HAW HAMBURG

BACHELOR THESIS
Zafar Akhtar

Raum-Koordinatenbestimmung durch Deep-Learning-Analyse von rasterbasierten Grundrissplänen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Zafar Akhtar

Raum-Koordinatenbestimmung durch Deep-Learning-Analyse von rasterbasierten Grundrissplänen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stephan Pareigis
Zweitgutachter: Prof. Dr. Michael Neitzke

Eingereicht am: 30.09.2022

Zafar Akhtar

Thema der Arbeit

Raum-Koordinatenbestimmung durch Deep-Learning-Analyse von rasterbasierten Grundrissplänen

Stichworte

Instance-Segmentierung, Deep-Learning, rasterbasierte Grundrisspläne, Harris-Corner-Detector

Kurzzusammenfassung

Ein Grundrissplan ist ein Bestandteil von Bauzeichnungen und stellt eine maßstabsgetreue zeidimensionale Abbildung der Räumlichkeiten einer Etage da. In dieser Arbeit wurde die Anwendbarkeit von Deep-Learning mit Schwerpunkt Instanz-Segmentierung auf rasterbasierte Grundrisspläne untersucht, um so Eckpunkte eines Raums zu bestimmen. Es wurde ein mehrstufiges System konzipiert, bei welchem das Mask R-CNN Modell Masken von Räumen erzeugt, welche anschließend mit dem Harris-Corner-Detector nach Ecken abgesucht werden. Grundlage der Daten bilden dabei frei zugängliche als auch ein selbst annotierter Datensatz. In den Experimenten wurde das Mask R-CNN quantitativ evaluiert und Vergleiche zwischen unterschiedlichen Konfigurationen vorgenommen. Anschließend wurden Ergebnisse qualitativ ausgewertet. Es stellte sich heraus, dass das System überwiegend Räume als solches detektieren und segmentieren kann. Jedoch sind Grundrisspläne mit vielen Störquellen ein Hindernis, weshalb Masken in ungleichmäßigen Dimensionen erzeugt werden. Werden Masken gleichmäßig erzeugt, können Ecken effizienter lokalisiert werden.

Zafar Akhtar

Title of Thesis

Room coordinate determination through deep learning analysis of bitmap floor plans

Keywords

Instance segmentation, Deep learning, Bitmap floor plan, Harris-Corner-Detector

Abstract

A floor plan is part of architectural drawings and represents a true-to-scale, two-dimensional representation of the premises on a floor. In this work, the applicability of deep learning with a focus on instance segmentation on bitmap floor plans was examined in order to determine corner points of a room. A multi-stage system was designed in which the Mask R-CNN model generates masks of rooms and these are then searched for corners with the Harris-Corner-Detector. The data is based on freely accessible and self-annotated data sets. In the experiments, the Mask R-CNN was evaluated quantitatively and comparisons were made between different configurations. The results were then evaluated qualitatively. It turned out that the system can predominantly detect and segment rooms as such. However, floor plans with many sources of interference are an obstacle, which is why masks are generated in non-uniform dimensions. If masks are generated evenly, corners can be located more efficiently.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

CNN Convulutional-Neural-Networks.

DNN Deep-Neural-Network.

FCN Fully-Convolutional-Neural-Network.

MLP Multi-Layer Perceptron.

RPN Region-Proposal-Network.

ROI Region of Interest.

1 Einleitung

1.1 Problemstellung und Motivation

Die Software "GebäudeCheck" der PLAN4 Software GmbH vereinfacht den Alltag von Architekten, Bauingenieuren, Bestandshalter und Kommunen. Um weiterhin im Markt den Anwendern ein optimales Benutzererlebnis zu ermöglichen, steht die Digitalisierung der Daten aus den Grundrissen als Nächstes in der Entwicklung. Grundrisspläne sind zweidimensionale technische Zeichnungen, welche ein Gebäude in abstrakter Form darstellen und von menschlichen Akteuren interpretiert werden. Erzeugt werden diese nach heutigem Standard mit Einsatz von moderner Zeichenprogramme und werden als CAD-Dateien [1] abgespeichert. Jedoch existiert eine Vielzahl von Grundrissplänen, die nicht auf dieser Technologie beruhen. Überwiegend handelt es sich um technische Zeichnungen, welche ausgedruckt, optional händisch beschriftet und anschließend wieder eingescannt wurden. Dabei liegen diese als Bilddateien in Form von Rastergrafiken (z.B. JPG, PNG) vor. Im Gegensatz zu Vektorgrafiken beinhalten Rastergrafiken keine semantischen Informationen wie geometrische Primitiven oder Daten zum topologischen Aufbau, sondern bestehen ausschließlich aus Pixeln. Für PLAN4 besteht damit der Bedarf für eine Lösung zur automatisierten Verarbeitung von rasterbasierten Grundrissplänen.

Computer Vision (zu dt. maschinelles Sehen), welches ein Feld innerhalb der künstlichen Intelligenz darstellt, beschäftigt sich damit, aus digitalen Bildern bzw. Rastergrafiken aussagefähige Informationen zu gewinnen. Die Forschungsgemeinde demonstriert laufend in Wettbewerben die aktuelle Leistungsfähigkeit von neuronalen Netzen. Besonders hervorzuheben ist die ImageNet-Challenge, welche 1000 Klassen beinhaltet und jährlich veranstaltet wird, um neueste Errungenschaften aus der Forschung zu präsentieren [19]. Erprobt werden neuronale Netze bereits in den unterschiedlichsten Alltagsszenarien wie z.B. Autonomes Fahren, Überwachungssysteme und Gesichtserkennung.

[1] Spezielles Vektorgrafik-Format für die Entwicklung von 2D und 3D Objekten.

1.2 Zielsetzung

Die vorliegende Arbeit untersucht, wie neuronale Netze zur Extraktion von Raumkoordinaten aus Rastergrafiken verwendet werden können.

Der Forschungs-Schwerpunkt liegt dabei auf der Optimierung der automatisierten Koordinatenerkennung von Raumecken. Die in der Praxis erforderliche Unterstützung für nachträgliche manuelle Fehlerkorrekturen wird dagegen nicht weiter betrachtet.

Entscheidend für die Resultate ist unter anderem die Anzahl und die Güte der Datensätze, die als Beispiele zur Parametrisierung der neuronalen Netze verwendet werden. Diese wurden dankenswerterweise von PLAN4 Software GmbH zur Verfügung gestellt, mit welcher eine Kooperation für diese Arbeit besteht. Darüber hinaus wurden überwiegend frei zugängliche Datensätze verwendet, welche entsprechend gekennzeichnet sind.

1.3 Abgrenzung

Im Rahmen der vorliegenden Arbeit wurde nicht angestrebt, eine neue Architektur für neuronale Netze zu definieren. Vielmehr sollte aus praktischen Gründen eine bestehende Architektur aus dem Bereich der Instanz-Segmentierung für die Erreichung des obigen Ziels eingesetzt werden.

1.4 Anforderungen

Aus den Grundrissplänen, welche als Rastergrafiken vorliegen, sollen automatisiert Pixel-Koordinaten von Räumen, in Form von Eckpunkten, ermittelt und im JSON-Format der "GebäudeCheck" Software zur Verfügung gestellt werden. Um die Raumbestimmung des neuronalen Netzes möglichst effizient zu machen, werden im Rahmen der Untersuchung Trainingsdaten mit Räumen verwendet, welche beliebig viele Raumecken aufweisen. Die Auflösung der Rastergrafiken ist stets größer oder gleich 1024x1024x3.

2 Grundlagen

Dieses Kapitel bietet einen Überblick über die verwendeten Konzepte und Techniken, beginnend mit einer Einführung in die Struktur eines Grundrissplans und die Grundlagen neuronaler Netze. Dargestellt wird der Unterschied zwischen reinen Klassifizierungsnetzen und Segmentierungsnetzen sowie die Definition von Fehlerfunktionen und Metriken. Abschließend wird der Harris-Corner-Detector erläutert.

2.1 Grundrisspläne

Ein Grundrissplan ist ein Bestandteil von Bauzeichnungen und stellt einen frühen Entwurf da, welcher iterativ bis zur Finalisierung mit dem Bauherren und dem Architekten überarbeitet wird. Er ist eine maßstabsgetreue zweidimensionale Abbildung, die in einer Aufsicht alle Räumlichkeiten einer Etage enthält. Gewählt wird dabei eine senkrechte Ansicht auf die Etage mit einem waagerechten Schnitt in einem Meter Höhe über dem Fußboden [13]. Ein Grundrissplan beinhaltet viele semantische Informationen, problematisch dabei ist die fehlende einheitliche Semantik, wodurch Grundrisspläne in Teilen unterschiedlich aussehen können, nachfolgend dazu einige Beispiele. So wird die Raumgröße wahlweise textuell im Raum selbst oder als Maßgabe in Form von Länge und Breite angegeben. Optional kann eine Raumbezeichnung weggelassen und beispielsweise durch markante Möbel ersetzt werden. Wandsegmente werden sowohl als dicke oder als auch dünne Linien dargestellt. Eine weitestgehend einheitliche Symbolik findet sich bei Türen und Fenstern.

In Abbildung 2.1 wird ein einfacher Grundrissplan eines Schwimmbades dargestellt, dabei sind Wandsegmente als dicke graue Linien gezeichnet. Raumbezeichnungen sind in den entsprechenden Räumen hinterlegt und die Raumgröße ist bei wenigen Räumen als Längen und Breiten angegeben. Fenster werden durch Lücken in den Wänden und Türen mit Öffnungsschlag symbolisiert. Eine Besonderheit sind die schraffierten Wände in

Rot und Gelb, die mögliche weitere Räume andeuten. Es handelt sich nicht um einen endgültigen Plan, sondern einen frühen Entwurf.

