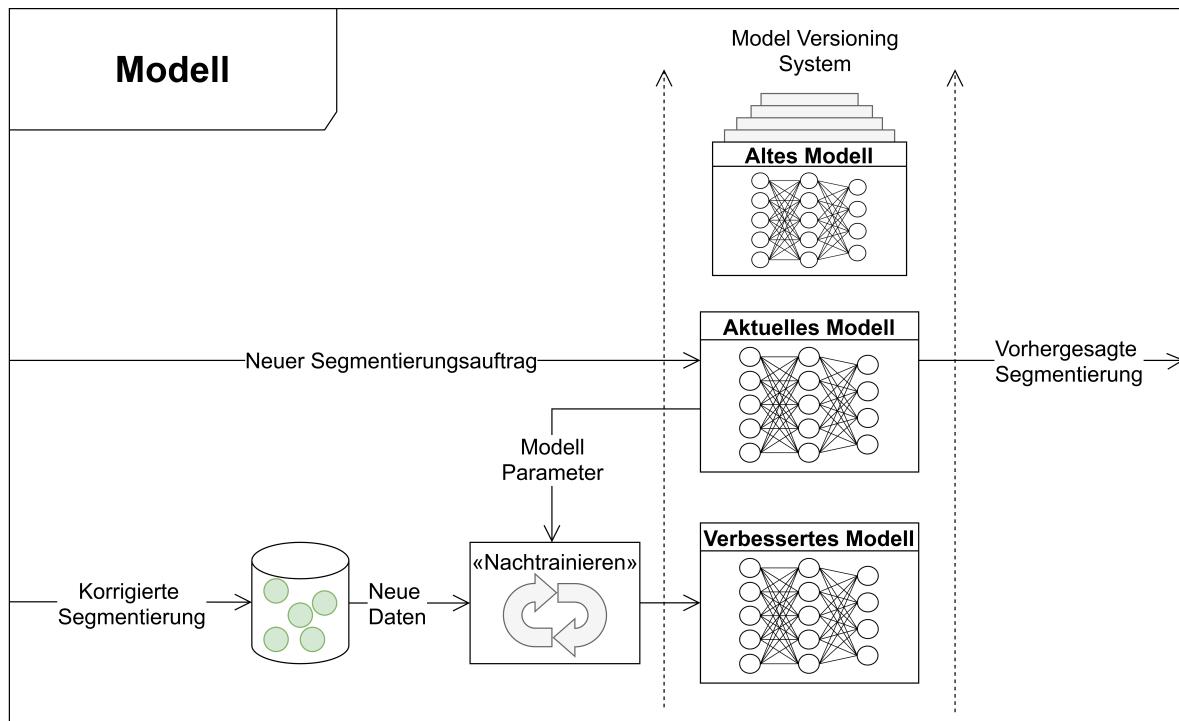


Abbildung 6: Verbesserung des Modells durch die Anwendung des Human-in-the-loop Konzepts.

Dabei werden die korrigierten Segmentierungen, welche vom Mensch kommen, gespeichert. Auf Grundlage des aktuellen Modells wird mit den neuen Daten ein neues Modell trainiert. Der Zeitpunkt des Nachtrainierens kann durch eine erreichte Anzahl neuer Daten ausgelöst werden. Das resultierende Modell sollte das aktuelle Modell übertreffen. Um dies sicherzustellen, kann mit Metriken gearbeitet oder das Modell kann von einem Experten überprüft werden. Schlussendlich wird das aktuelle Modell durch das neue Modell ersetzt.

### 3.2. Model-Version-Control

Der Umgang mit einem Modell in einem produktiven Umfeld ist entscheidend. Dabei besitzt jedes Modell einen Lebenszyklus, in dem es entwickelt und gepflegt wird. In der folgenden Abbildung 7 ist ein möglicher Lebenszyklus abgebildet [2].

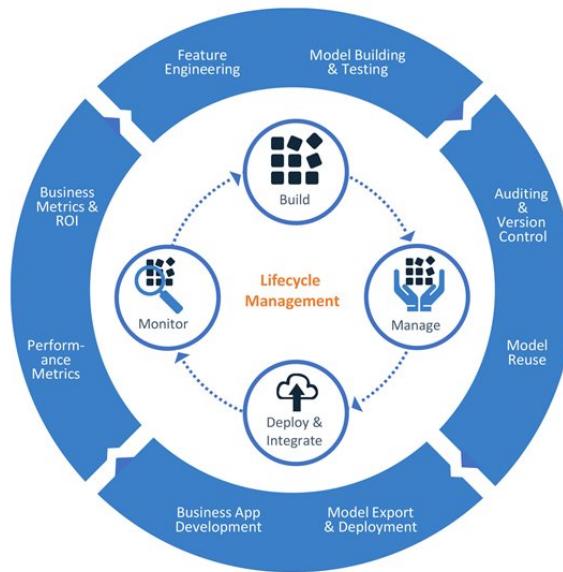


Abbildung 7: Lebenszyklus eines Machine-Learning Modells [2].

Die einzelnen Schritte, welches ein Modell durchläuft, sind Erstellung, Verwaltung, Bereitstellung und Überwachung. Zu der Erstellung gehört eine Datenaufbereitung, Feature-Engineering und Tests, um die Modellgüte zu bewerten. Ist das Modell erstellt, wird es in einem Verwaltungssystem aufgenommen. Dieses kümmert sich um die verschiedenen Versionen der Modelle um eine Wiederverwendung zu ermöglichen. Die Bereitstellung umfasst die Implementierung und Integration in die produktiven Geschäftssysteme und Anwendungen. Beim Überwachen werden die Leistungen des Modells überprüft, um das Modell bei Abweichungen nachzutrainieren oder zu ersetzen [2].

## 4. Frontend

Die Benutzeroberfläche für die Bildsegmentierung ist als Webapplikation realisiert. Diese ermöglicht einem Benutzer ein Bild in die Webseite zu laden und dort mittels Polygonen zu segmentieren. Um den Benutzer dabei zu unterstützen wird das Bild gleichzeitig an das Backend geschickt, welches einen Segmentierungsvorschlag ausarbeitet. Dieser Vorschlag wird dem Benutzer zur Weiterbearbeitung präsentiert. Ist dieser mit dem Resultat zufrieden, kann er die Segmentierung als JSON exportieren. Falls das Backend nicht erreichbar ist, schaltet die Applikation in einen Offline-modus. Auf dies wird mit einer entsprechenden Meldung hingewiesen. In diesem Modus können Bilder annotiert werden, die Unterstützung durch eine Segmentierungs-AI und Kommunikation zum Backend sind jedoch deaktiviert. Die Applikation unterstützt keine Datenverwaltung, dies liegt in der Verantwortung des Nutzers. Das System ist für einzelne und kurze Arbeitssessionen gedacht und nicht für das Segmentation von ganzen Datensätzen.

Die Webapplikation ist als eine React-Funktionskomponente implementiert. Dieser Ansatz wird für neu entwickelte React-Anwendungen empfohlen. Dabei wird auf einen Komponentenbaum mit gerichtetem Datenfluss gesetzt. Die Daten werden an Kindkomponenten weitergeleitet, die wiederum über Callbacks entsprechende Events in den Elternkomponenten auslösen. Über die Events können Nutzeraktionen abgebildet werden, wie zum Beispiel das Hochladen eines Bildes. Durch den gerichteten Informationsfluss kann React das Rendering optimieren und dadurch eine gute Performance erzielen. Es werden nur die Stellen der Webseite neu gerendert, wo sich die Informationen verändert haben [3].

Die Umsetzung und der gerichtete Informationsfluss werden im folgenden Abschnitt thematisiert. Ebenfalls wird die Bedienoberfläche beschrieben und wie das System in eine Webseite integriert werden kann. Dieses Kapitel wird mit den verwendeten Technologien abgeschlossen.

### 4.1. Umsetzung

Die Webapplikation besteht aus Funktionskomponenten in einer Baumstruktur. Jede dieser Komponenten gibt ein JSX (JavaScript XML) zurück. Diese werden in das DOM (Document Object Model) integriert und so die eigentliche Webseite aufbaut. Da Funktionen grundsätzlich keine persistenten Daten haben, ist dabei der Informationsfluss ein wichtiges Grundkonzept.

In Abbildung 8 wird der grobe Aufbau der Implementation gezeigt. Für die Darstellung des Aufbaus mit Funktionskomponenten wird ein angepasstes Klassendiagramm verwendet. Im mittleren Feld der Funktionskomponenten sind die jeweiligen Funktionsparameter gelistet. Das untere Feld beschreibt den Zweck und die Verantwortlichkeit der jeweiligen Funktionskomponente.

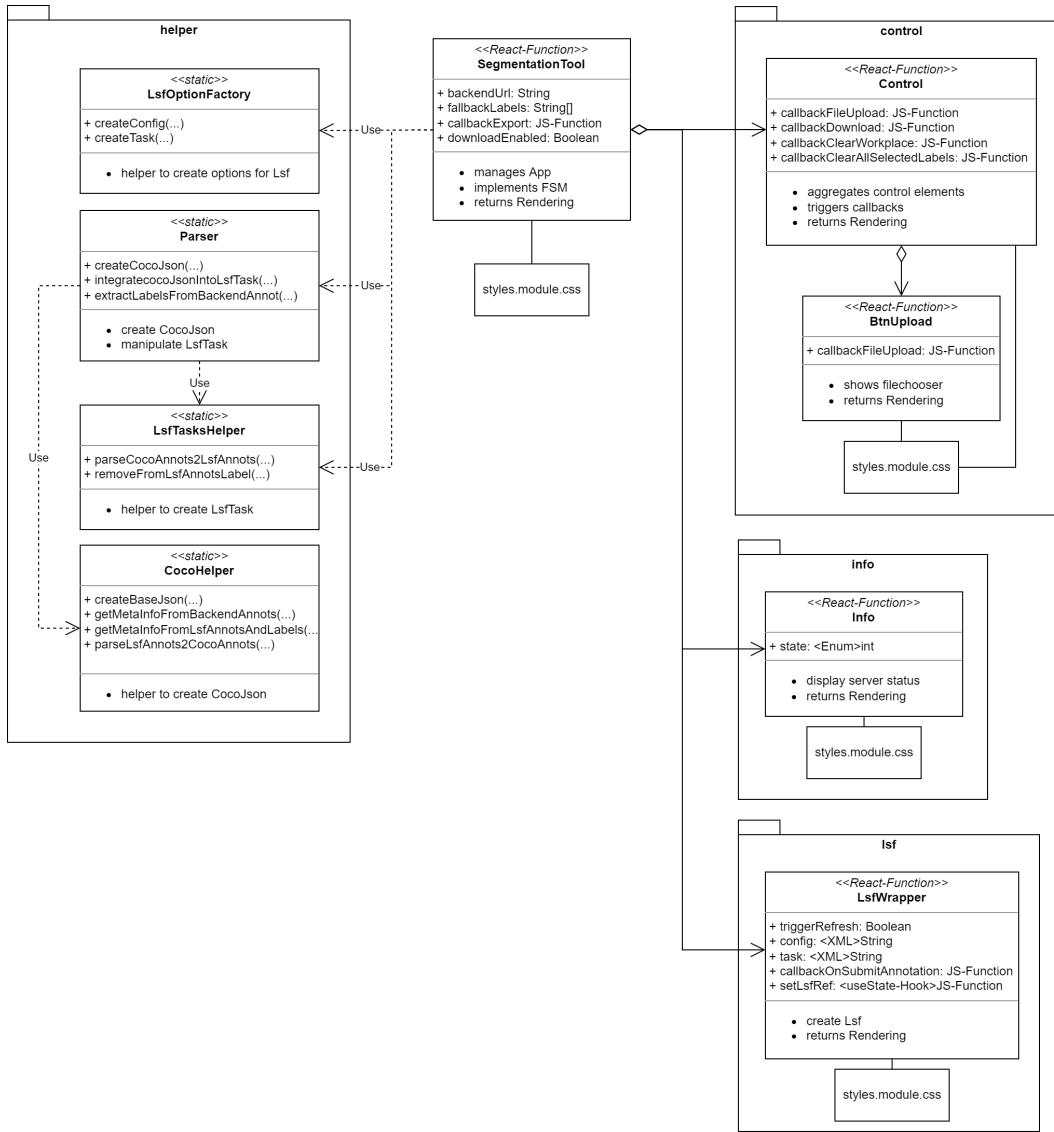


Abbildung 8: Übersicht über den Aufbau des Frontends.

Das Hauptelement ist das SegmentationTool. Dieses baut die App auf und steuert deren Verhalten während der Segmentation. Dazu wird eine FSM (Finite State Machine) genutzt. Hilfsfunktionen sind in statische Klassen ausgelagert worden. Ihre Hauptaufgabe ist das Übersetzen zwischen verschiedenen Annotierungsformaten und die Erstellung von Konfigurationen für das LSF (Label-Studio-Frontend). Über diese Konfigurationen wird die visuelle Präsentation der Segmentierung gesteuert. Im 'LsfWrapper' werden die Konfigurationen genutzt, um eine LSF-Instanz zu erzeugen. Diese ermöglicht das grafische Bearbeiten der Segmentierung.

#### 4.1.1. Informationsfluss

Ein Nachteil des funktionellen Ansatzes sind die flüchtigen Daten. Um Funktionskomponenten mit persistenten Daten zu erweitern, wird die Hook-API von React genutzt. Über Hook können Life-Cycles nachgebildet oder bestimmte Daten über mehrere Renderzyklen persistent gehalten werden. Die persistenten Daten werden fortan als 'State' bezeichnet. In Abbildung 9 wird der Informationsfluss des Frontends dargestellt. Dem 'SegmentationTool' wird beim Einbinden in eine Webseite die URL des Backends mitgegeben. Außerdem werden hier Labels definiert, auf welche das Frontend zurückfallen soll, wenn das Backend nicht erreicht werden kann. Über 'downloadEnabled' und 'callbackExport' kann gesteuert werden, was passiert, falls ein Benutzer die Segmentation exportiert. In der Komponente 'Control' werden alle Kontrollelemente zusammengefasst und bilden dadurch eine Bedienfläche. Informationen und Anweisungen für den Benutzer werden mittels 'Info' dargestellt. Für die interaktive Bearbeitung der Segmentierung durch Polygone ist 'LsfWrapper' verantwortlich. Diese Komponente initialisiert die Arbeitsfläche und stellt Funktionen des LSF dem 'Segmentationtool' zur Verfügung. [4].

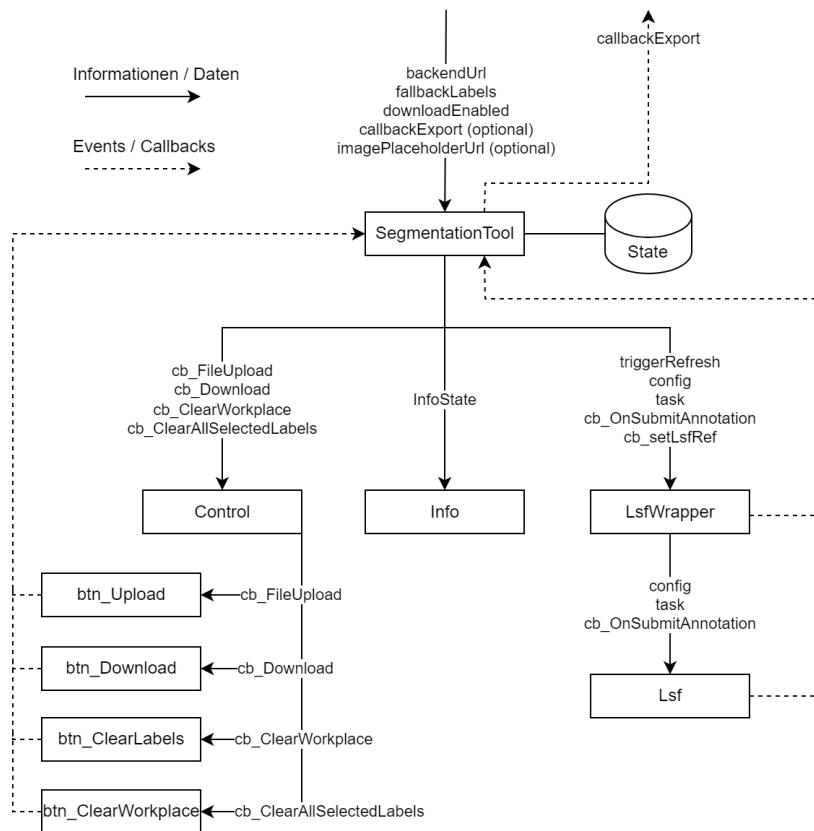


Abbildung 9: Der Informationsfluss innerhalb des Frontends.

Der Präfix 'cb\_...' steht für Callback. Dies sind Funktionen, mit welchen Events in den Elternkomponenten ausgelöst werden. Elemente welche der Benutzer durch Klicken aktivieren kann, sind mit 'btn\_...' für Button gekennzeichnet.

#### 4.1.2. Finite State Machine

Die Komponente 'SegmentationTool' implementiert eine FSM. Diese steuert das Verhalten der Applikation während der Nutzung. Die FSM ist in Abbildung 10 dargestellt. Sie verfügt über zwei Hauptpfade, welche sich in einer Schleife befinden. Die Pfade sind in einen offline und online Pfad aufgeteilt. Ist das Backend nicht verfügbar oder kommt es zu einem Kommunikationsfehler, schaltet das Tool in den offline Modus. In diesem Modus ist die Kommunikation zum Backend deaktiviert und somit auch die Unterstützung durch das neurale Netzwerk. Eine manuelle Nutzung ist immer noch möglich und der aktuelle Arbeitsstand wird übernommen.

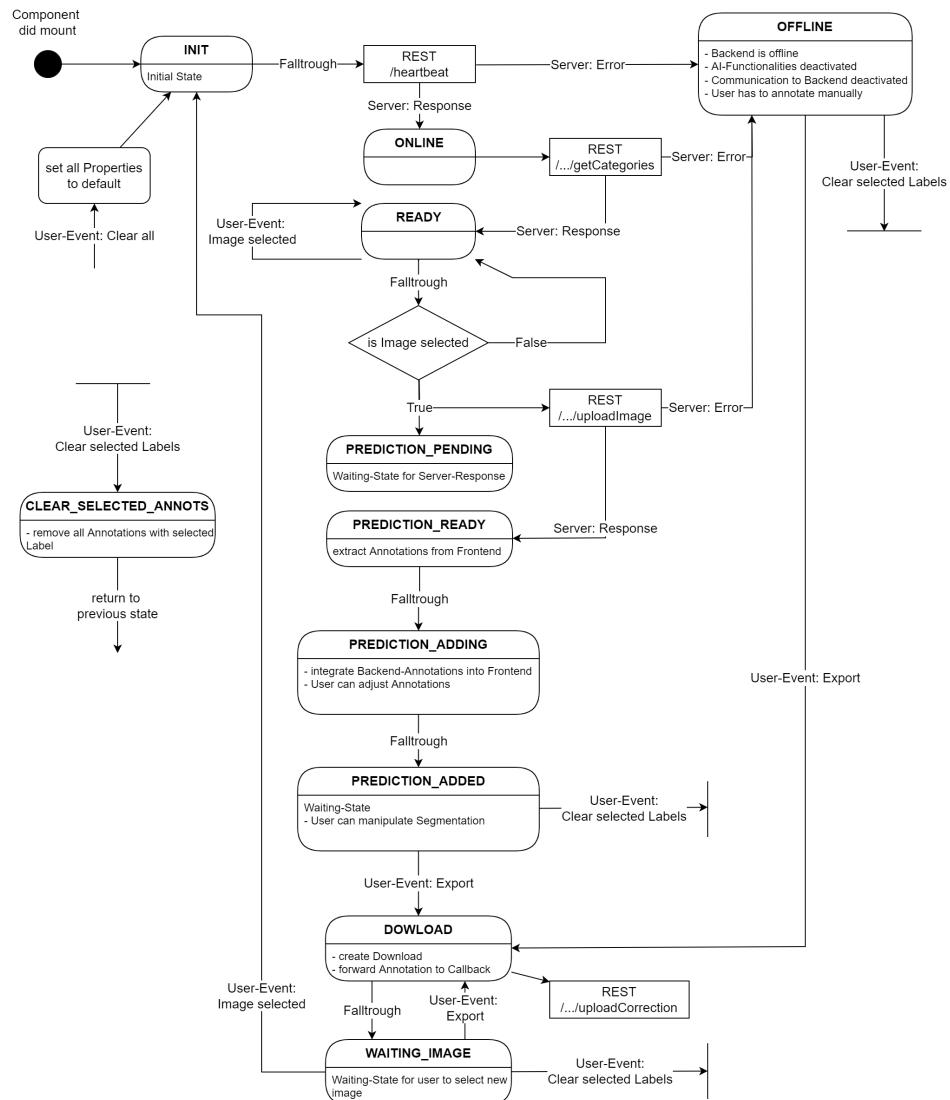


Abbildung 10: Aufbau der genutzten State-Machine im Frontend.

Beim Laden der Webapplikation befindet sich die FSM im INIT-State. Bei jedem abgeschlossenen Segmentierungsvorgang befindet es sich wieder in diesem State. Von diesem State aus wird über eine REST-Schnittstelle ein Heartbeat vom Backend angefordert. Durch diesen wird überprüft, ob das Backend verfügbar und bereit ist. Diese Anfrage ist mit einem Timeout ausgestattet, so

dass bei einem überlasteten Backend in den offline Modus gewechselt werden kann. In den States 'ONLINE' bis 'PREDICTION\_ADDING' befindet sich das Tool nur für kurze Zeit. Diese dienen der Vorbereitung, Anforderung und Integration eines Segmentierungsvorschlags. Während dieser Zeit kann ein Benutzer das Tool bedienen und eingezeichnete Segmentierungen werden bei einem Statewechsel beibehalten. Der State 'PREDICTION\_ADDED' dient dem Benutzer dazu die Segmentierungen anzupassen und zu verfeinern. Der Übergang zum nächsten State wird durch das Benutzer-Event 'Export' ausgelöst. Anschliessend wartet das Tool im State 'WAITING\_IMAGE' auf ein neues Bild. Hier können immer noch Anpassungen an der Segmentierung vorgenommen und bei Bedarf erneut exportiert werden. Die letzte exportierte Segmentierung wird dabei dem Backend geschickt und als Trainingsdaten abgelegt.

In Abbildung 11 ist der optimale Arbeitsablauf dargestellt. Dieser Beginnt mit dem Hochladen eines Bildes in das 'SegmentationTool' und endet mit dem Export der Segmentierung. Die Kommunikation mit dem Backend ist dabei asynchron, so dass der Benutzer beim Arbeiten nicht unterbrochen wird. Die Segmentierungsvorschläge des Backends werden nahtlos in die bestehende Segmentierung integriert.

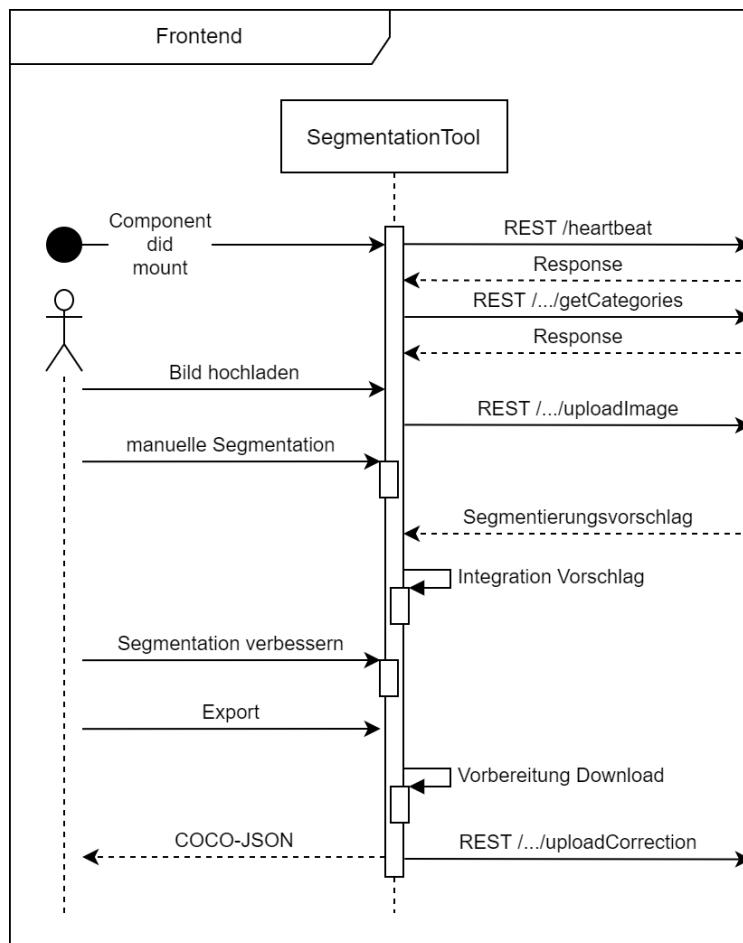


Abbildung 11: Typischer Ablauf eines Segmentierungsauftrages aus Sicht des Frontends.

## 4.2. Bedienoberfläche

In Abbildung 12 ist die grafische Benutzeroberfläche abgebildet. Diese soll dem Benutzer eine schnelle Segmentierung ohne grosse Einarbeitungszeit ermöglichen. Zu diesem Zweck ist die Oberfläche thematisch in einzelne Bereiche unterteilt.

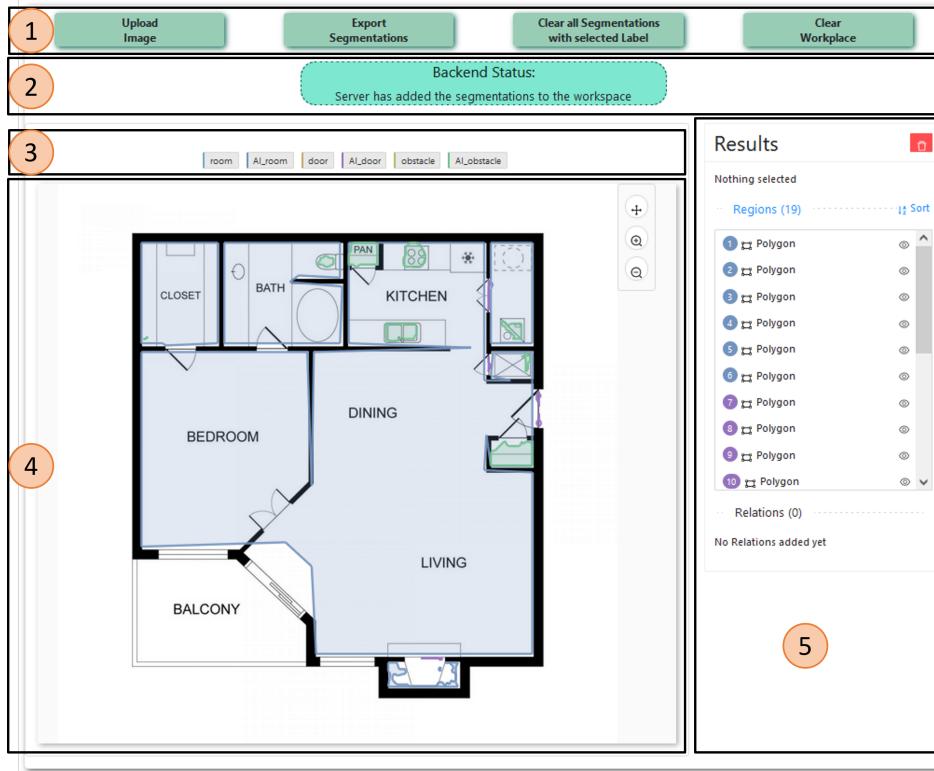


Abbildung 12: Grafische Oberfläche mit markierten Bereichen. Nummeriert sind (1) Kontrollelemente, (2) Informationsbereich, (3) Verfügbare Labels, (4) Arbeitsfläche und (5) Übersicht der Annotierungen.

### 4.2.1. Kontrollelemente

Der in Abbildung 12 mit (1) gekennzeichnete Bereich beinhaltet die Kontrollelemente. Diese dienen dem Benutzer zur Steuerung des Arbeitsablaufes und Zurücksetzen des Frontends.

- Upload Image

Dieses Element öffnet ein Dateiauswahldialog. Der Benutzer kann in diesem das Bild für die Segmentierung auswählen. Alternativ steht dem Benutzer auch eine Drag-And-Drop Funktionalität zur Verfügung, die es ihm ermöglicht, ein Bild mit der Maus in die Arbeitsfläche zu ziehen. Dies führt zum gleichen Ergebnis wie der Weg über den Auswahldialog.

- Export Segmentation

Ist der Benutzer zufrieden mit der Segmentierung des Bildes, kann er über dieses Element die Segmentierung als COCO-JSON exportieren. Bei der Einbindung des Frontends als Element in eine Webseite kann bestimmt werden, ob die Segmentierung heruntergeladen und/oder über einen Callback an die implementierende Webseite weitergeleitet wird.

- Clear all Annotations with selected Label

Es werden alle Annotationen des selektierten Typs entfernt. Dadurch ist es dem Benutzer möglich eine bestimmte Kategorie unkompliziert zu entfernen. Dies ist hilfreich, um einen Annotierungsvorschlag schnell wieder zu entfernen.

- Clear Workplace

Mit diesem Element kann das Frontend wieder auf den Anfangszustand gesetzt werden. Dabei werden das hochgeladene Bild und die bestehenden Annotationen verworfen.

#### 4.2.2. Informationsbereich

In der Abbildung 12 mit (2) gekennzeichneten Bereich befinden sich Informationen zum Verbindungszustand zum Backend. Es wird angezeigt, ob das Backend beschäftigt ist oder welcher Arbeitsschritt als nächstes angedacht wäre. Die möglichen States werden in Abbildung 13 abgebildet.

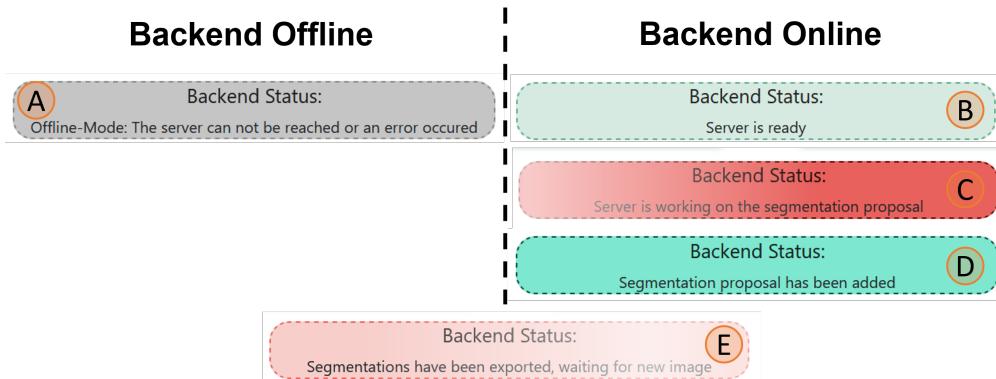


Abbildung 13: Die verschiedenen Informationsanzeigen für den Benutzer.

- Das Backend ist offline oder eine REST-Anfrage hat zu einer Fehlermeldung geführt. Die Unterstützung durch die Segmentierungs-AI ist deaktiviert. Ein Segmentierungsauftrag kann gestartet werden, jedoch muss manuell gearbeitet werden. Wird das Frontend neu geladen oder zurückgesetzt, wird erneut versucht eine Verbindung mit dem Backend aufzubauen.
- Das Backend ist online und das Frontend ist bereit für die Auswahl eines Bildes. In diesem State wartet das Frontend auf einen Segmentierungsauftrag. Dieser wird durch eine Bildauswahl gestartet.
- Das Backend arbeitet an einem Segmentierungsvorschlag. Es kann parallel mit der manuellen Segmentierung gestartet werden. Ein leichter Schimmereffekt macht auf diesen State aufmerksam.
- Der Segmentierungsvorschlag wurde in das Frontend integriert. Schon bestehende Segmente werden beibehalten. Der Benutzer kann den Vorschlag überarbeiten.
- Die Segmentierung wurde exportiert und es kann ein weiteres Bild ausgewählt werden. In diesem State wartet das Frontend auf einen neuen Segmentierungsauftrag, ähnlich wie der State (B). Ein leichter Schimmereffekt macht auf diesen State aufmerksam.

#### 4.2.3. Verfügbare Labels

In dem Abbildung 12 mit (3) gekennzeichneten Bereich werden die Labels angezeigt, welche zur Verfügung stehen. Diese werden über eine REST-Schnittstelle vom Backend angefordert. Ist dieses nicht erreichbar, wird auf vordefinierte Labels zurückgegriffen. Die Segmentierungsvorschläge von der Segmentierungs-AI werden mit dem Präfix 'AI\_' gekennzeichnet. Dieser Präfix wird nicht exportiert und dient nur der Differenzierung zwischen Segmentierungsvorschlag und manueller Segmentierung.

#### 4.2.4. Arbeitsfläche

Der in Abbildung 12 mit (4) gekennzeichnete Bereich beinhaltet die Arbeitsfläche der eigentlichen Segmentierung. Die Segmentierung wird mittels Polygonen über die unterschiedlichen Labels visualisiert. Diese können verschoben oder über Eckpunkte in der Form angepasst werden. Für die Realisierung dieses Bereiches wurde das Label-Studio-Frontend Framework verwendet.

#### 4.2.5. Übersicht Annotierungen

Der in Abbildung 12 mit (5) gekennzeichnete Bereich beinhaltet die einzelnen Polygone. Über ein Mülleimer-Symbol können einzelne Polygone gezielt gelöscht werden.

### 4.3. Integrationsbeispiel

Die Integration des Frontends in eine bestehende React-Webapplikation ist über wenige Zeilen Code realisierbar. Wie dies in der folgenden Abbildung 14 ersichtlich ist.

---

```
1 import SegmentationTool from "SegmentationTool";
2
3 ... weiterer React-Code ...
4
5 <SegmentationTool
6   backendUrl={"http://127.0.0.1:8000/"}
7   fallbackLabels={[ "label_1", "label_2"] }
8   downloadEnabled={true}
9   callbackExport={undefined}
10  imagePlaceholderUrl={undefined}
11 />
```

---

Abbildung 14: Beispielcode für die Integration in eine React-Applikation.

Erklärungen zum in Abbildung 14 gezeigten Code-Ausschnitt und dessen Parametern.

- Zeile 1: Import des Frontends.
- Zeile 5: Erstellen der React Komponente.
- Zeile 6: Verknüpfung zum Backend über eine URL.
- Zeile 7: Falls das Backend nicht verfügbar ist, wird auf diese Labels für die Segmentierung zurückgegriffen.
- Zeile 8: Einstellen, ob die Segmentation als COCO-JSON heruntergeladen werden kann.
- Zeile 9: Übergabe einer Funktion als Callback. Diese wird beim Exportieren aufgerufen und die Segmentation als COCO-JSON übergeben. Dieser Parameter ist optional.
- Zeile 10: Es kann eine URL zu einem Bild übergeben werden. Dieses wird anschliessend als Platzhalter verwendet, solange kein Bild zur Segmentation ausgewählt wurde. Dieser Parameter ist optional.

## 4.4. Verwendete Technologien

Für das Frontend werden mehrere Technologien genutzt. Die wichtigsten werden in den folgenden Abschnitten vorgestellt.

### 4.4.1. COCO-JSON

Bei der Formatierung der Segmentierung wird auf das COCO-Format (Common Objects in Context) gesetzt. Dieses definiert wie eine Bildsegmentierung in einem JSON (JavaScript Object Notation) abgelegt wird. Es handelt sich dabei um ein Format, welches durch den gleichnamigen Datensatz populär gemacht wurde. In Abbildung 15 ist die Definition des Formates ersichtlich. Die Abbildung 16 zeigt, wie das COCO-Format für diese Arbeit verwendet wird [5].

```

1  {
2    "info" :info,
3    "images" :[image],
4    "annotations" :[annotation],
5    "categories" :[category],
6    "licenses" :[license],
7  }
```

Abbildung 15: Der grundlegende Aufbau eines COCO-JSON.

Über dieses Format können in "info" Informationen zum Datensatz definiert werden. Bilder werden als Array in "images" und Segmentierungen in "annotations", ebenfalls als Array, gespeichert. Kategorien und Lizenzen werden in den entsprechenden Feldern abgelegt. Der genaue Aufbau des COCO-JSON ist im Anhang F ersichtlich. In dieser Arbeit wird pro Segmentierungsauftrag ein COCO-JSON generiert. Diese könnten später auch zu einem JSON zusammengefasst werden, was durch den Aufbau des Formates ohne Schwierigkeiten möglich ist [6].

```
1  {
2      "info": {
3          "description": "Human-in-the-loop Image-Segmentation",
4          "ModelVersion": "1"
5      },
6      "categories": [
7          {
8              "id": 0,
9              "name": "room"
10         },
11         ...
12     ],
13     "images": [
14         {
15             "id": 0,
16             "file_name": "grundriss1.jpg",
17             "height": 377,
18             "width": 474
19         }
20     ],
21     "annotations": [
22         {
23             "id": 0,
24             "image_id": 0,
25             "category_id": 0,
26             "segmentation": [
27                 4,
28                 315,
29                 10,
30                 437,
31                 ...
32             ]
33         },
34         ...
35     ]
36 }
```

Abbildung 16: Ein in dieser Arbeit eingesetztes COCO-JSON.

Für diese Arbeit wurde das Format den Bedürfnissen entsprechend angepasst. Auf einige Angaben, wie zum Beispiel die Lizizen, wurde verzichtet. Bei anderen Punkten wurden Informationen hinzugefügt, zum Beispiel die Modelversion des neuronalen Netzwerks. Für jedes Polygon wird ein separater Eintrag in 'annotations' generiert. Die einzelnen Punkte des Polygones sind im Unterbereich 'segmentation' abgelegt. Die X- und Y-Koordinate der Punkte sind abwechslungsweise in einem Array eingetragen.

#### 4.4.2. React

React ist ein weit verbreitetes JavaScript-Framework, welches zum Entwickeln von performanten Webapplikationen eingesetzt wird. In der Regel wird es verwendet, um Single-Page Applikationen zu erstellen. Es nutzt das Konzept des Einwegdatenflusses. Dies sorgt für stabilen Code und performante Webseiten. Durch diesen und einen virtuellen DOM kann React die Webseite optimieren, sodass nur aktualisierte Bereiche der Webseite vom Browser neu geladen werden müssen. Das Framework wurde ursprünglich von und für Facebook entwickelt. Später wurde es unter einer Open-Source Lizenz allen Entwicklern zur Verfügung gestellt. React-Code kompiliert zu einer JavaScript-Datei, welche von einem Browser interpretiert werden kann. Dadurch ist es möglich, kompilierten React-Code in einer traditionellen HTML-Webseite (Hypertext Markup Language) einzubinden [3].

#### 4.4.3. Label-Studio-Frontend

Für die Darstellung und Bearbeitung der Segmentierungspolygone wird das LSF eingesetzt. Dieses React-Package ist das Frontend des Label-Studios. Dieses Daten-Annotierungs-Tool richtet sich an Projekte im Bereich Data-Science und bietet eine Komplettlösung für entsprechende Aufgaben. Das Frontend kann aber unabhängig vom restlichen Label-Studio verwendet werden. Durch den flexiblen Aufbau wurde es auf dieses Projekt angepasst und ermöglicht dadurch eine einfache grafische Bearbeitung der Polygone [7, 8].

#### 4.4.4. Axios

Für die Nutzung der REST-API des Backends wird das React-Package Axios verwendet. Es handelt sich dabei um einen promisebasierten HTTP-Client. Ein weiterer Vorteil ist die automatische Transformation der JSON-Daten vom Server [3, 9].

#### 4.4.5. CSS-Module

Für die Steuerung der visuellen Darstellung werden CSS-Modules eingesetzt. Gegenüber reinem CSS (Cascading Style Sheet) ist der Hauptunterschied, dass die Modules nur für die jeweilige React-Komponente gültig sind. Dadurch wird der Scope des CSS auf die Komponenten beschränkt, welche das CSS-Module nutzen. Dadurch können Konflikte mit Webseiten und deren CSS vorgebeugt werden. Solche Konflikte können entstehen, falls von mehreren Stellen CSS mit gleichen Identifikatoren genutzt werden. [3, 10].

## 5. Backend

Das Backend-Modul bietet eine REST-API und eine Anbindung zur Image-Segmentierung. Über die REST-API werden Anfragen angenommen, welche mit der Image-Segmentierung bearbeitet werden. Für die Image-Segmentierung bietet sich die Programmiersprache Python an, da für diese umfangreiche Bibliotheken im Bereich Datenverarbeitung zur Verfügung stehen. Um diese Schnittstelle möglichst einfach zu halten, ist auch der Backendserver in Python implementiert. Für die REST-API wurde das FastAPI-Framework verwendet. Dieses nimmt Einfluss auf die Architektur, welche in der folgenden Abbildung 17 ersichtlich ist [11, 12].

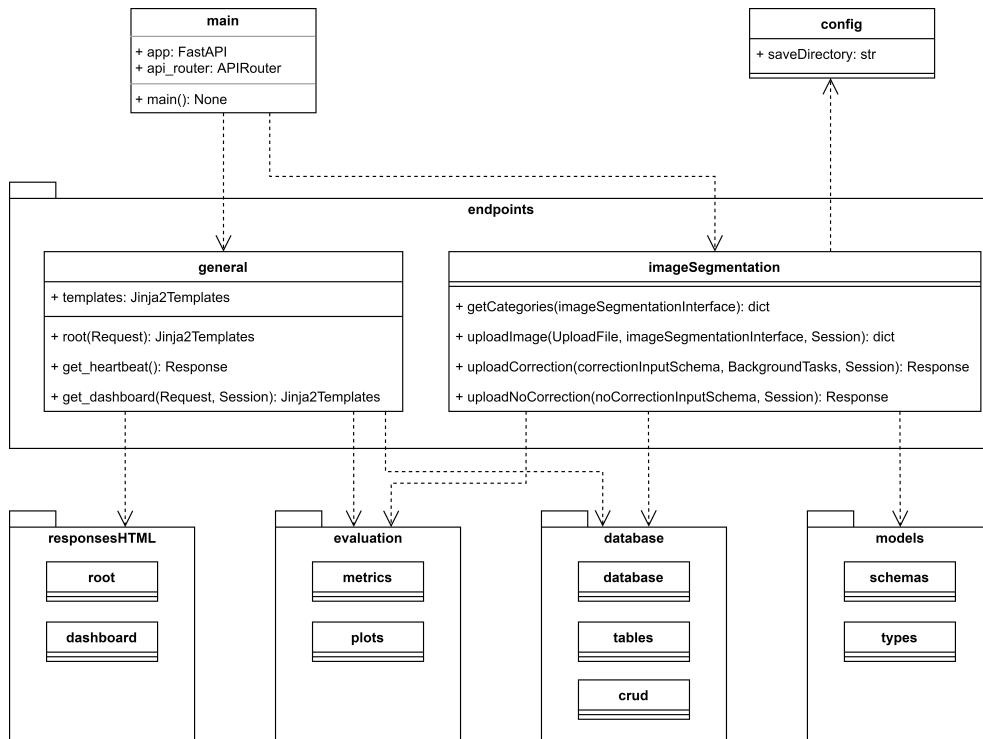


Abbildung 17: Klassendiagramm des Backends.