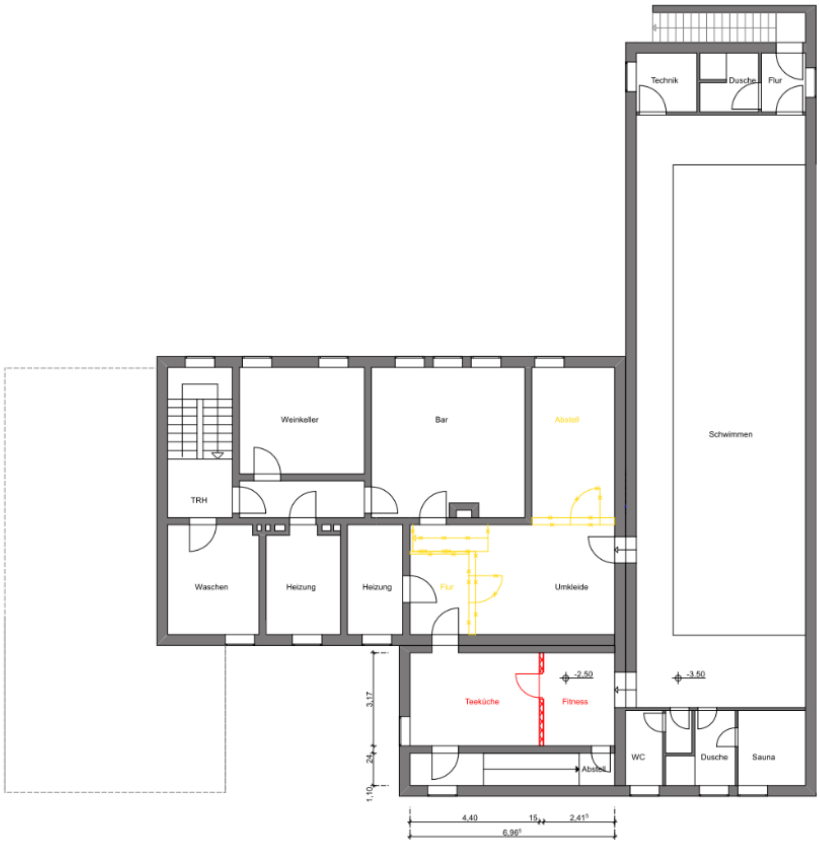

Abbildung 2.1: Beispiel eines frühen Entwurfs aus der PLAN4 Datensammlung. Die schraffierten Wände in Rot und Gelb stellen weitere mögliche Räume da. Name: PLAN-191

2.2 Neuronale Netze

Die Anfänge neuronaler Netze liegen in den 40er Jahren des 20. Jahrhunderts, erste Fortschritte machte Frank Rosenblatt im Jahre 1958. Rosenblatt [18] veröffentliche seine Forschungsarbeit über das Perzeptron, welches aus einem einzigen Neuron mit Gewichten, einem Schwellenwert und Ausgabevektor bestand. Dabei war es in seiner Möglichkeit limitiert und konnte ausschließlich linear separierbare Mengen wie AND, OR und NOT klassifizieren. Da Rechenleistung zur damaligen Zeit nur in geringem Maße zur Verfü-

gung stand, konnten keine komplexeren Modelle wie z.B. das DNN (deutsch: Tiefes-Neuronales-Netz) aufgebaut werden. Ein DNN besteht aus einer Eingangsschicht, gefolgt von zahlreichen Zwischenschichten, welche anschließend in eine Ausgabeschicht münden. Diese Anordnung erlaubt die Extraktion von beliebigen Features (deutsch: Merkmale) und qualifiziert das DNN für komplexere Anwendungsgebiete.

Die in neuronalen Netzen gespeicherten Gewichte werden grundsätzlich mit Hilfe selbstlernender Algorithmen bestimmt. Dabei unterscheidet man zwischen den folgenden drei Lernverfahren. Für diese Arbeit ist ausschließlich das Überwachte Lernen von Bedeutung.

1. Überwachtes Lernen: Das Training verläuft anhand von gelabelten Daten, bei der die Ausgabe des Modells mit dem zu erwartendem Wert verglichen wird. Der Nachteil dabei ist, dass das Modell nur so gut sein kann, wie die Trainingsdaten selbst.
2. Unüberwachtes Lernen: Das Modell passt sich selbst an, indem es ohne Vorgaben versucht, Muster in den Daten zu erkennen.
3. Bestärkendes Lernen: Das Modell wird trainiert, in dem es danach strebt, das Ergebnis einer Belohnungsfunktion zu optimieren. Im Unterschied zum Überwachten Lernen ist keine große Menge an Trainingsdaten erforderlich.

2.2.1 Convolutional-Neural-Networks

CNNs haben sich bei der Verarbeitung von Bildern als besonders effizient erwiesen. Popularität gewannen CNNs, welches eine Form von DNNs darstellen, durch die Forschungsarbeit von LeCun et al. [9]. Sie zeigten quantitativ die Überlegenheit von CNNs gegenüber reinen MLP am Beispiel der Klassifikation von handschriftlichen Zahlen. Ein CNN erlaubt die Codierung von Merkmalen in Form einer Matrix. Das Erlernen von räumlichen Merkmalen wird damit gegenüber MLP mit eindimensionalen Eingabevektoren ermöglicht. In Abbildung 2.2 wird eine einfache CNN-Architektur dargestellt. Der Aufbau ist dabei in zwei Abschnitte unterteilt. Der erste Abschnitt dient dazu, um Features zu erlernen und der zweite, um aus diesem Feature-Vektor mögliche Klassen aus einer diskreten Menge zu prognostizieren. Die Extraktion von Features erfolgt dabei auf Grundlage einzelner CNN-Schichten, welche die Eingabe mittels Kernel-Filter bzw. Filterbänken durchlaufen. Kernel-Filter führen Faltungsoperationen durch, um Features zu identifizieren. Erzeugt werden Feature-Maps (zu dt. Merkmalsmatrix), welche die Eingabe für

die jeweilige Folgeschicht darstellen. Eine CNN-Schicht besteht dabei aus Filtern, welche durch ihre Aktivierungsfunktion, z.B. einer Relu-Funktion, ein Ausgabe-Signal erzeugen. Nach einer CNN-Schicht folgt eine Pooling-Schicht, mit deren Hilfe die Dimension der erkannten Features reduziert wird. Dazu kann zum Beispiel Maximal-Pooling, Minimal-Pooling oder Mittelwert-Pooling eingesetzt werden. Beim Übergang in den Abschnitt Klassifikation, werden die Feature-Maps in einen eindimensionalen Vektor transformiert und die Klassifikation mittels eines MLP vorgenommen. Dabei wird in der Regel in der Ausgabeschicht eine Softmax-Aktivierungsfunktion eingesetzt, um den allen Klassen eine Identifikations-Wahrscheinlichkeit zuzuordnen.

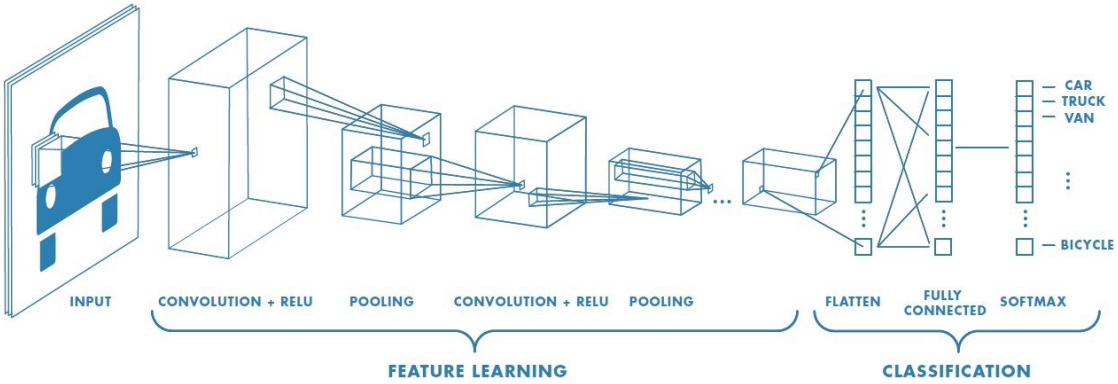

Abbildung 2.2: Architektur eines Convolutional Neural Network. Extraktion von Features über Faltungsschichten, welche anschließend in ein MLP zur Klassifizierung übergeben werden [20].

2.2.2 Segmentierungsnetze

Während bei Klassifizierungsnetzen, welche überwiegend aus CNNs bestehen, angestrebt wird der Eingabe-Matrix Klassen aus einer diskreten Menge zuzuordnen, werden bei Segmentierungsnetzen einzelne Pixel klassifiziert. Vorgenommen wird also eine Pixel-bei-Pixel Klassifikation. Der Vorteil besteht darin, dass den Klassen Bildsegmente zugewiesen werden. Bei diesem Typ von neuronalem Netz werden zwei Arten unterschieden: Semantische Segmentierung und Instanz-Segmentierung. Beide Typen werden nachfolgend erläutert, für diese Arbeit ist jedoch ausschließlich Instanz-Segmentierung von Bedeutung.

2.2.2.1 Semantische Segmentierung

Die semantische Segmentierung strebt die Trennung von Hintergrund- und Vordergrund-Pixeln an. Vernachlässigt wird dabei die Unterscheidung einzelner Objekte des selben Klassentyps. Long et al. [12] bewiesen durch ihre Forschungsarbeit, dass ihr Modell für semantische Segmentierung akkuratere Ergebnisse als z.B. das bekannte Klassifikationsnetz VGG-16, erzielt und gleichzeitig performanter ist. Sie entwickelten eine Encoder-Decoder-Architektur, welche als FCN benannt und im Unterschied zu CNNs ausschließlich aus Faltungsschichten besteht. Zu beachten ist, dass jede CNN-Architektur in eine FCN-Architektur überführt werden kann (s. Abbildung 2.3). Der Encoder-Abschnitt ist bei CNN und FCN identisch, es werden Features durch Faltungen extrahiert. Während bei CNNs zur Weiterverarbeitung die Feature-Maps in einem eindimensionalen Vektor überführt werden, werden bei FCN ausschließlich Faltungsoperationen durchgeführt. Dabei kommen 1x1-Filter in der Anzahl der Klassen zum Einsatz, welche Heatmaps bzw. Masken erzeugen und im Anschluss upgesampelt werden.

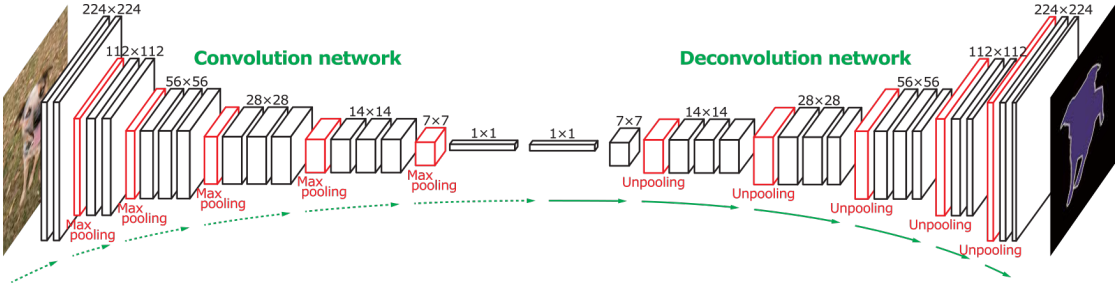

Abbildung 2.3: VGG-16 als FCN mit Encoder zur Feature Extraktion, anschließend 1x1 Filter zur Klassifizierung, welche anschließend mit Decoder upgesampelt wird. [14].

2.2.2.2 Instanz-Segmentierung

Bei der prinzipiell sehr ähnlichen Instanz-Segmentierung, wird im Unterschied zur semantischen Segmentierung, allen Objekten derselben Klasse eine eindeutige Instanz zugewiesen. Dadurch können einzelne Objekte ausfindig gemacht und möglichst pixel-genau extrahiert werden. Kombiniert wird dazu FCN mit Objekt-Detektion, welches ein Bestandteil des Computer-Vision darstellt. Während bei der Klassifikation dem Eingabebild eine Klasse aus einer diskreten Menge zugewiesen wird, strebt die Detektion zusätzlich an, Koordinaten für einzelne Instanzen zu bestimmen und damit eine Lokalisierung der

zugehörigen Objekte zu ermöglichen. Im Ergebnis liegen Höhe, Breite und Koordinaten des zentralen Punkt der Box vor, welche anschließend mit einem FCN weiterverarbeitet werden. Umgesetzt werden kann die Objekt-Detektion mit unterschiedlichen Konzepten, besonders hervorzuheben ist das RPN, welches nachfolgend beschrieben wird.

2.2.3 Region-Proposal-Network

He et al. [15] forschten daran, die Erkennung von Objekten performanter zu machen, da bekannte Lösungen eine vergleichsweise hohe Inferenzzeit in Anspruch nahmen. Dabei stellten sie das RPN vor, welches ein besonders kleines Modell darstellt und zur Kategorie der FCN angehört. Ziel war die Erkennung von "Regions of Interest", die anschließend mit anderen Verfahren weiter bearbeitet werden (z.B. zur Bestimmung von Koordinaten oder zur Klassifizierung). Auch wenn es ein eigenständiges Netz abbildet, wird es in größere Modelle eingebettet. Illustriert wird die Architektur in Abbildung 2.4. Erwartet wird als Eingabe eine Feature-Map mit 128-, oder 256-Dimensionen, welche anschließend mit einem $3x3x256$-Filter oder $3x3x512$-Filter durchlaufen wird. In dieser Phase werden pro Pixel k Anchor-Boxen generiert.

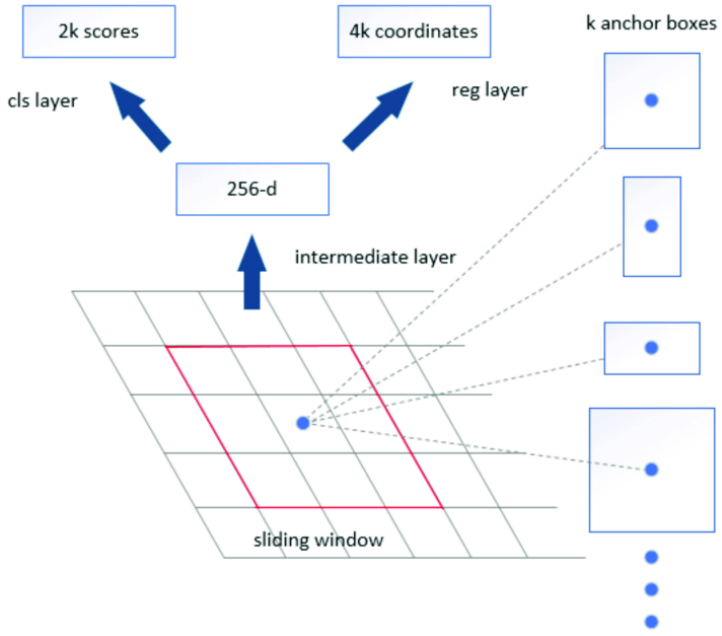

Abbildung 2.4: Architektur eines RPN [15]. Zu jedem Pixel werden Anchor-Boxen erzeugt. *Cls layer* evaluiert, ob ein Objekt existiert. *Reg layer* optimiert die Anchor-Boxen.

Bei ihrer Forschung entschieden sie sich dabei für 3 verschiedene Seitenverhältnisse und Skalierungen, wodurch $3x3 = 9$ Anchor-Boxen in Form von Rechtecken pro Pixel erzeugt werden. Anschließend verlaufen diese in zwei parallel liegenden Zweigen, ohne die Nutzung einer Pooling-Schicht, welche als *cls layer* und *reg layer* bezeichnet werden. Eingesetzt werden bei beiden $1x1$-Filter. Der *cls layer* führt eine binäre Klassifikation durch: jedem Pixel innerhalb einer Box wird dabei eine Wahrscheinlichkeit mittels Softmax-Aktivierungsfunktion zugewiesen, dass ein Objekt in der betreffenden Box liegt; es werden somit $2 * k$ Feature-Maps erzeugt. Gleiches gilt für den *reg layer*, dieser optimiert Breite, Höhe und den zentralen Koordinatenpunkt (x, y) der Anchor-Boxen und erzeugt $4 * k$ Feature-Maps.

2.2.4 Feature-Pyramid-Network

Bei CNNs werden die Features anhand der Bottom-Up-Methode extrahiert, was zur Folge hat, dass die Auflösung immer weiter verkleinert wird, weshalb räumliche Informationen verloren gehen. Dabei erlernen die ersten Faltungsschichten grobe und die tieferen Schichten feinere semantische Strukturen. Zur Erzielung besserer Resultate, stellten Lin et al. [10] mittels Feature-Pyramid-Network (FPN) allen Schichten dieselben hochwertigen Feature-Maps zur Verfügung. FPN lösen dieses Problem, indem sie einen zusätzlichen Top-Down-Layer einsetzen (s. Abbildung 2.5). Das Prinzip beruht darauf, dass Features-Maps kleinerer Auflösung mit größeren elementweise addiert werden, um damit räumliche Informationen beizubehalten. Bezogen auf das Beispiel in Abbildung 2.5 bedeutet dies für M4, dass es sich aus der Addition der Faltungsschicht C4 mit M5 ergibt. Jedoch muss M5 auf die Auflösung von M4 upgesampelt werden, dies gelingt durch Interpolation mittels Nearest-Neighbor. Dies wird konsequent vollzogen, bis auf C1, da dass Feature-Map zu groß ist. Es entstehen dabei Feature-Maps P2-P5, welche beispielsweise einem RPN als Eingabe übermittelt werden können.

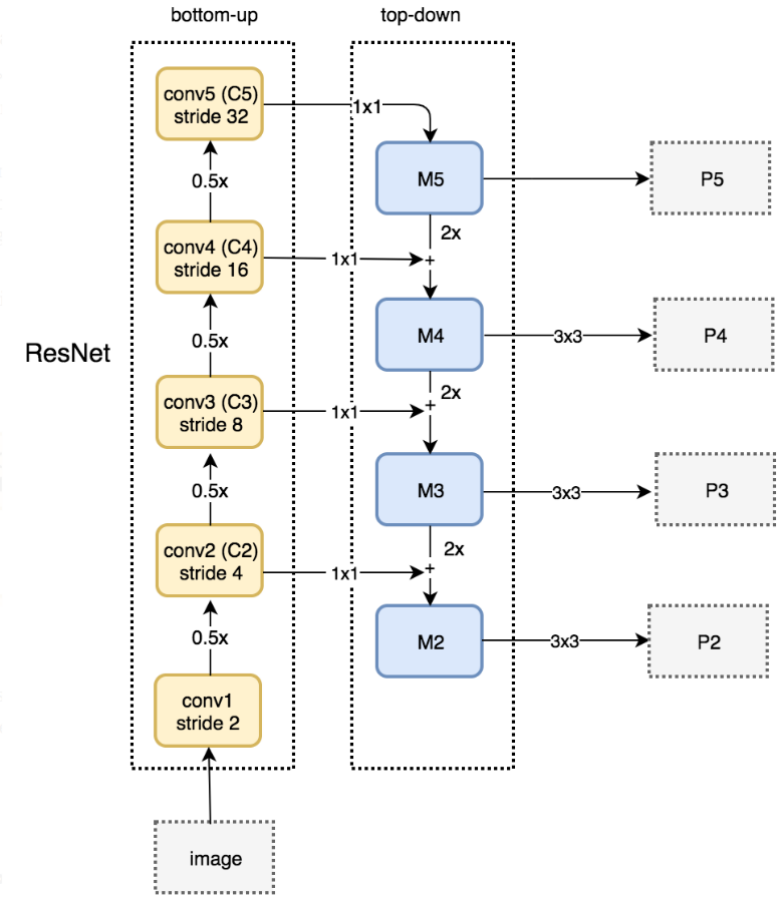

Abbildung 2.5: ResNet mit FPN, M2-M5 repräsentieren die Feature-Maps aus der jeweiligen Faltungsschicht (C2-C5) und P2-P5 sind die Ausgabe Feature-Maps. [15].

2.2.5 Fehlerfunktion

Die Fehlerfunktion ist ein Maß dafür, wie akkurat ein Modell im Training den erwarteten Wert vorhersagen kann. Dazu wird der Ausgabewert des Modells und der Ground-Truth-Wert (zu dt. Wahrheitswert) als Parameter für die Funktion verwendet, um damit den Verlust, also die Abweichung zueinander, zu bemessen. Nachfolgend werden gebräuchliche Fehlerfunktionen erläutert.

2.2.5.1 MQA und MAF

Die Mittlere-Quadratische-Abweichung (MQA) und der Mittlerer-Absoluter-Fehler (MAF) werden überwiegend in der Regressionsanalyse eingesetzt. Die Funktionsweise ist dabei wie folgt: es wird der Mittelwert über die Summe der Trainingsdaten N gebildet, indem die Differenz zwischen dem prognostiziertem Wert y_i und dem Ground-Truth-Wert $\hat{y}_i$ berechnet und quadriert wird.

$$MQA = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.1}$$

Der Wert bleibt wegen der Quadrierung immer positiv. Als Nachteil ist zu erwähnen, dass bei kleinen Fehlern bzw. Ausreißern der Verlust als groß gewichtet wird, wodurch das Modell nicht optimal auf die Daten trainiert werden kann.