Der Einstiegspunkt des Backends ist in der `main`-Klasse zu finden. Diese konfiguriert den Server und die Middleware. Im Verzeichnis 'endpoints' sind die Endpunkte der REST-API definiert. Die Anfragen und Antworten dieser Endpunkte werden im Verzeichnis 'models' definiert und überprüft. Das Backend besitzt auch eine Homepage und ein Dashboard für die Darstellung der Performance der Image-Segmentierung. Bei diesen Endpunkten wird im HTML-Format geantwortet, welche im Verzeichnis 'responsesHTML' definiert sind. Für die Plots und Berechnungen der Metriken, welche im Dashboard dargestellt werden, stehen im Verzeichnis 'evaluation' einige Hilfsfunktionen bereit. Bei einem Segmentierungsauftrag wird das angefragte Bild, die geantwortete Segmentierung und die korrigierte Segmentierung in einem lokalen Verzeichnis gespeichert. Die Speicherorte, wie auch weitere Informationen zum Segmentierungsauftrag, werden in einer SQL-Datenbank gespeichert. Die Datenbankanbindung ist im Verzeichnis 'database' definiert und der lokale Speicherpfad der Dateien wird in der `config`-Klasse festgelegt. In den folgenden Abschnitten werden die einzelnen Komponenten im Detail beschrieben.

## 5.1. REST-API

REST (Representational State Transfer) ist ein Web-Architekturstil, welcher beschreibt, wie das Web funktionieren sollte. Dazu definiert REST eine einheitliche Kommunikation, welche zustandslos funktioniert. Somit werden auf dem Server keine Daten vom Client gespeichert und alle nötigen Informationen werden in die Kommunikation integriert. Dies ermöglicht eine hohe Skalierbarkeit vom Dienst, da der Client nicht an einen Server gebunden ist. Die Kommunikation in REST wird mit den Methoden 'GET' für die Anfrage von Ressourcen, 'POST' für das Erstellen von Ressourcen, 'PUT' für das Update von Ressourcen und 'DELETE' für das Löschen von Ressourcen beschrieben. Ein Webservice, welcher die REST Architektur umsetzt, wird REST-API genannt und ist RESTful [13].

### 5.1.1. FastAPI

FastAPI ist ein Web-Framework, welches es ermöglicht eine REST-API in Python zu implementieren. Dabei zeichnet sich dieses Framework durch seine Einsteigerfreundlichkeit, automatische Dokumentation und hohe Performance aus. Die automatische Dokumentation wird über den Backendserver zur Verfügung gestellt und ist unter der Route './docs' zu finden. Diese beschreibt die verwendeten Schemas, welche benutzt werden, um die Kommunikation für die verschiedenen Endpunkte zu definieren. Die Routen für die einzelnen Endpunkte sind ebenfalls ersichtlich und es können einzelne Anfragen ausgeführt werden. Die hohe Performance wird durch die Benutzung des Starlett-Frameworks und der Pydantic-Bibliothek erreicht. Dabei verwendet FastAPI das Starlett-Framework für den Webserver. Dieses ermöglicht die asynchrone Bearbeitung von Aufträgen. Mit Pydantic können Schemas definiert werden, welche eine effiziente Datenprüfung durchführen [14].

### 5.1.2. Endpunkte

Die Endpunkte der REST-API stellen Service zu Verfügung, welche über das Internet erreicht werden können. Eine Übersicht aller Endpunkte und deren Routen ist in der folgenden Abbildung 18 zu sehen.

General		
GET	/ Root	^
GET	/heartbeat Get Heartbeat	^
GET	/dashboard Get Dashboard	^
Image Segmentation		
GET	/ImageSegmentation/getCategories Get Categories	^
POST	/ImageSegmentation/uploadImage Upload Image For Segmentation	^
POST	/ImageSegmentation/uploadCorrection Upload Correction	^
POST	/ImageSegmentation/uploadNoCorrection Upload No Correction	^

Abbildung 18: Endpunkte der REST-API.

Die Endpunkte, welche zur Kategorie 'General' gehören, werden für die Zustandsbeschreibung des Backends verwendet. Bei der Kategorie 'Image Segmentation' handelt es sich um Endpunkte, welche für die Bild-Segmentierung benötigt werden. Eine vollständige Auflistung der Anforderungs- und Antwortschemas, welche die Benutzung des jeweiligen Endpunkts definiert, ist im Anhang E zu finden.

## 5.2. Middleware

Eine Middleware fungiert als versteckte Übersetzungsebene zwischen einem Betriebssystem und den Anwendungen, welche darauf ausgeführt werden. Somit wird die Datenverwaltung und Kommunikation verteilter Anwendungen ermöglicht. Einer FastAPI-Applikation können mehrere Middleware-Schichten hinzugefügt werden. Diese werden jeweils vor einer Anfrage auf einen Endpunkt und auf die Antwort vom Endpunkt angewendet. In der folgender Abbildung 19 wird dies dargestellt [15, 16].

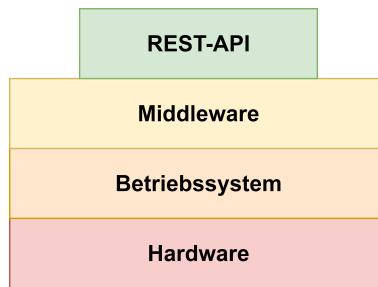


Abbildung 19: Middleware-Schicht innerhalb des Backendservers. In Anlehnung an [17].

Die eingesetzte CORS-Middleware (Cross-Origin Resource Sharing) ermöglicht die Kommunikation zwischen unterschiedlichen Protokollen, Domänen und Ports. Diese Middleware konfiguriert die Sicherheitseinstellungen der Applikation. Es ist von Vorteil den Zugriff auf Ressourcen von unterschiedlichen Protokollen, Domänen und Ports zu verhindern. Trotzdem müssen bestimmte Kommunikationen ermöglicht werden, wie dies zwischen Frontend und Backend der Fall ist [18, 19].

## 5.3. Segmentierungsauftrag

Beim Human-in-the-loop Konzept kann auf einen Segmentierungsvorschlag mit einer Korrektur geantwortet werden, welche dazu verwendet wird, die Image-Segmentierung zu verbessern. Damit zu einem späteren Zeitpunkt eine Korrektur vom Frontend nachgeliefert werden kann, wird mit Tokens gearbeitet. Ein Token identifiziert einen Segmentierungsauftrag. Dieser wird mit dem Segmentierungsvorschlag vom Backend an das Frontend geschickt. Sobald der Nutzer seine Korrekturen an der Segmentierung vorgenommen hat, wird diese mit dem Token an das Backend übergeben. Somit kann die Korrektur dem entsprechendem Segmentierungsauftrag zugewiesen werden.

Eine Identifikation des Segmentierungsauftrags ist für die Umsetzung des Human-in-the-loop Konzepts nötig. Darüberhinaus kann somit auch die Performance der Image-Segmentierung verfolgt werden. Dazu werden während des Segmentierungsauftrags die notwendigen Daten im Backend gespeichert. Der Ablauf bei einer Anfrage für einen Segmentierungsvorschlag ist in der folgenden Abbildung 20 ersichtlich.

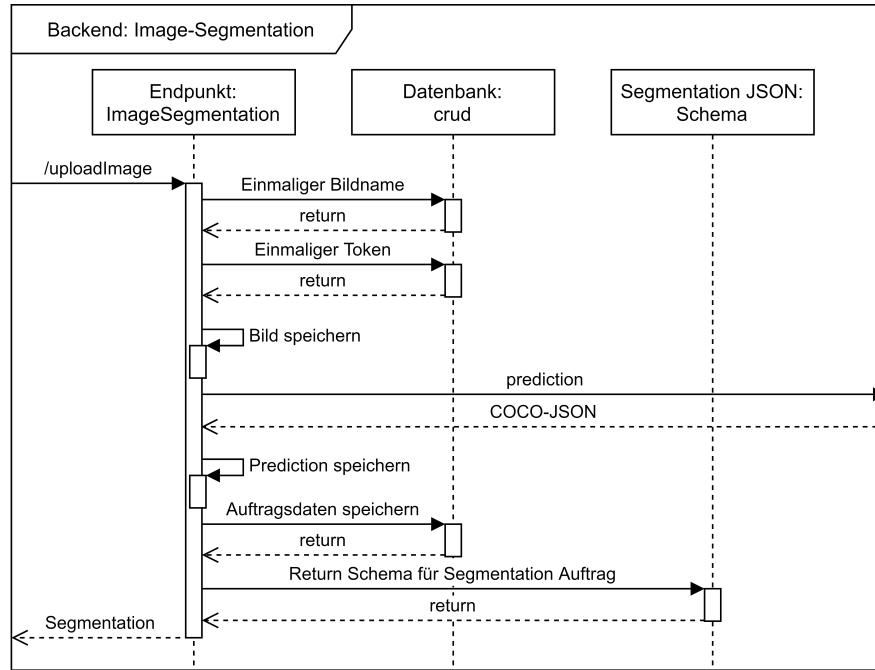


Abbildung 20: Sequenzdiagramm zum Ablauf einer Segmentierung im Backend.

Über den Endpunkt '/uploadImage' kann ein neuer Segmentierungsauftrag gestartet werden. Als Erstes wird für das hochgeladene Bild ein einmaliger Name und Token gesucht. Das Bild wird unter dem neuen Namen abgespeichert. Dann wird mit dem Image-Segmentation Modul eine Segmentierung erstellt. Diese wird unter demselben Namen wie das Bild gespeichert. Die vorgeschlagene Segmentierung wird benötigt, um in einem späteren Schritt die Metriken zu berechnen. Schlussendlich werden die Auftragsdaten gespeichert und die Segmentierung mit dem Token an das Frontend zurückgegeben.

Hier kommt der Mensch ins Spiel. Er kann die Segmentierung anpassen oder übernehmen. In beiden Fällen wird die Segmentierung mit dem Token an das Backend geschickt, um damit die Image-Segmentierung zu verbessern. Damit der Verbesserungsprozess möglichst autonom funktioniert, wird nach einer festgelegten Anzahl von korrigierten Segmentierungen das Nachtrainieren ausgelöst. Die einzelnen Schritte sind in der folgenden Abbildung 21 zu sehen.

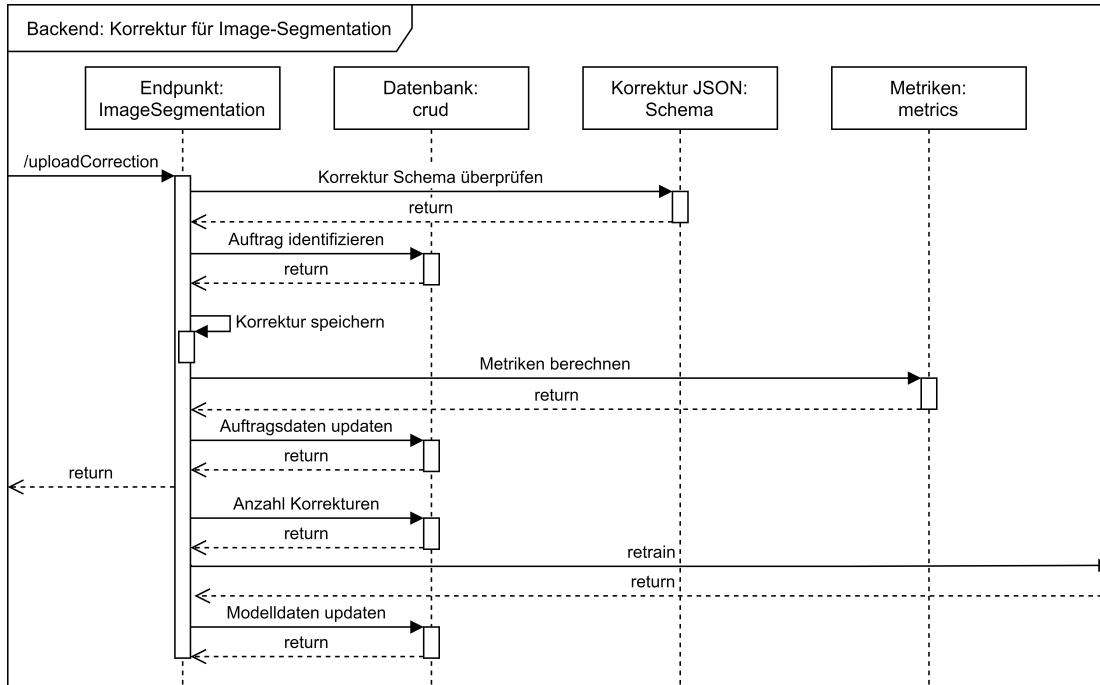


Abbildung 21: Sequenzdiagramm einer Segmentierungskorrektur im Backend.

Die korrigierte Segmentierung wird mit dem Token dem Endpunkt '/uploadCorrection' übergeben. Als Erstes wird überprüft, ob die Segmentierung im richtigen Format mit gültigem Inhalt vorliegt. Dann wird der Segmentierungsauftrag mit dem Token identifiziert. Falls ein gültiger Token vorliegt, wird die Korrektur gespeichert. Mit der gespeicherten Korrektur und der vorgeschlagenen Segmentierung werden dann die Metriken errechnet. Anschliessend werden die Auftragsdaten abgespeichert und der Segmentierungsauftrag abgeschlossen mit einer Rückmeldung an das Frontend. Jetzt wird geprüft, ob genügend Korrekturen vorliegen, um die Image-Segmentierung nachzutrainieren. Falls dies der Fall ist, werden die neusten Korrekturen und Bilder der Image-Segmentierung übergeben. Die Informationen des Nachtrainierens werden mit dem Namen der neuen Modelversion im Backend abgelegt, um diese im Dashboard darzustellen zu können.

Ein weiterer Fall wäre, dass der Benutzer die erhaltene Segmentierung weder anpasst noch übernimmt. Dies kann dem Backend mitgeteilt werden, somit wird der entsprechende Auftrag geschlossen. Zur Optimierung des Speicherplatzes wäre es in diesem Fall möglich die angelegten Daten der Segmentierungen zu löschen. Da diese Daten für einen Anwendungsfall wichtig sein könnten, wird zurzeit darauf verzichtet. Der Ablauf dieses Falls ist in der folgenden Abbildung 22 zu sehen.

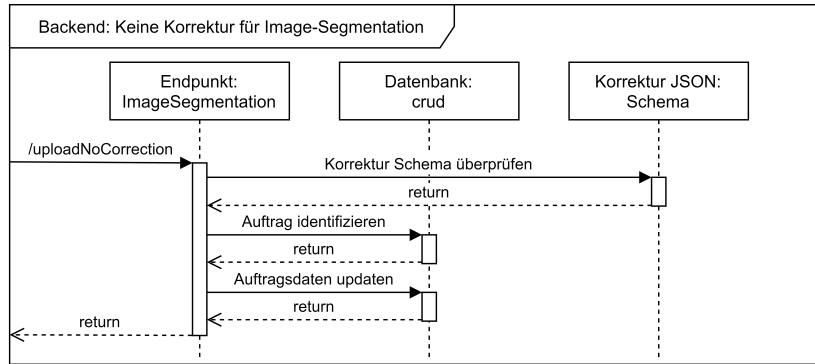


Abbildung 22: Sequenzdiagramm über den Auftragsabschluss ohne eine Segmentierungskorrektur im Backend.

Bei einem Abschluss eines Segmentierungsauftrags ohne Rückmeldung wird der Endpunkt 'uploadNoCorrection' aufgerufen. Es wird mit dem Token der Segmentierungsauftrag identifiziert und als abgeschlossen markiert.

## 5.4. Datenablage

Alle Daten, welche das Backend speichert, werden auf dem lokalen Dateisystem abgelegt. Somit wird kein externer Service benötigt. Dabei werden Bilder im erhaltenen Datenformat gespeichert, sowie Segmentierungen und Korrekturen im JSON-Format abgelegt. Diese Speicherorte und die restlichen anfallenden Daten werden in einer db-Datei mittels SQLite gespeichert. Dabei wurde eine relationale Datenbank gewählt, da die Daten in Beziehungen zueinander stehen und zu unterschiedlichen Zeiten ergänzt werden. Die Modellierung der Datenbank ist in der folgenden Abbildung 23 zu sehen [20].

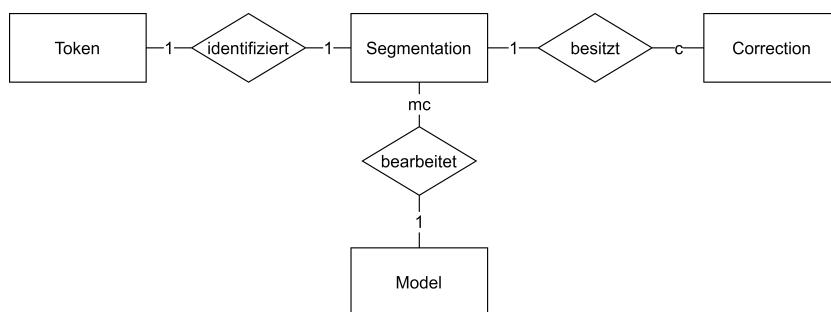


Abbildung 23: Entity-Relationship-Modell der Datenbank im Backend.

Es gibt für jede Segmentierung genau einen Token, da diese eindeutig identifizierbar sein muss. Die Segmentierung wurde von einem künstlichen neuronalen Netzwerk durchgeführt, wobei ein solches Modell mehrere Segmentierungen erstellen kann. Zu einem späteren Zeitpunkt kann eine Korrektur für eine Segmentierung nachgeliefert werden, wobei eine Segmentierung nur eine oder keine Korrektur besitzen kann. Anhand dieser Beziehungen wurde das folgende Datenbankschema in Abbildung 24 umgesetzt.

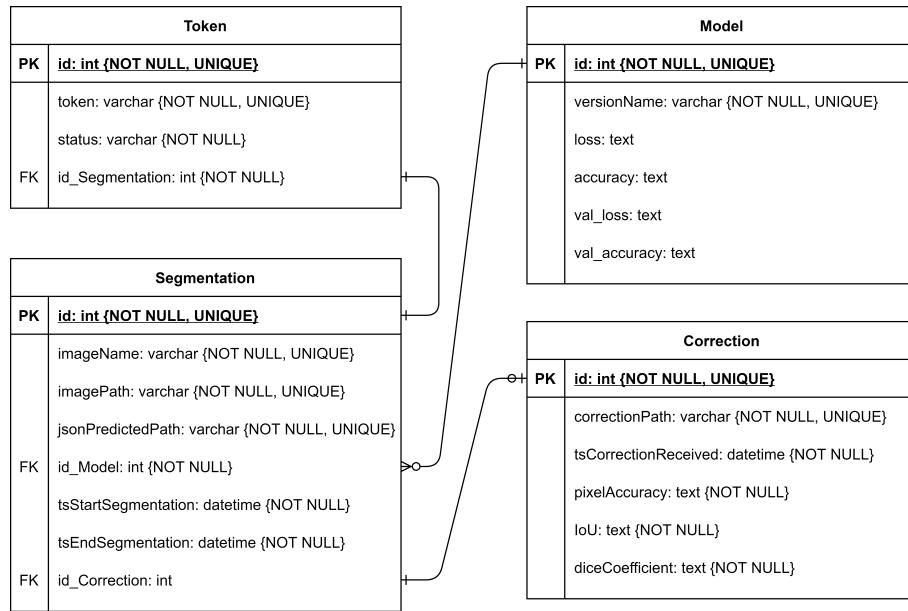


Abbildung 24: Datenbankschema der Datenbank im Backend.

In der Tabelle 'Token' wird der Token und der Status des Segmentierungsauftrags abgelegt. Der Status gibt an, ob noch eine Korrektur für die Segmentierung erwartet wird.

Die Tabelle 'Segmentation' besitzt die Speicherorte zum Bild und der Segmentierung. Darüberhinaus wird die benötigte Zeit für die Segmentierung mit einem Start- und Endzeitpunkt gespeichert. Bei der Tabelle 'Model' handelt es sich um die Daten der Image-Segmentierung. Es wird der Name der Modellversion, sowie die Werte der Lossfunktion und der Accuracy während dem Nachtrainieren gespeichert.

Wenn eine Korrektur für eine Segmentierung bereit steht, werden diese Werte in die Tabelle 'Correction' geschrieben. Dort werden der Speicherort der Korrektur, der Zeitpunkt der Korrektur und die Metriken, welche für die Bewertung der Image-Segmentierung verwendet werden, abgelegt. Die Übersetzung der Pythonobjekte in die relationale Datenbank übernimmt SQLAlchemy. Diese ORM (Object Relational Mapper) kann mit den Pydantic Schemas umgesetzt werden, was die Handhabung mit den Daten vereinfacht [21].

## 5.5. Evaluierung der Image-Segmentation

Um die Performance des Image-Segmentation Moduls zu evaluieren wurden verschiedene Metriken eingesetzt und über ein Dashboard visualisiert. Dies wird in den folgenden Abschnitten behandelt.

### 5.5.1. Metriken

Bei den Metriken für die Performance Bewertung handelt es sich um die Accuracy, die IoU (Intersection-over-Union) und den Dice-Coefficient. Diese haben sich im Bereich der Segmentierung bewährt und werden häufig genutzt [22].

Bei der Accuracy handelt es sich um den prozentualen Anteil richtig erkannter Pixel. Um diese Metrik zu berechnen, wird die Anzahl richtig erkannter Pixel durch die gesamte Anzahl Pixel geteilt. Die richtig erkannten Pixel bestehen aus den richtig gesetzten (TP: True Positive) und den richtig nicht gesetzten Pixel (TN: True Negative). Die gesamten Pixel schliessen jedoch noch die falsch gesetzten (FP: False Positive) und die falsch nicht gesetzten Pixel (FN: False Negative) mit ein. In der folgenden Formel (1) ist die Berechnung der Accuracy zu sehen [22].

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP} \quad (1)$$

Um die richtig erkannte Flächendeckung zu beurteilen, wird die IoU eingesetzt. Diese errechnet die Überschneidung der vorgeschlagenen Fläche mit der korrekten Fläche geteilt durch die aufgespannte Fläche der beiden. Dies ist in der folgenden Abbildung 25 ersichtlich [22].

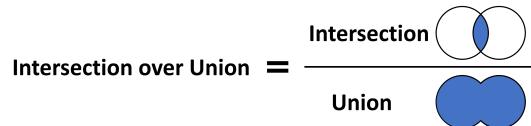


Abbildung 25: Darstellung der Intersection-over-Union.

Mit dem Dice-Coefficient wird ebenfalls die richtig erkannte Flächendeckung bewertet. Er wird durch die doppelte Überschneidung der vorgeschlagenen Fläche mit der korrekten Fläche geteilt durch die Summe der vorgeschlagenen und korrekten Flächen berechnet, wie in der folgenden Abbildung 26 zu sehen ist [22].

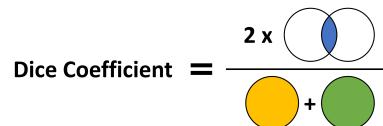


Abbildung 26: Darstellung des Dice-Coefficient.

Diese drei Metriken werden für jede Kategorie der Image-Segmentation berechnet. Somit kann auch eine Verbesserung in einer einzelnen Kategorie erkannt werden. Für eine Übersicht der gesamten Performance wird der Mittelwert über die Kategorien errechnet. Die IoU und der Dice-Coefficient wurden beide verwendet, um Vergleiche mit Modellen von Dritten zu ermöglichen. Da diese teilweise nur eine der beiden Metriken implementieren.

### 5.5.2. Dashboard

Das Dashboard ermöglicht einen Überblick über die Nutzung des Backends. Bei einer Anfrage des Endpunkts '/dashboard' werden Grafiken mittels der Plotly-Bibliothek erstellt und diese über die Jinja2-Bibliothek in eine HTML-Vorlage eingebettet. Die HTML-Vorlage verwendet für das Design die Bootstrap4-Bibliothek. Schlussendlich wird mit dem entstandenen HTML-Dokument auf die Anfrage geantwortet. Der erste Abschnitt des Dashboards ist in der folgenden Abbildung 27 zu sehen [23, 24, 25].



Abbildung 27: Bereich für die Performancebewertung der Image-Segmentation im Dashboard.

In der ersten Grafik wird der Verlauf der erreichten Accuracy, der IoU und des Dice-Coefficient für die einzelnen Kategorien dargestellt. Diese Metriken werden beim Erhalten einer korrigierten Segmentierung errechnet. Mit dem Dropdown-Menü können die Metriken stündlich, täglich oder wöchentlich gemittelt werden. Beim Boxplot handelt es sich um eine Darstellung der benötigten Zeit für eine Segmentierung durch den Segmentierungsalgorithmus. Im nächsten Abschnitt des Dashboards wird die Benutzungsstatistik der REST-API aufgezeigt, wie in der folgenden Abbildung 28 ersichtlich.

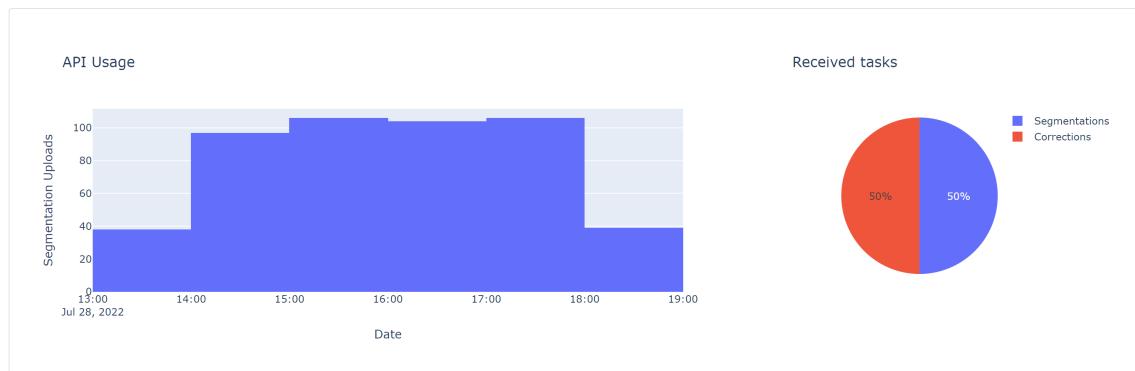


Abbildung 28: Bereich für die Benutzung der REST-API im Dashboard.

Im Histogramm werden die getätigten Segmentierungsaufrufe der REST-API über die Zeit aufgetragen. Die Behälter des Histogramms sind stündlich aufgeteilt. Standardmäßig werden die letzten 24 Stunden angezeigt. Nebenan befindet sich ein Kuchendiagramm, welches die Anzahl der Segmentierungsaufrufe mit den erhaltenen korrigierten Segmentierungen vergleicht. Daraufhin folgt im Dashboard eine Ansicht über das Nachtrainieren der Image-Segmentation. Dies ist in der folgenden Abbildung 29 zu sehen.



Abbildung 29: Bereich für das Verbessern der Image-Segmentierung im Dashboard.

Auf der linken Seite befindet sich ein Boxplot, welcher die verschiedenen Versionen der Segmentierungsmodelle über die gemittelte IoU vergleicht. Darunter wird die derzeitige Modellversion angezeigt. Rechts werden die Informationen fortlaufenden des Nachtrainierens dargestellt. Für jede Epoche des Nachtrainierens wird der Wert der Verlustfunktion und Accuracy aufgezeigt. Dabei wird zwischen Trainings- und Validierungsdaten unterschieden.

Durch diese Grafiken lassen sich die Perfomance der Image-Segmentierung, die Benutzung der REST-API und der Stand des Nachtrainierens bewerten. Darüberhinaus kann eine Voraussage über die zukünftige Performance erstellt werden.

## 5.6. Softwaretests

Das Backend wurde umfangreich mithilfe der Pydantic-Bibliothek getestet. Mittels Unit- und Integrationstests wird eine Testabdeckung von 98 % erreicht. Dabei wird auch die Datenbank mit der Dateiablage von der Applikation getrennt getestet. Ein vollständiger Testbericht ist im Anhang D zu finden [26].

## 6. Image-Segmentation

Das Image-Segmentation Modul befasst sich mit der Segmentierung von Bildern. Dabei ist es mühelos möglich das künstliche neuronale Netzwerk auszutauschen, um verschiedenste Anwendungsbereiche zu ermöglichen. In einer 'config' Datei lassen sich dazu der Speicherort des neuronalen Netzwerks und dessen Kategorien definieren. Um diese Anpassungsfähigkeit zu testen, wurde in dieser Arbeit die Segmentierung von Katzen und das Einzeichnen von Räumen, Türen und Hindernissen in Gebäudeplänen realisiert. Die Schnittstelle zum Backend ist durch ein Interface implementiert, wie in der folgenden Abbildung 30 ersichtlich.

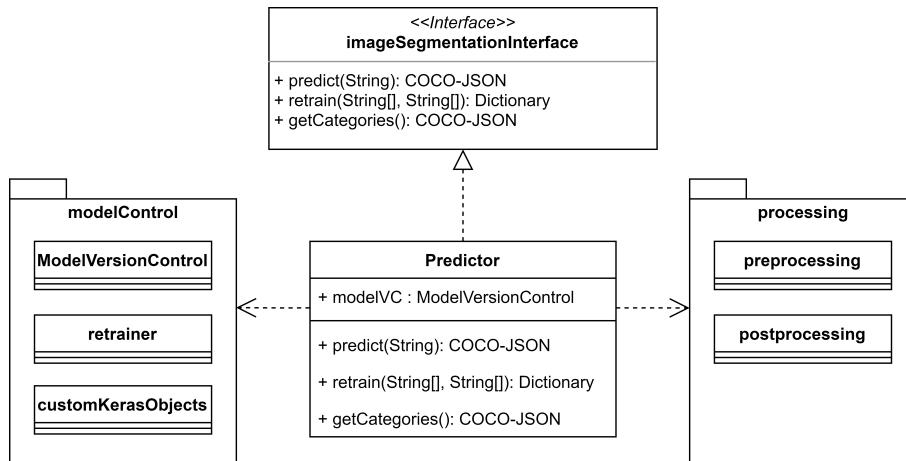


Abbildung 30: Klassendiagramm der ImageSegmentation.

Das Interface definiert Methoden, um ein Bild zu segmentieren, das Nachtrainieren auszulösen und die Kategorien der Segmentierung zu erhalten. Dieses Interface wird von der Klasse 'Predictor' implementiert. Dabei greift die Klasse auf den Ordner 'processing' zu. Dieser enthält Hilfsfunktionen, um Bilder zu bearbeiten. Des Weiteren wird der Ordner 'modelControl' vom 'Predictor' verwendet. Darin werden das Nachtrainieren und die verschiedenen Versionen des neuronalen Netzwerks verwaltet. In den folgenden Abschnitten werden der Segmentierungsablauf, die entwickelten Modelle und das Nachtrainieren thematisiert.

## 6.1. Ablauf der Segmentierung

Um eine Segmentierung für ein Bild zu erstellen, wird die Methode 'predict' des Interface verwendet. Diese erwartet den Speicherort des Bildes als String und antwortet mit der Segmentierung im COCO-JSON Format.

Als erstes wird das Bild in den Arbeitsspeicher geladen. Dannach wird es auf die Inputgrösse des neuronalen Netzwerks skaliert. Die Skalierung ist in folgender Abbildung 31 zu sehen.



Abbildung 31: Skalierung eines Inputbildes.

Das skalierte Bild wird dem neuronalen Netzwerk übergeben. Dieses erstellt für jeden Pixel eine Wahrscheinlichkeit, ob er der jeweiligen Kategorie zugehört. Auf diese Wahrscheinlichkeitsebenen wird ein Schwellwert angewendet. Befindet sich die Wahrscheinlichkeit des jeweiligen Pixel oberhalb dieses Schwellwerts, wird dieser der entsprechenden Kategorie zugeordnet und falls die Wahrscheinlichkeit darunter liegt nicht. Mit Hilfe des Algorithmus von Suzuki und Abe [27] werden die Konturen von den entstandenen Binärbildern gefunden. Dabei wird aus jedem Randpixel ein neuer Punkt. Die einzelnen Konturen werden durch einen minimalen Abstand zwischen den Punkten approximiert, um vereinfachte Polygone zu erhalten. Bei den entstandenen Polygonen wird überprüft, ob ein Polygon innerhalb eines anderen Polygons derselben Kategorie liegt. Dieses wird aus den gefundenen Polygonen entfernt. Die gefundenen Polygone der jeweiligen Kategorie werden wieder auf die Originalgrösse des Bildes skaliert. Die Resultate dieser einzelnen Schritte sind in der folgenden Abbildung 32 ersichtlich [27].

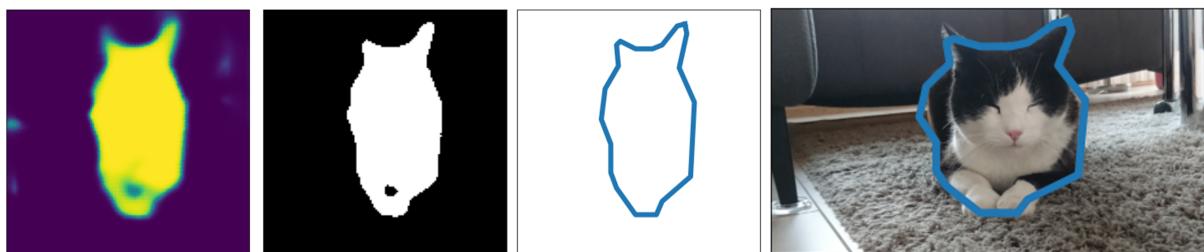


Abbildung 32: Polygone aus einer Wahrscheinlichkeitsebene generieren. Von links nach rechts: Wahrscheinlichkeitsebene, Binärbild, Polygon, skaliertes Polygon.

Schlussendlich wird mit den gefundenen Polygone ein COCO-JSON Dokument erstellt, mit welchem auf die Anfrage geantwortet wird. Die Polygone werden dabei der jeweiligen Kategorie zugeordnet. Es ist möglich, dass die Polygone einer Kategorie die Polygone einer anderen Kategorie überschneiden.

## 6.2. Künstliche neuronale Netzwerke für die Image-Segmentierung

In der klassischen Programmierung werden Daten mittels Regeln in Antworten umgewandelt. Diese Herangehensweise setzt ein Verständnis des Problems voraus, welches Expertenwissen benötigt. Bei komplexen und hochdimensionalen Problemen ist es schwierig alle Einflüsse zu erfassen und mit Regeln abzubilden. In Abbildung 33 ist der Unterschied zum maschinellen Lernen dargestellt. Neuronale Netzwerke können aus Daten und Antworten komplexe Regeln ableiten. Diese Regeln können später zur Problembewältigung genutzt werden. Dadurch sind Lösungen für hochkomplexe Probleme möglich, wie dies bei der Bildsegmentierung der Fall ist. Der Nachteil dabei ist, dass die entstandenen Regeln nur schwer nachvollziehbar sind. [28].

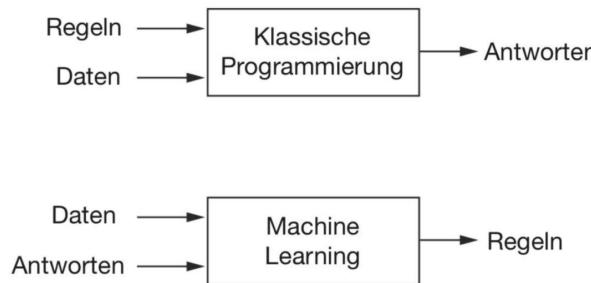


Abbildung 33: Unterschied zwischen Programmieren und Machine-Learning [28].

Um ein künstliches neuronales Netz zu erstellen, werden folgende Grundzutaten benötigt. [28, 29]

1. Dataset:  
Ein qualitativ hochwertiger und umfangreicher Datensatz, welcher die Problemstellung abdeckt.
2. Model:  
Ein Aufbau mit welchem die Problemstellung modelliert werden kann.
3. Verlustfunktion (Cost-Function oder loss):  
Eine Zielfunktion, deren Wert minimiert werden soll.
4. Optimization Procedure:  
Ein Algorithmus zur Suche der optimalen Parameter.

### 6.2.1. Dataset und Data-Augmentation

Für das Training eines neuronalen Netzwerks werden umfangreiche Datensätze benötigt. Dabei gilt, je grösser und qualitativ hochwertiger der Datensatz, umso bessere Ergebnisse können erzielt werden. Durch eine Datenaugmentation kann ein Datensatz künstlich vergrössert werden und es wird die Gefahr der Überanpassung vermindert. Dabei bleibt aber der Grundsatz erhalten, dass ein neuronales Netzwerk nur so gut ist, wie die ihm zugrundeliegenden Daten [30, 31].

Für das Training der Modelle wurden die Daten mittels Spiegelung, Verzerrung, Rotationen und Vergrösserungen augmentiert. Beispielhaft sind einige der Augmentationen in Abbildung 34 dargestellt. Die jeweilige Ground-Truth wurde ebenfalls augmentiert.



Abbildung 34: Data-Augmentation eines Katzenbildes mit dazugehörigen Ground-Truth.

Durch die Augmentation entstehen Daten, welche so nicht im Datensatz vorhanden wären. Zum Beispiel gäbe es keine Katzen, welche auf der Seite liegen (Abbildung 34 oben links) oder nicht komplett im Bild sind (Abbildung 34 oben rechts). Durch diese zusätzlichen Daten wird das trainierte Modell robuster und kann besser generalisieren. Dies ist ein weiterer Vorteil neben der vergrösserten Datenmenge.

### 6.2.2. Modellarchitekturen

Durch die Festlegung einer Modellarchitektur wird die Menge von Möglichkeiten (Hypothesenraum) auf eine Reihe bestimmter Tensoroperationen beschränkt. Geeignete Architekturen für die Image-Segmentierung sind unter anderem das U-Net, HRNet und FastFCN [28, 32].

Das U-Net besteht aus einem Encoder und Decoder. Der Encoder verkleinert die Auflösung und extrahiert somit wichtige Features im Bild. Danach folgt der Decoder, welcher die Auflösung wieder erhöht. Dabei werden die Details durch Verbindungen zum Encoderpfad mitgegeben. Die Architektur ist symmetrisch, wodurch die Auflösung des Inputs dem des Outputs entspricht. Die folgende Abbildung 35 bildet die Architektur des U-Nets ab [33].

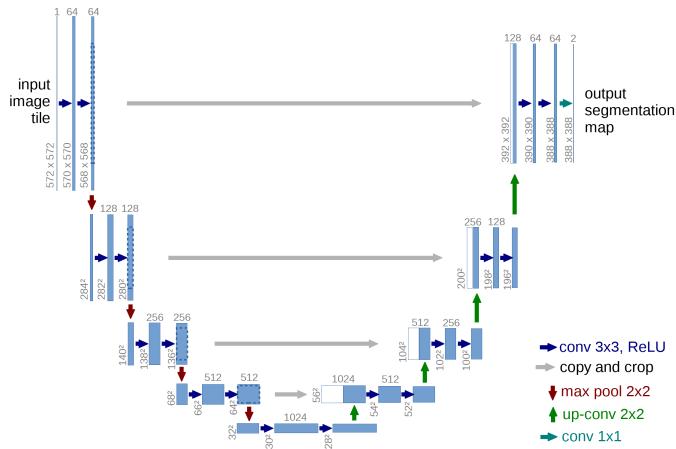


Abbildung 35: Architektur des U-Net [33].

Das HRNet (High-Resolution Network) behält eine hohe Auflösung des Bildes über den ganzen Bearbeitungsprozess bei. Parallel dazu sind Pfade, welche eine niedrigere Auflösung besitzen. Die Informationen der Pfade mit hohen und niedrigen Auflösungen werden wiederholt ausgetauscht. Somit entsteht eine Architektur, welche in der folgenden Abbildung 36 zu sehen ist [34].

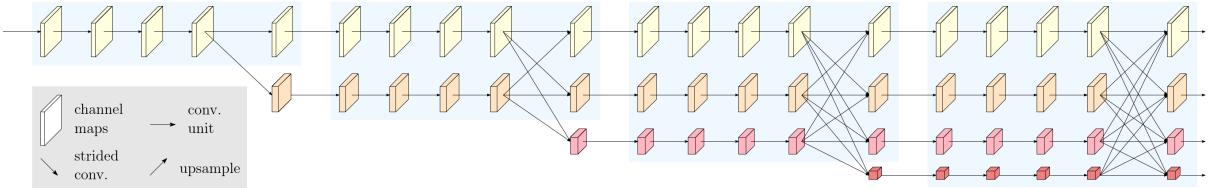


Abbildung 36: Architektur des HRNet [34].

Der Aufbau der FastFCN Architektur beginnt mit einem Encoderpfad. Danach wird mit einer JPU (Joint Pyramid Upsampling) wieder eine höhere Auflösung erreicht. Dazu werden zuerst mehrere Stufen des Encoderpfads auf dieselbe Auflösung dekodiert. Diese durchlaufen vier Faltungsebenen mit verschiedenen Dilatationsraten. Die Dilatationsrate vergrößert den Kernel der jeweiligen Faltungsebene. Danach werden die vier Faltungsebenen wieder verbunden. Die folgende Abbildung 37 zeigt das Herzstück dieser Architektur, die JPU [35].

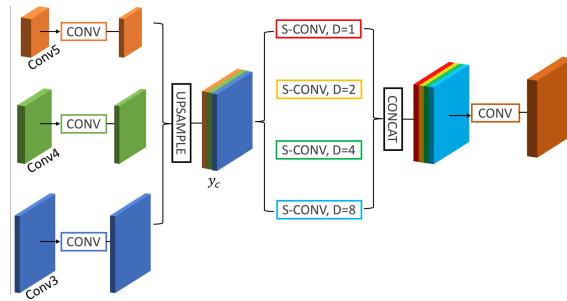


Abbildung 37: Die JPU der FastFCN Architektur [35].

Es gibt viele Architekturen, welche für die Image-Segmentierung eingesetzt werden können. Diese Auswahl an Architekturen wurde getroffen, weil sie einen Bekanntheitsgrad besitzen und eine gute Performance versprechen [32].

### 6.2.3. Verlustfunktionen

Der Wert der Verlustfunktion soll während dem Trainieren des neuronalen Netzwerks minimiert werden. Bei einem trainierten Netzwerk kann der erzielte Wert der Verlustfunktion als Bewertung für die Erreichung der gegebenen Aufgabe eingesetzt werden. Für die Image-Segmentierung Aufgabe sind die binäre Kreuzentropie und der Dice-Loss als Verlustfunktion geeignet [28, 36]. Die binäre Kreuzentropie wird benutzt, um die Differenz zwischen zwei Wahrscheinlichkeitsverteilungen zu messen. Somit wird versucht die Unterschiede im Informationsgehalt zwischen den tatsächlichen und den vorhergesagten Bildmasken zu messen. Die Berechnung der binären Kreuzentropie ist in der folgenden Gleichung (2) zu sehen, wobei mit  $\hat{y}$  der vorhergesagte und mit  $y$  der tatsächliche Pixel beschrieben wird [36].

$$L(y, \hat{y}) = -y \cdot \log(\hat{y}) - (1 - y) \cdot \log(1 - \hat{y}) \quad (2)$$

Der Dice-Loss ist eine Adaption der Dice-Coefficient Metrik zu einer Verlustfunktion. Es wird der prozentuale Anteil der überlappenden Bildmasken ermittelt. In der folgenden Gleichung (3) ist die Berechnung ersichtlich [36].

$$DL(y, \hat{y}) = 1 - \frac{2 \cdot y \cdot \hat{y} + 1}{y + \hat{y} + 1} \quad (3)$$

Verschiedenste Verlustfunktionen können benutzt werden, um eine Image-Segmentierung Aufgabe zu bewältigen. Bei den ausgewählten handelt es sich um häufig benutzte Verlustfunktionen für die Image-Segmentierung [36, 37].

### 6.2.4. Optimierungsprozess

Mit einem Optimierer wird das neuronale Netzwerk anhand der Verlustfunktion aktualisiert. Dabei implementiert der Optimierer eine bestimmte Variante des stochastischen Gradientenabstiegsverfahrens [28, 36].

Der benutzte Adam-Optimierer (adaptive moment estimation) verwendet einen Ansatz, bei dem mit einem Momentum gearbeitet wird. Dies ermöglicht eine Anpassung der Lernrate, um in der Verlustfunktion nicht bei lokalen Minimas oder Sattelpunkten zu verweilen. Der Adam-Optimierer ist weit verbreitet und gilt als vielseitiger und robuster Optimierer [29].

### 6.3. Modell zur Segmentierung von Katzenbildern

Für das Training des Netzwerkes zur Katzenbildersegmentierung wurde der 'Oxford-IIIT Pet Dataset' verwendet. Dieser Datensatz wurde erstmals in der Publikation 'Cats and Dogs' vorgestellt. Der komplette Datensatz umfasst Bilder mit einer zugehörigen Ground-Truth von Katzen und Hunden in 37 verschiedenen Kategorien mit jeweils 200 Bildern. Für das neuronale Netzwerk wurden nur die Katzenbilder verwendet. Dadurch standen für das Training noch 2'371 Bilder zur Verfügung [38, 39].

In Abbildung 38 ist ein Katzenbild mit dazugehöriger Ground-Truth dargestellt. Die Ground-Truth ist dabei in Katze und Hintergrund zu interpretieren.



Abbildung 38: Links das Bild einer Katze mit der dazugehörigen Ground-Truth rechts [39].

Für die Katzenbildersegmentierung wurden die drei vorgestellten Architekturen eingesetzt. Diese wurden solange trainiert, bis der Wert der Verlustfunktion konstant über 10 Epochen anstieg. Die Ergebnisse sind im Anhang G ersichtlich. Die U-Net Architektur hat auf der Segmentierung von Katzenbildern die besten Ergebnisse erzielt. Dabei wurden auf den Validierungsdaten eine Accuracy von 93 % und IoU von 70 % erreicht. Die Metriken während dem gesamten Trainingsprozess sind in der folgenden Abbildung 39 ersichtlich.

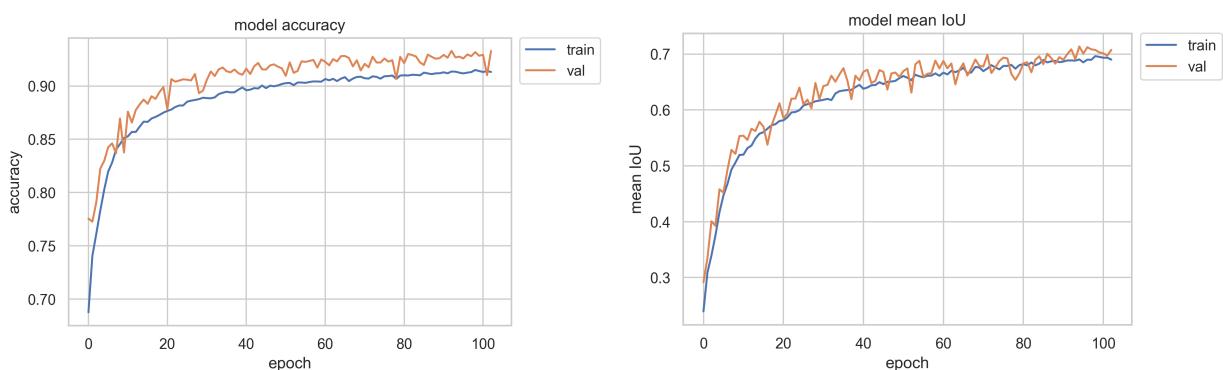


Abbildung 39: Metriken der U-Net-Architektur bei Katzenbildern.

Wie die Metriken erahnen lassen, kann das Modell Wahrscheinlichkeitsebene erzeugen, welche nahe an der Ground-Truth liegen. Dies zeigt auch die Stichprobe in Abbildung 40.

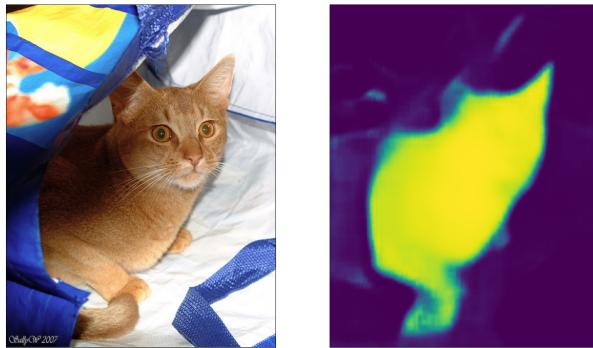


Abbildung 40: Links das Bild einer Katze mit der erzeugten Wahrscheinlichkeitsebene rechts [39].

Das Modell erkennt Katzen verschiedener Arten an den unterschiedlichsten Positionen. Es wurde somit eine hohe Modellgüte erreicht.

#### 6.4. Modell zur Segmentierung von Gebäudeplänen

Datensätze für die Segmentierung von Gebäudeplänen sind nicht so verbreitet, wie jene für klassische Segmentierungsaufgaben. Es existieren mehrere kleinere Datensätze für Symbol-Spotting Aufgaben mit Umfängen von 200-500 Bildern (siehe Fachmodul im Anhang I). Der umfangreichste Datensatz für die Aufgabe dieser Arbeit stellt der Datensatz 'CubiCasa5k' zur Verfügung. Dieser Datensatz wird mit der Publikation 'CubiCasa5K: A Dataset ...' vorgestellt. Er ist dabei als Beiproduct bei der Erzeugung von Marketingmaterial im Bereich Immobilien entstanden. Zu diesem Zweck wurden 5'000 reale Grundrisspläne digitalisiert und auf deren Grundlage Pläne im Vector-Format erstellt. Veröffentlicht wurde der Datensatz, um die Lücke von zu Verfügung stehenden Datensätzen im Bereich der Grundrissplansegmentierung zu schliessen. In den Plänen ist die Ground-Truth im Vector-Format eingebettet. Für diese Arbeit wurde das Vector-Format in eine Bitmap umgewandelt. Dadurch konnten Trainingsziele wie Räume, Hindernisse und Türen voneinander getrennt werden. In Abbildung 41 ist ein Plan mit den drei dazugehörigen Ground-Truths abgebildet [40, 41].

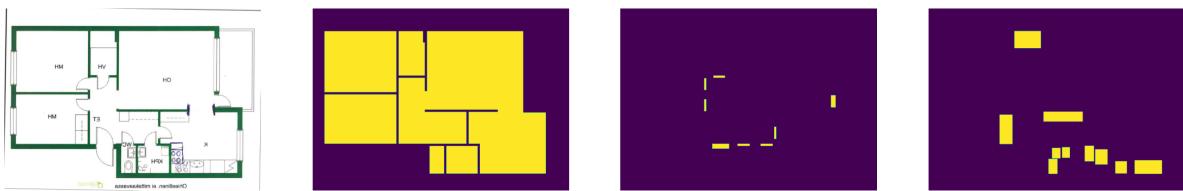


Abbildung 41: Ausschnitt des Datensatzes 'CubiCasa5K'. Von links nach rechts: Grundrissplan, Raum-, Hindernis- und Tür-Ground-Truth [39].

Für die Gebäudeplansegmentierung wurden verschiedene Einstellungen des U-Nets getestet. Das Training wurde gestoppt, wenn der Wert der Verlustfunktion über 10 Epochen konstant anstieg. Die Ergebnisse sind im Anhang H ersichtlich. Das Modell mit den besten Ergebnissen besitzt eine verdoppelte Anzahl von Featuremaps im Vergleich zum Baseline-Modell. Dadurch konnte eine IoU von 57 % auf den Validierungsdaten erreicht werden. Beim Training blieb die Accuracy konstant auf einem hohen Niveau, besitzt jedoch keine hohe Aussagekraft. Der gesamte Verlauf der Metriken während der Trainingsphase ist in der folgenden Abbildung 42 zu sehen.

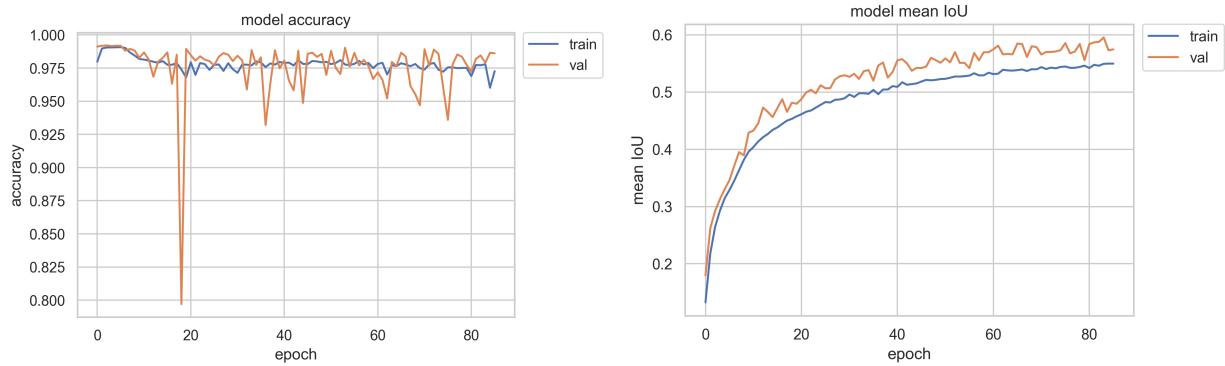


Abbildung 42: Metriken der U-Net-Architektur bei Grundrissplänen.

Die Aufteilung in die drei Wahrscheinlichkeitsebenen (Räume, Türen und Hindernisse) werden im letzten Teil des U-Nets durchgeführt. Dabei werden die Räume besser erkannt als die Türen und Hindernisse. Dies ist auch in der folgenden Abbildung 43 ersichtlich.



Abbildung 43: Von links nach rechts: Grundrissplan, Wahrscheinlichkeitsebene der Räume, Hindernisse und Türen [40].

Folglich konnte ein Modell kreiert werden, welches Vorschläge von Räumen, Türen und Hindernissen für einen Gebäudeplan liefert. Diese sind jedoch qualitativ nicht ausreichend, um sie direkt verwenden zu können. Deshalb ist die Anwendung des Human-in-the-loop Konzepts für dieses Modell sinnvoll.

## 6.5. Nachtrainieren

Um das neuronale Netzwerk nachzutrainieren, kann die Methode 'retrain' benutzt werden, welche im Interface definiert ist. Der Methode werden die Speicherorte von Bildern und den dazugehörigen COCO-JSON Korrekturen übergeben, welche verwendet werden, um das neuronale Netzwerk weiterzutrainieren.

Zuerst müssen die Bilder, wie auch die COCO-JSONs, auf den Modelinput angepasst werden. Dazu werden die Bilder in den Arbeitsspeicher geladen. Dann werden sie auf die Inputgrösse skaliert.

Den COCO-JSONs werden die Polygone zu den jeweiligen Kategorien entnommen. Diese werden ebenfalls auf die Inputgrösse skaliert. Daraufhin wird für jede Kategorie pro COCO-JSON ein leeres Bild mit den Outputgrößen des neuronalen Netzwerks erstellt. In diese werden die jeweiligen Polygone gezeichnet und die entstehenden Segmente ausgefüllt. Diese werden als Ground-Truth verwendet. Die Vorbereitung des Nachtrainierens ist in der folgenden Abbildung 44 ersichtlich.

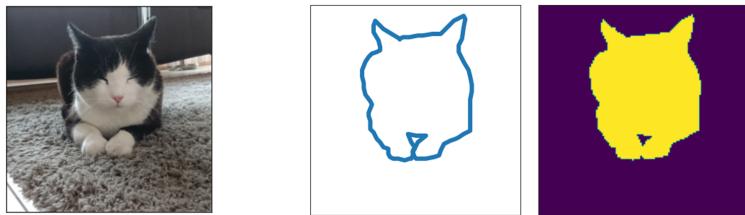


Abbildung 44: Vorbereitung für das Nachtrainieren. Von links nach rechts: skaliertes Inputbild, skaliertes Polygon des korrigierten COCO-JSONs und dasselbe Polygon als Ground-Truth.

Die Bilder mit den dazugehörigen Ground-Truths werden in einen Trainings- und Validierungsdatensatz unterteilt. Mit Datenaugmentation wird die Datenmenge des Trainingsdatensatzes vergrössert. Schlussendlich wird das neuronale Netzwerk mit den Daten weitertrainiert. Während dem Trainieren werden die Metriken und der Wert der Verlustfunktion mit den Validierungsdaten bestimmt. Die Trainingsdauer kann durch einen frei gewählten Parameter gesetzt oder abhängig von der Verlustfunktion bestimmt werden. Die erreichten Werte der Metriken und der Verlustfunktion jedes einzelnen Trainingsschritts werden dem Backend übergeben, um dort über das Dashboard dargestellt werden zu können.

## 6.6. Model-Version-Control

Die implementierte Model-Version-Control wird für die Verwaltung der einzelnen trainierten Modelle verwendet. Beim Nachtrainieren entsteht ein angepasstes Modell, welches über das Model-Version-Control abgelegt werden kann. Die Modelldaten werden im lokalen Dateisystem gespeichert. Im selben Verzeichnis wird eine CSV-Datei angelegt, welche eine Übersicht der Modelle beinhaltet. In dieser Datei ist auch das neuste Modell ersichtlich, welches für neue Segmentierungen benutzt wird. Die Version der einzelnen Modelle wird beim Speichern festgelegt.

## 7. Fazit

Das resultierende System dieser Arbeit implementiert das Human-in-the-loop Konzept für die Image-Segmentierung. Dabei wird durch die Aufteilung in Frontend, Backend und Image-Segmentierung Modul eine hohe Flexibilität erreicht. Falls ein Modul für einen Anwendungsbereich unpassend ist, kann dieses ausgetauscht werden.

Die Integration der einzelnen Module in eine bestehende Infrastruktur lässt sich problemlos durchführen. Das Frontend ist dazu als Bibliothek konzipiert und lässt sich in eine bestehende Webseite einfügen. Auf einem Server kann das Backend und Image-Segmentierung Modul betrieben werden. Im Backend kann mit der config-Datei ein Speicherort für die entstehenden Daten definiert werden. Die Image-Segmentierung besitzt eine weitere config-Datei für die Bestimmung des Baseline-Modells, der dazugehörigen Kategorien und den Speicherort für die entstehenden Modelle. Somit kann das Modell, welches für die Segmentierung der Bilder zuständig ist, mit Leichtigkeit ausgetauscht werden. Das vollumfängliche System ermöglicht eine AI-unterstützte Bearbeitung einer Segmentierungsaufgabe. Dabei wird der Nutzer unterstützt, was eine erhöhte Effizienz zur Folge hat. Zugleich wird im Hintergrund das Modell durch die Nutzereingaben verbessert. Dadurch wird das Modell in seiner unterstützenden Tätigkeit konstant verbessert.

Während dem Fachmodul gab es Bedenken bezüglich der Machbarkeit von einem Modell für die Segmentierung von Gebäudeplänen. Im Gegensatz zu natürlichen Bildern ist dort die Dichte der Bildinformationen geringer. Ein Gebäudeplan besteht mehrheitlich aus weißen Flächen und schwarzen Linien. Durch einen umfangreichen Datensatz konnte ein Modell trainiert werden, welches brauchbare Resultate erzielt. Dabei kann das Modell die Räume, Türen und Hindernisse in einem Gebäudeplan erkennen und einzeichnen.

Mit einem entwickelten Modell für die Katzenbilderkennung konnte das Nachtrainieren des Systems getestet werden. Dazu wurden mehrere Katzenbilder annotiert und dem System als Korrektur übergeben. Dabei konnte eine Verbesserung der IoU Metrik über die Modellversionen beobachtet werden. Dies ist in der folgenden Abbildung 45 ersichtlich.

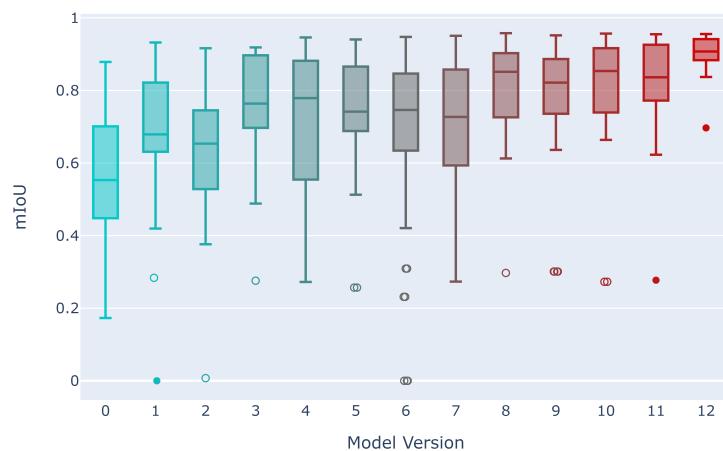


Abbildung 45: Durchschnittliche IoU über mehrere Modellversionen der U-Net Architektur für die Segmentierung von Katzenbildern.

## 8. Ausblick

Die grösste Herausforderung bei der Implementierung des Systems besteht in der Anpassung des Modells auf einen neuen Anwendungsfall. Das System muss so konfiguriert werden, damit sich das Modell über die Zeit verbessert. Um die Entwicklung des Modells steuern zu können, wäre eine Kontrolle durch einen Experten bei der Einführung einer weiteren Modellversion über das Dashboard möglich. Zur Zeit können die Lernrate und weitere Parameter des Nachtrainierens während des Betriebs nicht angepasst werden. Diese Einstellungen könnten direkt über eine grafische Oberfläche im Dashboard zur Verfügung gestellt werden, um eine höhere Anpassungsfähigkeit zu bieten. Sollte sich durch das Nachtrainieren das Modell extrem verschlechtern, wäre es von Vorteil zu einem vorherigen Modell zu springen oder mit einem neuen Modell zu starten. Diese Funktionalität könnte ebenfalls über das Dashboard zur Verfügung gestellt werden, um somit eine Kontrollzentrale für das Modell zu schaffen.

Im Image-Segmentation Modul gibt es eine Klasse, welche eine rudimentäre Model Version Control implementiert. Diese könnte zu einer Schnittstelle für ein vollumfängliches Versioning System ausgebaut werden. Somit kann ein kontrollierter Arbeitsablauf mit den unterschiedlichen Modellen erreicht werden. Falls das System in einem öffentlichen Umfeld benutzt wird, wäre eine Pipeline zur Datenüberprüfung möglich. Dadurch könnten die unseriösen Trainingsdaten gefiltert werden. Die Funktionalität zur Generierung der Polygone anhand der Wahrscheinlichkeitsebenen ist für einen allgemeinen Anwendungsfall konzipiert. In diesem Bereich könnten die Resultate pro Anwendungsfall optimiert werden. Zum Beispiel könnte bei Gebäudeplänen die Approximation des nächsten Polygons abhängig vom Winkel zur vorherigen Polygonlinie gemacht werden. In Plänen stehen die einzelnen Linien häufig in definierten Winkeln zueinander. Bei der Katzenbildsegmentation könnte man die Gesamtanzahl der Polygone empirisch festlegen.

Zu Beginn dieser Arbeit mussten einige Entscheidungen zur Implementierung des Systems getroffen werden. Durch die gesammelten Erfahrungen, sind bessere Möglichkeiten für einige dieser Entscheidungen bekannt. Das Frontend könnte dadurch auf die Performance optimiert werden. Grössere Anpassungsmöglichkeiten im Bereich Design und Darstellung wären ebenfalls eine Verbesserung. Um weitere Möglichkeiten zum Start eines Segmentierungsauftrags anzubieten, könnten noch weitere Schnittstellen zum Hochladen von Bildern implementiert werden.

Um das System in einer Produktionsumgebung verwenden zu können, müssten ebenfalls noch die Punkte behandelt werden, welche bewusst ausgeklammert wurden. Die wichtigsten davon sind die Sicherheit und Skalierbarkeit. Mit einer Authentifizierung könnte die REST-API geschützt werden. Dies verhindert einen Missbrauch der Schnittstelle. Skalierbarkeit könnte durch die Nutzung von mehreren Threads oder eine Veröffentlichung über einen Cloud-Service erreicht werden.

## A. Verzeichnisse

### Abbildungsverzeichnis

1.	Katzenbild-Segmentierung mit dem Human-in-the-loop Ansatz. . . . .	I
2.	Cat image segmentation using the human-in-the-loop approach. . . . .	III
3.	Deploymentdiagramm des Systems in bestehender Infrastruktur. . . . .	2
4.	Kommunikationdiagramm von Frontend, Backend und Image-Segmentation. . . . .	3
5.	Human-in-the-loop Konzept für die Image-Segmentation. In Anlehnung an [1, Seite 57]. . . . .	5
6.	Verbesserung des Modells durch die Anwendung des Human-in-the-loop Konzepts. .	6
7.	Lebenszyklus eines Machine-Learning Modells [2]. . . . .	7
8.	Übersicht über den Aufbau des Frontends. . . . .	9
9.	Der Informationsfluss innerhalb des Frontends. . . . .	10
10.	Aufbau der genutzten State-Machine im Frontend. . . . .	11
11.	Typischer Ablauf eines Segmentierungsauftrages aus Sicht des Frontends. . . . .	12
12.	Grafische Oberfläche mit markierten Bereichen. Nummeriert sind (1) Kontrollemente, (2) Informationsbereich, (3) Verfügbare Labels, (4) Arbeitsfläche und (5) Übersicht der Annotierungen. . . . .	13
13.	Die verschiedenen Informationsanzeigen für den Benutzer. . . . .	14
14.	Beispielcode für die Integration in eine React-Applikation. . . . .	15
15.	Der grundlegende Aufbau eines COCO-JSON. . . . .	16
16.	Ein in dieser Arbeit eingesetztes COCO-JSON. . . . .	17
17.	Klassendiagramm des Backends. . . . .	19
18.	Endpunkte der REST-API. . . . .	20
19.	Middleware-Schicht innerhalb des Backendservers. In Anlehnung an [17]. . . . .	21
20.	Sequenzdiagramm zum Ablauf einer Segmentierung im Backend. . . . .	22
21.	Sequenzdiagramm einer Segmentierungskorrektur im Backend. . . . .	23
22.	Sequenzdiagramm über den Auftragsabschluss ohne eine Segmentierungskorrektur im Backend. . . . .	24
23.	Entity-Relationship-Modell der Datenbank im Backend. . . . .	24
24.	Datenbankschema der Datenbank im Backend. . . . .	25
25.	Darstellung der Intersection-over-Union. . . . .	26
26.	Darstellung des Dice-Coefficient. . . . .	26
27.	Bereich für die Performancebewertung der Image-Segmentation im Dashboard. .	27
28.	Bereich für die Benutzung der REST-API im Dashboard. . . . .	27
29.	Bereich für das Verbessern der Image-Segmentation im Dashboard. . . . .	28
30.	Klassendiagramm der ImageSegmentation. . . . .	29
31.	Skalierung eines Inputbildes. . . . .	30
32.	Polygone aus einer Wahrscheinlichkeitsebene generieren. Von links nach rechts: Wahrscheinlichkeitsebene, Binärbild, Polygon, skaliertes Polygon. . . . .	30
33.	Unterschied zwischen Programmieren und Machine-Learning [28]. . . . .	31
34.	Data-Augmentation eines Katzenbildes mit dazugehörigen Ground-Truth. . . . .	32
35.	Architektur des U-Net [33]. . . . .	32
36.	Architektur des HRNet [34]. . . . .	33

37.	Die JPU der FastFCN Architektur [35]. . . . .	33
38.	Links das Bild einer Katze mit der dazugehörigen Ground-Truth rechts [39]. . . . .	35
39.	Metriken der U-Net-Architektur bei Katzenbildern. . . . .	35
40.	Links das Bild einer Katze mit der erzeugten Wahrscheinlichkeitsebene rechts [39]. . . . .	36
41.	Ausschnitt des Datensatzes 'CubiCasa5K'. Von links nach rechts: Grundrissplan, Raum-, Hindernis- und Tür-Ground-Truth [39]. . . . .	36
42.	Metriken der U-Net-Architektur bei Grundrissplänen. . . . .	37
43.	Von links nach rechts: Grundrissplan, Wahrscheinlichkeitsebene der Räume, Hindernisse und Türen [40]. . . . .	37
44.	Vorbereitung für das Nachtrainieren. Von links nach rechts: skaliertes Inputbild, skaliertes Polygon des korrigierten COCO-JSONs und dasselbe Polygon als Ground-Truth. . . . .	38
45.	Durchschnittliche IoU über mehrere Modellversionen der U-Net Architektur für die Segmentierung von Katzenbildern. . . . .	39
46.	Der grundlegende Aufbau eines COCO-JSON. . . . .	52
47.	Die einzelnen Bestandteile eines COCO-JSONs im Detail. . . . .	53

## Literaturverzeichnis

- [1] R. Monarch and C. D. Manning, *Human-in-the-Loop Machine Learning: Active Learning and Annotation for Human-Centered AI*, Sherlter Island, NY, 2021.
- [2] "Was ist MLOps und wie funktioniert es? | Alteryx." [Online]. Available: <https://www.alteryx.com/de/glossary/mlops>
- [3] S. Springer, *React: das umfassende Handbuch*, 1st ed., ser. Rheinwerk Computing. Bonn: Rheinwerk Verlag, 2020.
- [4] "Hooks API Reference – React." [Online]. Available: <https://reactjs.org/docs/hooks-reference.html>
- [5] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," 2014, publisher: arXiv Version Number: 3. [Online]. Available: <https://arxiv.org/abs/1405.0312>
- [6] "COCO - Common Objects in Context." [Online]. Available: <https://cocodataset.org/#format-data>
- [7] "Label Studio – Open Source Data Labeling." [Online]. Available: <https://labelstud.io/>
- [8] "Label Studio Frontend .," Aug. 2022, original-date: 2020-01-08T13:45:26Z. [Online]. Available: <https://github.com/heartexlabs/label-studio-frontend>
- [9] "Getting Started | Axios Docs." [Online]. Available: <https://axios-http.com/docs/intro>
- [10] "CSS Modules," Aug. 2022, original-date: 2015-05-25T21:54:47Z. [Online]. Available: <https://github.com/css-modules/css-modules>

- [11] “FastAPI.” [Online]. Available: <https://fastapi.tiangolo.com/>
- [12] A. Boschetti and L. Massaron, *Python data science essentials: become an efficient data science practitioner by thoroughly understanding the key concepts of Python*, 1st ed., ser. Community experience distilled. Birmingham Mumbai: Packt Publishing, 2015.
- [13] M. H. Massé and M. Massé, *REST API design rulebook: designing consistent RESTful Web Service Interfaces*. Beijing Köln: O'Reilly, 2012.
- [14] S. Ramírez, “tiangolo/fastapi,” Jul. 2022. [Online]. Available: <https://github.com/tiangolo/fastapi>
- [15] “Was ist Middleware – Definition | Microsoft Azure.” [Online]. Available: <https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-middleware/>
- [16] “Middleware - FastAPI.” [Online]. Available: <https://fastapi.tiangolo.com/tutorial/middleware/?h=middle>
- [17] “Client, Server & Middleware.” [Online]. Available: [https://www.dbs.ethz.ch/education/dom/dom\\_seminary.html](https://www.dbs.ethz.ch/education/dom/dom_seminary.html)
- [18] “CORS (Cross-Origin Resource Sharing) - FastAPI.” [Online]. Available: <https://fastapi.tiangolo.com/tutorial/cors/>
- [19] “Cross-Origin Resource Sharing (CORS) - HTTP | MDN.” [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
- [20] “SQLite Home Page.” [Online]. Available: <https://www.sqlite.org/index.html>
- [21] “SQLAlchemy - The Database Toolkit for Python.” [Online]. Available: <https://www.sqlalchemy.org/>
- [22] E. Tiu, “Metrics to Evaluate your Semantic Segmentation Model,” Oct. 2020, publication Title: Medium. [Online]. Available: <https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>
- [23] “plotly.py,” Jul. 2022. [Online]. Available: <https://github.com/plotly/plotly.py>
- [24] “Jinja — Jinja Documentation (3.1.x).” [Online]. Available: <https://jinja.palletsprojects.com/en/3.1.x/>
- [25] J. T. contributors and M. Otto, “Bootstrap.” [Online]. Available: <https://getbootstrap.com/>
- [26] “pydantic.” [Online]. Available: <https://pydantic-docs.helpmanual.io/>
- [27] S. Suzuki and K. be, “Topological structural analysis of digitized binary images by border following,” *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, Apr. 1985. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/0734189X85900167>
- [28] F. Chollet, *Deep Learning mit Python und Keras: das Praxis-Handbuch: vom Entwickler der Keras-Bibliothek*, 1st ed. Frechen: mitp, 2018.

- [29] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*, first edition ed. Beijing ; Boston: O'Reilly Media, 2017, oCLC: ocn953432302.
- [30] J. Krohn, G. Beyleveld, and A. Bassens, *Deep Learning illustriert: eine anschauliche Einführung in Machine Vision, Natural Language Processing und Bilderzeugung für Programmierer und Datenanalysten*. Heidelberg: dpunkt.verlag, 2020.
- [31] "Data Science and Machine Learning Resources." [Online]. Available: <https://www.jonkrohn.com/resources>
- [32] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," Nov. 2020, arXiv:2001.05566 [cs]. [Online]. Available: <http://arxiv.org/abs/2001.05566>
- [33] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," May 2015, arXiv:1505.04597 [cs]. [Online]. Available: <http://arxiv.org/abs/1505.04597>
- [34] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep High-Resolution Representation Learning for Visual Recognition," Mar. 2020, arXiv:1908.07919 [cs]. [Online]. Available: <http://arxiv.org/abs/1908.07919>
- [35] W. Huikai, "FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation," Aug. 2022, original-date: 2019-03-27T09:08:22Z. [Online]. Available: <https://github.com/wuhuikai/FastFCN>
- [36] "3 Common Loss Functions for Image Segmentation." [Online]. Available: [https://dev.to/\\_aadidev/3-common-loss-functions-for-image-segmentation-545o](https://dev.to/_aadidev/3-common-loss-functions-for-image-segmentation-545o)
- [37] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, Oct. 2020, pp. 1–7, arXiv:2006.14822 [cs, eess]. [Online]. Available: <http://arxiv.org/abs/2006.14822>
- [38] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Providence, RI: IEEE, Jun. 2012, pp. 3498–3505. [Online]. Available: <http://ieeexplore.ieee.org/document/6248092/>
- [39] "Visual Geometry Group - University of Oxford." [Online]. Available: <https://www.robots.ox.ac.uk/~vgg/data/pets/>
- [40] A. Kalervo, J. Ylioinas, M. Häikiö, A. Karhu, and J. Kannala, "CubiCasa5K: A Dataset and an Improved Multi-Task Model for Floorplan Image Analysis," Apr. 2019, arXiv:1904.01920 [cs]. [Online]. Available: <http://arxiv.org/abs/1904.01920>
- [41] "CubiCasa5k," Mar. 2019, type: dataset. [Online]. Available: <https://zenodo.org/record/2613548>

## B. Aufgabenstellung und Zieldefinition

Die Aufgabenstellung für die Bachelorarbeit wurde im Fachmodul (siehe Kapitel I) erarbeitet und aus diesem übernommen.

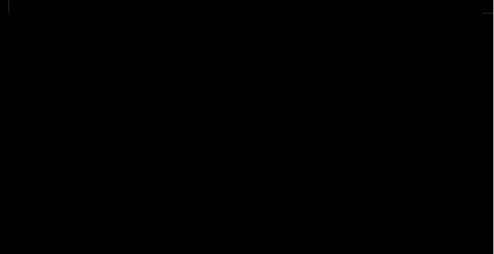
Das Ziel dieser Bachelorarbeit ist es, ein leicht integrierbares Labelling-Tool zu entwickeln. Dieses dient dem Labeln von Räumen und Hindernissen auf Grundrissplänen. Dabei soll das Labelling von einem Machine-Learner unterstützt werden. Dieser soll sich aufgrund der Daten kontinuierlich weiter verbessern können. Diese Daten entstehen durch die Nutzung des Tools. Dieses ist als eine Human-in-the-Loop-Applikation konzipiert. Der Mehrwert dieser Arbeit besteht daraus, die vorhandenen Lösungen und Technologien zu einem leicht integrierbaren Paket zusammenzuführen. Dieses Paket ist auf die Segmentation und das Labelling von Grundrissplänen spezialisiert. Die folgenden Anforderungen sollen mit dieser Bachelorarbeit erfüllt werden.

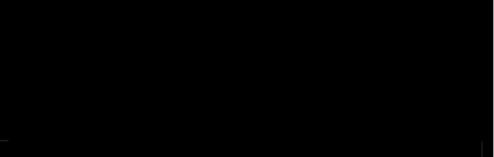
- Das Tool ist einfach in bestehende Webseiten zu integrieren.
- Das Tool implementiert den Arbeitsablauf 'Human in the Loop'.
- Das Tool implementiert alle nötigen Funktionen für eine Segmentierung von Gebäudeplänen.
- Der Machine-Learner kann einfach ausgetauscht werden.
- Das Frontend nutzt React.
- Es existiert eine Schnittstelle, um ein Data-Versioning-System anbinden zu können.
- Es können Grundrisspläne gelabelt werden.
- Das Labeln wird von einem Machine-Learner unterstützt. Die Qualität der Segmentierung ist zu diesem Zeitpunkt noch nicht abzuschätzen.
- Das Projekt wird unter der Open-Source-Lizenz veröffentlicht.

Die Verantwortung für ein funktionstüchtiges Frontend wird grösstenteils von [REDACTED] übernommen, wohingegen die Funktionalität des Backends hauptsächlich von [REDACTED] übernommen wird. Bei der Bildsegmentierung sind beide zu gleichen Teilen verantwortlich. Im Zeitplan (siehe Kapitel I) sind die genauen Verantwortlichkeiten definiert.

## C. Eidestattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbstständig und nur unter Verwendung der von uns angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Datum: \_\_\_\_\_ Unterschrift: 

Datum: \_\_\_\_\_ Unterschrift: 

## D. Testbericht Backend und Image-Segmentation

```
=====
test session starts =====
platform win32 -- Python 3.10.0, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\[REDACTED] Documents\Projekte\Bachelorarbeit\00_src\backend
plugins: asyncio-3.6.1, cov-3.0.0
collected 100 items

tests\Data\COCO_JSON\test_coco_handler.py ... [ 3%]
tests\Data\COCO_JSON\test_coco_parser.py ..... [ 9%]
tests\ImageSegmentation\test_predictor.py ..... [15%]
tests\ImageSegmentation\modelControl\test_custom_kearas_objects.py .... [19%]
tests\ImageSegmentation\modelControl\test_model_version_control.py .... [24%]
...
tests\ImageSegmentation\modelControl\test_retrainer.py ..... [27%]
tests\ImageSegmentation\processing\test_postprocess.py ..... [32%]
tests\ImageSegmentation\processing\test_preprocess.py ..... [44%]
tests\REST_Server\database\test_crud.py ..... [52%]
tests\REST_Server\endpoints\test_general.py ... [74%]
tests\REST_Server\endpoints\test_imageSegmentation.py ..... [77%]
tests\REST_Server\endpoints\test_metrics.py ..... [89%]
tests\REST_Server\evaluation\test_metrics.py ..... [100%]

----- coverage: platform win32, python 3.10.0-final-0 -----
Name                               Stmts  Miss  Cover
-----
C:\[REDACTED]\AppData\Local\Temp\__autograph_generated_file45yfsemq.py    17      3   82%
C:\[REDACTED]\AppData\Local\Temp\__autograph_generated_filealuhykqb.py    17      3   82%
C:\[REDACTED]\AppData\Local\Temp\__autograph_generated_filecwtpp51a.py    17      3   82%
Data\COCO_JSON\cocoHandler.py          24      0  100%
Data\COCO_JSON\cocoParser.py         25      0  100%
ImageSegmentation\Predictor.py       70      1   99%
ImageSegmentation\imageSegmentationInterface.py 14      3   79%
ImageSegmentation\modelControl\ModelVersionControl.py 56      3   95%
ImageSegmentation\modelControl\customKearasObjects.py 17      0  100%
ImageSegmentation\modelControl\retrainer.py     53      0  100%
ImageSegmentation\processing\postprocess.py 60      0  100%
ImageSegmentation\processing\preprocess.py    23      0  100%
REST_Server\config.py                1       0  100%
REST_Server\database\crud.py        115      0  100%
REST_Server\database\database.py    13       4   69%
REST_Server\database\tables.py     39       0  100%
REST_Server\endpoints\general.py   42       1   98%
REST_Server\endpoints\imageSegmentation.py 122      6   95%
REST_Server\evaluation\metrics.py   50       0  100%
REST_Server\evaluation\plots.py    104      1   99%
REST_Server\main.py                 21       1   95%
REST_Server\models\pydanticSchemas.py 56       5   91%
REST_Server\models\sqlAlchemyTypes.py 15       0  100%
tests\Data\COCO_JSON\test_coco_handler.py 19       0  100%
tests\Data\COCO_JSON\test_coco_parser.py 55       0  100%
tests\ImageSegmentation\modelControl\test_custom_kearas_objects.py 19       0  100%
tests\ImageSegmentation\modelControl\test_model_version_control.py 70       0  100%
tests\ImageSegmentation\modelControl\test_retrainer.py    99      0  100%
tests\ImageSegmentation\processing\test_postprocess.py 92       0  100%
tests\ImageSegmentation\processing\test_preprocess.py   50      0  100%
tests\ImageSegmentation\test_predictor.py   41      0  100%
tests\REST_Server\clientFastApi.py     30       0  100%
tests\REST_Server\database\dummyDatabase.py 20       0  100%
tests\REST_Server\database\test_crud.py   196      0  100%
tests\REST_Server\endpoints\test_general.py 17       0  100%
tests\REST_Server\endpoints\test_imageSegmentation.py 72       0  100%
tests\REST_Server\evaluation\test_metrics.py 68       0  100%

TOTAL                               1819     34   98%
```

===== 100 passed in 94.16s (0:01:34) =====

## E. Schemas der REST-Schnittstelle

### Antwortschema für den '/' Endpunkt

```
1 "string"
```

### Antwortschema für den '/heartbeat' Endpunkt

```
1 ""
```

### Antwortschema für den '/dashboard' Endpunkt

```
1 "string"
```

### Antwortschema für den '/getCategories' Endpunkt

```
1 {
2   "categories": [
3     {
4       "id": 0,
5       "name": "string"
6     }
7   ]
8 }
```

## Anforderungsschema für den '/uploadImage' Endpunkt

```
1  {
2      "file": "string <binary>"
3 }
```

## Antwortschema für den '/uploadImage' Endpunkt

```
1  {
2      "cocoJSON": {
3          "info": {
4              "description": "string",
5              "ModelVersion": "string"
6          },
7          "categories": [
8              {
9                  "id": 0,
10                 "name": "string"
11             }
12         ],
13         "images": [
14             {
15                 "id": 0,
16                 "file_name": "string",
17                 "height": 0,
18                 "width": 0
19             }
20         ],
21         "annotations": [
22             {
23                 "id": 0,
24                 "image_id": 0,
25                 "category_id": 0,
26                 "area": 0,
27                 "segmentation": [
28                     0
29                 ]
30             }
31         ],
32         "token": "string"
33     }
34 }
```

## Anforderungsschema für den '/uploadCorrection' Endpunkt

```
1  {
2      "cocoJSON": {
3          "info": {
4              "description": "string",
5              "ModelVersion": "string"
6          },
7          "categories": [
8              {
9                  "id": 0,
10                 "name": "string"
11             }
12         ],
13         "images": [
14             {
15                 "id": 0,
16                 "file_name": "string",
17                 "height": 0,
18                 "width": 0
19             }
20         ],
21         "annotations": [
22             {
23                 "id": 0,
24                 "image_id": 0,
25                 "category_id": 0,
26                 "area": 0,
27                 "segmentation": [
28                     0
29                 ]
30             }
31         ],
32     },
33     "token": "string"
34 }
```

## Antwortschema für den '/uploadCorrection' Endpunkt

```
1  ""
```

**Anforderungsschema für den '/uploadNoCorrection' Endpunkt**

```
1  {
2      "token": "string"
3 }
```

**Antwortschema für den '/uploadNoCorrection' Endpunkt**

```
1 """
```

## F. Aufbau des COCO-JSON Formates

```
1 {  
2   "info" :info,  
3   "images" :[image],  
4   "annotations" :[annotation],  
5   "categories" :[categorie],  
6   "licenses" :[license],  
7 }
```

Der grundlegende Aufbau eines COCO-JSON.

```
1 info {
2     "year" :int,
3     "version" :str,
4     "description" :str,
5     "contributor" :str,
6     "url" :str,
7     "date_created" :datetime,
8 }
9
10 image {
11     "id" :int,
12     "width" :int,
13     "height" :int,
14     "file_name" :str,
15     "license" :int,
16     "flickr_url" :str,
17     "coco_url" :str,
18     "date_captured" :datetime,
19 }
20
21 annotation {
22     "id" :int,
23     "image_id" :int,
24     "category_id" :int,
25     "segmentation" :[x1,y1,x2,y2,... (polygon)],
26 }
27
28 categorie {
29     "id" :int,
30     "name" :str,
31     "supercategory" :str,
32 }
33
34 license {
35     "id" :int,
36     "name" :str,
37     "url" :str,
38 }
```

Die einzelnen Bestandteile eines COCO-JSONs im Detail.