Der MAF hat einen fast identischen Aufbau, mit dem Unterschied den absoluten Wert statt der Quadrierung zu verwenden, wodurch allen Abweichungen eine gleiche Gewichtung gegeben wird und zusätzlich der Wert weiterhin positiv bleibt.

$$MAF = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{2.2}$$

2.2.5.2 Huber-Kostenfunktion

Die Huber-Kostenfunktion vereinigt die Stärken von MQA und MAF, dazu wird ein Parameter δ definiert, welcher als eine Art Schranke dient. Liegt die Differenz unter δ, so wird MQA und ansonsten bei großen Abweichungen MFA verwendet.

$$L_\delta = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & Wenn\,|(y-\hat{y})| < \delta \\ \delta((y-\hat{y}) - \frac{1}{2}\delta) & andernfalls \end{cases} \tag{2.3}$$

2.2.5.3 Binäre-Kreuzentropie

Die binäre Kreuzentropie wird eingesetzt, wenn die diskrete Menge aus zwei Klassen besteht, so wie es beispielsweise bei RPN der Fall ist, bei welchem klassifiziert wird, ob es sich um ein Objekt handelt. Die Funktionsweise dieser Fehlerfunktion ist dabei wie folgt zu interpretieren. Der Wahrheitswert y wird als 0 (nein) oder 1 (ja) angegeben und in einer logarithmischen Funktion ist die Wahrscheinlichkeit des Modells als p definiert. Das besondere dabei, abhängig vom Wahrheitswert wird ein Teil der Gleichung aktiviert und der andere nicht mit berücksichtigt. Wenn beispielsweise $y = 0$ ist, so verschwindet der erste Teil $0 * log(p)$, während $(1 - 0) * log(1 - p)$ für die Berechnung bleibt.

$$BK = -(y \log(p) + (1 - y) \log(1 - p)) \tag{2.4}$$

2.2.6 Metriken

Metriken dienen dazu, ein Modell quantitativ zu evaluieren. Grundlage bildet dabei die sog. Konfusionsmatrix, die sich auf folgenden Komponenten zusammensetzt.

1. True Positive (TP): Wie oft werden richtige Pixel auch als richtig klassifiziert?
2. False Negativ (FN): Wie oft werden richtige Pixel als falsch klassifiziert?
3. False Positive (FP): Wie oft werden falsche Pixel als richtig klassifiziert?
4. True Negativ (TN): Wie oft werden falsche Pixel als falsch klassifiziert?

2.2.6.1 Precision

Die Precision quantifiziert die Anzahl richtig klassifizierter Pixel zu den tatsächlich richtigen Pixel, indem alle Klassifizierungen ins Verhältnis zueinander gesetzt werden. Dadurch kann die Relevanz der Prognose beurteilt werden.

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

2.2.6.2 Recall

Der Recall quantifiziert die Anzahl positiver Klassifizierungen zu allen positiven Pixeln, dabei werden FN mit in Betracht genommen. Somit gibt dies Aufschluss darauf, ob alle richtigen Pixel gefunden werden.

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

2.2.6.3 IoU

Die Intersection Over Union ist eine Metrik zur Evaluierung von Segmentierungen. Dabei werden die prognostizierten Pixel des Modells den Ground-Truth-Pixel gegenübergestellt. Das Ergebnis gibt Aufschluss darüber, in wie fern beide Werte übereinstimmen. Der IoU wird auch als ein Schwellenwert eingesetzt, um nicht relevante Prognosen herauszufiltern.

$$IoU = \frac{TP}{(TP + FP + FN)} \tag{2.7}$$

2.2.6.4 Mean-Average-Precision

Der Mean-Average-Precision (mAP) ist eine skalare Größe, welche die durchschnittliche Genauigkeit eines Modells angibt.

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \tag{2.8}$$

Als Grundlage wird die Average-Precision (AP) jeder möglichen Klasse k kalkuliert, indem die Precision und der Recall durch Multiplikation kombiniert werden, welche auch Precision-Recall-Kurve genannt wird. Dabei wird die klassifizierte Pixelmenge P iteriert und der AP berechnet.

$$AP = \sum_{j=0}^{z=P-1} [recall(j) - recall(j+1)] * precision(j) \tag{2.9}$$

2.3 Harris-Corner-Detector

Eine Ecke ist ein Punkt, dessen Kanten in zwei unterschiedlichen Richtungen zueinander stehen. Sie gehören zu den wichtigsten Features eines Bildes und sind gegenüber Translation, Rotation und Beleuchtung invariant. Harris et al. [3] entwickelten eine mathematische Herangehensweise, mit welcher Eckepunkte eines Bildes detektiert werden. Über jeden Pixel wird ein Fenster W bzw. eine Matrix der Größe mxm gelegt, welche in (u, v) Richtung bewegt wird, um dann die Summe der Abweichungsquadrate zu berechnen (SSD). Der Pixel, dessen SSD in allen Richtungen am größten ist, wird bei Überschreiten eines Schwellwerts als Eckpunkt definiert.

$$SSD(u,v) = \sum_{(x,y)\in W} \underset{Fenster}{w(x,y)} - [\ \underset{Intensitaet}{I(x,y)} - \underset{Verschobene-Intensitaet}{I(x+u,y+v)}\ ]^2 \tag{2.10}$$

3 Verwandte Arbeiten

Im Rahmen der Literaturrecherche wurden wissenschaftlíche Arbeiten analysiert, welche sich auf verwandte Themengebiete und Teilbereiche beziehen. Unterschieden werden grundsätzlich für rasterbasierte Grundrisspläne zwei Herangehensweisen, so werden regel-basierte oder lern-basierte Algorithmen eingesetzt, um geometrische und semantische Informationen zu extrahieren, wodurch aus diesem dann z.B. ein 3D-Modell generiert werden kann. In Abschnitt 3.1 werden zunächst regel-basierte und in Abschnitt 3.2 lern-basierte Algorithmen betrachtet, es wird die Funktionsweise untersucht und welche möglichen Einschränkungen in den Systemen vorhanden sind.

Die Grundlage dieser Arbeit sind rasterbasierte Grundrisspläne. Zur Vollständigkeit werden in Abschnitt 3.3 kurz digitale Grundrisspläne angeschnitten, welche im Gegensatz zu rasterbasierten Grundrissplänen bereits eine Menge von Informationen zur Verfügung stellen.

3.1 Regel-basierte Algorithmen

Yin et al. [22] untersuchten verschiedene Arbeiten zur Generierung von 3D-Modellen auf Basis von Grundrissplänen. In Abbildung 3.1 wird ein ideales System illustriert, auf welches nachfolgend näher eingegangen wird.

Es ist von Bedeutung, ob es sich bei der Eingabe um Rastergrafiken oder um digitale CAD-Dateien handelt. Bei ersteren sind komplexe Verarbeitungsschritte notwendig, welche in der Abbildung 3.1 als “image parsing” zusammengefasst werden. Dabei werden mithilfe von Morphologie-Filtern unbrauchbare Informationen wie z.B. Möbel und Textsegmente entfernt. Yin et al. erwähnen als Nachteil des verwendeten Filters, dass dickere Linien gegenüber dünneren Linien besser segmentiert werden. Dies macht es erforderlich, dass Wände in Grundrissplänen explizit als solche gekennzeichnet sein müssen.

Als nächstes wird eine Vektorisierung unter anderem mit Hough Transformation vorgenommen, wodurch geometrische Primitiven in Form von Polylinien und Bögen vorliegen. Aufbauend darauf werden semantische Informationen, wie die Erkennung von Wänden, Türen und Fenstern mit Unterstützung von Symbol-Algorithmen durchgeführt, bevor schließlich durch Extrudieren von polygonalen Flächen die Gebäudebestandteile eine dreidimensionale Form annehmen. Yin et al. nehmen an, dass eine vollständige Automatisierung ohne manuelle Eingriffe mit diesem System nicht möglich ist. Dabei liegen mögliche Fehlerquellen vor allem in der Bildverarbeitung (image parsing), welche kontinuierlich überwacht und gegebenenfalls nachträglich manuell korrigiert werden müssen. Die Ursachen für Fehler können dabei unterschiedlich sein, wie z.B. eine unzureichende Vektorisierung von Zeichnungen.

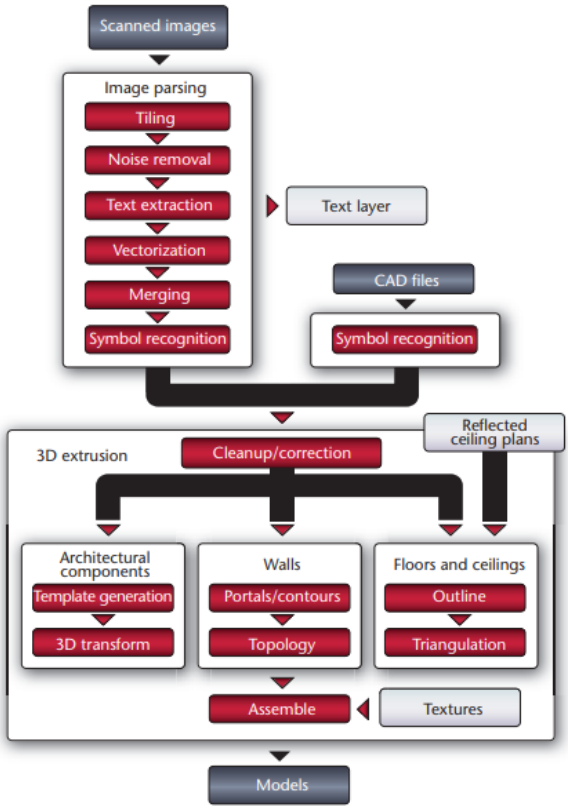

Abbildung 3.1: Vollständiger Aufbau des idealen Systems bis zur Erstellung des 3D-Models nach Yin et al. Grundrissplan wird im Block *Image parsing* von Störquellen befreit und vektorisiert. Im Block *3D extrusion* dienen Symbol-Algorithmen zur Erkennung von Wänden, Türen und Fenster. [22]

3.2 Lern-basierte Algorithmen

Dodge et al. [2] beschreiben ein System, bei welchem im ersten Schritt ein FCN eingesetzt wird, um Wandpixeln zu segmentieren. Anschließend wird die Detektion von Objekten wie Türen, Öfen, Badewannen und Schiebetüren mit einem Faster R-CNN vorgenommen. Als letztes wird Optical Character Recognition (OCR) zum Auslesen von Raumgrößen in Form von Textsegmenten verwendet. Sie evaluierten ihr System mit regel-basierten Algorithmen und konnten bessere Ergebnisse auf Grundlage desselben Datensatzes hervorbringen.

Liu et al. [11] verwenden in ihrer Arbeit eine DNN-Architektur auf Grundlage von ResNet-152, um Wandverbindungen (s. Abbildung 3.2) zu identifizieren.

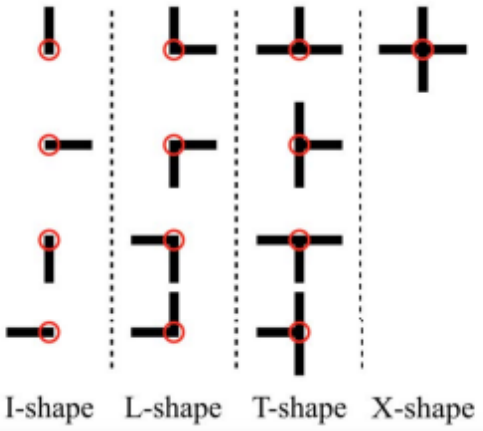

Abbildung 3.2: Liu et al. definieren grundsätzlich 4 Formen: I, L, T und X. Mit Einbeziehung der Ausrichtungen ergeben sich somit 13 (=4 + 4 + 4 +1) Typen von Wandverbindungen. [11]

In Abbildung 3.3 werden sämtliche Verarbeitungsschritte mit einem Beispiel dargestellt. So sind Wände in Grundrissen als dicke Linienzeichnungen vorausgesetzt. Nach Identifizierung von Wandverbindungen durch Deep-Learning, werden diese miteinander verbunden, um Wände zu lokalisieren. Schließlich wird als letzter Schritt auch vektorisiert. Mit der *Manhattan-World-Assumption* [2] beschränken Liu et al. ihr System auf die Verarbeitung von vertikal oder horizontal ausgerichteten Wänden. Daher kann es nur Grundrisse mit rechteckigen Räumen und gleichmäßig dicken Wänden erkennen.

[2] Annahme besagt, dass Räume und Stadtpläne auf Grundlage eines kartesischen Gitters gebaut wurden.

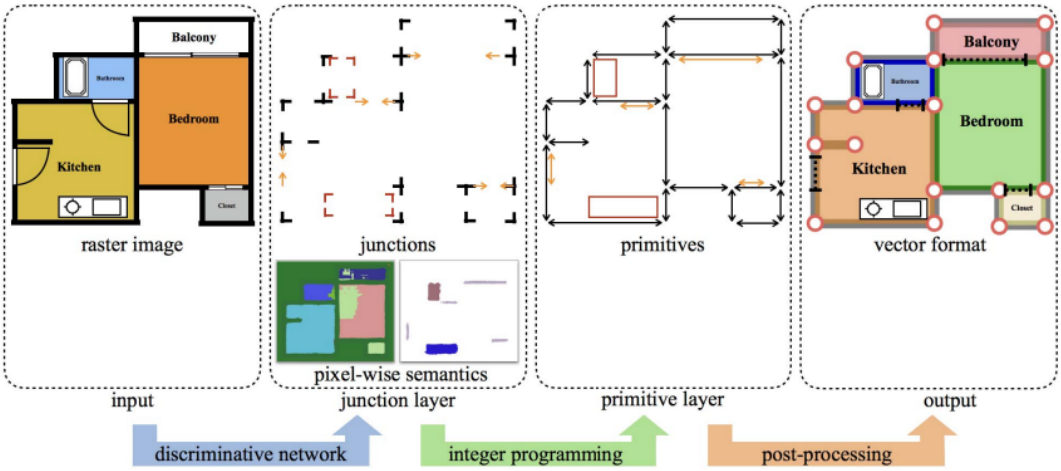

Abbildung 3.3: Verarbeitungsschritte von der Eingabe bis zur Erstellung eins Vektor-Formats. Grundrissplan wird mittels Deep-Learning (*discriminative network*) nach Wandverbindungen untersucht, welche anschließend miteinander verbunden werden (*integer programming*). Im *post processiong* werden die Wandverbindungen vektorisiert. [11]

Zeng et al. [23] untersuchten das Problem, ob Wände mit unterschiedlicher Dicke und Räume unterschiedlicher Formen in Grundrissplänen verarbeitet werden können. Dazu entwickelten sie ein Deep Multi-Task Network, welches Raumgrenzen (Wände, Türen und Fenster) und parallel auch den Raumtyp (Schlafzimmer, Küche, WC, etc.) klassifizieren kann. In Abbildung 3.4 wird der schematische Aufbau der Architektur dargestellt.

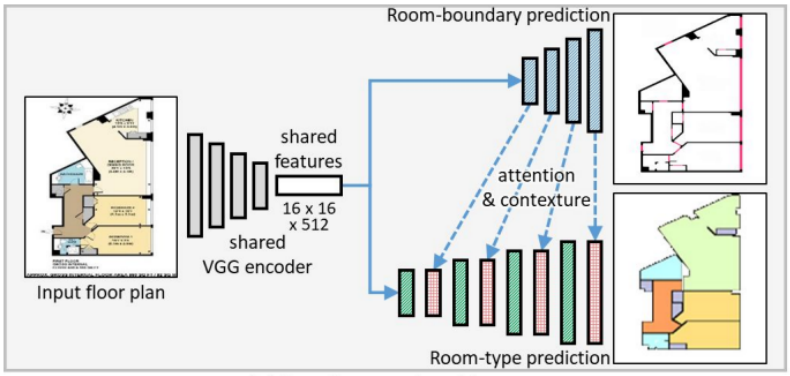

Abbildung 3.4: Das Multi-Task Network von Zeng et al., welches Raumgrenzen und Raumtyp ermitteln kann. VGG-Encoder als Feature-Extraktor und 2 VGG-Decoder für Raumgrenze und Raumtyp. [23]

Die Grundlage hierbei bilden Segmentierungsnetze in Form von *VGG Encoder-Decoder Architektur* [21]. Die Feature-Ausgabe des VGG-Encoders wird als Eingabe für den jeweiligen VGG-Decoder verwendet. Um die Erkennung des Raumtyps zu verbessern, werden die Features des Raumgrenzen-Decoders mit einbezogen. Zeng et al. weisen jedoch darauf hin, dass ihr System Schwierigkeiten hat, z.B. gebogene Korridore zu erkennen oder dass es größere Icons (wie z.B. einen Nordpfeil) als vermeintliche Wände klassifiziert.

3.3 Digitale Grundrisspläne

Digitale Grundrisspläne liegen als CAD-Dateien vor, dazu gehören *Data Exchange Format (DXF)*, *AutoCAD Drawing (DWG)* oder *building-information-modeling (BIM)* [22]. Es werden eine Reihe von Informationen von diesen Dateien zur Verfügung gestellt. So sind geometrische Informationen als Linien, Bögen und Punkte vorhanden und müssen nicht wie bei rasterbasierten Grundrissplänen mit komplexen Operationen erzeugt werden. Dadurch ist die Erkennung im Verhältnis trivial und wird deshalb im Rahmen dieser Arbeit nicht weiter betrachtet.

4 Implementierung

In diesem Kapitel wird die Implementierung beschrieben, welche eine mögliche Lösung für die Problemstellung darstellen soll. Beginnend mit der Funktionsweise des Systems mit anschließender Vertiefung auf die jeweiligen Komponenten, insbesondere der verwendeten Architektur für Instanz-Segmentierung. Abschließend wird ein Überblick über die Datengrundlage gegeben, unter anderem zählen dazu frei zugängliche als auch bereitgestellte Grundrisspläne der PLAN4 Software GmbH. Darüber hinaus soll der Trainingsaufbau erläutert werden.

4.1 Das System

Das System besteht hauptsächlich aus drei Komponenten. Der Vorverarbeitung, der Bildverarbeitung mit Mask R-CNN und einer abschließenden Nachbearbeitung. Dabei sind diese stark miteinander gekoppelt und beruhen auf gegenseitigen Ausgaben. Entwickelt wurde das System in *Python 3.7* und beruht auf eine freiverfügbare Umsetzung des Mask R-CNN in *Keras* [1]. Nachfolgend werden die Komponenten des Systems ausführlich erläutert.

4.1.1 Vorverarbeitung

Die Vorverarbeitung dient dazu, dass Eingabebild in eine für das neuronale Netz verarbeitbare Form zu transformieren. Dazu gehört in erster Linie die Auflösung auf die einheitliche Größe von $1024x1024$ zu skalieren und bei einem Graustufenbild eine Überführung in RGB durchzuführen, somit beinhaltet das Eingabebild 3.145.728 Pixel. In Abbildung 4.1 ist der komplette Prozess der Vorverarbeitung als Ablaufdiagramm dargestellt. Anschließend folgt ein Ausschnitt aus dem Quelltext, bei welchem das Skimage-Paket [1] zur Umsetzung des Ablaufs verwendet wurde.

[1] https://scikit-image.org/docs/stable/install.html - Zugrissdatum: 17.09.2022

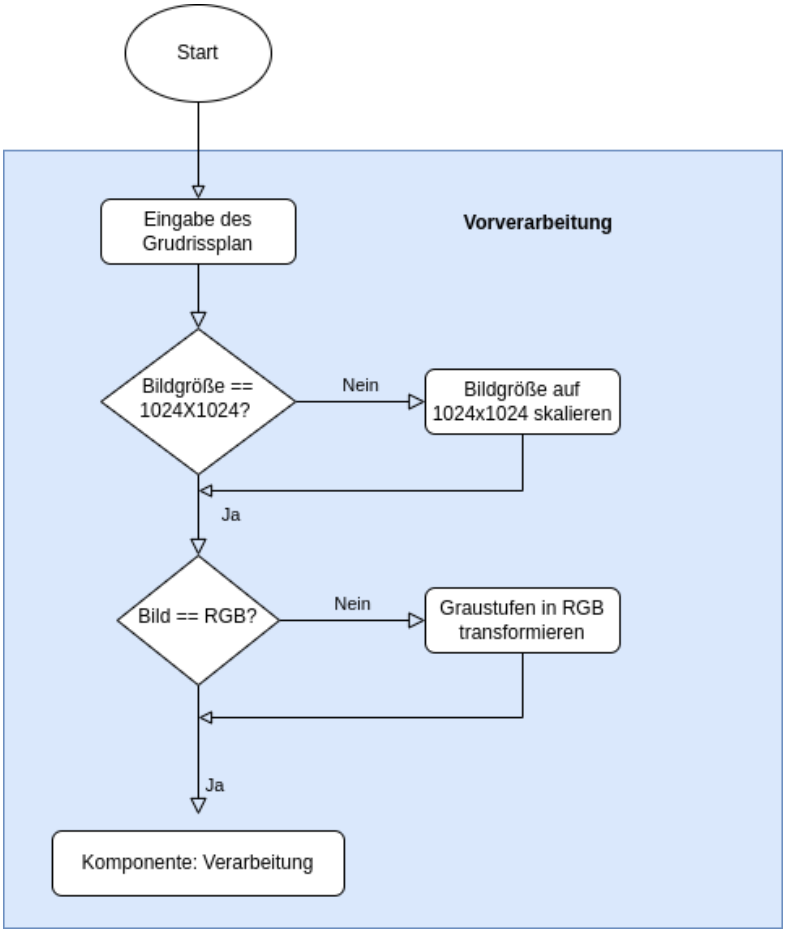

Abbildung 4.1: Die Vorverarbeitung dient zur Anspassung des Eingabebildes für die anschließende Verarbeitung.