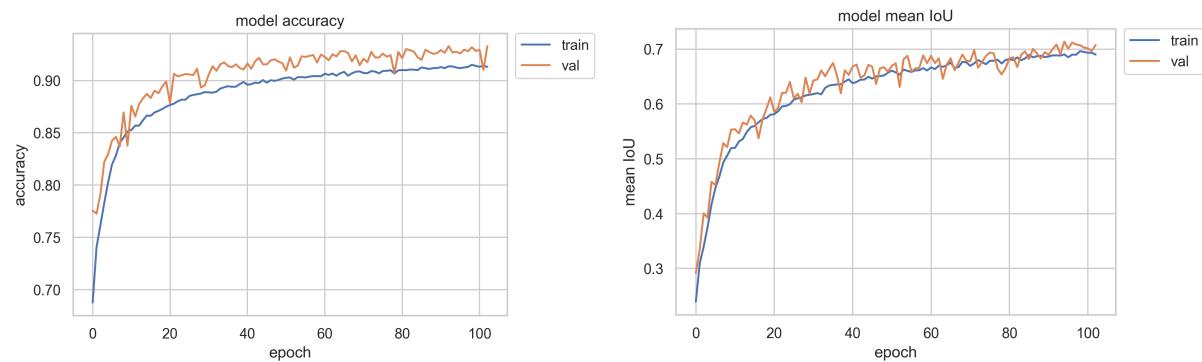
## G. Image-Segmentation Katzen

- Bezeichnung: Die Bezeichnung des jeweiligen Modelles.
- Hyperparameter: Welcher Parameter oder Einstellung wurde bei dieser Version untersucht.
- Dim.: Welche Dimension haben die Input- und Output-Tensoren des Modells.
- Train. Parameter: Wieviele Parameter können im Modell trainiert werden (trainable parameter).
- Train. Epochen: Wieviele Epochen lang wurde das Modell trainiert bis das Early-Stopping über den Wert der Verlustfunktion ausgelöst wird.
- Val. Acc.: Die Accuracy auf den Validierungsdaten.
- Val. Loss: Der Loss auf den Validierungsdaten.
- Val. M.-IOU: Die Mean-Intersection-Over-Union auf den Validierungsdaten.
- th: threshold, Schwellwert für die Aktivierungsfunktion

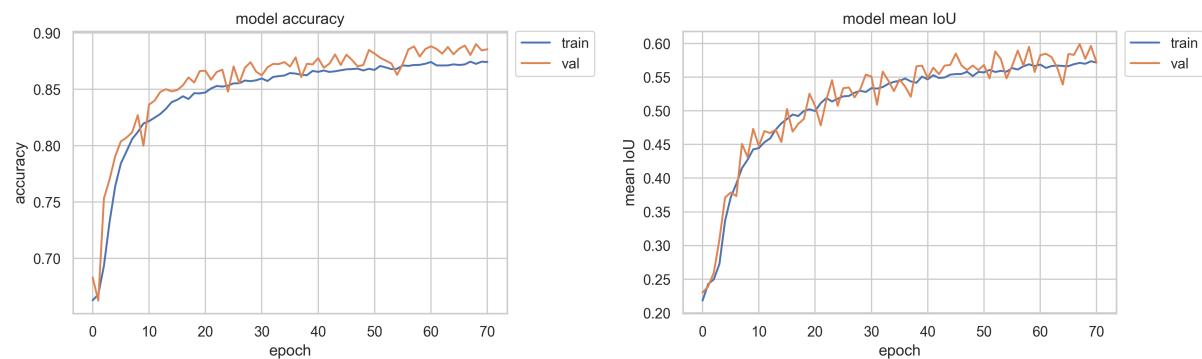
### Modellübersicht

Bezeichnung	Architektur	Dim.	Train. Parameter	Train. Epochen	Val. Acc.	Val. Loss	Val. M.-IOU
CatVersion1	U-Net	128x128	1'941'105	103	0.933	0.172	0.708
CatVersion2	HR-Net	128x128	141'057	71	0.886	0.269	0.572
CatVersion3	FastFCN	128x128	1'891'361	67	0.926	0.185	0.690

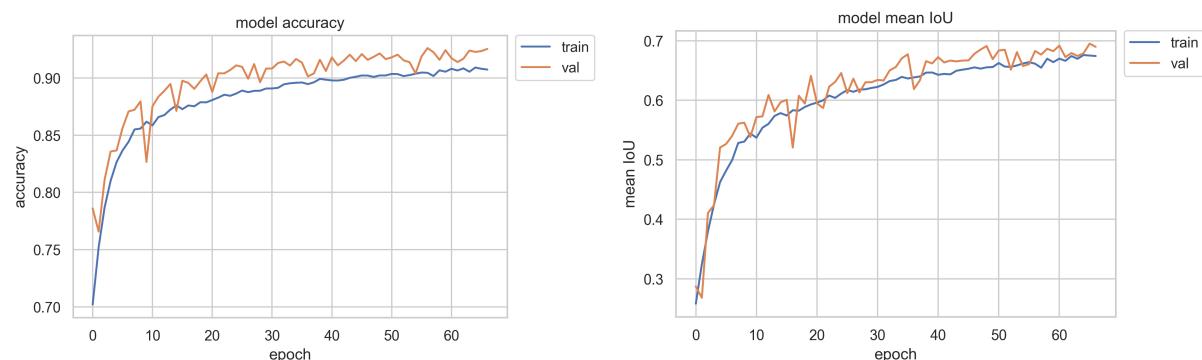
## Vergleich Metriken



CatVersion1

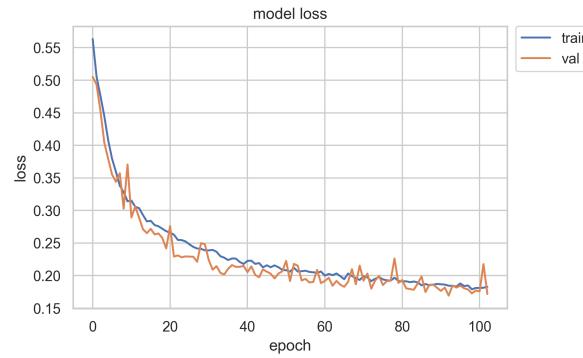


CatVersion2

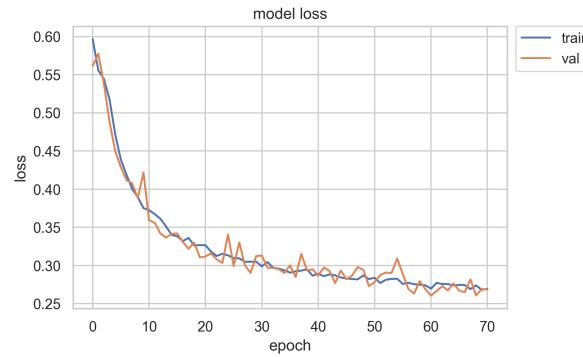


CatVersion3

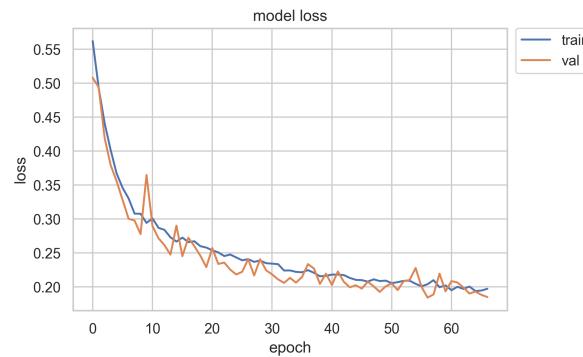
## Vergleich Verlustfunktion



CatVersion1

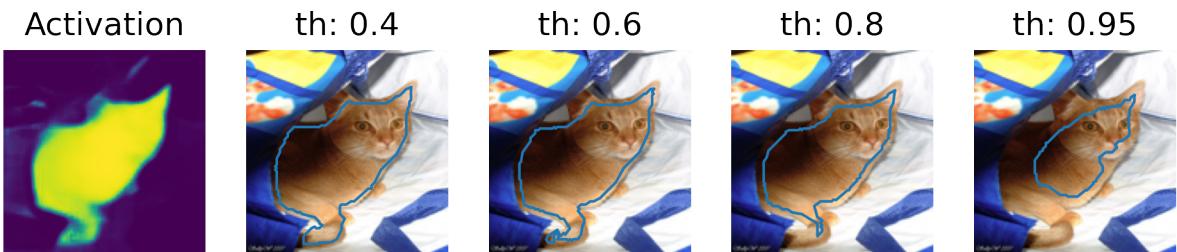


CatVersion2

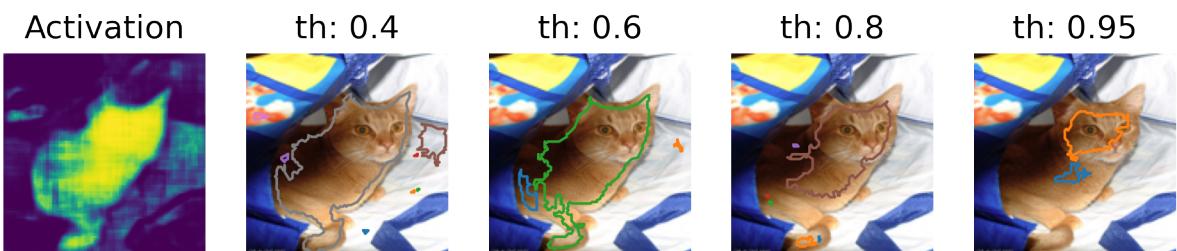


CatVersion3

## Qualitative Vergleiche



CatVersion1

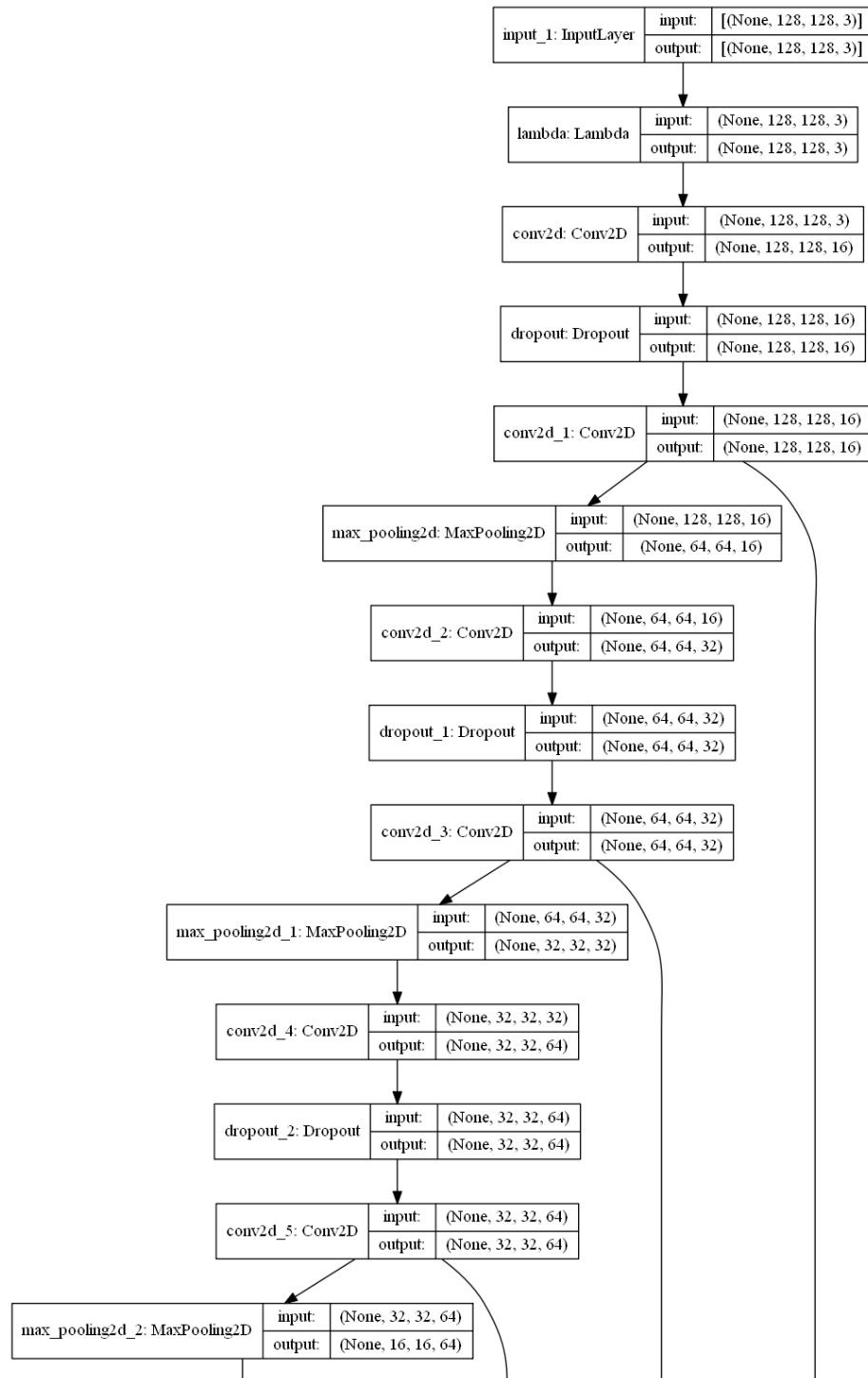


CatVersion2

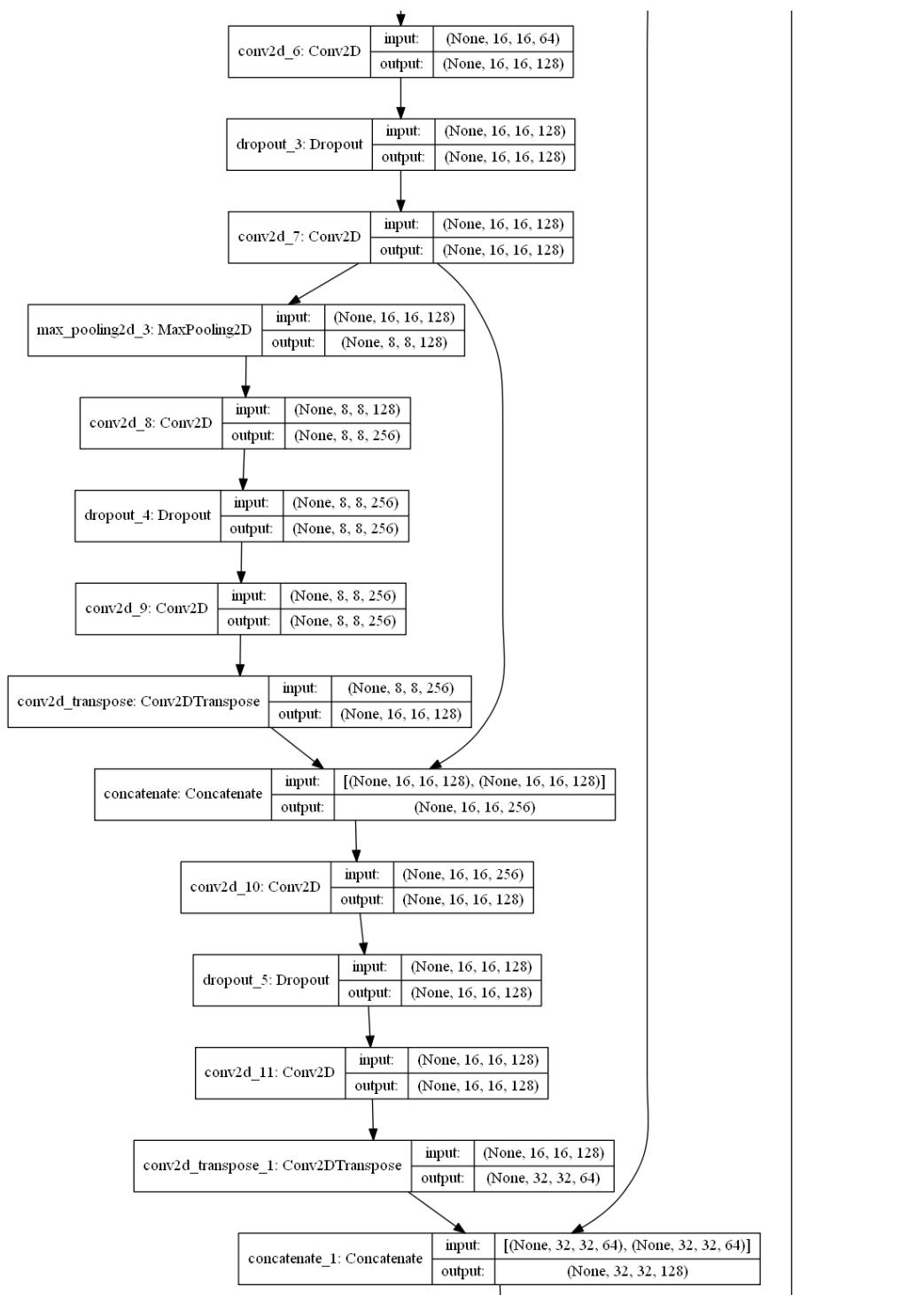


CatVersion3

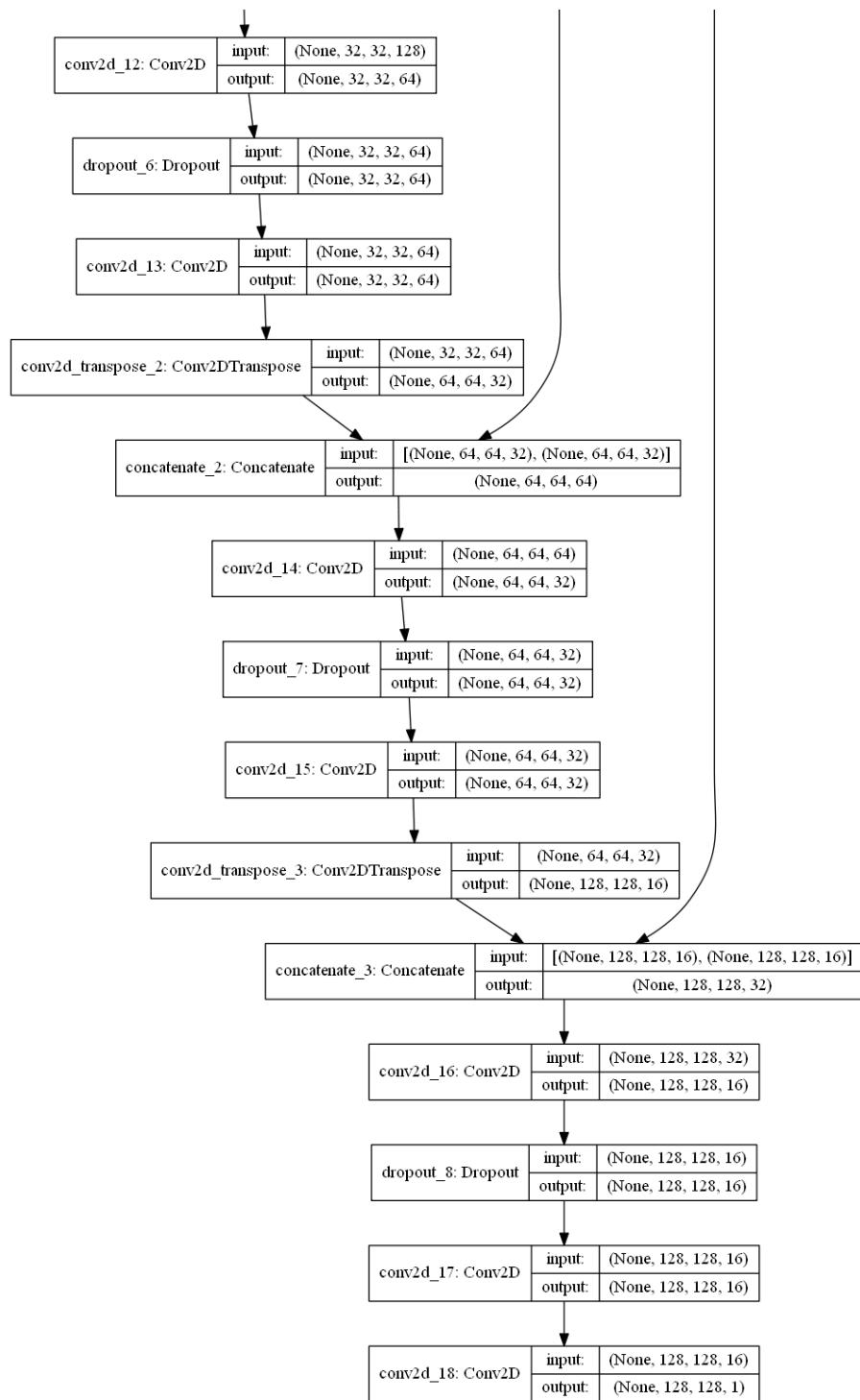
## Modellarchitekturen



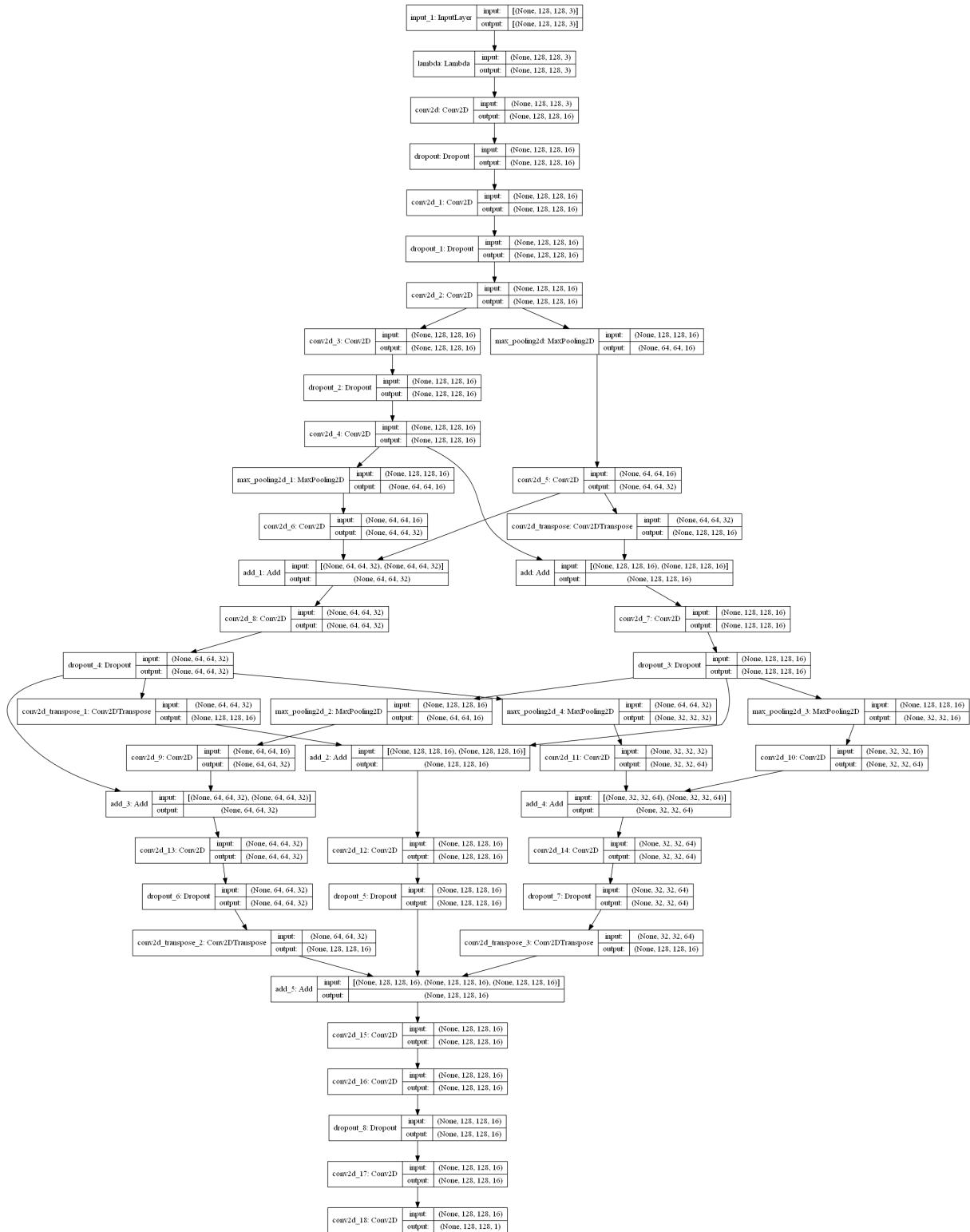
CatVersion1 (Teil 1 von 3)



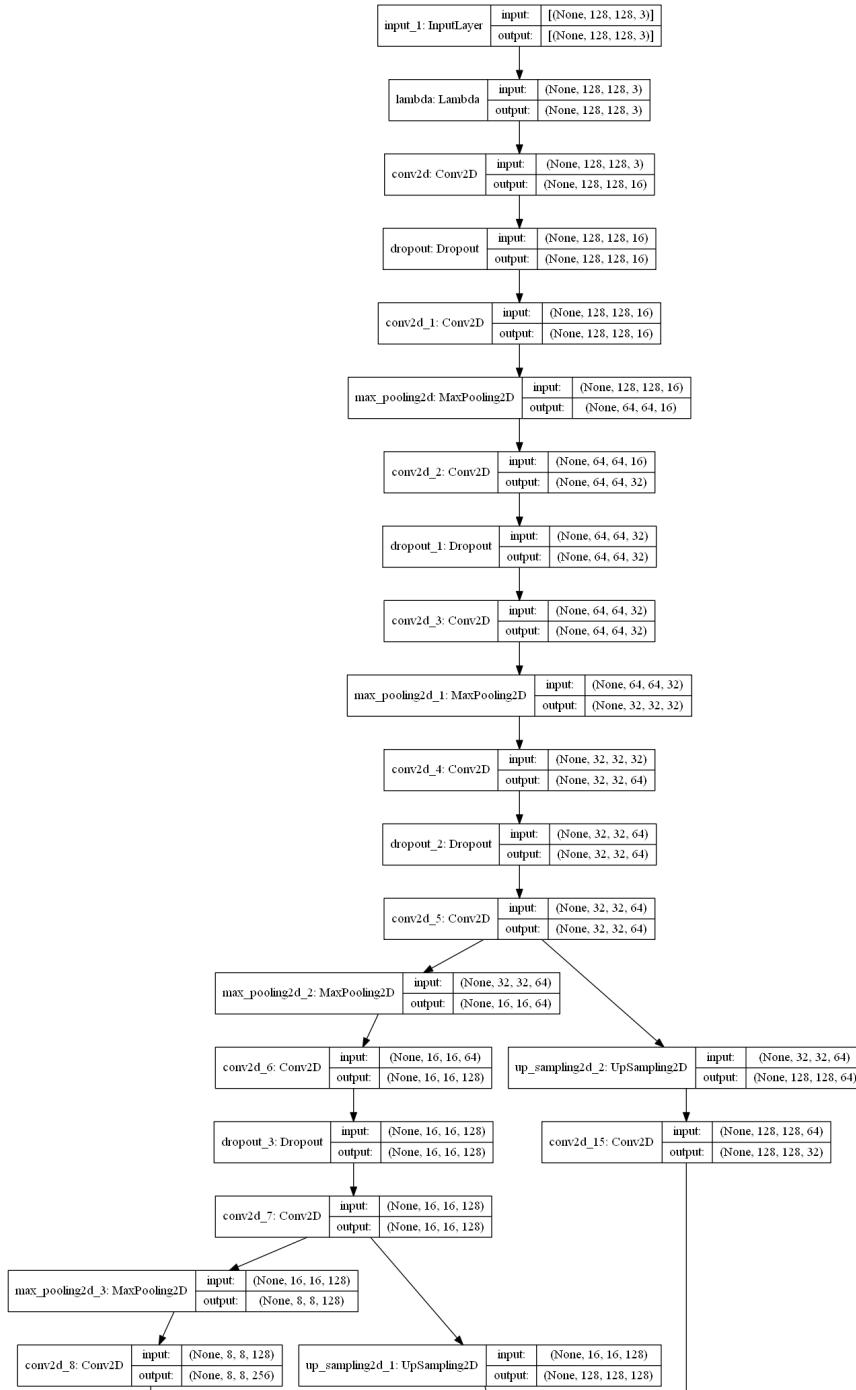
CatVersion1 (Teil 2 von 3)



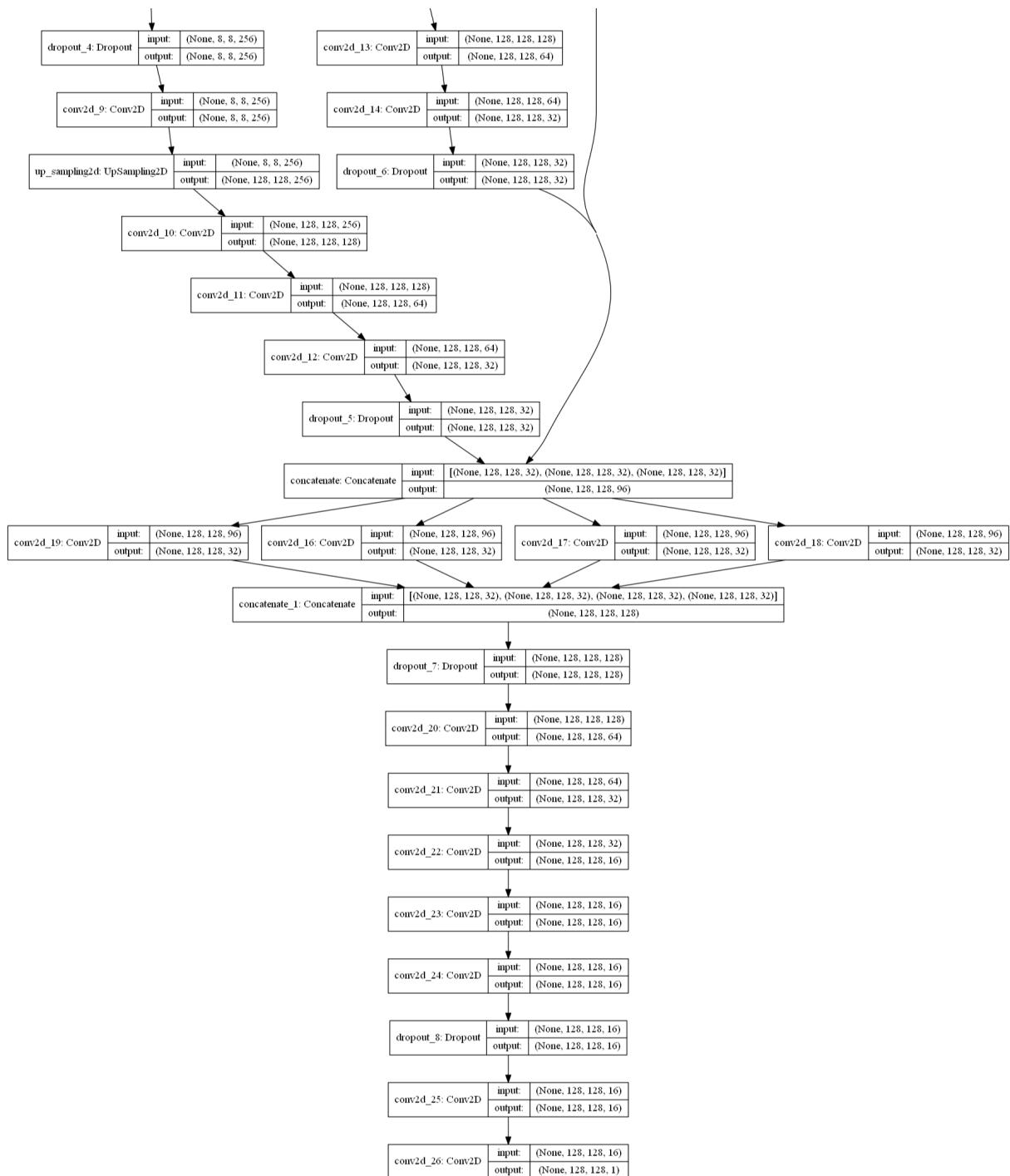
CatVersion1 (Teil 3 von 3)



CatVersion2



CatVersion3 (Teil 1 von 2)



CatVersion3 (Teil 2 von 2)

## H. Image-Segmentation Floorplans

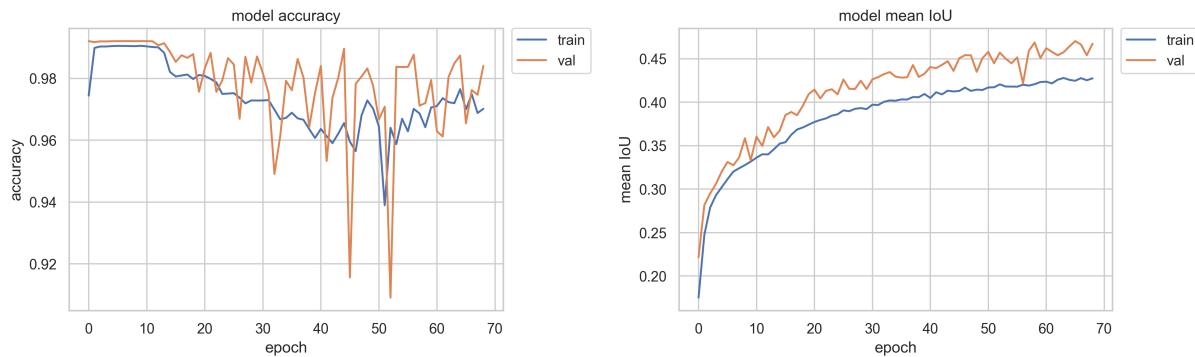
- Bezeichnung: Die Bezeichnung des jeweiligen Modelles.
- Hyperparameter: Welcher Parameter oder Einstellung wurde bei dieser Version untersucht.
- Dim.: Welche Dimension haben die Input- und Output-Tensoren des Modells.
- Train. Parameter: Wieviele Parameter können im Modell trainiert werden (trainable parameter).
- Train. Epochen: Wieviele Epochen lang wurde das Modell trainiert bis das Early-Stopping über den Wert der Verlustfunktion ausgelöst wird.
- Val. Acc.: Die Accuracy auf den Validierungsdaten.
- Val. Loss: Der Loss auf den Validierungsdaten.
- Val. M.-IOU: Die Mean-Intersection-Over-Union auf den Validierungsdaten.
- th: threshold, Schwellwert für die Aktivierungsfunktion

### Modellübersicht

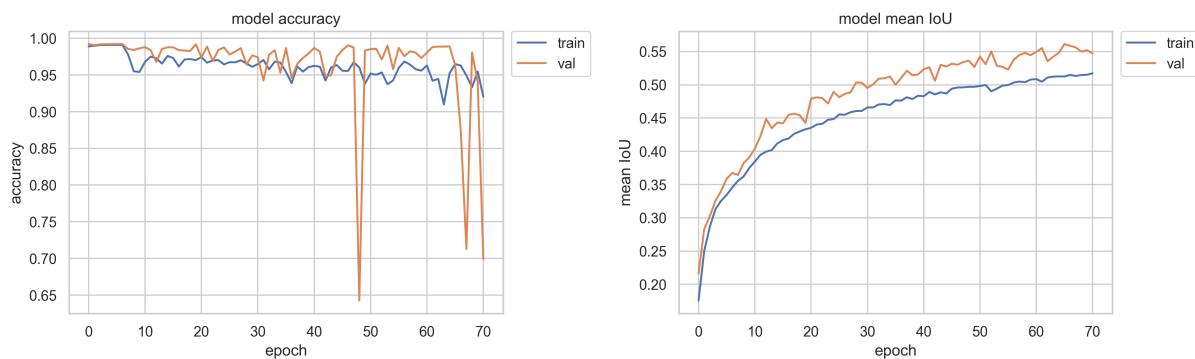
Bezeichnung	Hyperparameter	Dim.	Train. Parameter	Train. Epochen	Val. Acc.	Val. Loss	Val. M.-IOU
FpVersion1	Baseline-Model	128x128	1'941'139	69	0.984	0.069	0.467
FpVersion2_10	Dimension verändert	256x256	1'941'139	71	0.940	0.059	0.552
FpVersion3	Nutzung 3 Decoderpfade	256x256	3'465'779	45	0.992	0.098	nan
FpVersion4	Nutzung 2 Decoderpfade	256x256	2'703'459	50	0.992	0.097	nan
FpVersion5	Dimension verändert	512x512	1'941'139	61	0.984	0.081	0.544
FpVersion6	'binary-crossentropy-loss' statt 'Dice-loss'	256x256	1'941'139	52	0.982	0.079	0.478
FpVersion7_11	Featuremaps für jede Ebene verdoppelt	256x256	7'760'163	86	0.986	0.056	0.575
FpVersion8	Encoderpfad um eine Ebene erweitert	256x256	7'775'635	116	0.983	0.054	0.571
FpVersion9	Kernelsize von 5x5 statt 3x3	256x256	5'079'443	89	0.965	0.058	0.530

Modelle 2 und 7 lieferten die besten Ergebnisse. Diese wurden für das Backend angepasst und als 10 beziehungsweise 11 erneut trainiert. Es wurden dabei keine Parameteranpassungen vorgenommen und beide Varianten sind deshalb in der Tabelle zusammengefasst.