Listing 4.1: Python Code für die Anpassung der Auflösung und Graufstufen nach RGB

```
# read image
image = skimage.io.imread(self.image_info[image_id]['path'])
h, w, d = image.shape
# resize image to 1024x1024 with bilinear interpolation
if h != 1024 or w != 1024:
    skimage.transform.resize(image, (1024, 1024))

# If grayscale. Convert to RGB for consistency.
if d != 3:
    image = skimage.color.gray2rgb(image)
```

4.1.2 Mask R-CNN

He et al. [4] forschten zum Thema Instanz-Segmentierung und stellten ihre Ergebnisse in Form eines Modells mit dem Namen Mask R-CNN vor. Das Mask R-CNN ist eine Erweiterung des weitläufig bekannten Objekt-Detektors Faster-RCNN [15] und erzielte im COCO-Wettbewerb eine mAP von 35%. Somit etablierte es sich zu einer State-of-the-Art Architektur. Die einzigen Modelle, welche noch bessere Ergebnisse erzielen, sind Erweiterungen des Mask R-CNN, wie z.B. das Mask Scoring R-CNN [7]. Nachfolgend wird die Architektur als auch Hyperparameter in ausführlicher Form beleuchtet.

4.1.2.1 Architektur

In Abbildung 4.2 wird die gesamte Architektur illustriert. Die Architektur besteht aus den Komponenten: Backbone, RPN, ROI Alignment, MLP und einem FCN.

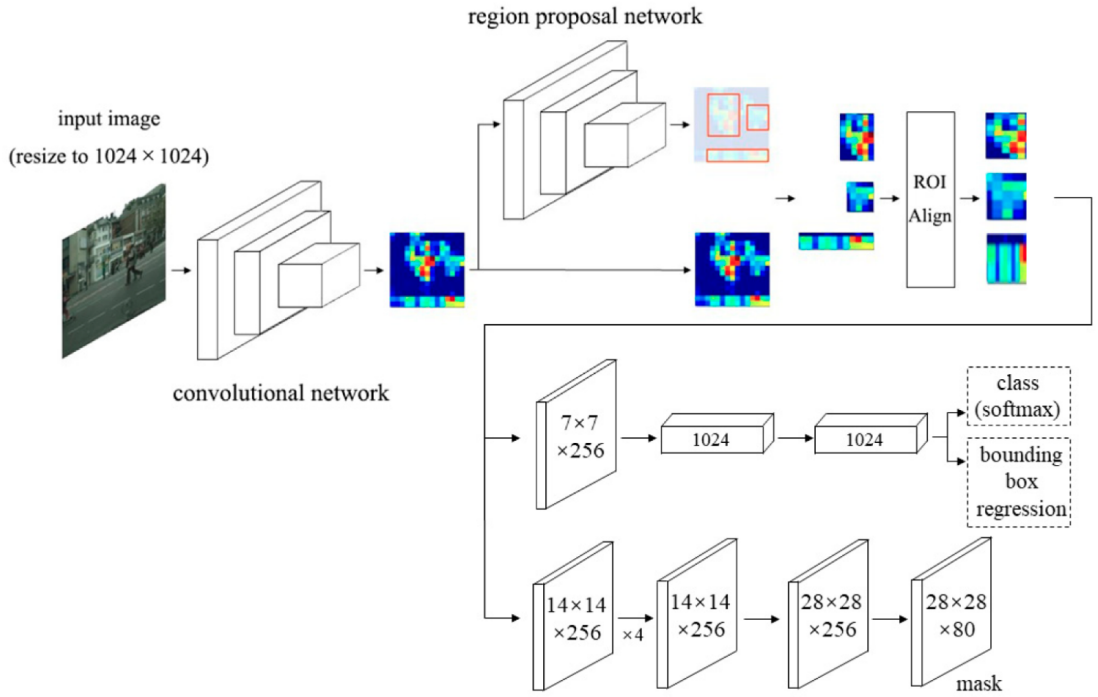

Abbildung 4.2: Die Architektur des Mask R-CNN [24]. Schritt-1: Feature Extraktion mittels Backbone mit FPN. Schritt-2: Erzeugen von ROIs mit einem RPN. Schritt-3: ROI Alignment. Schritt-4: Klassifizierung, Detektion und Segmentierung.

Das Backbone bildet eine CNN-Architektur, welche wahlweise auf das ImageNet,- oder COCO-Datensatz vor trainiert wurde. Dabei werden Faltungsschichten in das neue Modell transferiert und zur Extraktion von Features verwendet. Dies ist möglich, da erste Faltungsschichten grobe Muster wie Ecken, Farbverläufe und Kanten erlernen. Die Autoren entschieden sich bei Mask R-CNN das FPN (Abschnitt 2.2.4) in Kombination mit ResNet-50 oder ResNet-101 einzusetzen, für diese Arbeit wurde mit beiden experimentiert. Erzeugt werden $P2$, $P3$, $P4$, $P5$, $P6$ Feature-Maps in der Größe $16x16x256$, $32x32x256$, $64x64x256$, $128x128x256$, $256x256x256$, welches dem RPN (Abschnitt 2.2.3) als Eingabe dienen.

Das RPN erzeugt Anchor-Boxen auf jedes 256-Dimensionale Feature-Map. Zum Einsatz kommt dabei eine Faltungsschicht mit einem $3x3x512$-Filter, ReLU-Aktivierungsfunktion, Stride = 1 und Zero-Padding. Es werden für $P2$, $P3$, $P4$, $P5$, $P6$ Skalierungen von 32^2, 64^2, 128^2, 256^2, 512^2 und für jeden Feature-Map Seitenverhältnisse von 1:2, 1:1, 2:1 verwendet. Somit ergeben sich insgesamt 261.888 Anchor-Boxen für das Eingabebild.

$$Anchor-Boxen_{gesamt} = 16x16x3+32x32x3+64x64x3+128x128x3+256x256x3 \quad (4.1)$$

Anschließend durchlaufen die Anchor-Boxen zwei parallel liegende Zweige, der Objekt-Klassifizierung und Regression. Bei der Objekt-Klassifizierung wird ein 1x1x$(c*s)$-Filter, bei welchem $c = 2$ für die Klassen Vordergrund-, und Hintergrundpixel und $s = 3$ für die Anzahl der Seitenverhältnisse, verwendet. Es wird die Lineare-Aktivierungsfunktion eingesetzt. Anschließend wird die Ausgabe in eine Softmax-Schicht überführt, um den prozentualen Wert zu evaluieren. Die Regression verwendet ebenfalls einen 1x1x$(k*s)$-Filter, bei welchem $k = 4$ für Breite, Höhe, der zentrale Pixel (x,y) und $s = 3$ für die Anzahl der Seitenverhältnisse. Es wird die Lineare-Aktivierungsfunktion verwendet. Um die Anzahl der ROIs zu reduzieren, wird Non-Maximum-Suppression [6] eingesetzt, bei welchem am Ende die 200 ROIs mit der höchsten Punktzahl beibehalten werden.

Das ROI Alignment: Da ROIs in unterschiedlichen Formen vorliegen, wird eine einheitliche Größe für die nachfolgende Klassifizierung und Detektion ($7x7x256$) sowie der Segmentierung ($14x14x256$) angestrebt. Eingesetzt wird dazu die *RoI-Align-Schicht*, bei welcher in erster Instanz das optimale Feature-Map aus $P2-P5$, für jedes ROI berechnet wird. Als Indikatoren dienen dazu die Breite w und Höhe h des jeweiligen ROIs.

$$P_k = 4 + log_2(\sqrt{wh}/224) \quad (4.2)$$

Daraufhin wird das ROI aus der errechneten FPN-Schicht in den Zellen $7x7$ und $14x14$ aufgeteilt und mit Hilfe der Bilinearen Interpolation angepasst. Anschließend durchläuft jede Zelle eine Maximal-Pooling. Die erzeugten Feature-Maps dienen als Eingabe der in parallel liegenden Zweige: Klassifikation, Detektion und Segmentierung.

Die Klassifikation und Detektion bilden einen MLP ab, bei welchem das Eingabe Feature-Map in der Größe $7x7x256$ vorliegt. Anschließend werden mit $3x3x1024$-Filter Features extrahiert und der darauffolgenden Schicht $1x1x1024$-Filter in einen Vektor überführt. In dem Vektor liegen somit 1024 Features vor. Im Gegensatz zum RPN, wird nun die konkrete Klasse mittels einer Softmax-Aktivierungsfunktion evaluiert und die Klasse mit der höchsten Wahrscheinlichkeit als Ergebnis betrachtet. Die Regression dient dazu Höhe, Breite und zentralen Pixel (x,y) der Objekt-Detektion anzupassen, dabei existiert für jede Klasse eine Regression-Schicht. Die Lineare-Aktivierungsfunktion wird eingesetzt.

Die Segmentierung bildet einen FCN ab, bei welchem das ROI als Feature-Map der Größe $14x14x256$ vorliegt. Es folgen vier Faltungsschichten mit $3x3x256$-Filter, Zero-Padding und ReLU-Aktivierungsfunktion. Anschließend wird mit einem $2x2x256$-Filter, ReLU-Aktivierungsfunktion und Stride von 2 upgesampelt. Angestrebt wird die Erzeugung einer binären Maske für jeder Klasse mit einem $1x1xKlassen$-Filter und Sigmoid-Aktivierungsfunktion. Die binäre Maske hat die Größe von $28x28$. Abschließend wird die Maske anhand der Detektion skaliert, um dem Eingabebild zu entsprechen. He et al. [4] betrachten als Vorteil, dass im Vergleich zu herkömmlichen FCN, kein Wettbewerb zwischen den Klassen vorherrscht. Stattdessen wird das Ergebnis aus dem Zweig der Klassifikation als Indikator verwendet, um die korrekte Maske herauszufiltern.

Fehlerfunktion und Parameter: Trainiert wird das Modell über eine Ende-zu-Ende Beziehung und konstruiert sich aus den folgenden 5 Fehlerfunktionen. Das rpn_{cls}, welches das Objekt identifiziert. Das $faster_{cls}$, welches die tatsächliche Klasse evaluiert. Das $faster_{mask}$, welches die binäre Maske erzeugt. Diese spiegeln jeweils die Binäre-Kreuzentropie (Abschnitt 2.2.5.3) wieder. Das rpn_{loc} sowie $faster_{loc}$ sind Regressionen basierend auf der Huber-Kostenfunktion (Abschnitt 2.2.5.2).

$$L = rpn_{cls} + faster_{cls} + rpn_{loc} + faster_{loc} + faster_{mask} \tag{4.3}$$

Das Mask R-CNN mit ResNet-101 beinhaltet insgesamt **63.733.406** Parameter, davon werden **21.069.086** trainiert und **42.664.320** nicht.

4.1.2.2 Hyperparameter

Hyperparameter stellen Parameter da, dessen Werte den Lernprozess beeinflussen. Das Mask R-CNN beinhaltet unterschiedliche Parameter, nachfolgend werden einige aufgelistet. Unterstrichene Parameter wurden im Rahmen dieser Arbeit im *Kapitel 5 Experimente* untersucht. Darüber hinaus dient der Stochhastic Gradient Descent (SGD) [16] als Optimierung der Kostenfunktion.

Backbone: Feature-Extraktion des Eingabebildes mit ResNet-101 oder ResNet-50.

Skalierungen: Im RPN können Anchor-Boxen unterschiedlich skaliert werden. Standard sind 32^2, 64^2, 128^2, 256^2, 512^2.

Momentum: Der Momentum fügt einen Bruchteil der vorherigen Gewichtsaktualisierung zur aktuellen hinzu. Standard ist 0.9.

Learning-Rate: Learning-Rate gibt die Schrittgröße an. Standard ist 0.001.

Epoche: Anzahl der Durchgänge des gesamten Trainings-Datensatz. Standard sind 30 Epochen.

Seiteverhältnisse: Definition der Seitenverhältnisse im RPN. Es wurden die Standardwerte 1:2, 1:1, 2:1 beibehalten.

Weight Decay: Aktualisierung des Gewichts wird geschwächt. Standardwert von 0.0001 wurde beibehalten.

Batch-Size: Anzahl der Bilder, welche in einem Trainingsschritt verarbeitet werden. Abhängig von der Hardware, ist dieser in der Arbeit mit 3 definiert wurden.

Trainingsschritte: Anzahl der Schritte jeder Epoche beläuft sich auf 1000.

Validierungsschritte: Anzahl der Schritte jeder Epoche beläuft sich auf 200.

NMS-Zuversicht: Prozentualer Wert, ab welchem eine Anchor-Box als richtig gilt. Standardwert ist mit 70% definiert

Postive-Anchor: Anzahl der Anchor-Boxen, welche anschließend klassifiziert, detektiert und segmentiert werden. Standardwert mit 200.

Gewichtsinitalierung: Die Gewichte der Filter wurden mit Glorot-Uniform initialisiert.

4.1.3 Nachbearbeitung

In Abbildung 4.3 ist der komplette Prozess der Nachbearbeitung als Ablaufdiagramm dargestellt. Die Nachbearbeitung dient dazu, aus den generierten Masken ($28x28$) des Mask R-CNN, sämtliche Eckpunkte anhand des Harris-Corner-Detector (Abschnitt 2.3) zu lokalisieren. Anschließend werden lokalisierte Eckpunkte optimiert. Abschließend werden Koordinaten der Eckpunkte von Räumen in Form eines JSON-Format (siehe Listing 4.2) abgespeichert.

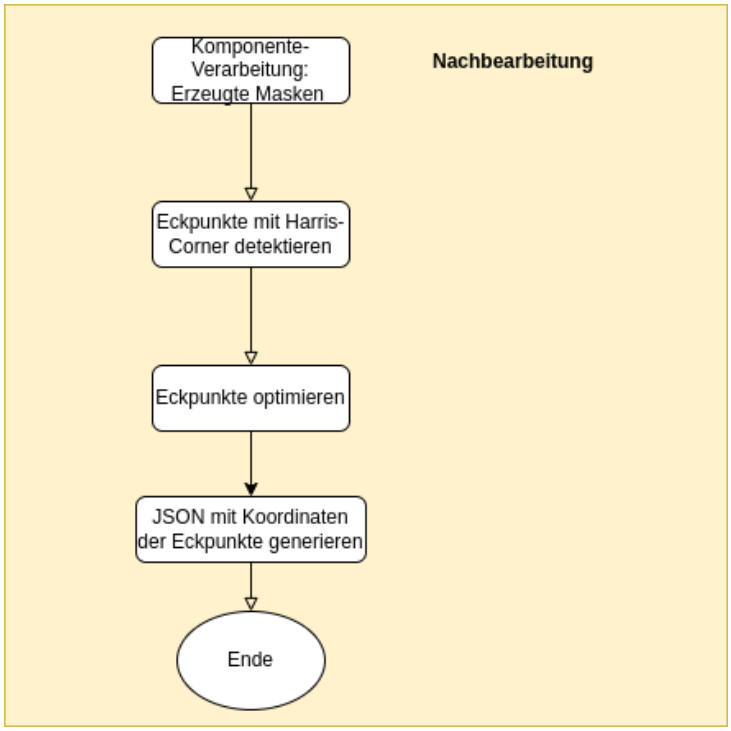

Abbildung 4.3: Die Nachbearbeitung lokalisiert Eckpunkte, optimiert und speichert sie in ein JSON-Format ab.

Listing 4.2: JSON-Sturktur für Raumkoordinaten

```
[ {
"room_0..N": [{
    "corner_0": "(x,y)"
    ...
    "corner_N": "(x,y)"
    }],
}]
```

Für jeden maskierten Raum wird eine Instanz in der Größe des Eingabebildes erzeugt (s. Abbildung 4.4). Dabei sind es Segmentierungen, welche keine einheitlichen Konturen aufweisen.

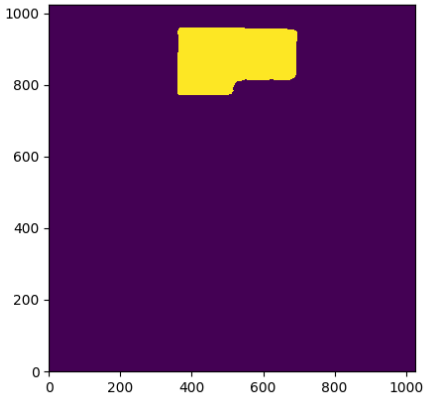

Abbildung 4.4: Beispiel eines maskierten Raums mit 6 Eckpunkten. Die Konturen sind ungleichmäßig.

Das OpenCV-Paket [2] beinhaltet eine implementierte Fassung des Harris-Corner-Detector. Dabei wird ein Graustufenbild als Eingabe vorausgesetzt. Als Parameter für die Funktion wird die Block-Größe, welches die Nachbarschaft für die Detektion eines Eckpunkt definiert, vorausgesetzt. In Listing 4.3 ist der entsprechende Auszug aus dem Quelltext abgebildet. Als Block-Größe sind 15 Pixel angegeben. Anschließend werden Eckpunkte anhand eines dynamischen Schwellwerts gefiltert. Der Schwellwert setzt sich aus der Multiplikation vom größten detektierten Wert mit $0,1$ zusammen.

Listing 4.3: Python Code für die Lokalisierung von Eckpunkten mit Harris-Corner-Detector

```
if show_corners:
    import cv2
    dst = cv2.cornerHarris(src=mask, blocksize=15)
    ret, dst = cv2.threshold(src=dst, thresh=0.1 * dst.max(), maxval=255)
    dst = np.uint8(dst)
```

[2] https://docs.opencv.org/3.4/dc/d0d/tutorial$_{p}y_{f}eatures_{h}arris.html, Zugriffsdatum$: 25.09.2022

Die in Abbildung 4.5 dargestellten roten Punkte repräsentieren die detektierten Eckpunkte bei einer Maske. Dabei werden korrekt 6 Punkte gefunden, welche jedoch nicht parallel gegenüber ihrer Nachbar-Eckpunkte liegen.

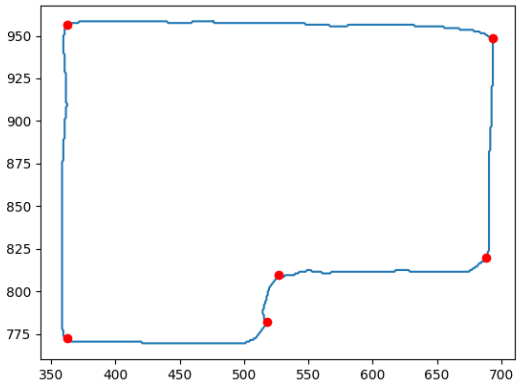

Abbildung 4.5: Beispiel einer Detektion. Die roten Punkte spiegeln Ecken wieder, welche durch den Harris-Corner-Detector detektiert wurden sind.

Um der Problematik entgegenzuwirken, wird ein Algorithmus über sämtliche Eckpunkte iteriert, um eine Angleichung der Koordinaten vorzunehmen. Ausgehend von jedem Eckpunkt, werden Nachbar-Eckpunkte berücksichtigt, welche eine Abweichung von höchstens 10-Pixeln in horizontaler oder vertikaler Richtung aufweisen. Anschließend wird der Durchschnitt über alle Treffer ermittelt und als neuer Koordinatenwert erachtet. In Listing 4.4 wird der Quelltext des beschriebenen Algorithmus abgebildet.

Listing 4.4: Python Code für die Angleichuung von Eckpunkten

```
tolerable = 10
for c in corners:
    x_axis_matching = [c[0]]
    y_axis_matching = [c[1]]

    for cc in corners:
        if (c == cc).all():
            continue
        x_axis = c[0] - cc[0]
        y_axis = c[1] - cc[1]

        if -abs(tolerable) <= x_axis <= tolerable:
            x_axis_matching.append(cc[0])
        if -abs(tolerable) <= y_axis <= tolerable:
            y_axis_matching.append(cc[1])
    x_round_mean = np.round(np.mean(x_axis_matching))
    y_round_mean = np.round(np.mean(y_axis_matching))
    corners_clean = np.append(corners_clean, [x_round_mean, y_round_mean])
```

Eine Angleichung mit dem Algorithmus, wird in Abbildung 4.6 illustriert. Zu beobachten ist, dass sämtliche Eckpunkte möglichst parallel gegenüber der Nachbar-Eckpunkte liegen.