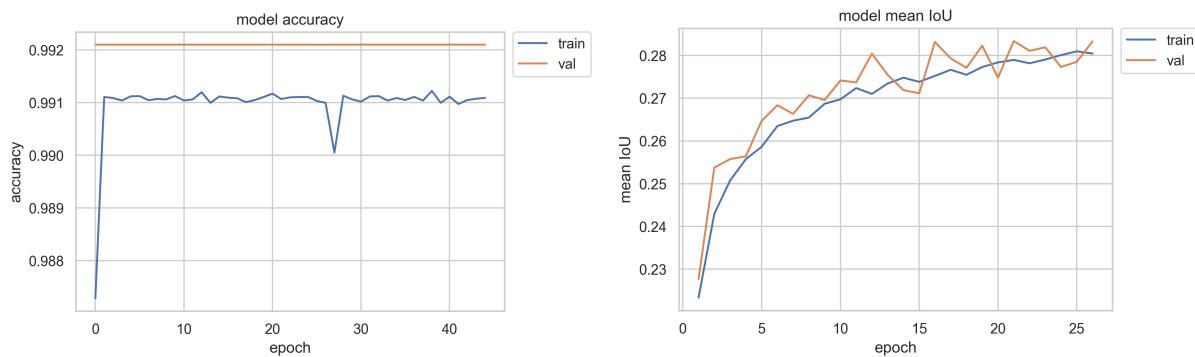
## Vergleich Metriken



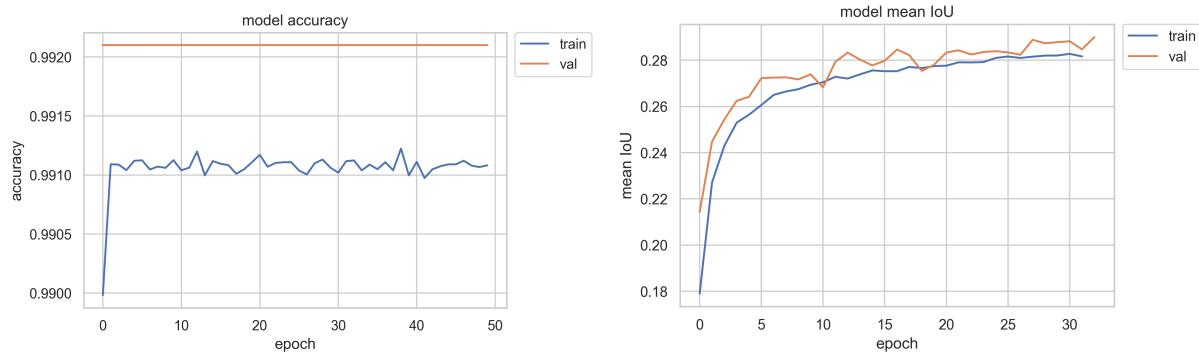
FpVersion1



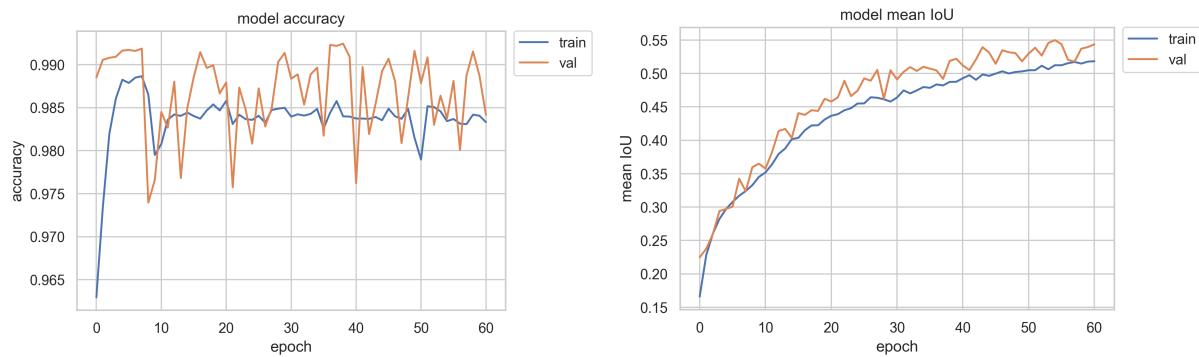
FpVersion2\_10



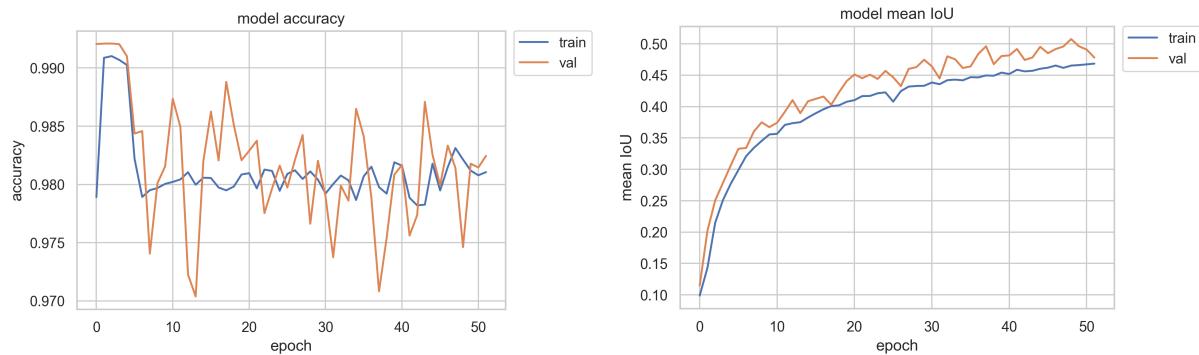
FpVersion3



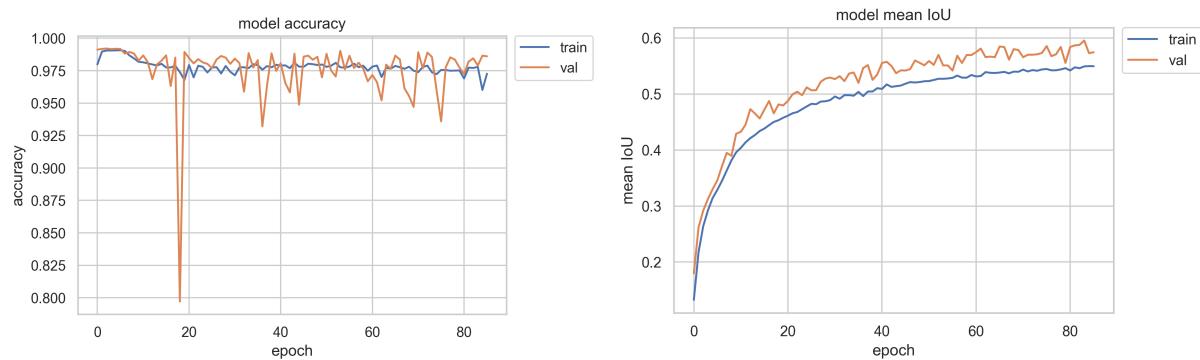
FpVersion4



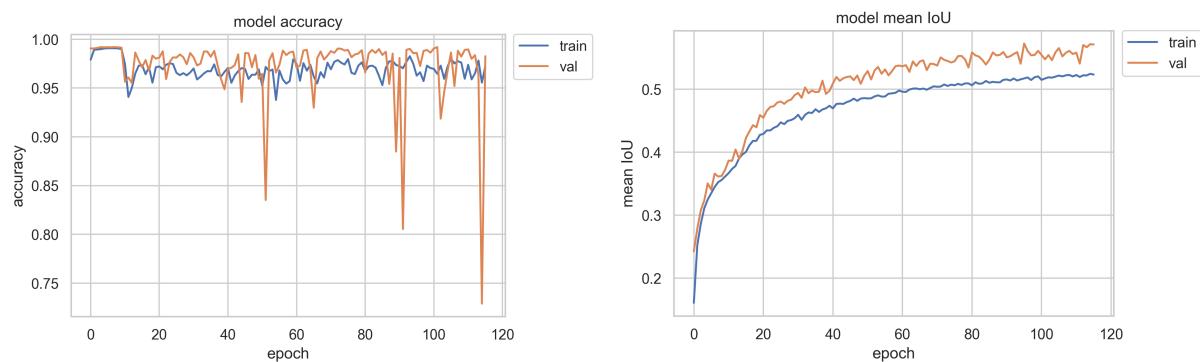
FpVersion5



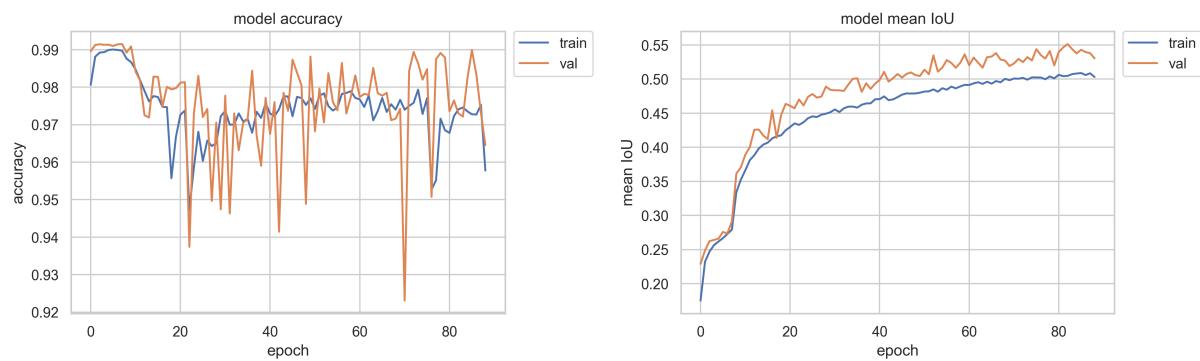
FpVersion6



FpVersion7\_11

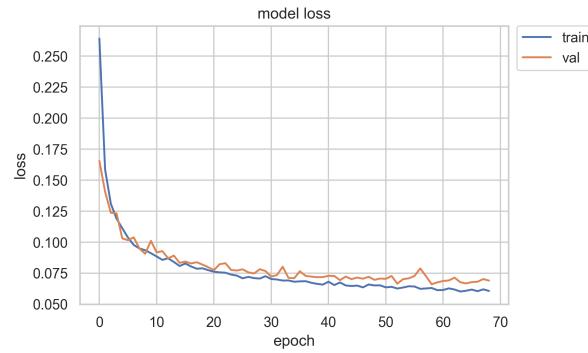


FpVersion8

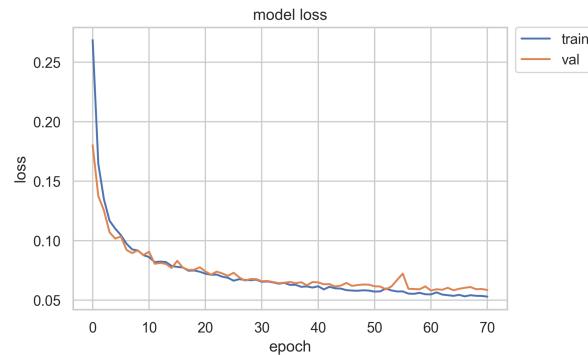


FpVersion9

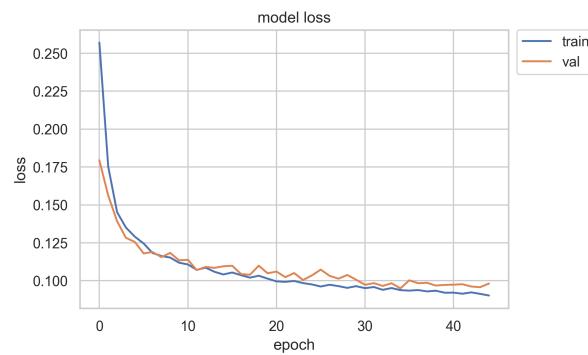
## Vergleich Verlustfunktion



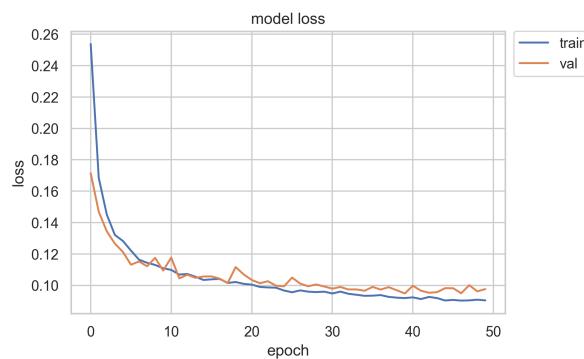
FpVersion1



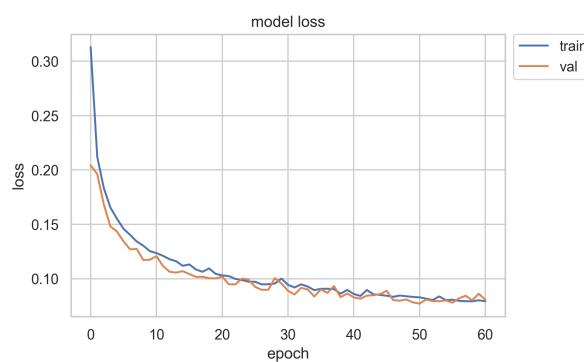
FpVersion2\_10



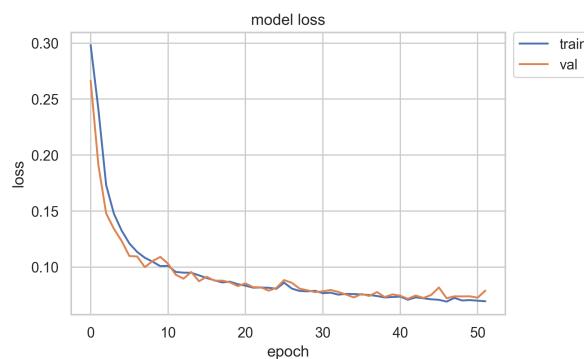
FpVersion3



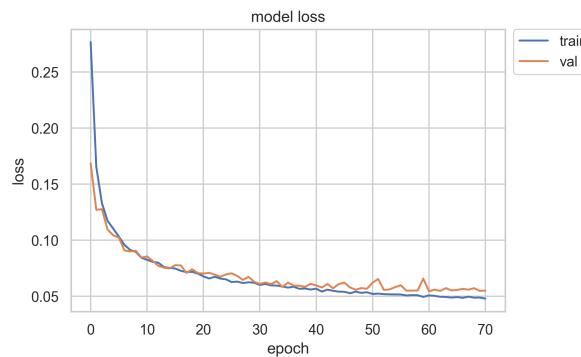
FpVersion4



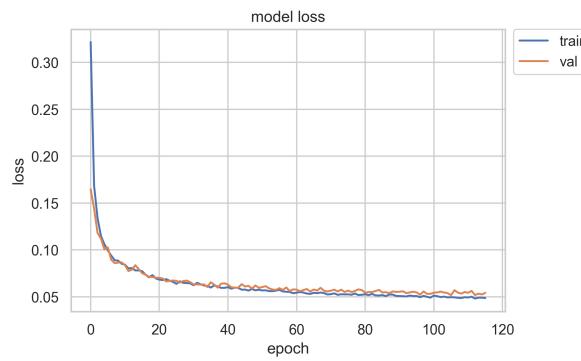
FpVersion5



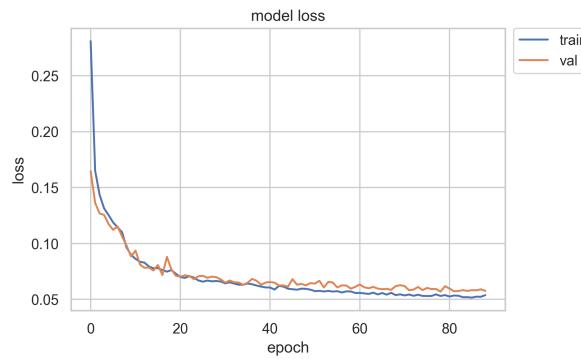
FpVersion6



FpVersion7\_11



FpVersion8



FpVersion9

## Qualitative Vergleiche



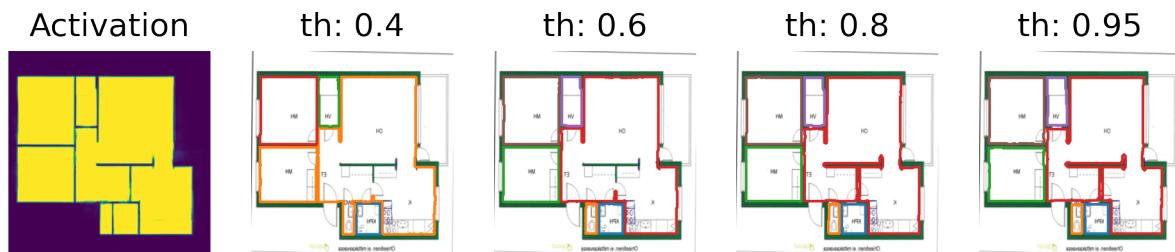
FpVersion1



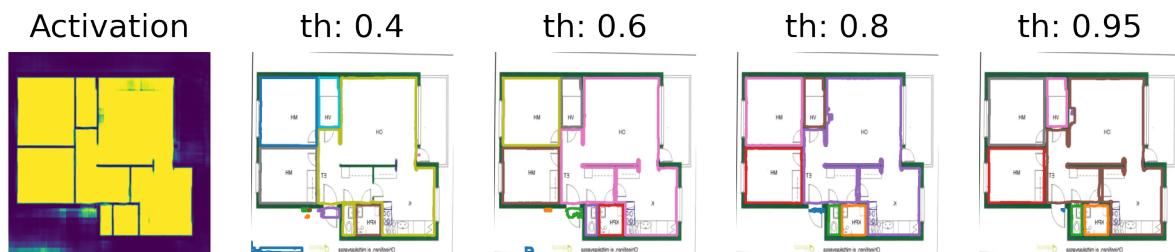
FpVersion2\_10



FpVersion3



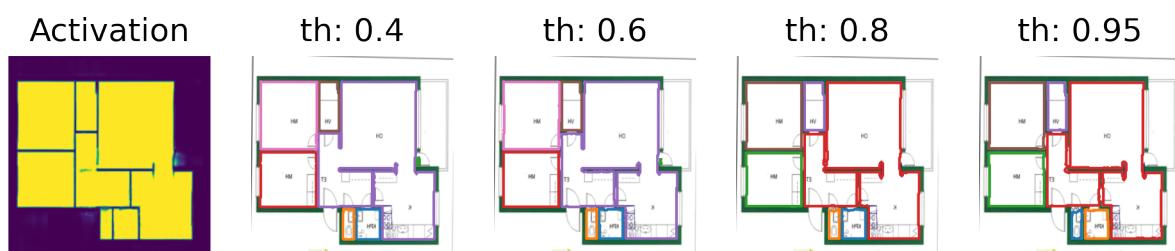
FpVersion4



FpVersion5



FpVersion6



FpVersion7\_11

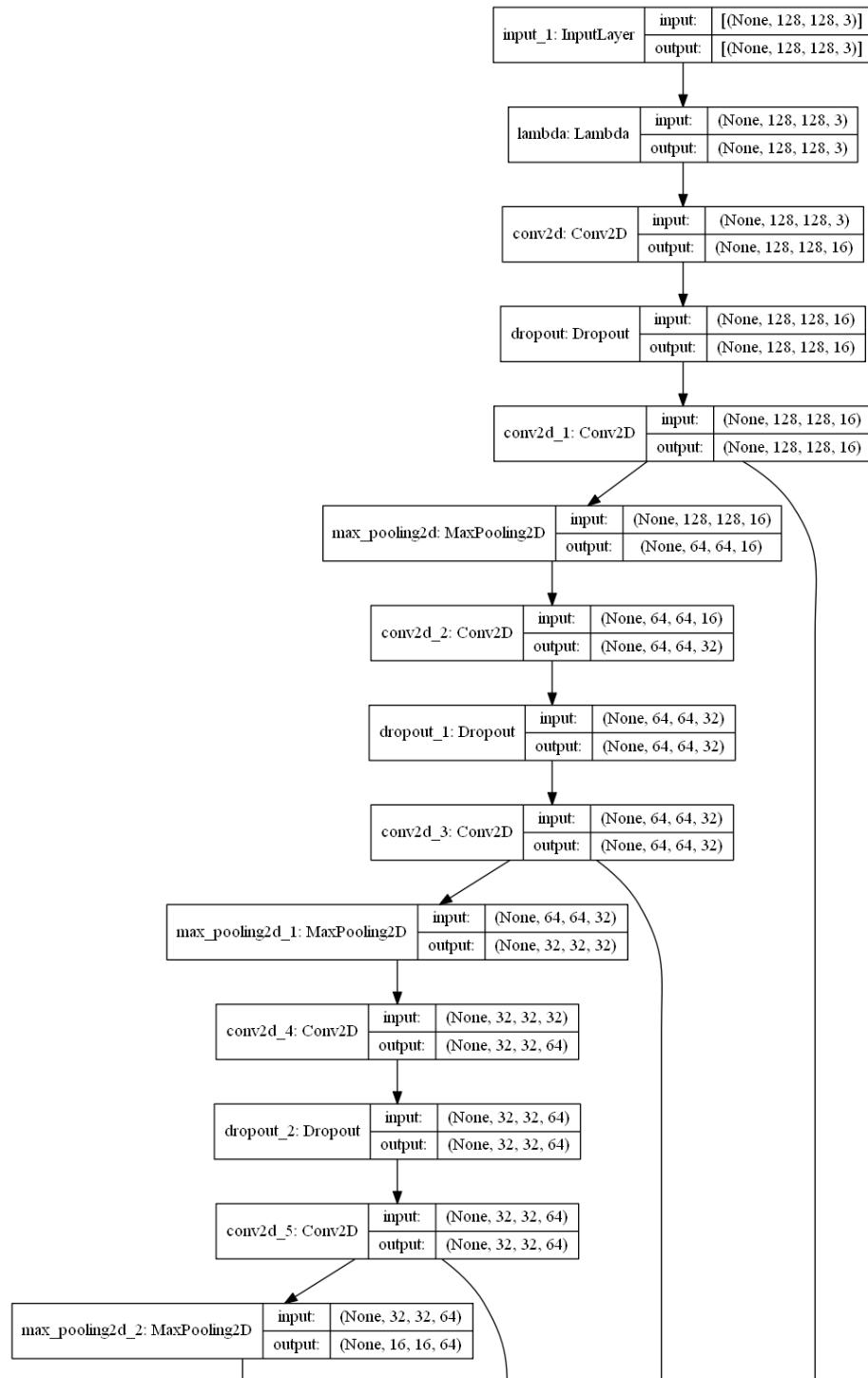


FpVersion8

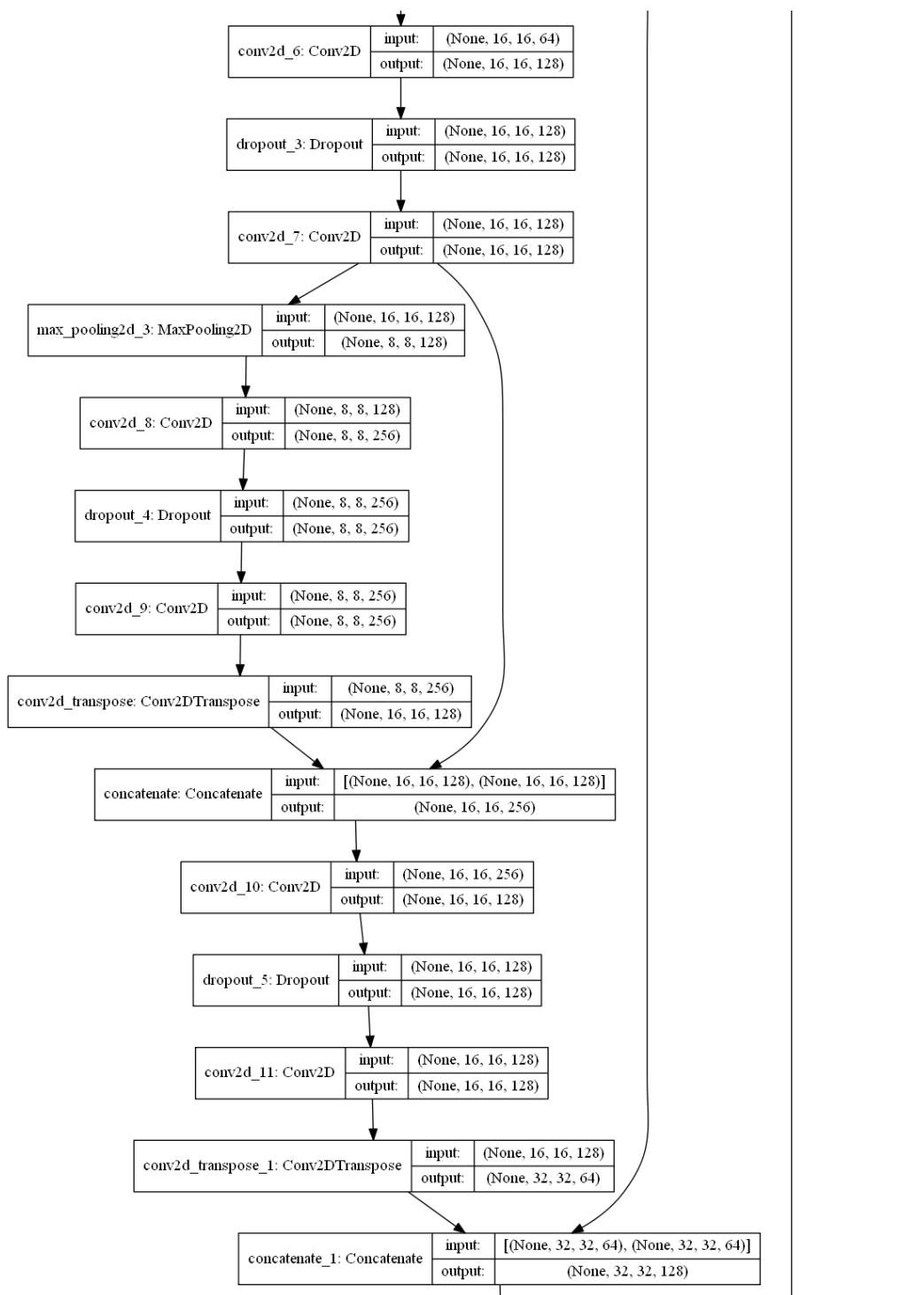


FpVersion9

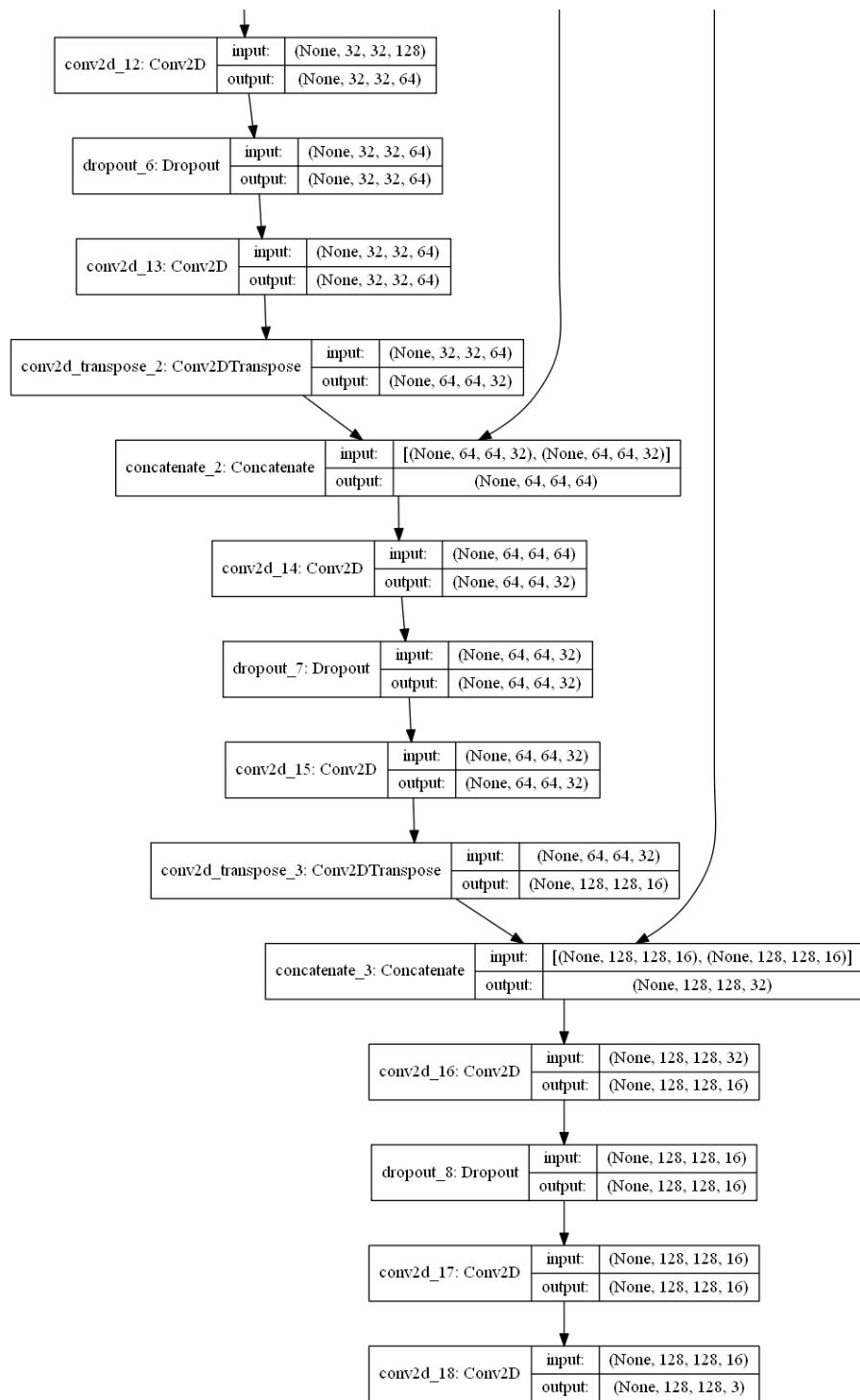
## Modellarchitekturen



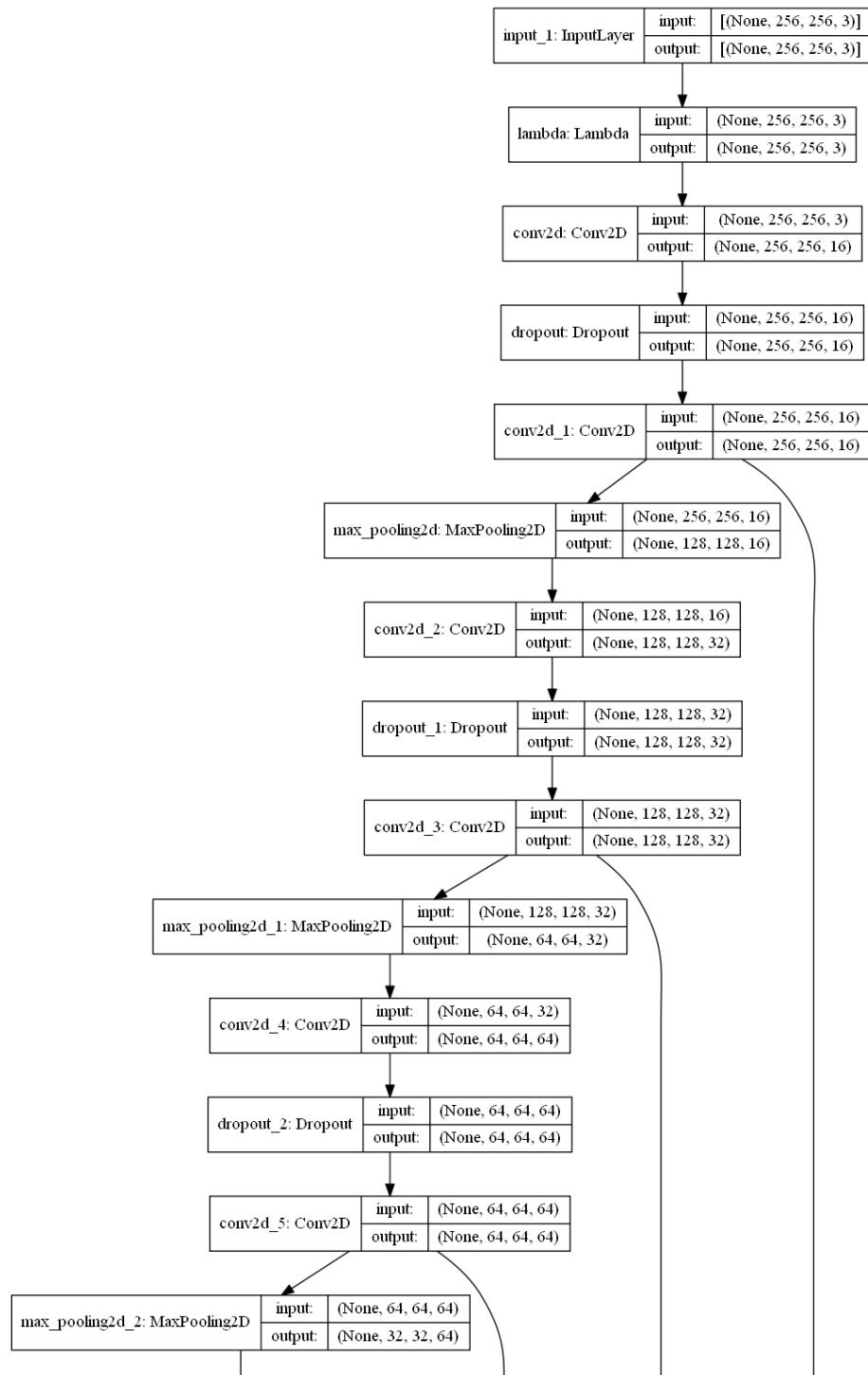
FpVersion1 (Teil 1 von 3)



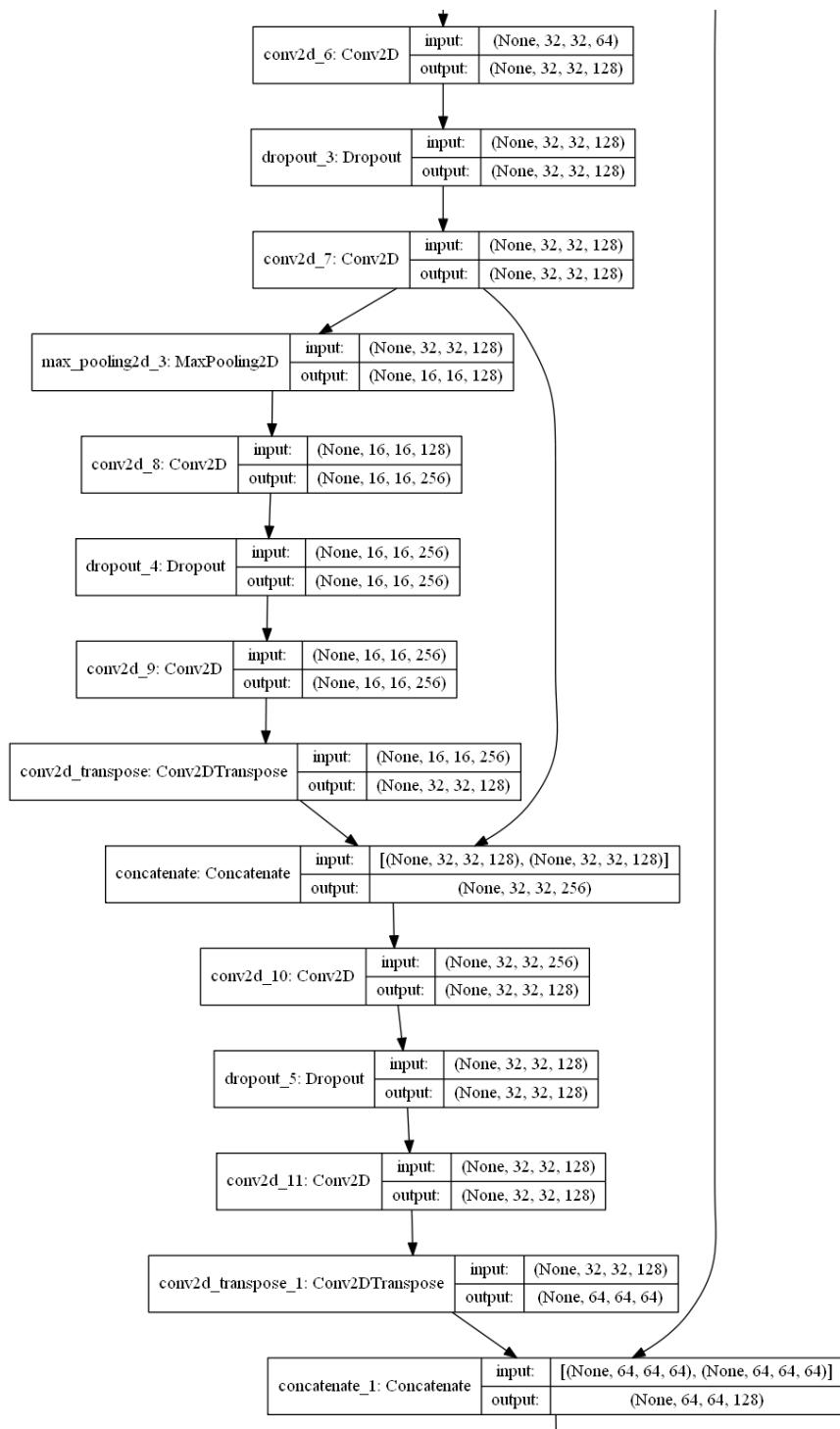
FpVersion1 (Teil 2 von 3)



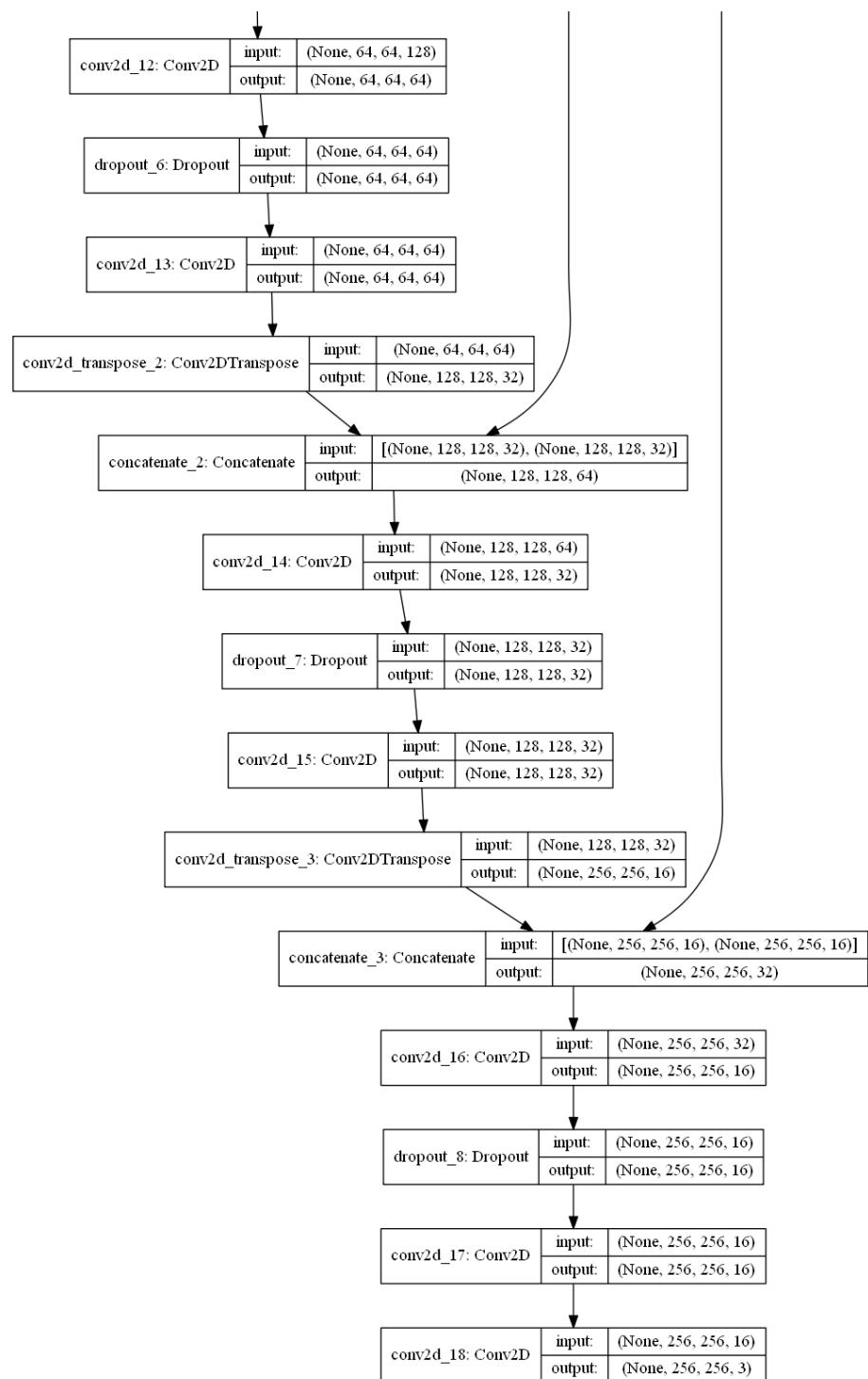
FpVersion1 (Teil 3 von 3)



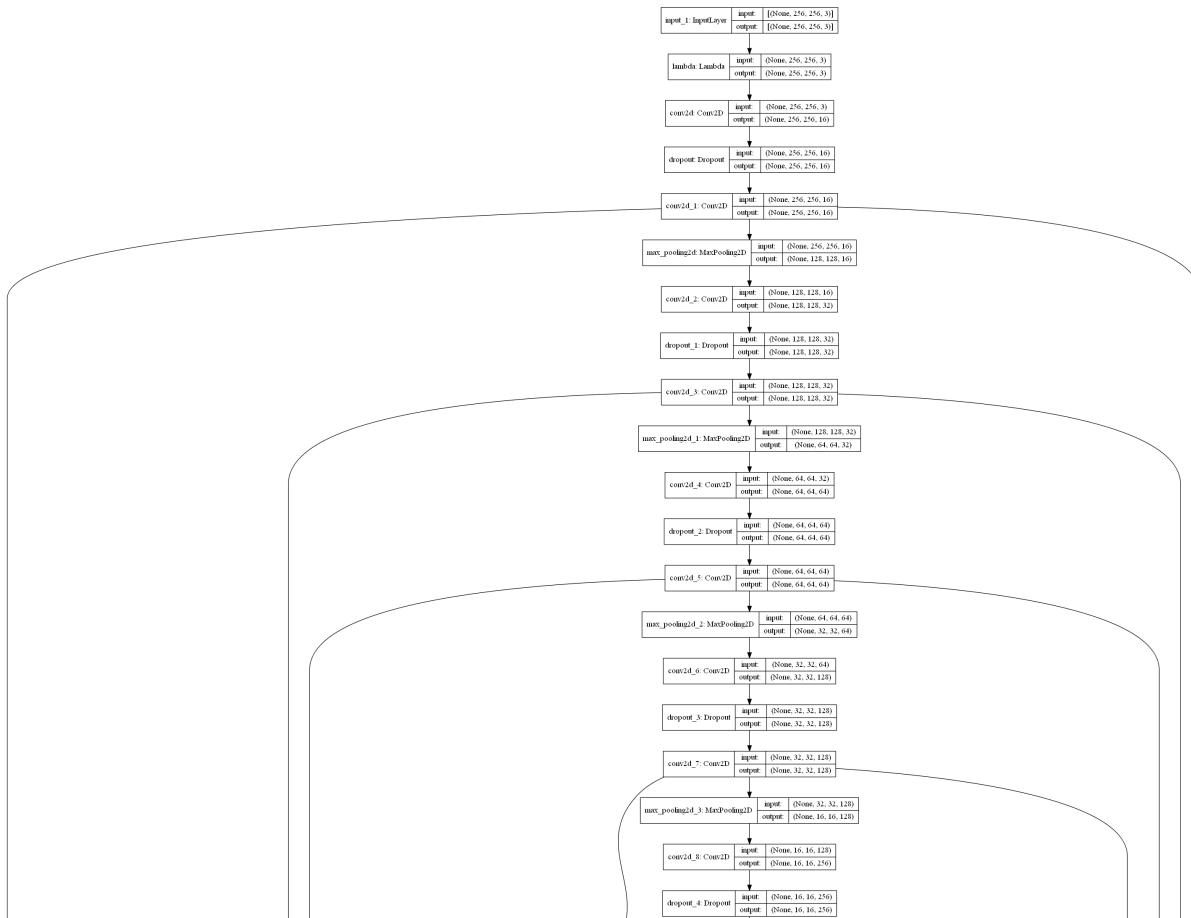
FpVersion2\_10 (Teil 1 von 3)



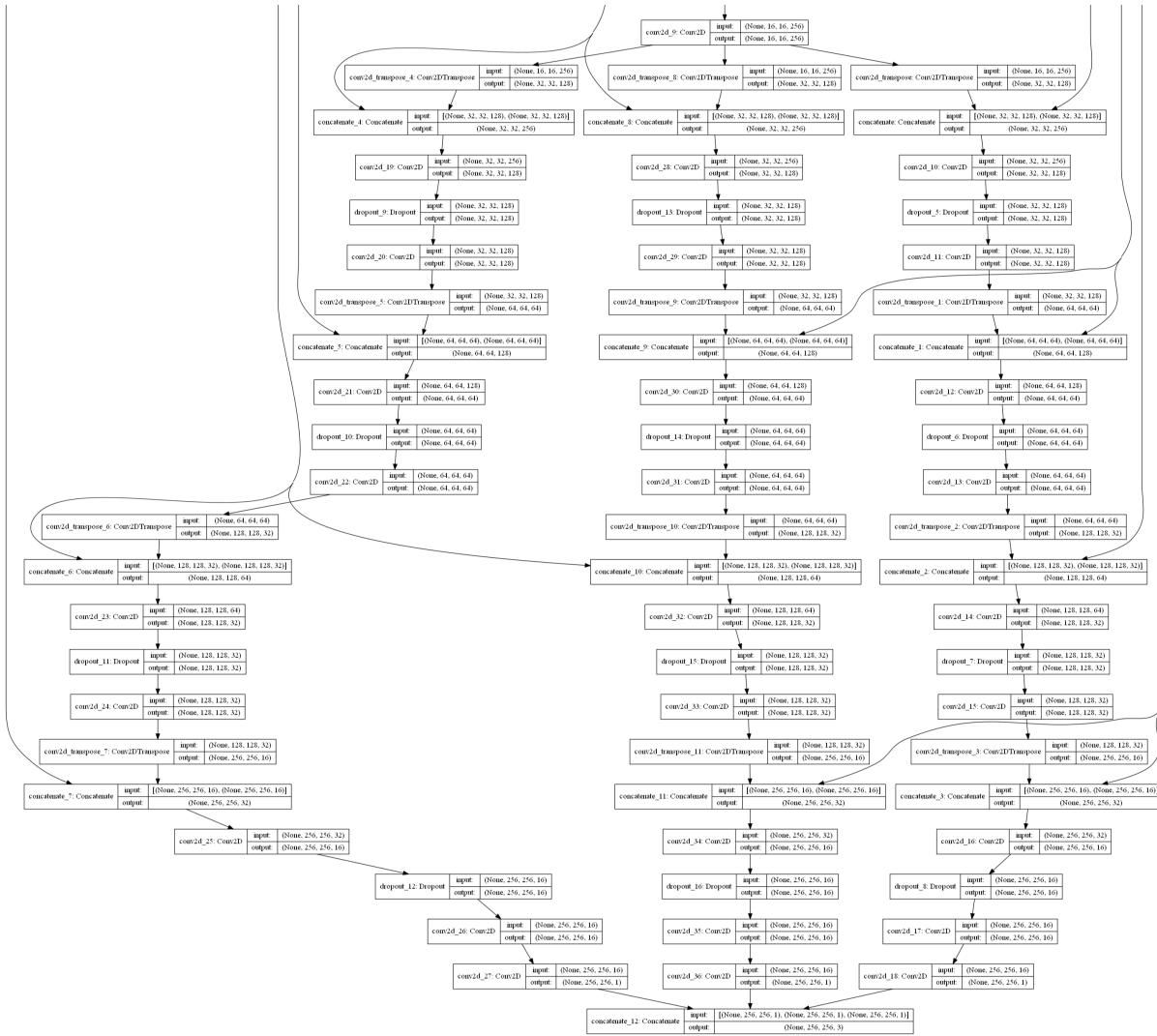
FpVersion2\_10 (Teil 2 von 3)



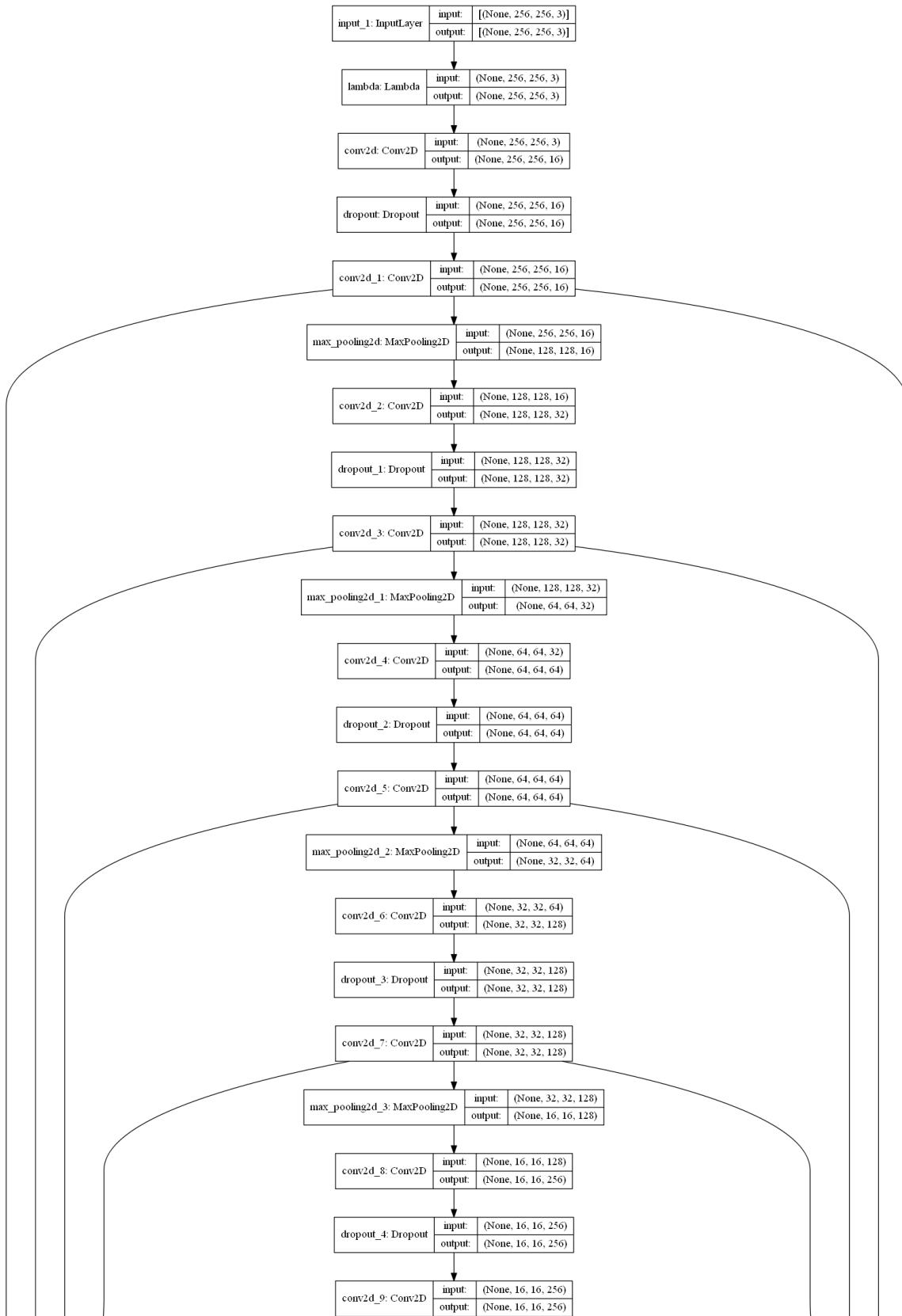
FpVersion2\_10 (Teil 3 von 3)



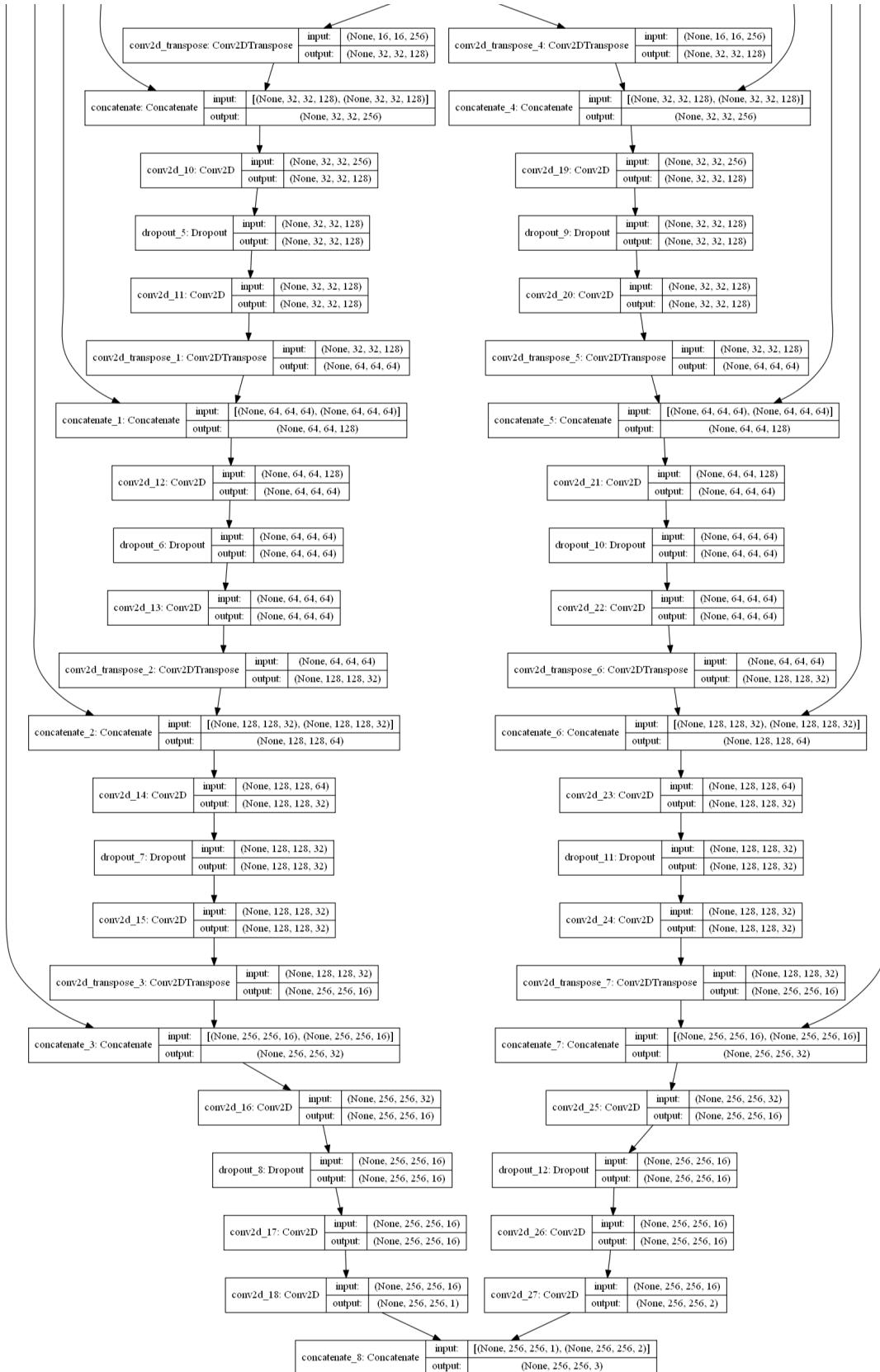
FpVersion3 (Teil 1 von 2)



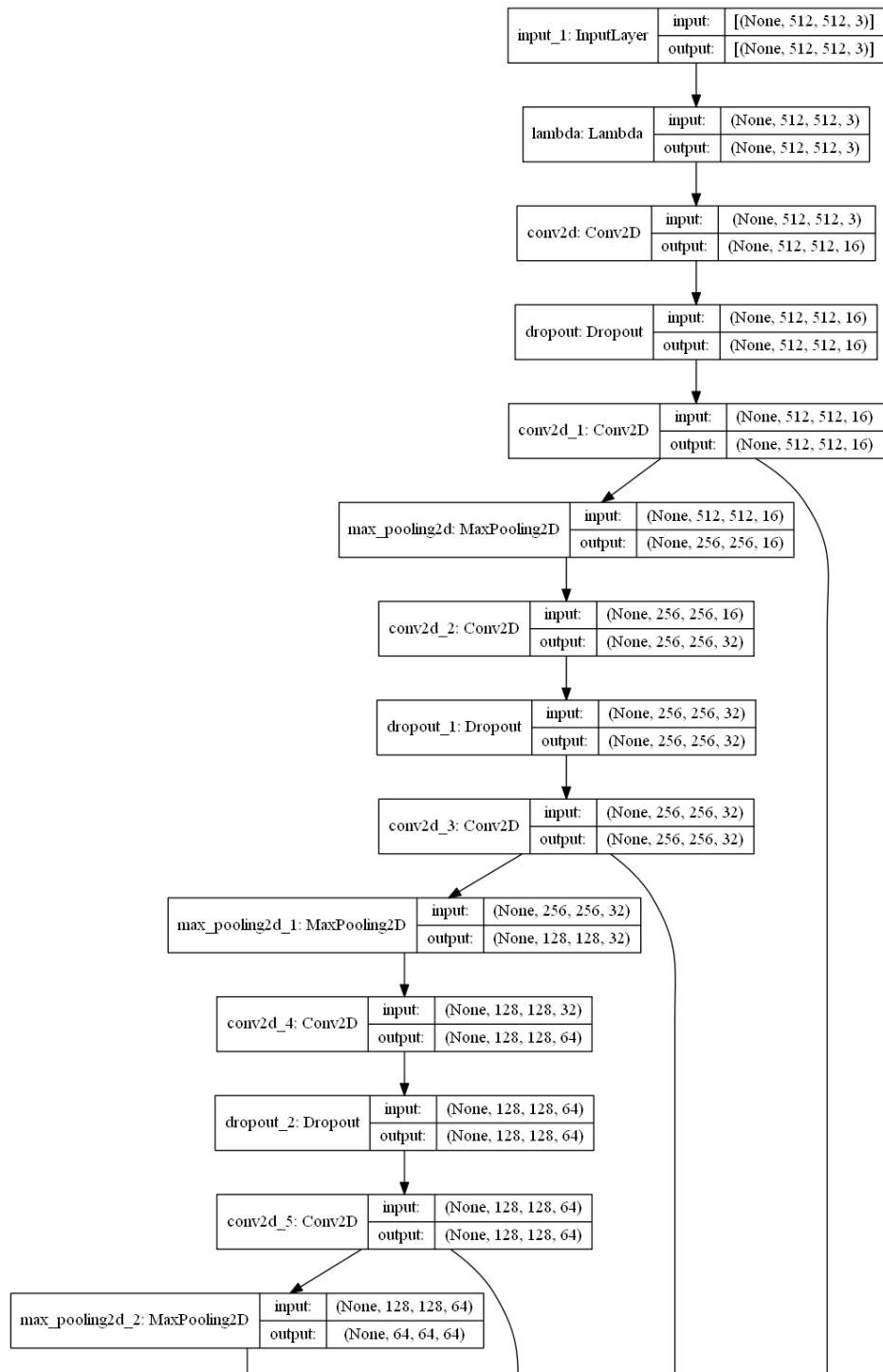
FpVersion3 (Teil 2 von 2)



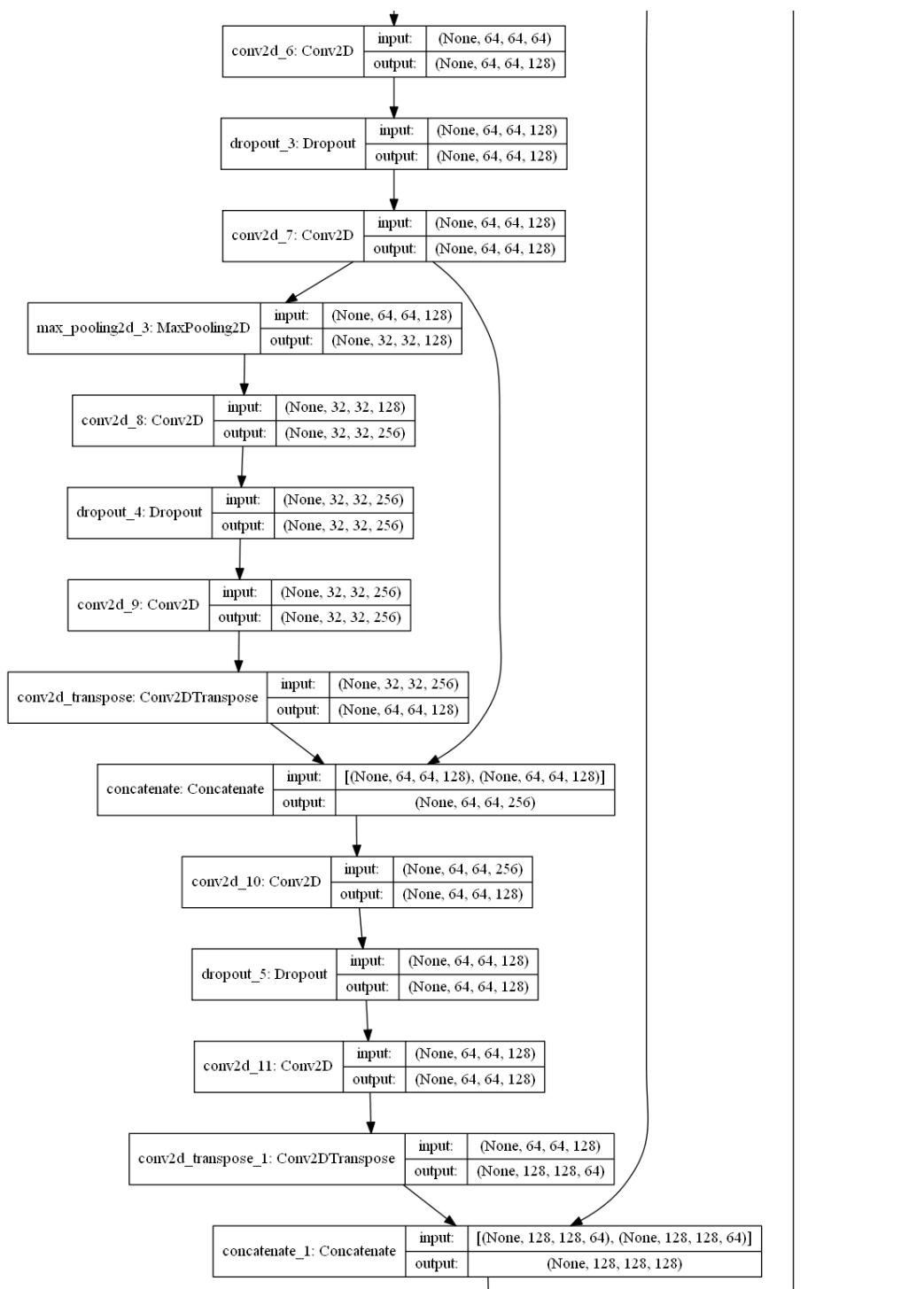
FpVersion4 (Teil 1 von 2)



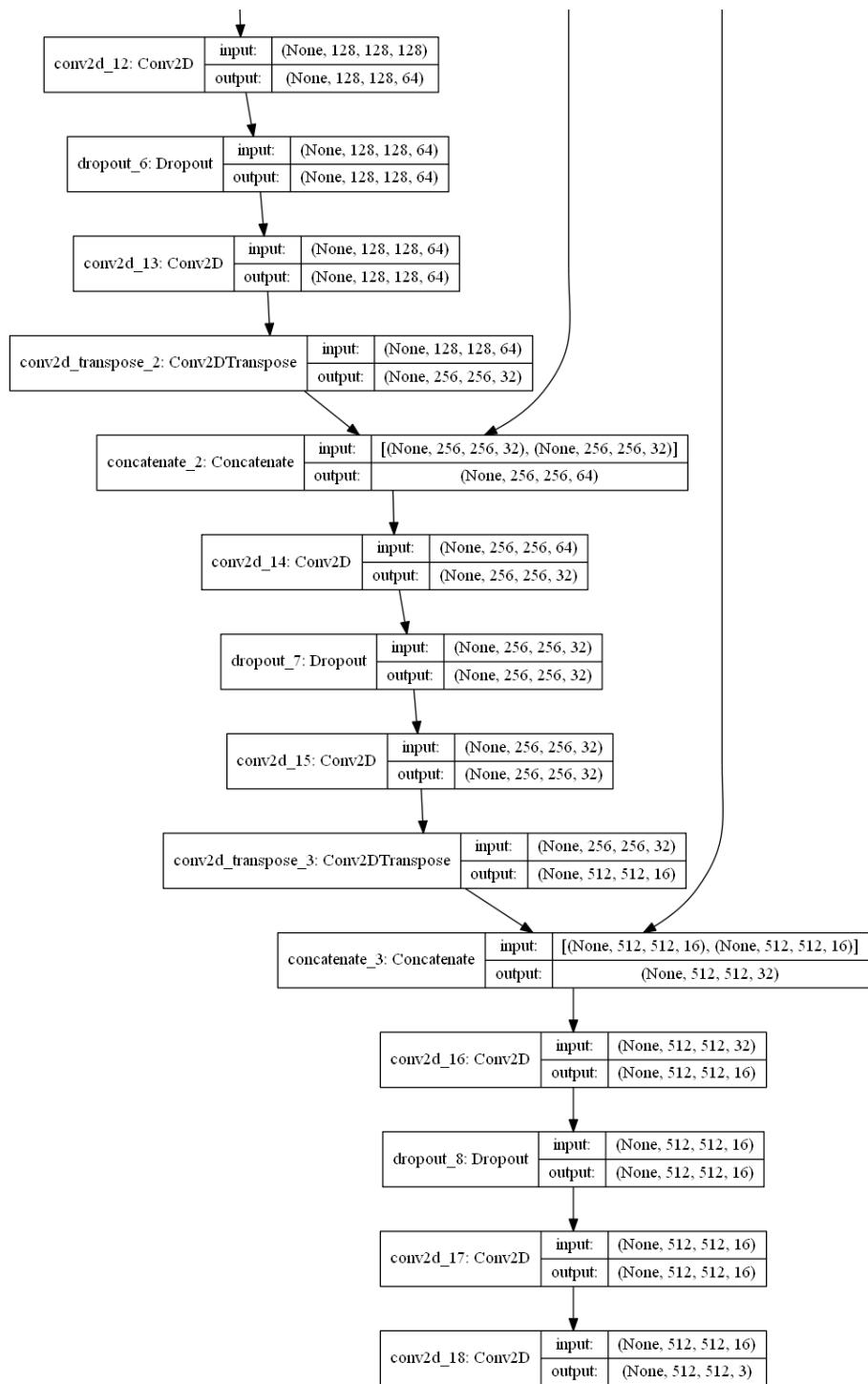
FpVersion4 (Teil 2 von 2)



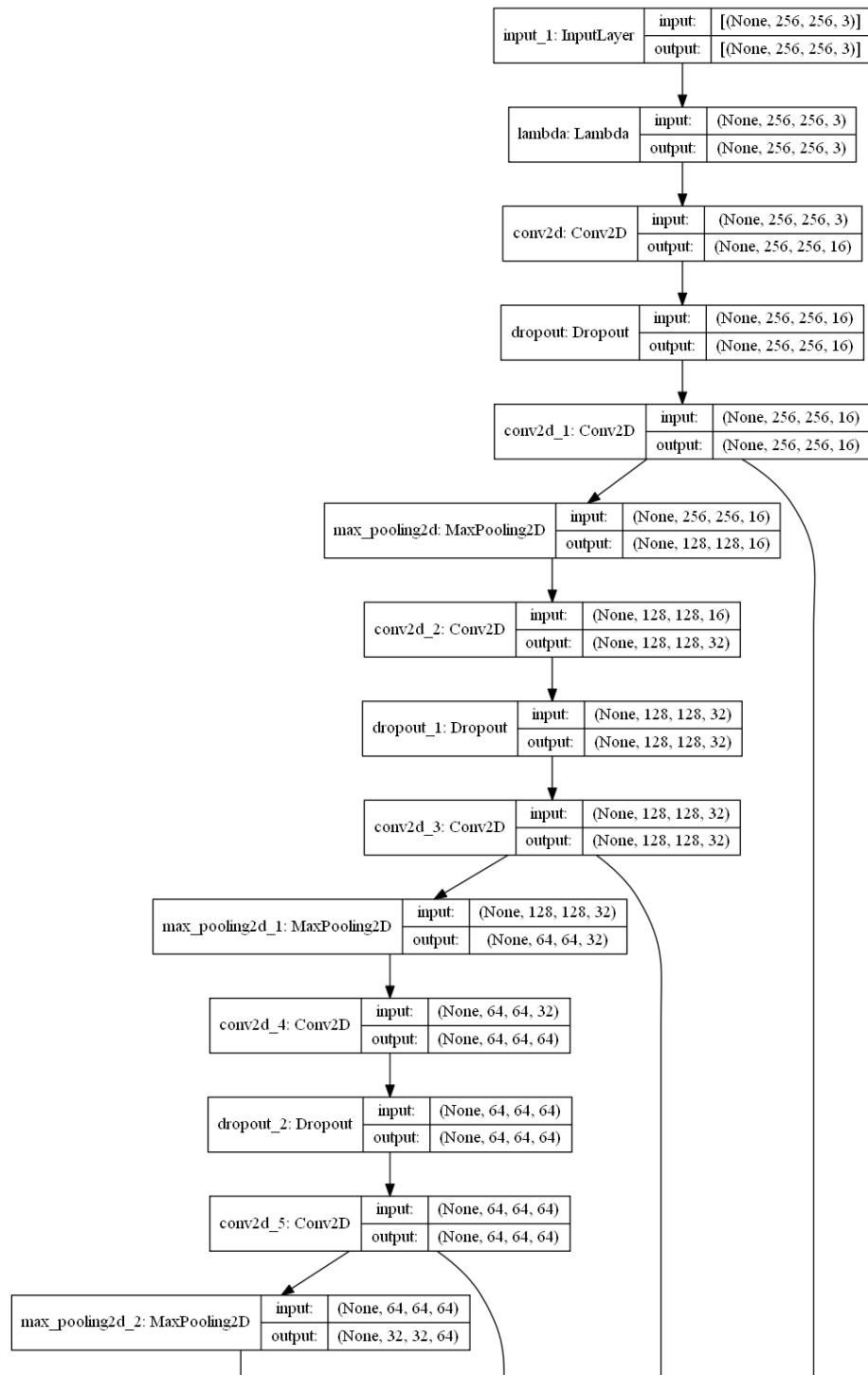
FpVersion5 (Teil 1 von 3)



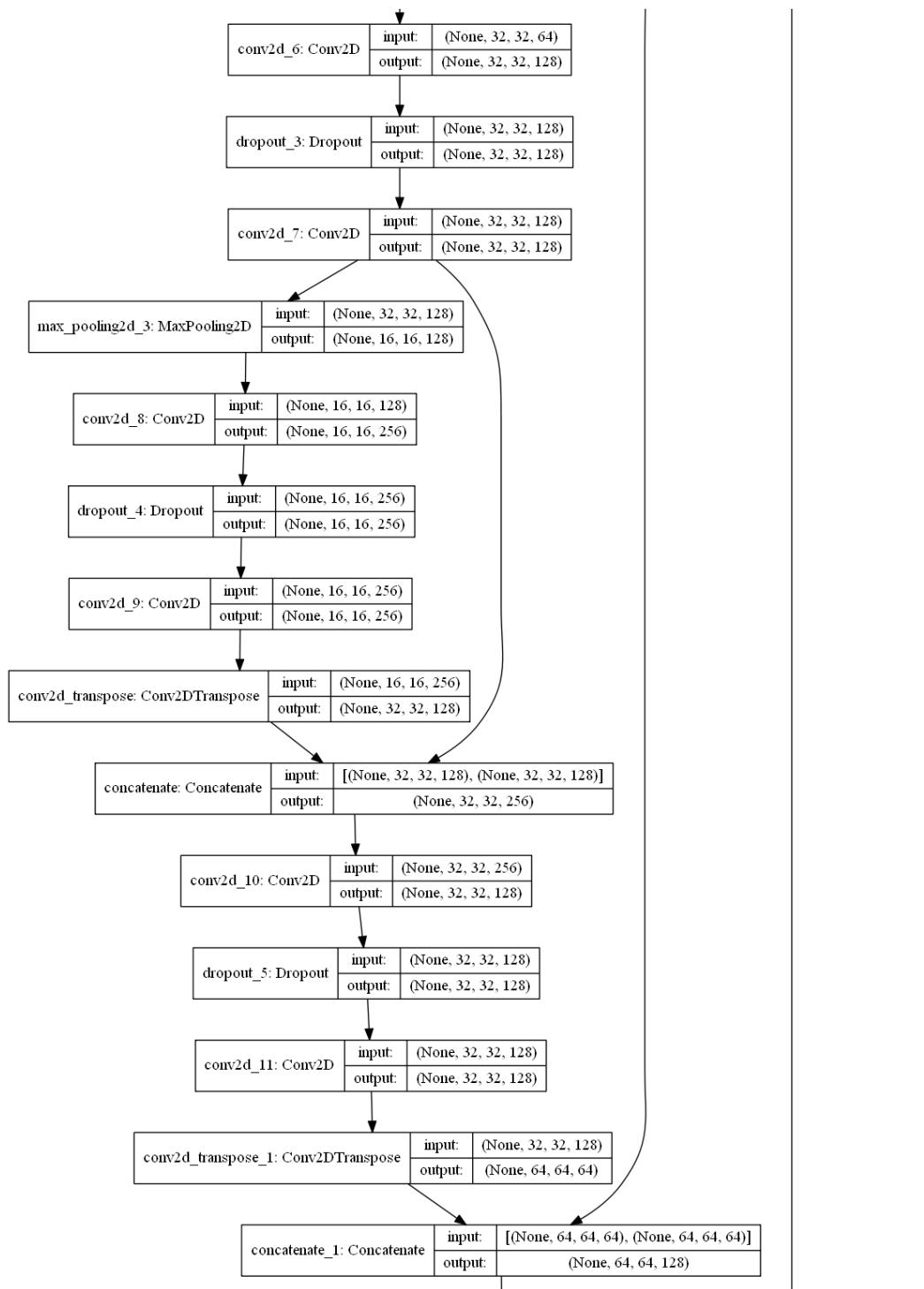
FpVersion5 (Teil 2 von 3)



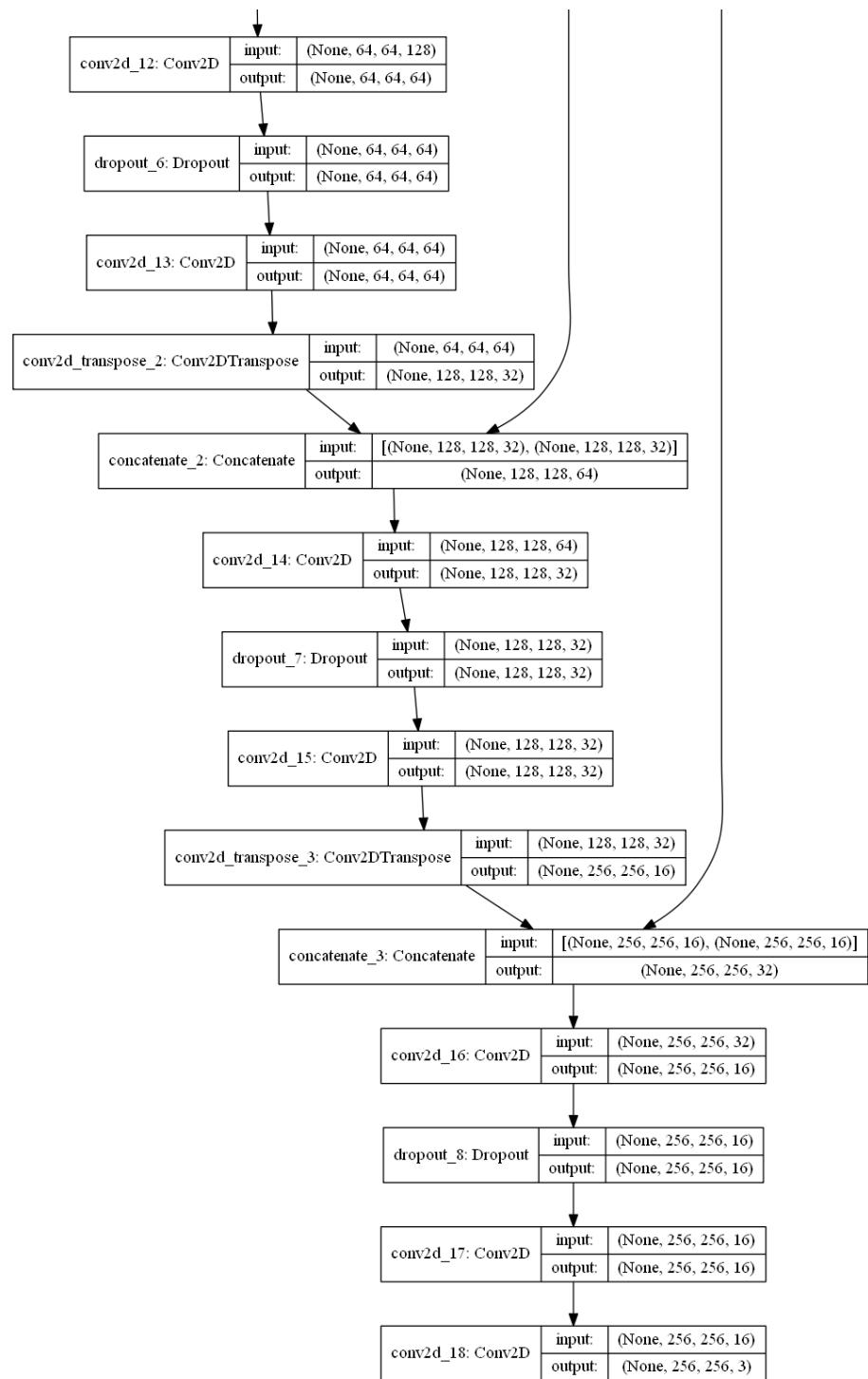
FpVersion5 (Teil 3 von 3)



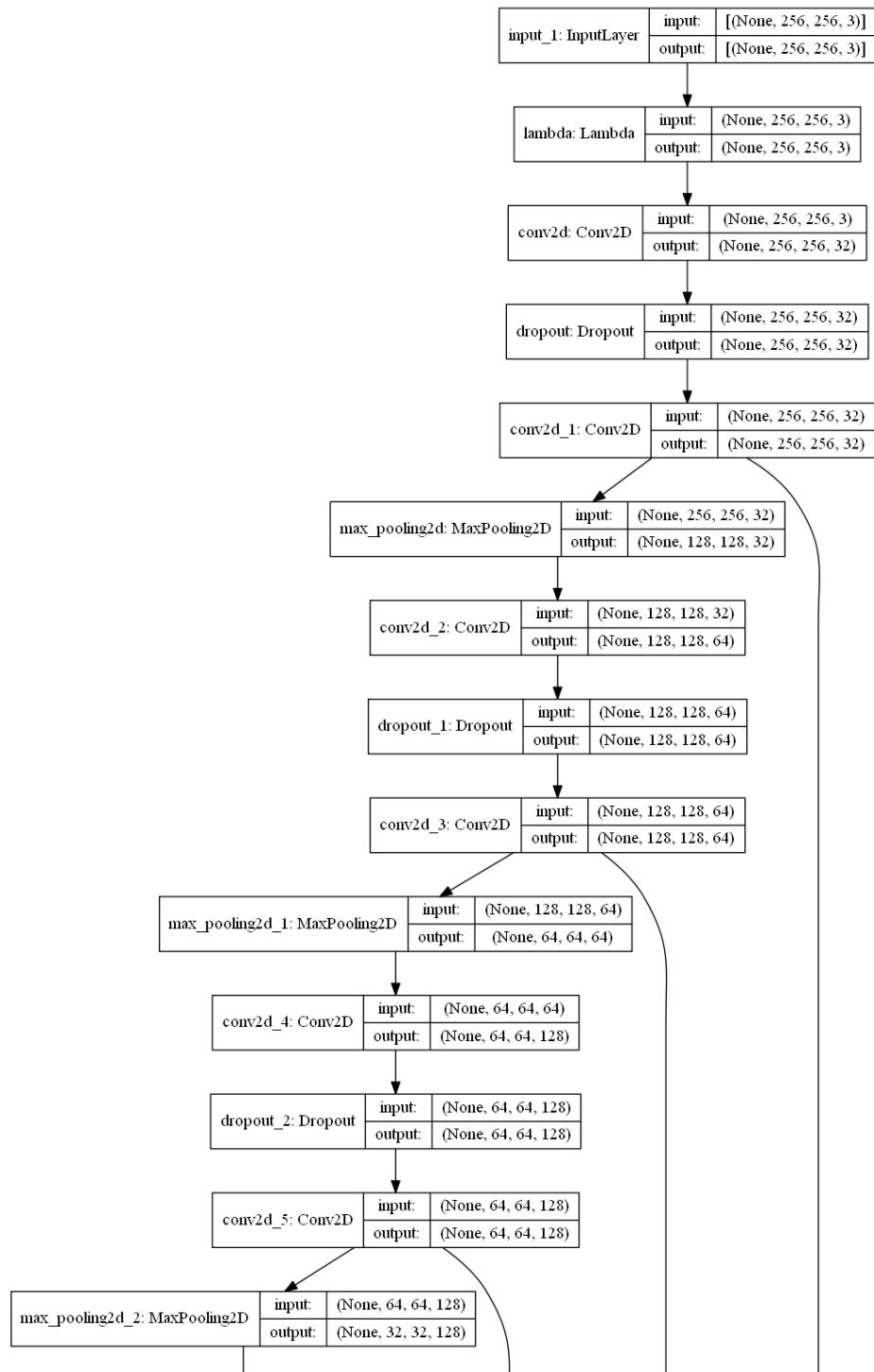
FpVersion6 (Teil 1 von 3)



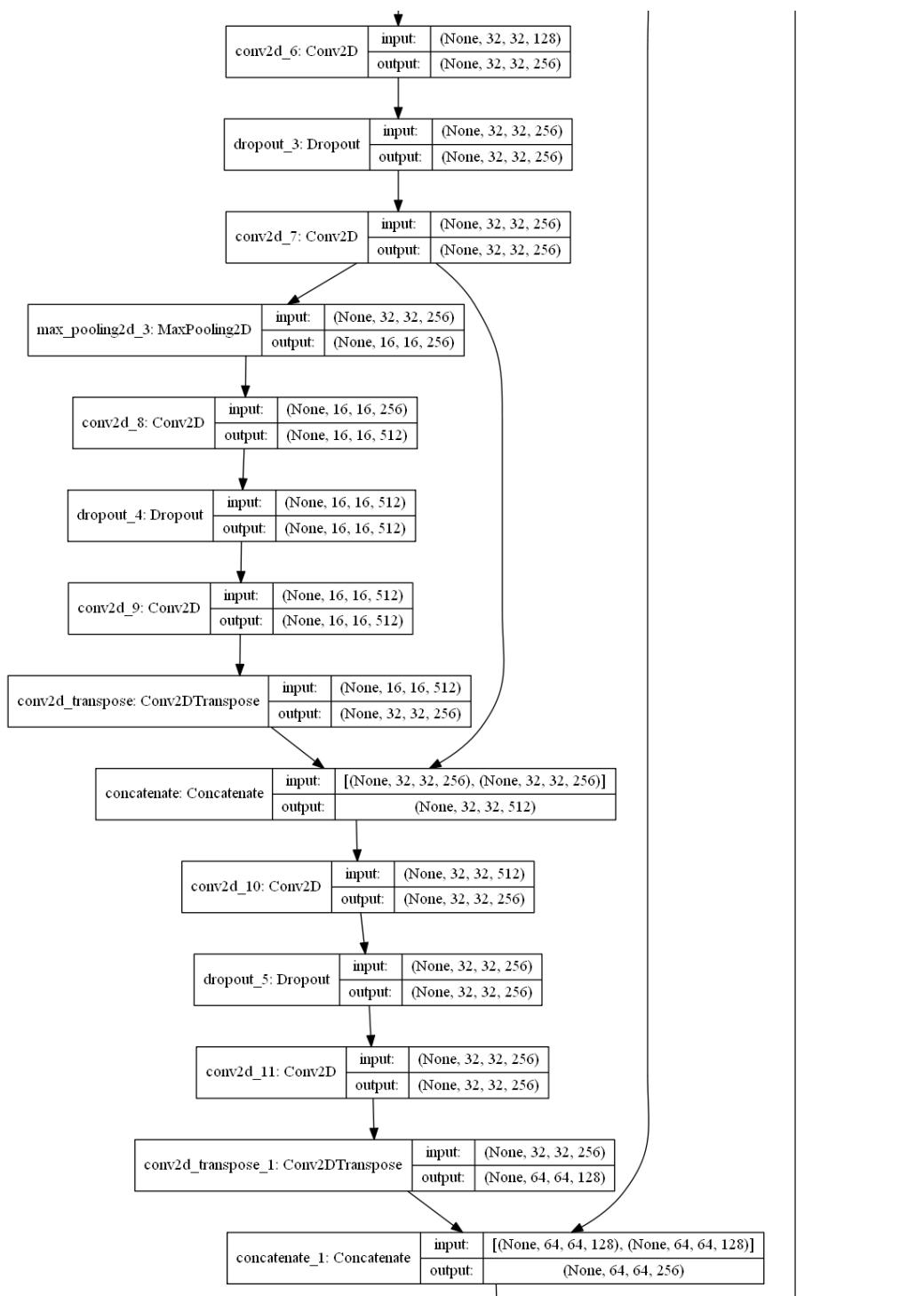
FpVersion6 (Teil 2 von 3)



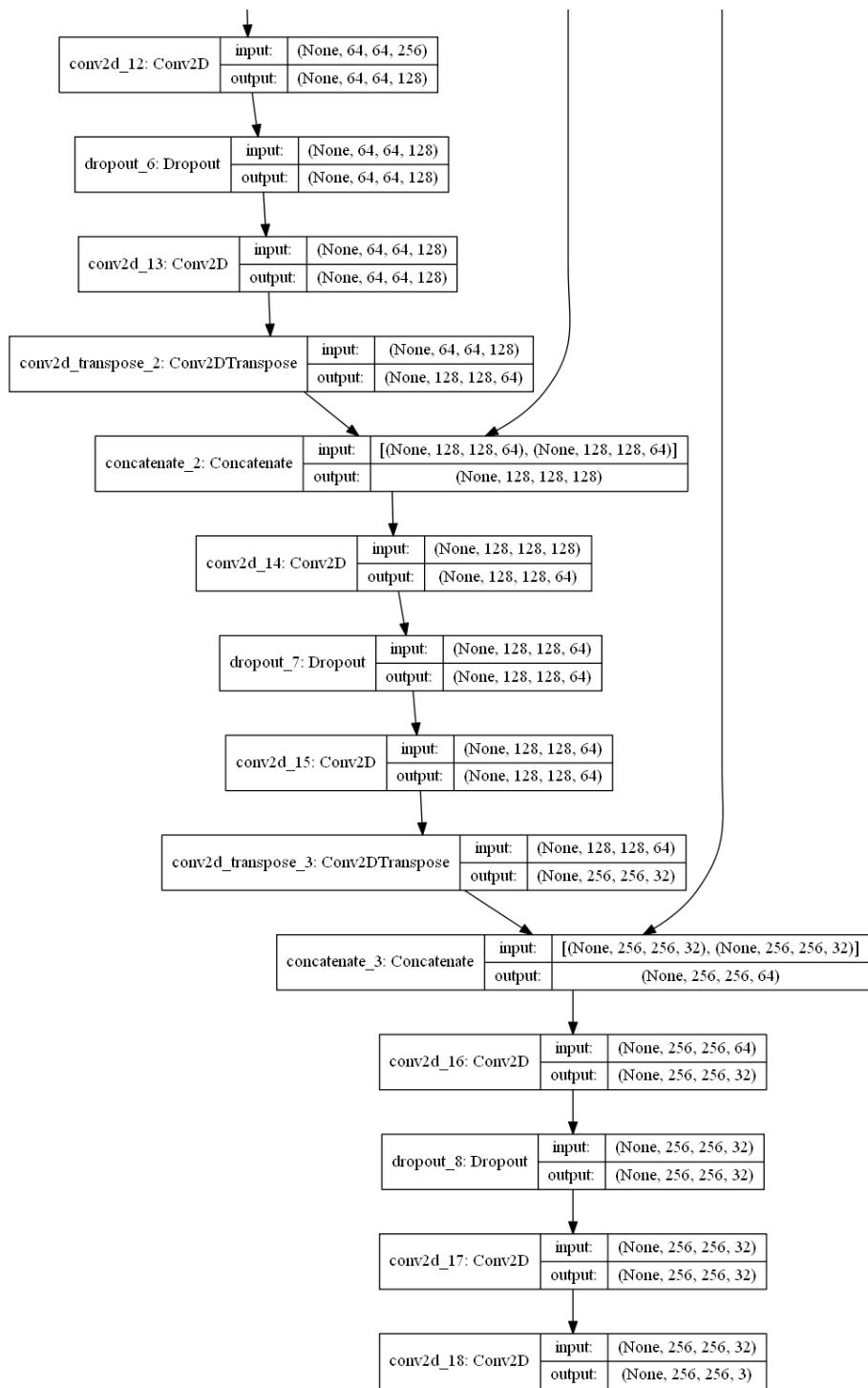
FpVersion6 (Teil 3 von 3)



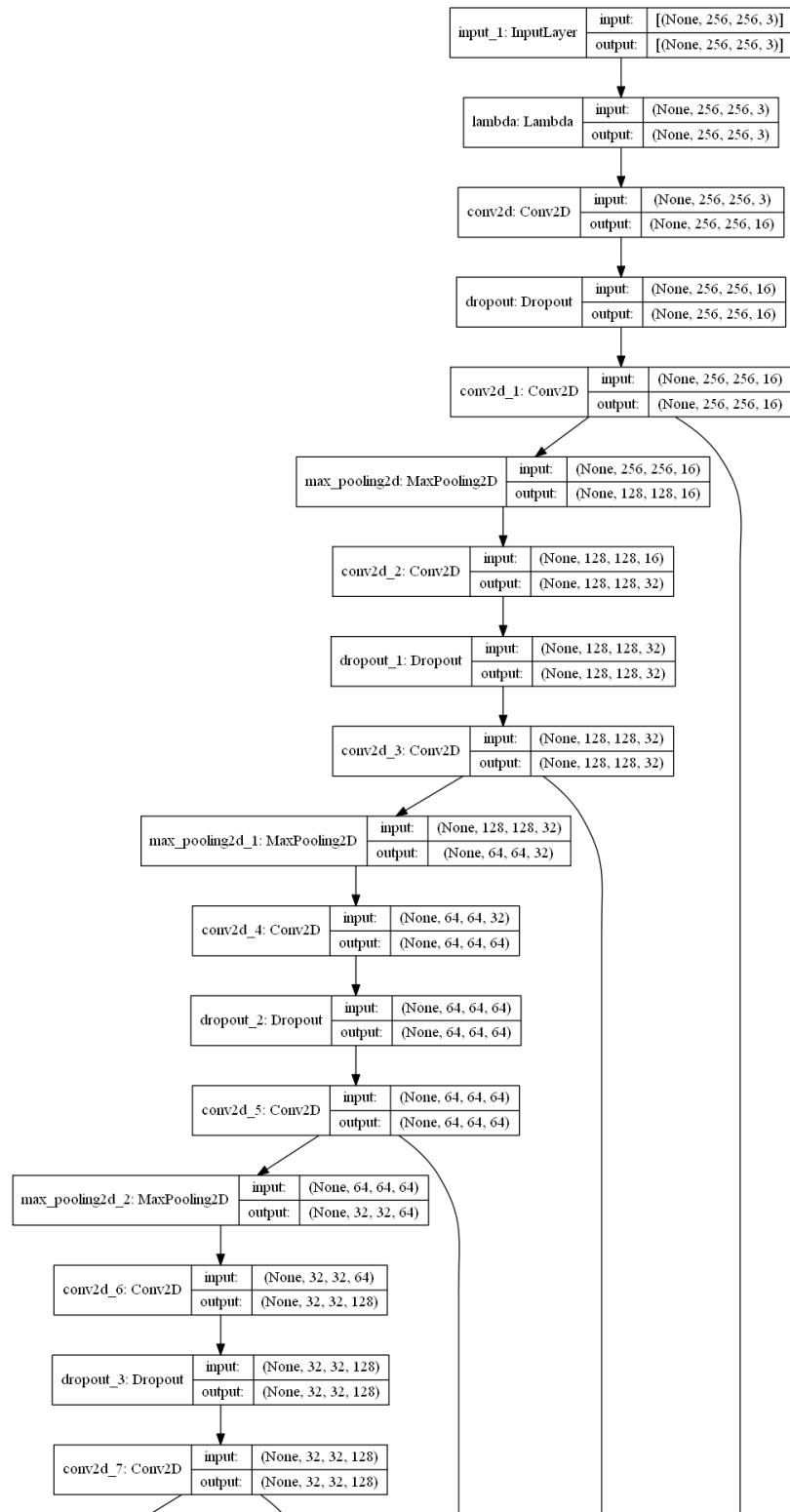
FpVersion7\_11 (Teil 1 von 3)



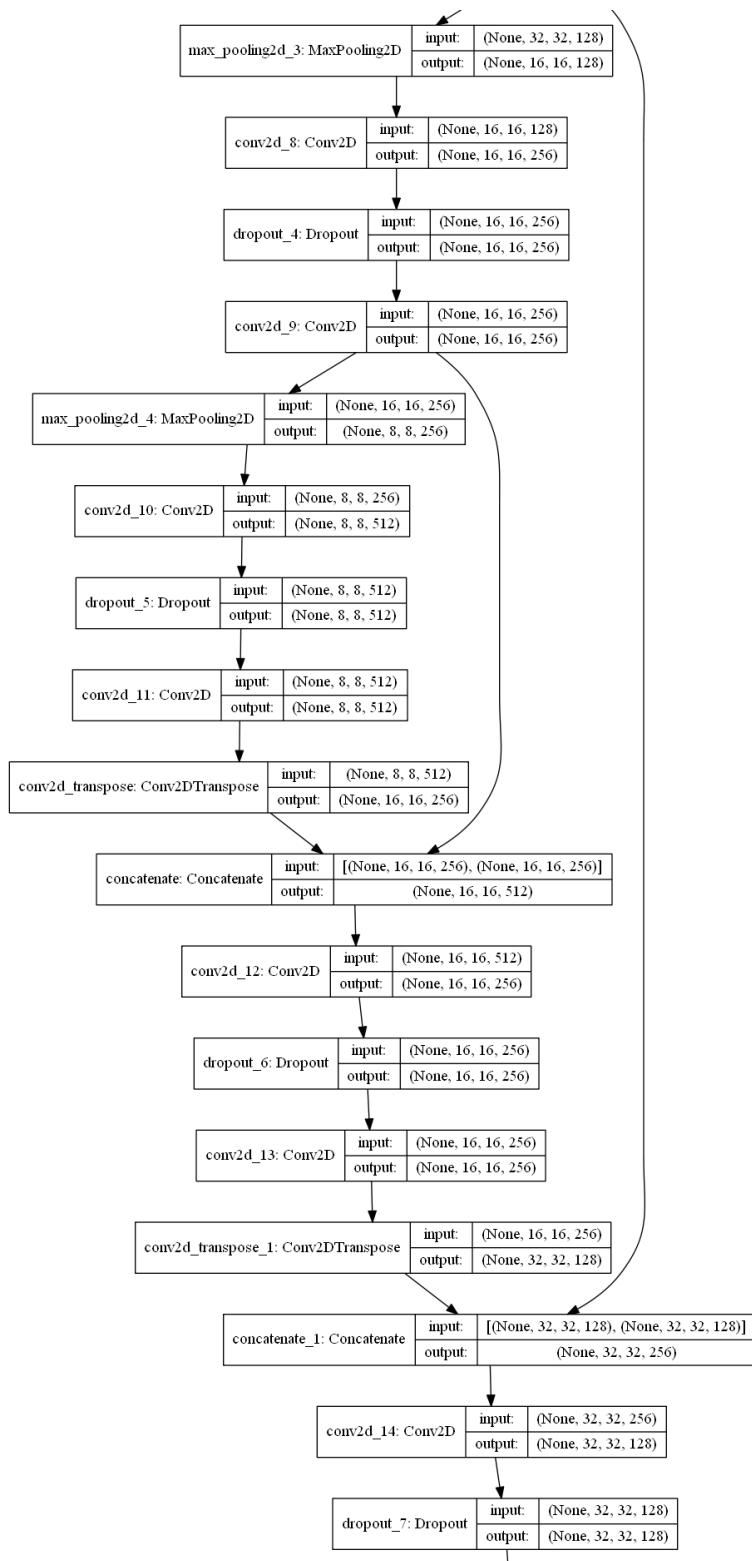
FpVersion7\_11 (Teil 2 von 3)



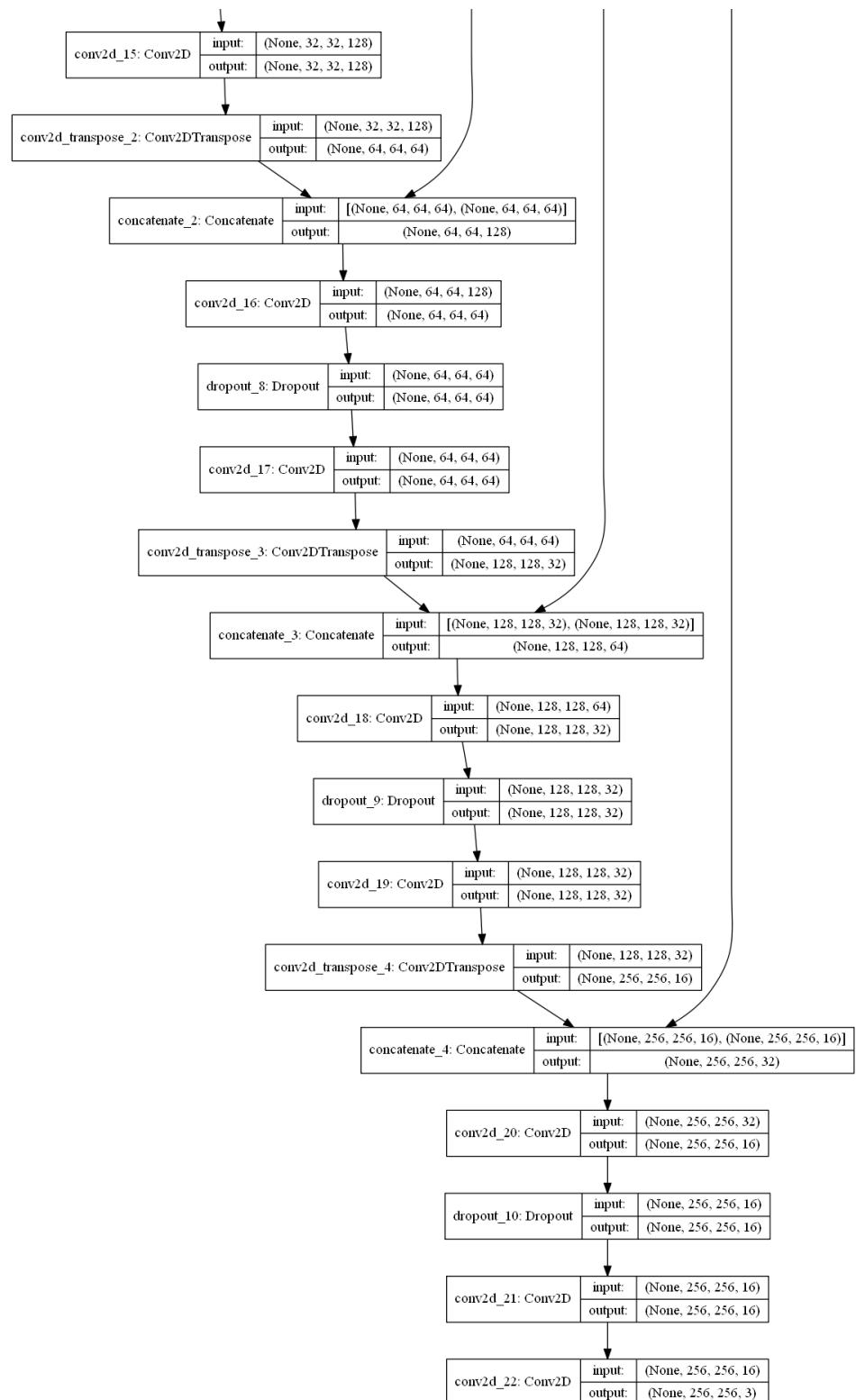
FpVersion7\_11 (Teil 3 von 3)



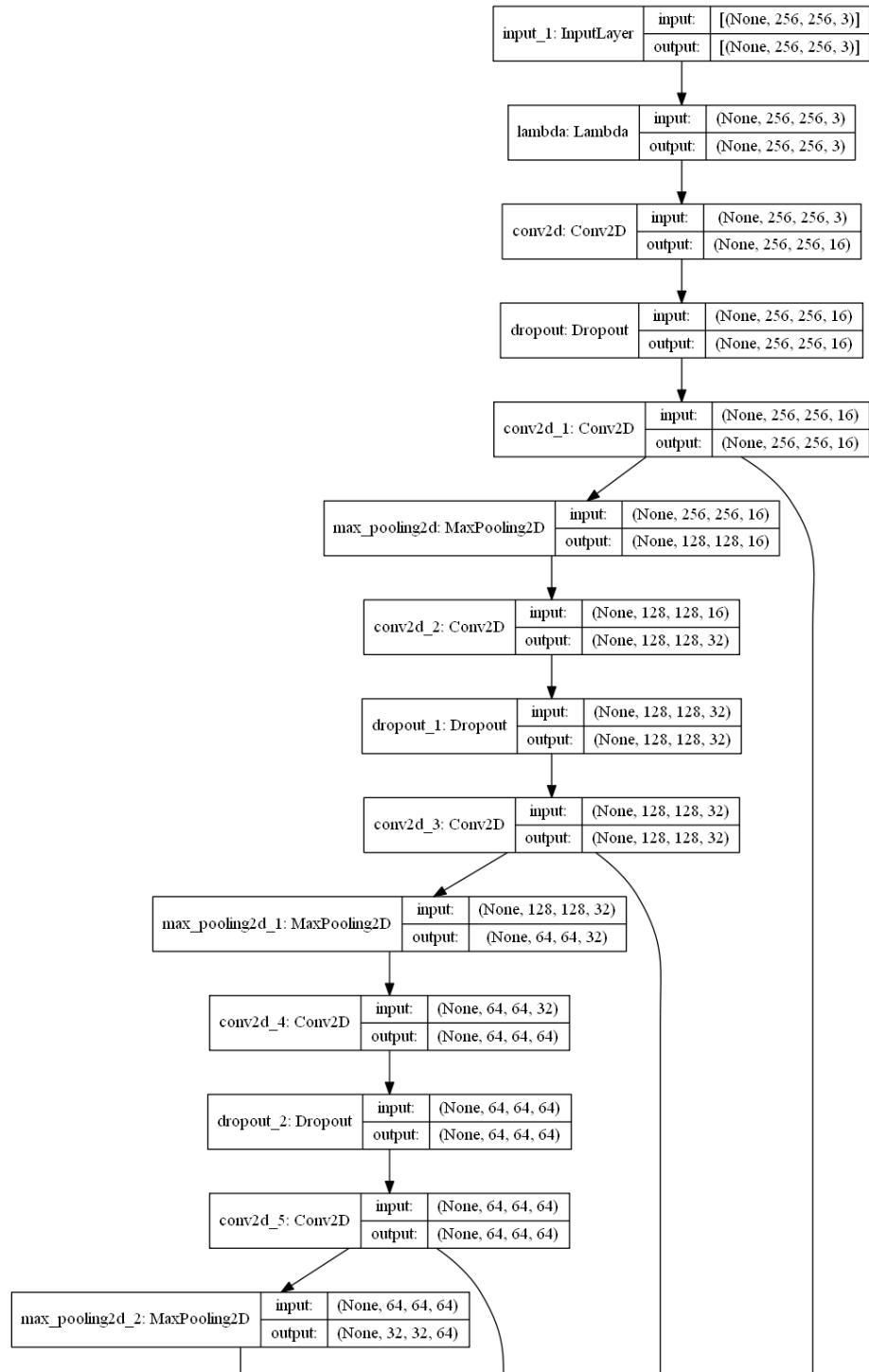
FpVersion8 (Teil 1 von 3)



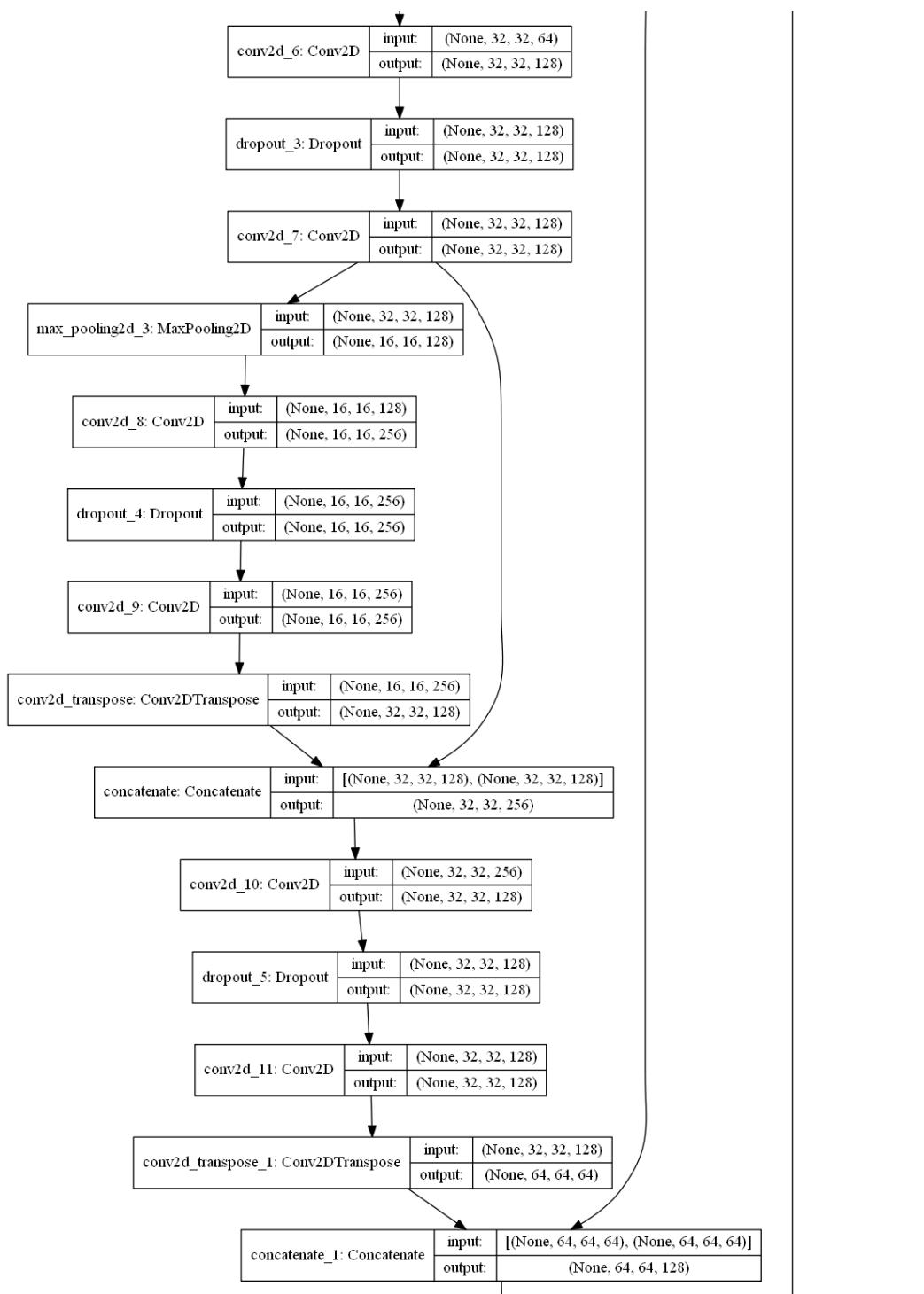
FpVersion8 (Teil 2 von 3)



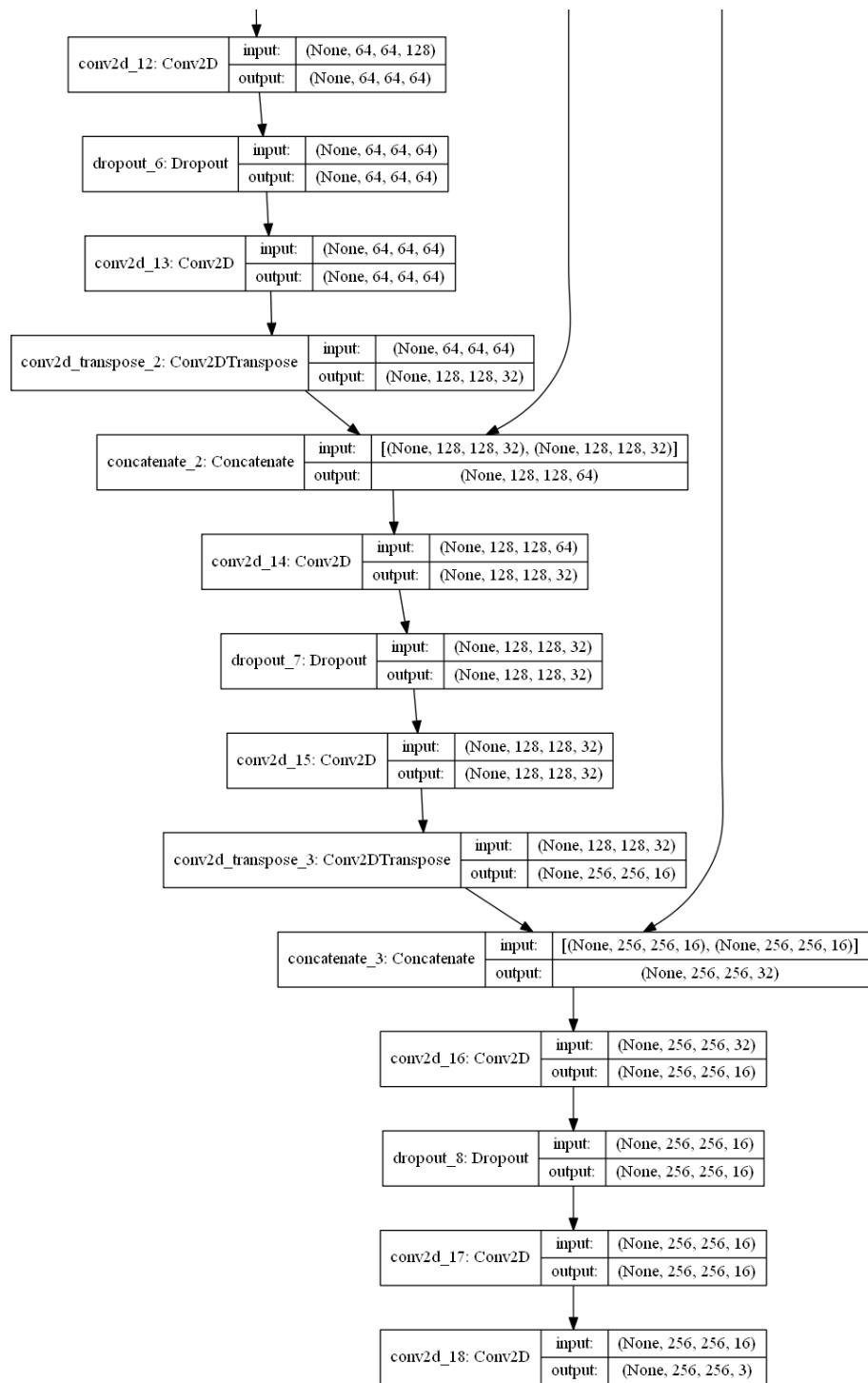
FpVersion8 (Teil 3 von 3)



FpVersion9 (Teil 1 von 3)



FpVersion9 (Teil 2 von 3)



FpVersion9 (Teil 3 von 3)

## I. Fachmodul



---

# Fachmodul

## Human-in-the-loop

## Image-Segmentation

---

Verfasser:

Referent:

Koreferent:

Abgabe: 07.01.2022



Bachelorstudiengang: Systemtechnik

Schule: Ostschweizer Fachhochschule

Departement: Technik

Institut: INF Institut für Ingenieurinformatik

## **Inhaltsverzeichnis**

<b>1. Einleitung</b>	<b>1</b>
<b>2. Architektur und Infrastruktur</b>	<b>2</b>
2.1. Softwarearchitektur . . . . .	2
2.2. Anwendungsfall: Bauplansegmentierung . . . . .	3
<b>3. Frontend</b>	<b>5</b>
3.1. View-Layer . . . . .	5
3.2. Labelling-Tools . . . . .	6
3.2.1. Make-Sense . . . . .	6
3.2.2. COCO-Annotator . . . . .	7
3.2.3. React-Annotation-Tool . . . . .	7
3.2.4. Label-Studio . . . . .	7
3.2.5. Auswahlbegründung . . . . .	8
3.3. Demoimplementation des Label-Studios . . . . .	8
<b>4. Backend</b>	<b>11</b>
4.1. Schnittstellen . . . . .	11
4.2. Framework für das Backend . . . . .	11
4.2.1. Django . . . . .	11
4.2.2. Flask . . . . .	12
4.2.3. FastAPI . . . . .	13
<b>5. Bildverarbeitung und Segmentierung</b>	<b>14</b>
5.1. Machine-Learning . . . . .	14
5.1.1. CNN: Convolutional Neural Network . . . . .	14
5.1.2. R-CNN: Region Based Convolutional Neural Network . . . . .	16
5.1.3. Mask R-CNN . . . . .	16
5.1.4. NLP: Natural-Language-Processing . . . . .	16
5.2. Datensätze . . . . .	17
5.2.1. SESYD Dataset . . . . .	18
5.2.2. ROBIN Dataset . . . . .	18
5.2.3. FPLAN-POLY Dataset . . . . .	19
5.2.4. CVC-FP Dataset . . . . .	19
5.3. Data-Versioning-System . . . . .	20
<b>6. Projektplanung</b>	<b>21</b>
6.1. Meilensteine . . . . .	21
6.2. Zeitplan . . . . .	22
6.3. Gefahren und Risiken . . . . .	23
<b>7. Aufgabenstellung und Zieldefinition</b>	<b>24</b>
<b>I. Verzeichnisse</b>	<b>25</b>

**II. Eidesstattliche Erklärung**

**27**

## 1. Einleitung

Während des Fachmoduls werden die benötigten Grundlagen in einem Selbststudium erarbeitet, um eine detaillierte Aufgabenstellung für die Bachelorarbeit erstellen zu können. Dazu wurden Recherchen durchgeführt, sich in verschiedene Teillbereiche eingearbeitet und verschiedene Technologien auf ihre Eignung für die Bachelorarbeit ausgetestet. Die verschiedenen Arbeitsschritte und deren Resultate wurden in diesem Bericht festgehalten.

Die Idee für die Bachelorarbeit und somit auch ein konkretes Fallbeispiel wurde von der Firma Kemaro<sup>1</sup> zur Verfügung gestellt. Auf dieser Grundlage wurde der Arbeitsauftrag für das Fachmodul definiert. Dieser beinhaltet die Einarbeitung in das Gebiet des Machine-Learnings und Webentwicklung. Gemäss dem Fallbeispiel wird die Applikation als Plug-in für eine Webseite, welche als React-Projekt implementiert wurde, angedacht. Für das Fachmodul werden mehrere Web-Frameworks, Backend-Frameworks und Labelling-Tools evaluiert. Die angedachte Architektur, Infrastruktur und Möglichkeiten für den Machine-Learning Bereich werden ebenfalls thematisiert. Abschliessend ist für die Projektplanung der Zeitplan und die Zieldefinition für die Bachelorarbeit im Fokus.

### Fallbeispiel

Ein Nutzer soll eine Webseite aufrufen und dort einen beliebigen Grundrissplan hochladen können. Der hochgeladene Plan wird von der Applikation mithilfe eines Machine-Learning-Algorithmus analysiert und es werden verschiedene Bereiche im Plan eingezeichnet und gelabelt. Diese Bereiche umfassen zum Beispiel Räume, Treppen, Hindernisse und Freiflächen. Dieser gelabelte Plan kann der Nutzer im nächsten Schritt akzeptieren oder nachbearbeiten. Abschliessend bestätigt der Nutzer den bearbeiteten Plan. Dieser kann unterschiedlich weiterverwendet werden, zum Beispiel als Orientierungsplan für autonom fahrende Roboter.

Die Applikation soll dabei möglichst gut in eine bestehende Web-Infrastruktur eingefügt werden können. Zudem ist eine kontinuierliche Verbesserung des Machine-Learners angestrebt. Dieser soll mit den hochgeladenen Plänen und den korrigierten Labels weiter trainiert werden können. Eine weitere Anforderung ist die Persistenz der generierten Daten. Es soll eine Möglichkeit bestehen, generierte Daten und Machine-Learner zu archivieren und auszutauschen.

---

<sup>1</sup>[www.kemaro.ch](http://www.kemaro.ch)

## 2. Architektur und Infrastruktur

### 2.1. Softwarearchitektur

Das Frontend (siehe Abbildung 1 links) der Applikation übernimmt die Darstellung in der Webseite und das Übermitteln des segmentierten Bauplans zur weiteren Verwendung. Dabei soll das React Plug-in möglichst einfach in eine bestehende React-Anwendung eingefügt werden können. Um dies zu erreichen, wird die Applikation zu einer JavaScript-Bibliothek gepackt. So kann sie in ein React-Projekt importiert und an einer idealen Stelle in der Webseite platziert werden.

Die Backend-API (siehe Abbildung 1 mittig) ermöglicht das Hochladen von Bildern, die segmentiert werden sollen. In einem späteren Schritt können die korrekt gelabelten Bauplaninformationen des Nutzers dem Backend übergeben werden. Diese werden mit dem Bauplan auf dem lokalen Dateisystem des Servers abgelegt.

Die Bildsegmentierung (siehe Abbildung 1 rechts) übernimmt die Segmentierung des Bauplans. Sie besitzt eine Schnittstelle zur Übergabe eines Bauplans und zur Anpassung des Machine Learning Algorithmus. Diese Anpassung ist noch nicht genau definiert, da dies Teil der folgenden Bachelorarbeit ist. Möglich wäre eine Übergabe der Daten oder der Austausch des Machine-Learning-Algoritmus.

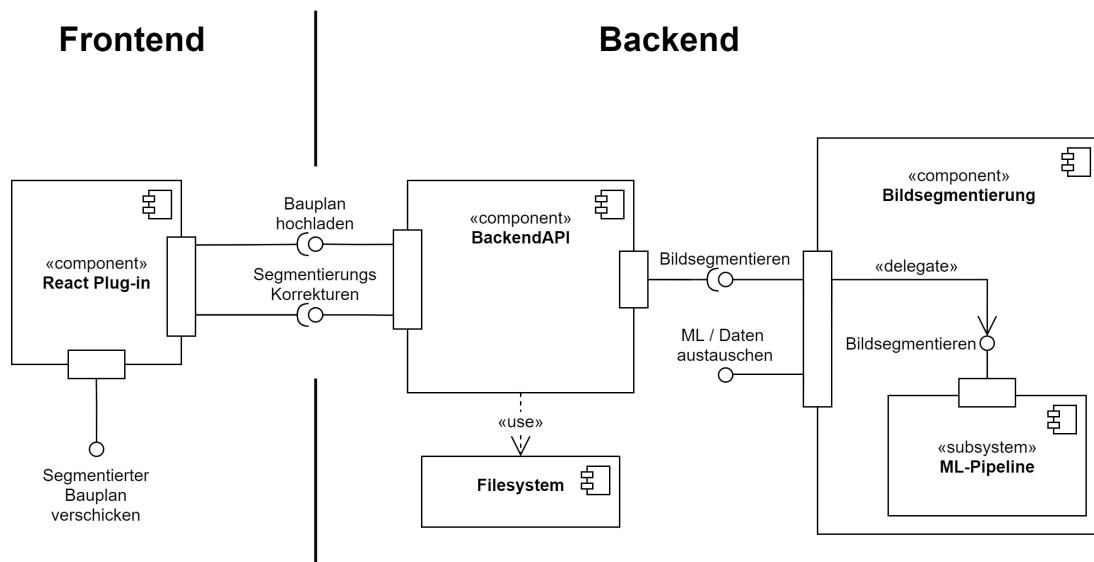


Abbildung 1: Komponentendiagramm des Front- und Backends

In den folgenden Kapiteln dieser Arbeit wird darauf eingegangen, welche Technologien für die jeweiligen Komponenten verwendet werden. Die Bildsegmentierung wird in der Bachelorarbeit genauer behandelt.

## 2.2. Anwendungsfall: Bauplansegmentierung

In diesem Anwendungsfall will ein Nutzer bei einer imaginären Firma X einen Bauplan einlesen. Dargestellt ist der Ablauf dieses Anwendungsfalls in der Abbildung 2 und wird folgend beschrieben.

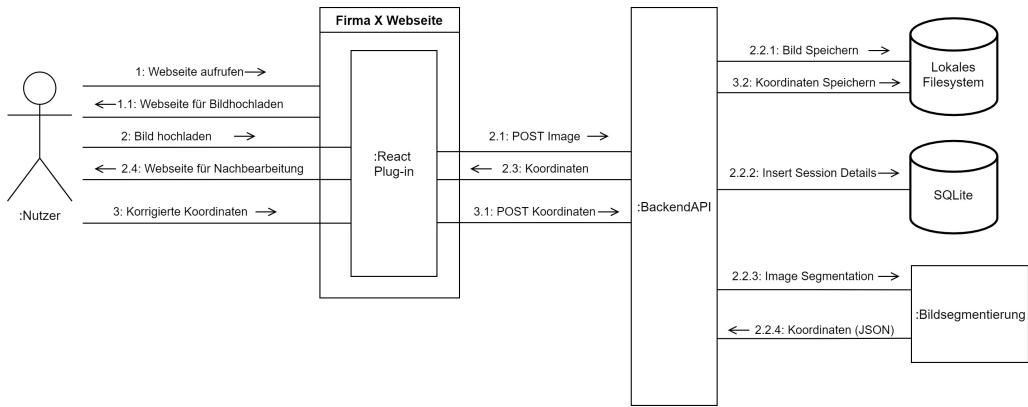


Abbildung 2: Kommunikationsdiagramm der Bauplansegmentierung

Der Nutzer ruft die Webseite der Firma X für die Bauplansegmentierung auf. Der Frontend-Server der Firma X schickt die Webseite mit dem eingebetteten React Plug-in an den Nutzer. Eine mögliche Webseitendarstellung ist in Abbildung 3 ersichtlich.

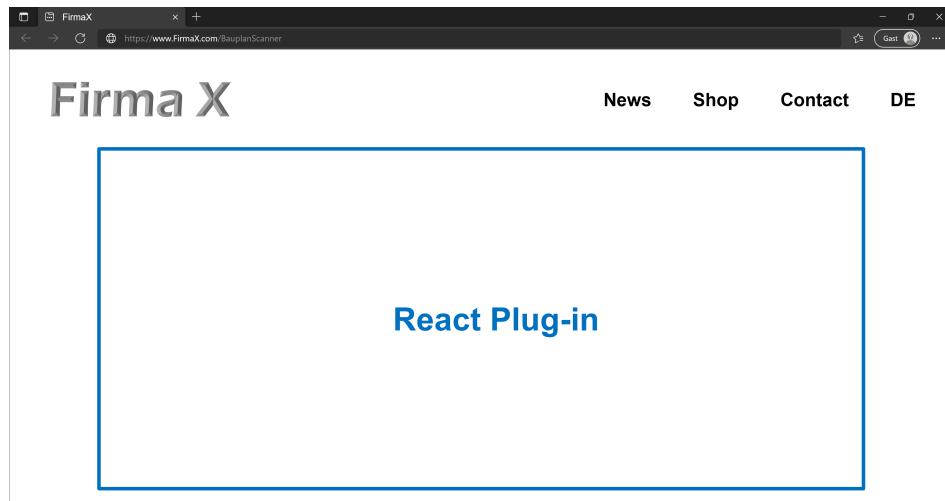


Abbildung 3: Mögliche Webseitendarstellung der Firma X mit dem React Plug-in

Der Nutzer kann über das React Plug-in einen Bauplan hochladen. Der Bauplan wird an den Backend-Server weitergeleitet. Dieser speichert den Bauplan, wie auch die Sessiondetails ab und übergibt der Bildsegmentierung den Bauplan. Als Ergebnis bekommt der Backend-Server die

Koordinaten der möglichen Labels auf dem Bauplan im JSON-Format. Die Koordinaten werden an den Frontend-Server geschickt. Dieser schickt eine Webseite zur Nachbearbeitung an den Nutzer. Der Nutzer korrigiert die Koordinaten auf dem Bauplan. Hier kann Firma X die Koordinaten und den Bauplan für ihre Zwecke weiterverwenden. Zusätzlich werden die korrigierten Koordinaten an den Backend-Server geschickt, um gespeichert zu werden.

Somit erhält die Firma X und das Backend der Applikation die Koordinaten und den Bauplan des Nutzers. Ein Übersetzungstool für die Koordinaten des Bauplans, damit diese von der Firma X genutzt werden können, wird nicht zur Verfügung gestellt. Die Daten im Backend werden später dazu verwendet die Bildsegmentierung stetig zu verbessern.

### 3. Frontend

Dieser Teil des Berichtes deckt den Bereich des Frontends ab. Das Frontend ist für die Darstellung der Applikation zuständig und der Teil eines Systemes, mit welchem ein Nutzer interagiert. Bei einer Interaktion des Nutzers mit dem Frontend werden die Änderungen an das Backend (siehe 4) geschickt. Dort werden diese verarbeitet und das Resultat dieser Bearbeitung wieder an den Nutzer geschickt, wie in Kapitel 2 beschrieben.

#### 3.1. View-Layer

Der Inhalt des View-Layers ist für Web-Applikationen ein Verbund von HTML-, CSS- und JS-Code. Heutige Bibliotheken ermöglichen eine abstraktere Implementation dieses Layers. Zum Beispiel können hier Bibliotheken wie React, Vue.js und Angular benutzt werden. Obwohl diese unterschiedliche Gebiete abdecken, können alle genannten für die Implementation eines View-Layers genutzt werden. Für diese Arbeit wird React verwendet, wie im Fachauftrag vorgegeben. Diese Bibliothek deckt nur den Bereich des View-Layers ab und dringt nicht in die anderen Bereiche der Applikation ein. Der daraus resultierende Freiheitsgrad ist für diese Arbeit von Vorteil. React wurde ursprünglich von Facebook entwickelt und wurde unter einer Open-Source-Lizenz veröffentlicht. Der grösste Nachteil dieser Lösung ist die Einstiegshürde. React erfordert eine höhere Einarbeitungszeit als andere vergleichbare Lösungen. Dafür werden moderne Programmierparadigmen unterstützt. React ist zudem leichtgewichtig, performant und besitzt eine gute Skalierbarkeit. Eine React-Applikation besitzt einen komponentenorientierten Aufbau, wobei der Datenfluss eine entscheidende Rolle spielt. Zudem aktualisiert der Renderer nur jene Elemente, welche verändert werden. Dadurch wird der View-Layer nicht immer komplett neu aufgebaut. Dieser Renderer kann auch ausgetauscht werden und so die Darstellung auf unterschiedlichste Arten ermöglicht werden. Ein Beispiel wäre eine interaktive Kommandozeilen-Applikation [1].

### 3.2. Labelling-Tools

Ein grosser Teilbereich in der Bilderkennung und -segmentierung ist das Labelling. Hierbei werden in den Bildern Polygone platziert die ein Label besitzen (Abbildung 4). Im Bereich des Labellings gibt es eine Vielzahl an Lösungen. Um eine möglichst passende zu finden, wurden mehrere ausprobiert und bewertet. Dabei wurden als Vorauswahl nur jene angeschaut, welche unter einer Open-Source-Lizenz veröffentlicht wurden.

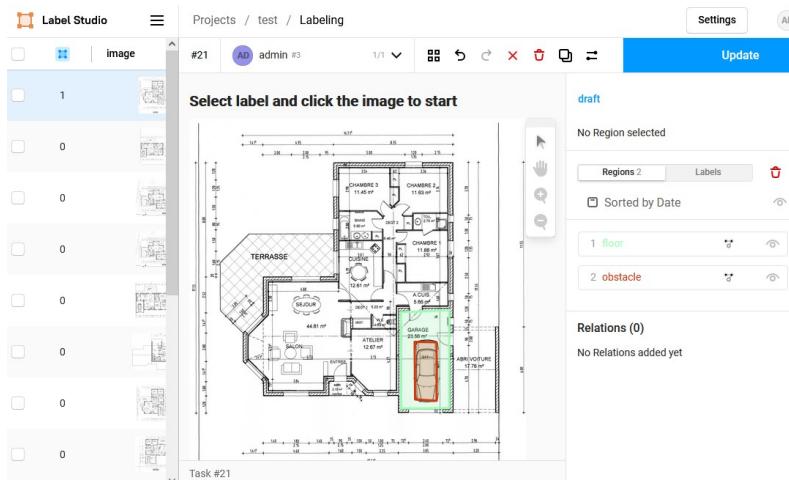


Abbildung 4: Labellingbeispiel eines Grundrisses

#### 3.2.1. Make-Sense

[makesense.ai](https://makesense.ai/) is a free to use online tool for labelling photos. Thanks to the use of a browser it does not require any complicated installation – just visit the website and you are ready to go. It also doesn't matter which operating system you're running on - we do our best to be truly cross-platform. It is perfect for small computer vision deeplearning projects, making the process of preparing a dataset much easier and faster. Prepared labels can be downloaded in one of multiple supported formats. The application was written in TypeScript and is based on React/Redux duo [2].

Das Ziel dieses Projektes ist, das Labelling eines Datensatzes möglichst zu vereinfachen [2]. Ein Nutzer kann durch wenige Eingaben mehrere Bounding-Boxes oder Polygone erstellen. Gesetzte Polygone können jedoch nicht einzeln entfernt werden. Ausserdem unterstützt es den Nutzer durch einen vtrainierten Machine-Learner beim Labelling. Dieser ist aber nicht auf Grundrisspläne trainiert und unterstützt auch kein Nachlernen. Eine Schnittstelle für das Austauschen des Learners ist nicht vorhanden. Das Projekt selbst richtet sich an Nutzer, welche für ein eigenes Machine-Learning-Projekt Bilder labeln möchten. Dies bedeutet auch, dass das Projekt als eine eigenständige Lösung gedacht ist. Die benötigte Dokumentation ist daher nicht zugänglich. Eine Weiterentwicklung zu einem Plug-in System würde sich schwierig gestalten.

### 3.2.2. COCO-Annotator

COCO-Annotator is a web-based image annotation tool designed for versatility and efficiently label images to create training data for image localization and object detection. It provides many distinct features including the ability to label an image segment (or part of a segment), track object instances, labeling objects with disconnected visible parts, efficiently storing and export annotations in the well-known COCO format. The annotation process is delivered through an intuitive and customizable interface and provides many tools for creating accurate datasets [3].

Auch dieses Tool richtet sich primär an Nutzer, welche für ein eigenes Machine-Learning-Projekt Daten labeln möchten. Es besitzt Werkzeuge für die Erzeugung von Polygonen und Bounding-Boxes. Des Weiteren ist ein Zauberstab<sup>2</sup> und Unterstützung durch Machine-Learner implementiert. Das Nachlernen wird hier nicht unterstützt, aber diese Lösung bietet eine vollständige API an. Die Dokumentation ist umfangreich und deckt viele Aspekte ab. Für die Verteilung der Software wird auf Docker<sup>3</sup> gesetzt.

### 3.2.3. React-Annotation-Tool

This tool allows to annotate images with polygons. Users could create new taxonomy if they feel the annotation does not fit into any default options. It is adopted by a CVPR 2019 paper, VizWiz-Priv: A Dataset for Recognizing the Presence and Purpose of Private Visual Information in Images Taken by Blind People [5].

Dieses Tool wurde als Plug-in konzipiert und befindet sich noch in aktiver Entwicklung. Dies macht sich auch beim Bedienen des Tools bemerkbar. Gewisse Funktionen sind noch nicht ausgereift. Bounding-Boxes können nicht gesetzt werden, es sind nur Polygone möglich. Diese lassen sich nach dem Setzen noch verschieben, aber nicht wieder löschen. Die Dokumentation ist spärlich und deckt nur einen kleinen Bereich des Projektes ab.

### 3.2.4. Label-Studio

Label Studio is an open-source, configurable data annotation tool.

Frontend, as its name suggests, is the frontend library developed using React and mobx-state-tree, distributed as an NPM package. You can include it in your applications and provide data annotation support to your users. It can be granularly customized and extended [6].

Das Label-Studio selbst ist eine Komplettlösung, das Frontend kann jedoch vom Backend getrennt verwendet werden. Dies ist auch so von den Projektentwicklern angedacht. Das komplette Frontend wird als CSS- und JS-File zugänglich gemacht. Eine Einbindung in ein eigenes Projekt ist dadurch möglich und es wird kein umfangreicher Abhängigkeitsbaum benötigt. Es unterstützt Bounding-Boxes und Polygone für das Labelling. Einzelne Polygone können nachträglich verändert und gelöscht werden. Die Dokumentation deckt das Projekt beinahe komplett ab und

<sup>2</sup>Bereichsmarkierer, welcher auf Ähnlichkeit des selektierten Punktes beruht

<sup>3</sup>Lösung, um containerisierte Applikationen zu generieren und zu verteilen [4]

ist auf eine Weiterentwicklung des Projektes ausgelegt. Das Paket aus Front- und Backend ist auf Machine-Learning unterstütztes Labelling ausgelegt. Ebenfalls ist dort eine Funktion für das Nachlernen vorgesehen.

### 3.2.5. Auswahlbegründung

Für die Bachelorarbeit kommen das Label-Studio und das React-Annotation-Tool in Frage. Beide können als Plug-in verwendet werden und besitzen einen genügend grossen Funktionsumfang. Ebenfalls sind beide unter einer Open-Source-Lizenz veröffentlicht worden.

Die Auswahl fiel auf das Label-Studio. Dieses Labelling-Tool ist ausgereifter, besitzt einen grösseren Funktionsumfang und eine bessere Dokumentation. Diese ist aber im Bereich Frontend noch lückenhaft. Für das Label-Studio spricht weiter, dass es in Kombination mit dem Label-Studio-Backend ähnliche Funktionen aufweist, wie sie für diese Bachelorarbeit benötigt werden. Namentlich das Austauschen des Machine-Learners und das Nachlernen. Ebenfalls ist das Label-Studio in bestehende Projekte integrierbar. Das Label-Studio als Komplett paket ist jedoch zu umfangreich für den in Kapitel 1 aufgezeigten Anwendungsfall. Ausserdem ist dieses Paket für die Segmentation und das Labelling von Grundrissplänen nicht ausgelegt. Das Label-Studio unterstützt ausser Bilderkennung auch Audio- und Textanalysen. Durch diesen enormen Funktionsumfang würde eine direkte Integration tief in bestehende Systeme eingreifen.

## 3.3. Demoimplementation des Label-Studios

Dieser Teil des Berichtes zeigt eine einfache Implementation des Label-Studios in eine eigene Webseite. Dies ist mit ein paar einfachen Schritten möglich. Zuerst müssen die CSS- und JS-Files in die eigene Webseite eingebunden werden. Dies kann entweder durch einen Importbefehl über das Web gelöst oder die Files von der Projektwebseite<sup>4</sup> heruntergeladen werden.

```

1 <!-- Include Label Studio stylesheet -->
2 <link
3   href="https://unpkg.com/label-studio@1.4.0/build/static/css/main.css"
4   rel="stylesheet">
5 </link>
6 <!-- Include the Label Studio library -->
7 <script
8   src="https://unpkg.com/label-studio@1.4.0/build/static/js/main.js">
9 </script>
```

Anschliessend wird der Label-Studio-Container initialisiert und die gewünschten Optionen gesetzt. Hier wird ebenfalls definiert, welche Labels zur Verfügung stehen.

```

1 <!-- Initialize Label Studio -->
2 <script>
3   var labelStudio = new LabelStudio('label-studio', {
4     config: '
5       <View>
6         <Image name="#img" value="$image"></Image>
```

---

<sup>4</sup>[https://github.com/heartexlabs/label-studio/tree/master/label\\_studio/frontend/dist/lst](https://github.com/heartexlabs/label-studio/tree/master/label_studio/frontend/dist/lst)

```
7      <!-- Label-Einstellungen -->
8      <RectangleLabels name="tag" toName="img">
9          <Label value="Hello"></Label>
10         <Label value="Label-Studio"></Label>
11     </RectangleLabels>
12
13     </View>
14 ,
15
16     interfaces: [
17     "panel",
18     "update",
19     "submit",
20     "controls",
21     "side-column",
22     "annotations:menu",
23     "annotations:add-new",
24     "annotations:delete",
25     "predictions:menu",
26   ],
27
28     user: {
29       pk: 1,
30       firstName: "James",
31       lastName: "Dean"
32     },
33
34     task: {
35       annotations: [],
36       predictions: [],
37       id: 1,
38
39       <!-- setzen des Grundrisses -->
40       data: {
41         image:
42           "https://upload.wikimedia.org/wikipedia/commons/thumb/b/ba/Graceland_Memphis_TN_Floor_Plan_2010.jpg"
43       }
44
45     },
46
47     onLabelStudioLoad: function(LS) {
48       var c = LS.annotationStore.addAnnotation({
49         userGenerate: true
50       });
51       LS.annotationStore.selectAnnotation(c.id);
52     }
53   );
```

10

3 FRONTEND

---

54 </script>

---

Das Einfügen in die Webseite erfolgt über einen HTML Div-Tag mit der ID "label-studio". Der Rest der Webseite kann wie gewohnt gestaltet werden.

---

1 <div id="label-studio"></div>

---

Dies führt zu folgendem Ergebnis. Mit wenigen Befehlen wurde das Label-Studio in eine Demo-Webseite eingefügt, wie in Abbildung 5 sichtbar ist.

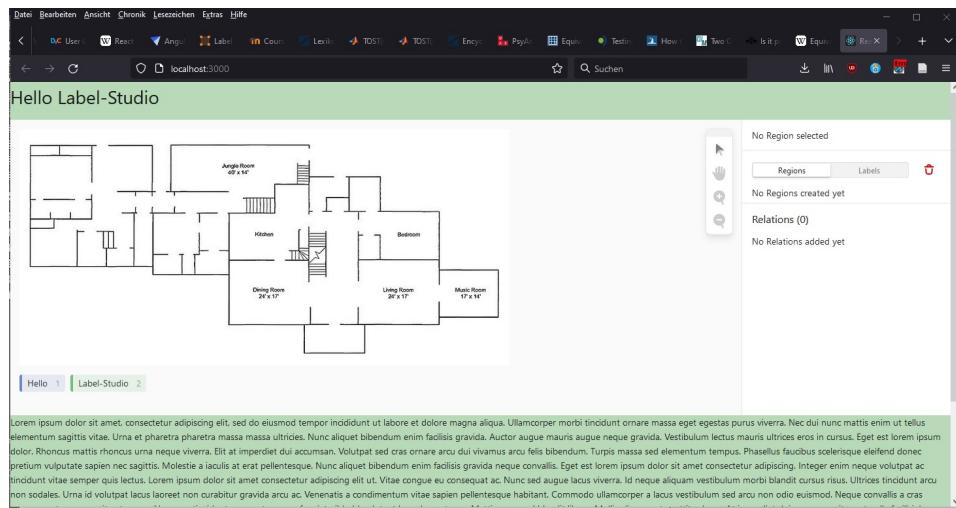


Abbildung 5: Demoimplementation von Label-Studio

## 4. Backend

Dieser Teil des Berichtes deckt den Bereich des Backends ab. Ein Backend ist mehrheitlich für die serverseitige Verarbeitung von Daten zuständig. Im Gegensatz zu einem Frontend (siehe Kapitel 3) hat der Client keinen direkten Kontakt mit diesem.

Für das Backend standen mehrere Schnittstellen und Technologien zur Auswahl. Zu Beginn wird auf die Kommunikation zwischen Front- und Backend sowie Schnittstellen nach aussen eingegangen. Darauffolgend werden verschiedene Technologien für das Backend vorgestellt. Diese wurden getestet und auf ihre Eignung für das Projekt beurteilt. Als Entscheidungskriterium für die Auswahl von Technologien wurde die Grösse der Abhängigkeitskette, Integrierbarkeit und die Lizenzierung verwendet.

### 4.1. Schnittstellen

Für die Kommunikation zwischen Front- und Backend wird REST verwendet. REST steht dabei für Representational State Transfer. Es ist kein Protokoll sondern eine Kommunikationsvereinbarung. Der Austausch von Informationen wird bei REST über eine HTTP-Kommunikation realisiert. Dabei muss der Anbieter der REST-Schnittstelle seine API definieren. Diese gibt an, welche Funktionen inklusive Attributen verarbeitet werden können. Als Alternativen zu REST wären SOAP<sup>5</sup>, WSDL<sup>6</sup> oder RPC<sup>7</sup> zur Diskussion gestanden. Diese sind jedoch abhängiger von den jeweiligen Implementationen. Dadurch sind solche Schnittstellen schwieriger in bestehende System einzufügen. Zudem ist eine spätere Änderung in der Gesamtarchitektur nicht ohne einen erheblichen Mehraufwand realisierbar. REST ist dagegen flexibler in der Implementation und besitzt eine gute Skalierbarkeit.

### 4.2. Framework für das Backend

Um ein Backend zu realisieren, stehen eine Vielzahl von Frameworks zur Verfügung. Die Auswahl wurde zunächst auf jene beschränkt, welche Python als Programmiersprache nutzen. Im Bereich Machine-Learning wird ebenfalls auf Python gesetzt. Dadurch werden mögliche Risiken für die Schnittstelle Backend / Machine-Learner minimiert. Weiter wurde die Auswahl auf Technologien beschränkt, welche unter einer Open-Source-Lizenz veröffentlicht wurden.

Als Framework für das Backend wird FastAPI verwendet. Dieses hat die beste Mischung zwischen Einfachheit und Funktionsumfang. Das Framework ist auf Geschwindigkeit optimiert und besitzt einen relativ kleinen Abhängigkeitsbaum.

#### 4.2.1. Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source [7].

---

<sup>5</sup>Simple Object Access Protocol

<sup>6</sup>Web Services Description Language

<sup>7</sup>Remote Procedure Call

Django ist eines der bekanntesten Backend-Frameworks. Es wird von vielen umfangreichen Webseiten als Backend genutzt. Für diese Zwecke ist es entsprechend ausgereift, dies bedeutet aber gleichzeitig auch einen grossen Abhängigkeitsbaum. Ebenfalls ist es fraglich, wie gut sich eine komplette Django-Umgebung in eine bestehende Web-Infrastruktur einpflegen lässt. Es ist dafür ausgelegt als Backend für ganze Webseiten zu dienen.

Eine baseline Implementation von Django lässt sich mit folgenden Kommandozeilenbefehlen realisieren:

---

```
1 pip install django djangorestframework
2 django-admin startproject [Name des Backend]
3 django-admin startapp [Name der App]
4 cd [Name des Backend]
5 .\manage.py makemigrations
6 .\manage.py migrate
7 .\manage.py runserver
```

---

Anschliessend kann man eine Demo-Webseite im Browser über <http://localhost:8000/> aufrufen.

#### 4.2.2. Flask

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy [8].

Flask ist ebenfalls ein Backend, welches Python nutzt. Es setzt dabei aber auf Minimalismus. Der Funktionsumfang wurde gering gehalten, kann aber durch Module erweitert werden. Für jede Funktionserweiterung muss ein entsprechendes Modul in das Projekt eingebunden werden. Dadurch ist es zwar ein simples und flexibles Framework, durch den modularen Aufbau können jedoch schnell Abhängigkeiten verletzt werden. Flask eignet sich daher eher für die Prototypen-Entwicklung oder kleine eigenständige Web-Dienste.

Eine baseline Implementation von Flask besteht aus Kommandozeilenbefehlen und einem Python-File.

HelloFlask.py:

---

```
1 from flask import Flask
2
3 app = Flask(__name__, static_url_path='', static_folder='frontend/build')
4 CORS(app) #comment this on deployment
5 api = Api(app)
6
7 @app.route('/')
```

---

---

```
8 def hello_world():
9     return "<p>Hello World with Flask</p>"
```

---

Kommandozeile:

---

```
1 pip install flask
2 set FLASK_ENV=development
3 set FLASK_APP=HelloFlask
4 flask run
```

---

Anschliessend kann man die Demo-Webseite im Browser über <http://localhost:5000/> aufrufen.

#### 4.2.3. FastAPI

FastAPI framework, high performance, easy to learn, fast to code, ready for production

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints [9].

Wie der Name schon erkennen lässt, ist dieses Framework auf Geschwindigkeit optimiert. Es ist neuer als die anderen zwei vorgestellten Frameworks. Das geringere Alter bedeutet jedoch nicht, dass es instabil ist. Dieses Framework wird erfolgreich in der Industrie (Uber und Netflix [9]) eingesetzt. Es besitzt eine gute Skalierbarkeit und Flexibilität. Im Vergleich zu den anderen zwei ist die Auswahl an Lernmaterial für dieses Framework jedoch noch begrenzter.

Eine baseline Implementation von FastAPI besteht aus Kommandozeilenbefehlen und einem Python-File.

HelloFastApi.py:

---

```
1 from fastapi import FastAPI
2 from fastapi.responses import HTMLResponse
3
4 app = FastAPI()
5
6
7 @app.get("/", response_class=HTMLResponse)
8 async def root():
9     return "<p>Hello World with FastAPI</p>"
```

---

Kommandozeile:

---

```
1 pip install fastapi
2 pip install "uvicorn[standard]"
3 uvicorn HelloFastApi:app --reload
```

---

Anschliessend kann man die Webseite im Browser über <http://localhost:8000/> aufrufen.

## 5. Bildverarbeitung und Segmentierung

Dieses Kapitel des Berichtes widmet sich der Bildverarbeitung mit Hilfe von Machine-Learning. Der Hauptzweck der Applikation ist die autonome Analyse und Segmentierung von Gebäudeplänen. Da Gebäudepläne in allen Varianten existieren, ist dieser Ansatz vielversprechender als konventionelle Methoden. Auf das Machine-Learning und dessen Vorteile wird im ersten Teil dieses Kapitels eingegangen. Anschliessend werden unterschiedliche Arten von Learnern behandelt. Ein weiterer Punkt sind die eigentlichen Daten und somit auch die zur Verfügung stehenden Datensätze in diesem Bereich. Es werden verschiedene Datensätze vorgestellt, welche für die Bachelorarbeit genutzt werden könnten. Abschliessend wird auf die Handhabung der Daten und deren Life-Cycle eingegangen.

### 5.1. Machine-Learning

Die Herangehensweise vor dem Machine-Learning waren Expertensysteme. Bei diesen wurden auf Grundlage von Datenanalysen Regeln definiert. Diese Regeln wurden mittels Algorithmen implementiert. Dadurch erhielt man ein System, welches eine klar definierte Aufgabe nach einer definierten Funktionsweise erledigen konnte. Solche Systeme sind jedoch für komplexere Problemstellungen nicht realisierbar. Systeme mit Machine-Learning nutzen stärker die Daten und das Aufstellen der Regeln tritt in den Hintergrund. Dadurch ist die effektive Funktionsweise des Systems zwar nicht genau definiert, es können dafür komplexere Aufgabenstellungen bewältigt werden.

Eine Auswahl von verschiedenen Arten von Machine-Learnern wurde im Fachmodul genauer angeschaut.

#### 5.1.1. CNN: Convolutional Neural Network

Ein CNN besteht aus einem Neuronalen Netz, welches sich besonders für das Klassifizieren von Bild- und Videodateien eignet. Wie jedes neuronale Netz besteht es aus einem Inputlayer, einem Outputlayer und mehreren Hiddenlayers. Einer der wichtigsten Hiddenlayer des CNN ist der Convolutionallayer. Bei diesem wird eine Matrix über jeden Bildpunkt gelegt, um daraus bestimmte Muster im Bild zu erkennen [10].

Mithilfe eines CNNs ist zum Beispiel die Erkennung von Zahlen möglich. Dazu wird hier der TMINST-Datensatz [11] verwendet, welcher verschiedene Google Fonts beinhaltet. Ein kleiner Ausschnitt der Daten ist in Abbildung 6 ersichtlich.

9	8	4	9	1
9	9	2	5	6
2	1	8	0	0
0	7	7	7	6
4	0	0	2	3

Abbildung 6: Auschnitt des TMINST-Datensatzes

Das Modell wird sequentiell aufgebaut. Es besteht aus zwei Convolutionallayern und einem Flattenlayer. Der Flattenlayer wird gebraucht, um die Bildmatrix in ein Array zu konvertieren. Das fertige Model ist in Abbildung 7 zusammengefasst. Weiterhin muss ein Optimierungsalgorithmus gewählt werden, welcher die Lernrate des Models steuert. Ist die Lernrate zu langsam, beansprucht das Trainieren des Modells sehr viel Zeit. Bei einer zu schnellen Lernrate wird das Modell keine konstante Verbesserung erreichen können. Mit der Verlustfunktion lässt sich ein Overfitting erkennen. Beim Overfitting schneidet das Modell auf ungesehnen Daten schlechter ab.

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
Conv2D_1 (Conv2D)	(None, 25, 25, 32)	544
Conv2D_2 (Conv2D)	(None, 23, 23, 64)	18496
flatten_1 (Flatten)	(None, 33856)	0
Ausgabe (Dense)	(None, 10)	338570

Total params: 357,610  
Trainable params: 357,610  
Non-trainable params: 0

Abbildung 7: CNN-Modell

Das erstellte Modell wird mit den Trainingsdaten trainiert und mit den Validierungsdaten überprüft. Die Trainingsphase wird gestoppt, wenn die Verlustfunktion anfängt anzusteigen. Eine visuelle Darstellung der Erkennung eines Bildes ist in Abbildung 8 zu sehen. Der Einfluss auf das Ergebnis jedes einzelnen Knotens wird mittels der Farbe dargestellt (Blau = wenig Einfluss / Rot = hoher Einfluss). Das fertige Modell erkennt die Zahl richtig als eine Sieben.

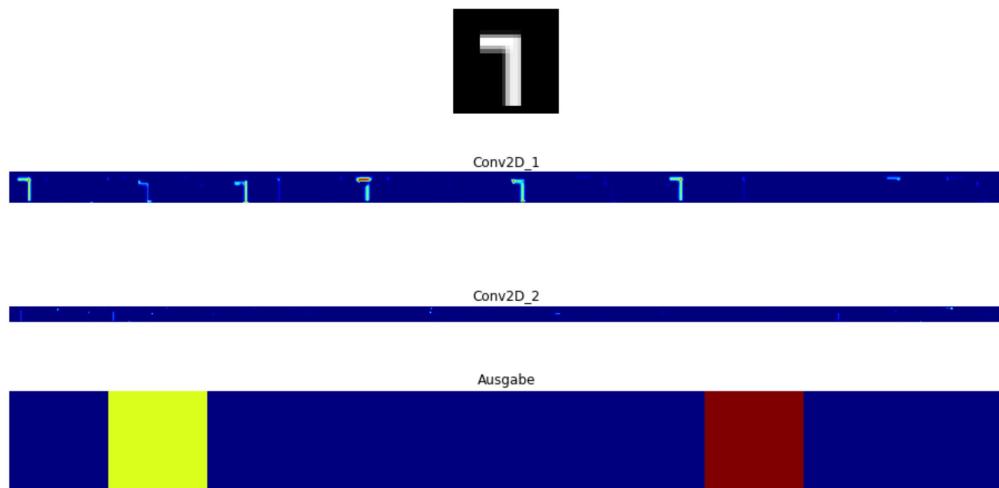


Abbildung 8: Visualisierung der Funktionsweise eines CNNs

### 5.1.2. R-CNN: Region Based Convolutional Neural Network

Beim R-CNN wird, wie der Name schon sagt, das Bild in Regionen aufgeteilt. Dabei kann aber nicht jeder Abschnitt eines Bildes klassifiziert werden, da dies eine enorme Rechenaufgabe wäre. Stattdessen werden mit Hilfe von Pixelhelligkeit, Farbe und weiteren Methoden verschiedene Regionen im Bild bestimmt. Die einzelnen Regionen werden dann klassifiziert. Der Fast R-CNN und der Faster R-CNN sind Nachfolger des R-CNNs und verbessern vor allem die Geschwindigkeit des Algorithmus [12].

### 5.1.3. Mask R-CNN

Der Mask R-CNN ist eine Erweiterung des Faster R-CNNs. Als Erstes werden auch hier die Regionen gesucht und diese klassifiziert. Für jede Region wird jeder Pixel klassifiziert, um daraus eine Maske zu generieren. Diese Maske wird dann auf die richtige Größe skaliert und am richtigen Ort im Bild eingesetzt. Das fertig segmentierte Bild (Beispiel Abbildung 9) enthält für jedes erkannte Objekt eine Erkennungswahrscheinlichkeit, eine Maske und eine Region (Bounding-Box)[13].



Abbildung 9: Segmentiertes Bild eines Mask R-CNNs [13]

Mit diesem Algorithmus wäre es möglich die einzelnen Labels auf einem Bauplan zu erkennen. Darüberhinaus können die einzelnen Pixel der Gegenstände zu einem Label zugewiesen werden. Dies ermöglicht das Erkennen von exotischen Formen auf dem Bauplan.

### 5.1.4. NLP: Natural-Language-Processing

Die bisher vorgestellten Learner basieren alle auf Bilderkennung. Bei Fotografien ist der Inhalt meist zufällig, da zufällige Situationen festgehalten werden. Bei einem Gebäudeplan werden jedoch bewusst alle nötigen Informationen festgehalten. Wie die Sprache werden Pläne zur Kommunikation genutzt. Ebenfalls können Teile eines Planes nicht korrekt interpretiert werden, wenn der Kontext fehlt. Pläne sind eine Art visuelle Sprache, welche Gebäude beschreiben. Ein Grundriss, welcher eine Wohnung abbildet, wird eine Nasszelle und eine Küche enthalten und diese Räume

werden eher klein sein. Eine Lagerhalle beinhaltet jedoch Regale und Säulen und es wird ein grosser zusammenhängender Raum existieren.

Daher wäre ein weiterer Ansatz, die Grundrisspläne auch wie eine Sprache zu betrachten. Dies würde bedeuten, dass die Informationen von einem Teil des Planes auf den Kontext des ganzen Planes angewiesen sind. Dieser Ansatz geht jedoch über den Umfang dieser Bachelorarbeit hinaus und wird nicht weiter verfolgt.

## 5.2. Datensätze

Für die Bildsegmentierung von normalen Motiven, wie zum Beispiel von Menschen oder Tieren, sind mehrere bekannte Datensätze vorhanden. Für die Segmentierung von Gebäudeplänen sind die Daten spärlicher. Dies weil zum Einen in diesem Gebiet noch wenige Forschungsarbeiten existieren und zum Anderen, weil Grundrisspläne hochkomplex und zum Teil auch sehr unterschiedlich sind. Je nach Planart werden Wände unterschiedlich dargestellt, zum Beispiel werden sie auf Bauplänen schraffiert und auf Einrichtungsplänen eher einheitlich eingefärbt. Des Weiteren werden oft unterschiedliche Farben für das Gleiche genutzt oder die Pläne beinhalten eine hohe Informationsdichte<sup>8</sup>. In der Abbildung 10 werden zwei Grundrisspläne gezeigt, welche die Problematik gut darstellen. Diese Beispiele wurden von der Firma Kemaro für diese Arbeit zur Verfügung gestellt. Im Verlauf des Fachmodules wurden weitere Datensätze gefunden, welche hier kurz vorgestellt werden. Ein optimaler Datensatz enthält möglichst viele unterschiedliche Pläne mit der dazugehöriger Ground-Truth. Diese gibt dem Machine-Learner das gewünschte Endresultat vor und ermöglicht erst einen Lernprozess. Die Ground-Truth ist in diesem Fall die korrekt gelabelten Räume und Hindernisse.



Abbildung 10: Beispiele von Grundrissplänen

<sup>8</sup>Architektur, Wasserleitungen, Stromleitungen und Einrichtungsgegenstände in einem Plan kombiniert.

### 5.2.1. SESYD Dataset

SESYD (Systems Evaluation SYnthetic Documents) is a database of synthetical documents, with the corresponding groundtruth, produced using the 3gT system [14].

Dieser Datensatz enthält Symbole und Grundrisspläne in verschiedenen Auflösungen. Die Bilder wurden mittels einer Applikation (3gT<sup>9</sup>) von einer XML-Definition erzeugt. Die dazugehörigen XML- und SVG-Files bilden somit auch die Ground-Truth. Der Umfang beträgt ungefähr 150 unterschiedliche Symbole und 1000 Pläne. In Abbildung 11 ist ein Symbol-Bag und ein Grundrissplan aus diesem Datensatz zu sehen.

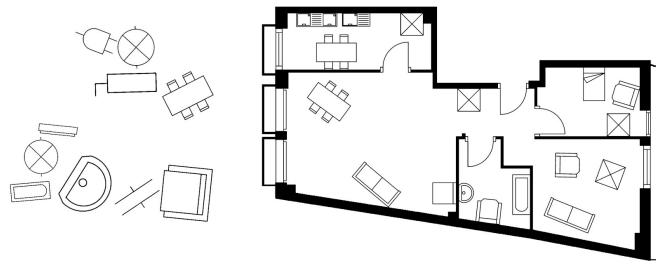


Abbildung 11: Beispiele von SESYD Dataset [14]

### 5.2.2. ROBIN Dataset

Dieser Datensatz besteht aus ungefähr 400 Grundrissplänen. Eine Ground-Truth ist nicht vorhanden. Der Datensatz selbst wurde in mehreren Forschungsarbeiten für eine automatische Raumerkennung zitiert. Die Grundrisse sind als genaue Zeichnungen und als Handzeichnungen vorhanden (siehe Abbildung 12). Die Abkürzung ROBIN steht für Repository Of Building plaNs. [15]

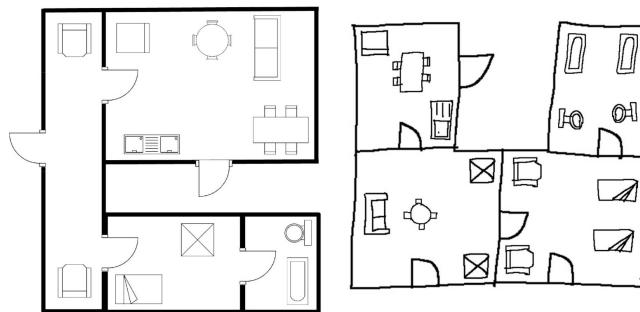


Abbildung 12: Beispiele von ROBIN Dataset [15]

<sup>9</sup>generation of graphical ground Truth

### 5.2.3. FPLAN-POLY Dataset

The main goal of this database is to provide a framework for the evaluation of different symbol spotting methods in vectorized graphic documents [16].

Dieser Datensatz besteht aus 42 Grundrissplänen und 38 Symbolen im DXF-Format. Die Ground-Truth steht im XML-Format für die Symbole zur Verfügung. Das Ziel dieses Datensatzes ist nicht die Raumerkennung selber, sondern die Symbolerkennung in Grundrissplänen. Ein Beispiel eines solchen Plans ist in Abbildung 13 zu sehen.

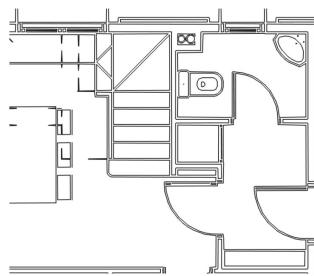


Abbildung 13: Beispiele von FPLAN-POLY Dataset [16]

### 5.2.4. CVC-FP Dataset

The collection consists of 122 scanned floor plan documents divided in 4 different subsets regarding their origin and style. It contains documents of different qualities, resolutions, and modeling styles, which is suitable to test the robustness of the analysis techniques. [17]

Dieser Datensatz besteht aus 122 Grundrissplänen mit der dazugehörigen Ground-Truth der Räume als Vektorgrafik. In der Vektorgrafik sind ebenfalls die verschiedenen Labels enthalten. Der Datensatz stammt von der Pattern Recognition and Document Analysis Group des Computer Vision Center der Universität Autònoma de Barcelona. In der Abbildung 14 ist links die Ground-Truth und rechts der ursprüngliche Grundrissplan zu sehen.

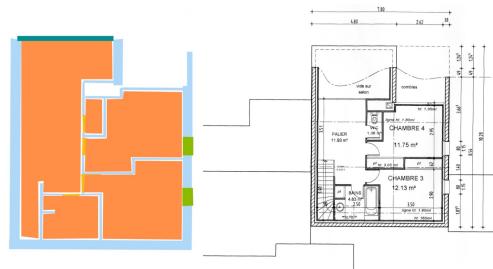


Abbildung 14: Beispiele von CVC-FP Dataset [17]

### 5.3. Data-Versioning-System

In der Softwareentwicklung ist der Einsatz von einem Versionsverwaltungssystem<sup>10</sup> Branchenstandard. Es dient zur Erfassung und Speicherung von Änderungen am Sourcecode. Dadurch ist es möglich auf frühere Versionen des Programmes zurückzugreifen. Es ist wie ein Backupsystem, nur werden nicht Kopien der Arbeitsdokumente verwaltet sondern die Änderungen an diesen. Der grösste Vorteil eines VCS ist, dass man jede Änderung in dem Code automatisch dokumentiert hat und man diese Änderungen bei Bedarf auch rückgängig machen kann.

Auch ein Machine-Learning-Projekt ist auf eine Versionskontrolle bei den Daten und Modellen angewiesen. Insbesondere falls das Modell kontinuierlich weiter trainiert werden soll. Ebenfalls ist ein Data-Version-Controlsystem<sup>11</sup> nützlich, um Daten und Modelle zu protokollieren und zu speichern. Dies ermöglicht einen einfachen Austausch der Daten- und Modellversionen. Des Weiteren können so die Metriken und Performance der unterschiedlichen Varianten verglichen werden. Auch können dadurch verschiedenen Nutzern verschiedene Modelle zur Verfügung gestellt werden [18]. Die Anbindung und Implementation eines solchen Systems ist nicht im Umfang dieser Arbeit enthalten. Es werden jedoch Schnittstellen implementiert, mit welchen man die aktuellen Daten und Modelle beziehen kann. Ebenfalls decken die Schnittstellen das Austauschen des Modells und der Daten ab. Eine mögliche Lösung für ein DVC ist das Open-Source Versionskontrollsystem für Machine-Learning-Projekte von iterative.ai<sup>12</sup>.

---

<sup>10</sup>VCS: Version Control System

<sup>11</sup>DVC

<sup>12</sup>[www.dvc.org](http://www.dvc.org)

## 6. Projektplanung

Der in Abbildung 15 vorgestellte Zeitplan mit den dazugehörigen Meilensteinen kann sich während der Bachelorarbeit noch verändern. Für eine exakte Planung fehlen Referenzpunkte und die nötige Erfahrung mit Machine-Learning-Projekten.

Das Projekt ist grob in vier Phasen aufgeteilt. Die Initialisierungsphase beinhaltet die Einarbeitung in weitere Themen sowie das Austesten von schon fertig trainierten Machine-Lernern. Sie ist kurz gehalten, damit man möglichst schnell mit der Prototypenphase beginnen kann. Hier werden Prototypen des Machine-Learner, Front- und Backends implementiert und evaluiert. Außerdem finden diverse Definitionen hier statt, die das UI/UX<sup>13</sup> und die Schnittstellen umfassen. Die Implementierungsphase folgt als nächstes, hier werden die Prototypen richtig implementiert und zu einem Paket zusammengeschürt. Die letzte Phase ist die Abschlussphase. Diese beinhaltet eine Demo-Webseite und die Fertigstellung des Berichtes. Dieser wird über die vier Phasen immer weiter ausgebaut. Diese Phase und somit auch die Bachelorarbeit wird mit der Präsentation und Abgabe des Berichtes abgeschlossen.

### 6.1. Meilensteine

Die Meilensteine dienen als Orientierungspunkte während der Bachelorarbeit. Sie sind nötig, um mit der jeweiligen nächsten Phase beginnen zu können. Zudem dienen sie zur Fortschrittskontrolle. Nachfolgend sind die jeweiligen Meilensteine kurz beschrieben.

#### 1. Abschluss Initialisierungsphase

Die letzten Anpassungen wurden am Pflichtenheft vorgenommen und es wurde abgesegnet. Es wurden ausgetestet, ob vortrainierte Machine-Lerner mit Plänen umgehen können.

#### 2. alle Prototypen erstellt

Die Prototypen für Machine-Learner, Back- und Frontend wurden erstellt.

#### 3. Abschluss Prototypenphase

Die Prototypen wurden evaluiert. Die Schnittstellen und das UI/UX wurden definiert.

#### 4. alle Implementationen fertig

Der Machine-Learner, Back- und Frontend wurden implementiert.

#### 5. Abschluss Implementierungsphase

Die Implementationen wurden getestet und zu einem Paket zusammengestellt.

#### 6. Dokumentation fertig

Die Dokumentation wurde fertig gestellt. Es sind alle Schreibarbeiten erledigt.

#### 7. Präsentation und Abgabe der Bachelorarbeit

Die Präsentation hat stattgefunden und der Bericht wurde abgegeben.

---

<sup>13</sup>User Interface / User Experience

## 6.2. Zeitplan

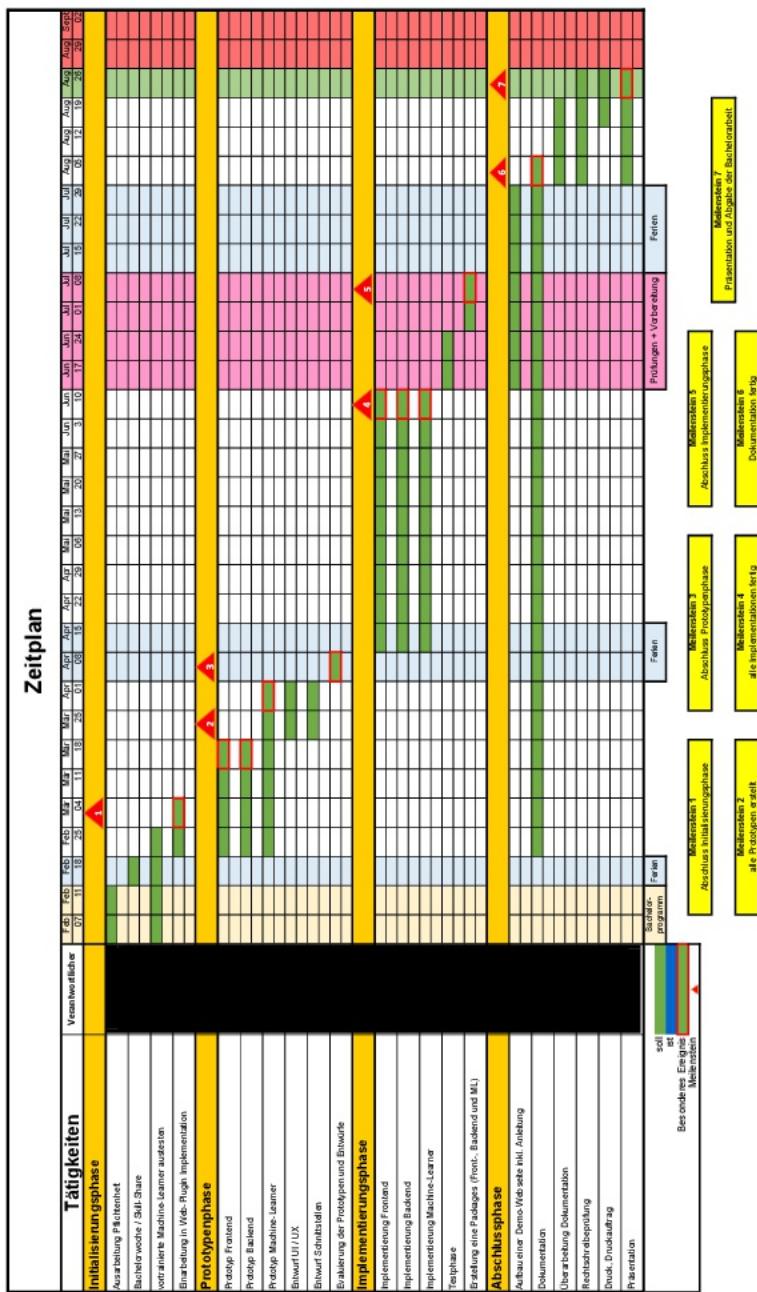


Abbildung 15: Zeitplan

### 6.3. Gefahren und Risiken

Die grösste Gefahr besteht in der Bildsegmentierung. Da diese erst in der Bachelorarbeit thematisiert wird, ist eine Risikoanalyse schwierig. Die komplexen Baupläne, die auch von einem Mensch nur schwer zu interpretieren sind, machen das Segmentieren zu einer Herausforderung. Der Prozess des Nachlernens könnte durch falsch gelabelte oder exotische Baupläne behindert werden.

Das Backend ermöglicht den Zugriff auf die Daten von verschiedenen Bauplänen. Falls das Backend nicht richtig abgesichert wird, könnte es einem Angreifer den Zugriff darauf ermöglichen. Beim Frontend ist der Einsatz eines Plug-ins für eine Webseite noch unsicher. Da sich mehrere verschiedene Seiten im Plug-in befinden müssen, könnte eine Implementation erschwert werden.

## 7. Aufgabenstellung und Zieldefinition

Das Ziel dieser Bachelorarbeit ist es, ein leicht integrierbares Labelling-Tool zu entwickeln. Dieses dient dem Labeln von Räumen und Hindernissen auf Grundrissplänen. Dabei soll das Labelling von einem Machine-Learner unterstützt werden. Dieser soll sich aufgrund der Daten kontinuierlich weiter verbessern können. Diese Daten entstehen durch die Nutzung des Tools. Dieses ist als eine Human-in-the-Loop-Applikation konzipiert. Der Mehrwert dieser Arbeit besteht daraus, die vorhandenen Lösungen und Technologien zu einem leicht integrierbaren Paket zusammenzuführen. Dieses Paket ist auf die Segmentation und das Labelling von Grundrissplänen spezialisiert. Die folgenden Anforderungen sollen mit dieser Bachelorarbeit erfüllt werden.

- Das Tool ist einfach in bestehende Webseiten zu integrieren.
- Das Tool implementiert den Arbeitsablauf 'Human in the Loop'.
- Das Tool implementiert alle nötigen Funktionen für eine Segmentierung von Gebäudeplänen.
- Der Machine-Learner kann einfach ausgetauscht werden.
- Das Frontend nutzt React.
- Es existiert eine Schnittstelle, um ein Data-Versioning-System anbinden zu können.
- Es können Grundrisspläne gelabelt werden.
- Das Labeln wird von einem Machine-Learner unterstützt. Die Qualität der Segmentierung ist zu diesem Zeitpunkt noch nicht abzuschätzen.
- Das Projekt wird unter der Open-Source-Lizenz veröffentlicht.

Die Verantwortung für ein funktionstüchtiges Frontend wird grösstenteils von [REDACTED] übernommen, wohingegen die Funktionalität des Backends hauptsächlich von [REDACTED] übernommen wird. Bei der Bildsegmentierung sind beide zu gleichen Teilen verantwortlich. Im Zeitplan (Abbildung 15) sind die genauen Verantwortlichkeiten definiert.

## I. Verzeichnisse

### Abbildungsverzeichnis

1.	Komponentendiagramm des Front- und Backends . . . . .	2
2.	Kommunikationsdiagramm der Bauplansegmentierung . . . . .	3
3.	Mögliche Webseitendarstellung der Firma X mit dem React Plug-in . . . . .	3
4.	Labellingbeispiel eines Grundrisses . . . . .	6
5.	Demoimplementation von Label-Studio . . . . .	10
6.	Auschnitt des TMINST-Datensatzes . . . . .	14
7.	CNN-Model . . . . .	15
8.	Visualisierung der Funktionsweise eines CNNs . . . . .	15
9.	Segmentiertes Bild eines Mask R-CNNs [13] . . . . .	16
10.	Beispiele von Grundrissplänen . . . . .	17
11.	Beispiele von SESYD Dataset [14] . . . . .	18
12.	Beispiele von ROBIN Dataset [15] . . . . .	18
13.	Beispiele von FPLAN-POLY Dataset [16] . . . . .	19
14.	Beispiele von CVC-FP Dataset [17] . . . . .	19
15.	Zeitplan . . . . .	22

### Literatur

- [1] S. Springer, *React: das umfassende Handbuch*, 1st ed., ser. Rheinwerk Computing. Bonn: Rheinwerk Verlag, 2020.
- [2] P. Skalski, “makesense.ai,” Jan. 2022. [Online]. Available: <https://github.com/SkalskiP/make-sense>
- [3] J. Brooks, “Features,” Jan. 2022. [Online]. Available: <https://github.com/jsbroks/coco-annotator>
- [4] Get started with docker | docker. [Online]. Available: <https://www.docker.com/get-started>
- [5] B. Lin, “react-annotation-tool,” Dec. 2021. [Online]. Available: <https://github.com/bennylin77/react-annotation-tool>
- [6] “Label Studio Frontend .,” Jan. 2022. [Online]. Available: <https://github.com/heartexlabs/label-studio-frontend>
- [7] “Django.” [Online]. Available: <https://www.djangoproject.com/>
- [8] “Flask.” [Online]. Available: <https://palletsprojects.com/p/flask/>
- [9] “FastAPI.” [Online]. Available: <https://fastapi.tiangolo.com/>
- [10] M. Deru and A. Ndiaye, *Deep Learning mit TensorFlow, Keras und TensorFlow.js*, 1st ed., ser. Rheinwerk Computing. Bonn: Rheinwerk Verlag, 2019.

- [11] “TMNIST (Typeface MNIST).” [Online]. Available: <https://kaggle.com/nimishmagre/tmnist-typeface-mnist>
- [12] U. Michelucci, *Advanced applied deep learning: convolutional neural networks and object detection*, ser. For professionals by professionals. New York: Apress, 2019.
- [13] “Mask R-CNN for Object Detection and Segmentation,” Jan. 2022, original-date: 2017-10-19T20:28:34Z. [Online]. Available: [https://github.com/matterport/Mask\\_RCNN](https://github.com/matterport/Mask_RCNN)
- [14] “Systems Evaluation SYnthetic Documents.” [Online]. Available: <http://mathieu.delalandre.free.fr/projects/sesyd/>
- [15] C. Chattopadhyay, “ROBIN,” Oct. 2021. [Online]. Available: <https://github.com/gesstalt/ROBIN>
- [16] M. Rusiñol, A. Borràs, and J. Lladós, “Relational indexing of vectorial primitives for symbol spotting in line-drawing images,” *Pattern Recognition Letters*, vol. 31, no. 3, pp. 188–201, Feb. 2010. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S016786550900275X>
- [17] “CVC-FP: Database for structural floor plan analysis.” [Online]. Available: <http://dag.cvc.uab.es/resources/floorplans/>
- [18] “Data Version Control · DVC.” [Online]. Available: <https://dvc.org/>

## II. Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbstständig und nur unter Verwendung der von uns angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Datum: 07.01.22 Unterschrift: 

Datum: 07.01.2022 Unterschrift: 