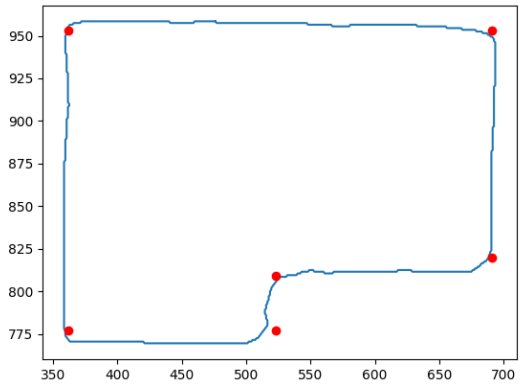

Abbildung 4.6: Beispiel mit 6 angeglichenen Eckpunkten. Eckpunkte liegen überwiegend parallel zueinander.

4.2 Datengrundlage

Die Datengrundlage bildet den essentiellen Rahmen für das Training ab. Dabei ist es zwingend erforderlich ein breite Menge an Daten aufzuweisen, um somit das Mask R-CNN effizient trainieren zu können. Ziel ist es das Phänomen des Overfittings zu vermeiden, bei welchem das Modell optimale Ergebnisse für die Trainingsdaten evaluieren, jedoch große Schwankungen gegenüber unbekannten Daten aufweist. Im nachfolgenden wird der Umfang an Daten für diese Arbeit behandelt.

4.2.1 Klassendefinition

Basierend der Ausgangslage, ist es nicht von Bedeutung spezifische Räumlichkeiten nach ihrem Typen wie beispielsweise Küche, Bad, Wohnzimmer oder Schlafzimmer zu klassifizieren, sondern den Raum als ganzes zu erfassen. Damit entsprochen existieren lediglich die Klassen **Raum** und **Hintergrundpixel**. Die Klasse Hintergrundpixel dient dazu Störquellen zu erfassen, um somit die Unterscheidung zum Raum zu ermöglichen.

4.2.2 Vorhandene Datensätze

Wie im Kapitel *Verwandte Arbeiten (Kapitel 3)* aufgezeigt wurde, existieren durchaus Ansätze, um rasterbasierte Grundrisspläne maschinell zu verarbeiten. Entsprechend existieren freizugängliche Datensätze, welche bereits annotiert sind. Für diese Arbeit wurde insbesondere der CVC-FP Datensatz [5] und CubiCasa5k Datensatz [8] verwendet. Dabei handelt es sich um etablierte Daten, welche in der Forschung als Metrik dienen.

4.2.2.1 CVC-FP Datensatz

Der CVC-FP Datensatz besteht aus 122 annontierten Grundrissplänen, basierend auf französischem Architektur-Stil. Dabei weisen diese Grundrisspläne eine überwiegende Übereinstimmung im Bezug auf Symbolen wie Türen, Fenster und Wandsegmenten gegenüber PLAN4-Daten wieder. Eine Unterscheidung besteht lediglich in der Beschriftung, welche in französisch erfolgt ist. Repräsentativ wird in Abbildung 4.7 ein Grundrissplan aus dieser Datensammlung dargestellt. Es sind insgesamt 11 Räume abgebildet. Als Gemeinsamkeit beinhalten sie jeweils die Raumgröße und den Raumtyp. Darüber hinaus

sollen Objekte wie markante Möbel oder das Fahrzeug den Raumtypen besonders untermalen.

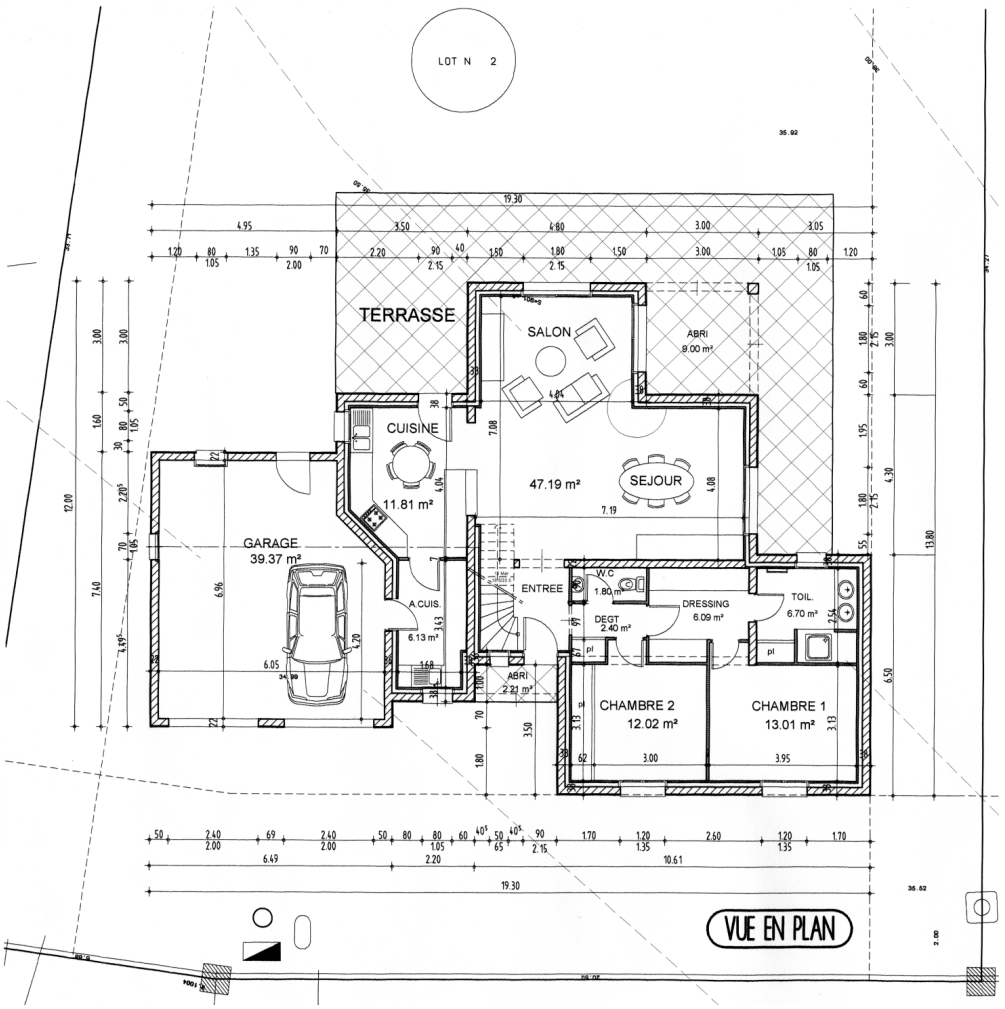

Abbildung 4.7: Grundrissplan aus dem CVC-FP Datensatz. 11-Räume mit detailreichen Informationen, wie Raumtypen, Raumgröße und Möbel. Name: CVC-5

Daneben gibt es weitere Informationen, wie Hilfslinien mit dazugehörigen Abmessungen, welche im inneren als auch äußeren eines Raums eingezeichnet sind. Relevant für das Erkennen von Räumen sind in erster Linie die Wandsegmente ausschlaggebend, diese bilden das umrahmende Gerüst. Die Wände sind als schraffierte dicke Linien eingezeichnet und heben sich dadurch von anderen Merkmalen ab. Um einen Raum als solches zu klassifizieren, gehören ebenso Türen als auch Fenster dazu. Zu bemerken ist, dass Räume unterschiedliche Formen und entsprechend unterschiedlich viele Eckpunkte aufweisen. In

Abbildung 4.8 werden die wichtigen Merkmale annotiert dargestellt, dabei ist zu deuten, dass überwiegende Semantiken für das Erkennen von Räumen von wenig Bedeutung sind.

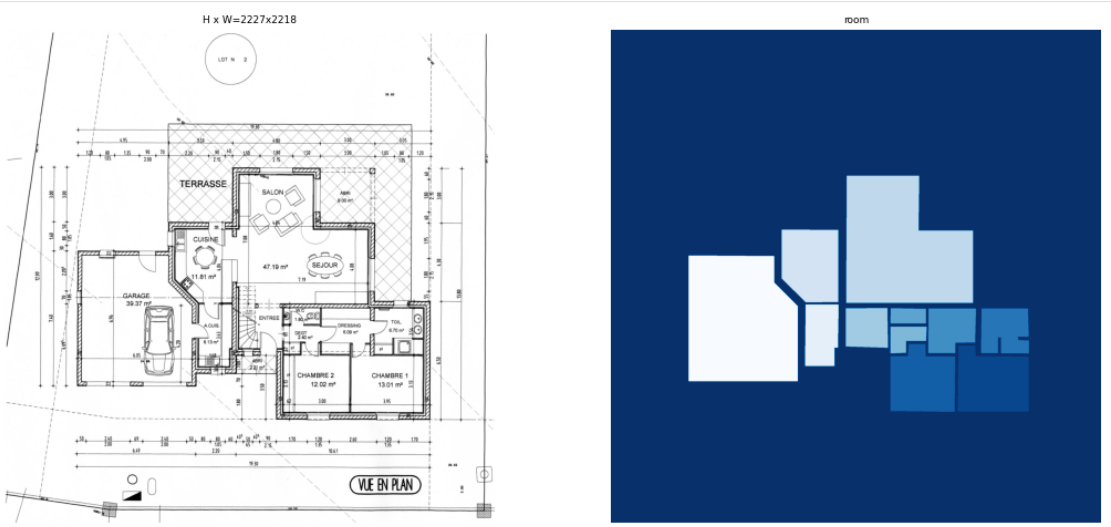

Abbildung 4.8: Grundrissplan mit Annotation aus dem CVC-FP Datensatz. Instanzen von Räumen werden durch unterschiedliche Farben repräsentiert. Name: CVC-5

4.2.2.2 CubiCasa5k Datensatz

Der CubiCasa5k Datensatz stellt mit seinen 5000 annotierten Grundrissplänen den größten Bestand für diese Arbeit da. Basierend auf finnischer Architektur, bietet es genauso wie der CVC-FP Datensatz, eine überwiegende Übereinstimmung mit dem PLAN4-Datensatz. Jeder Grundrissplan wurde dabei in einem mehrstufigen Verfahren händisch annontiert. Die Autoren unterteilen diese in folgende 3 Kategorien ein.

1. sehr hochwertige Grundrisspläne (3732)
2. hochwertige Grundrisspläne (992)
3. kolorierte Grundrisspläne (276)

Nachfolgend wird in Abbildung 4.9 ein sehr hochwertiger Grundrissplan gezeigt. Besonders hervorzuheben sind die eingezeichneten Details in sämtlichen Räumen. Dazu gehö-

ren besonders herausgearbeitete Möbel, aber auch Abmessungen innerhalb eines Raums. Umrahmt werden Räume auch hier mit dicken schraffierten Linien.

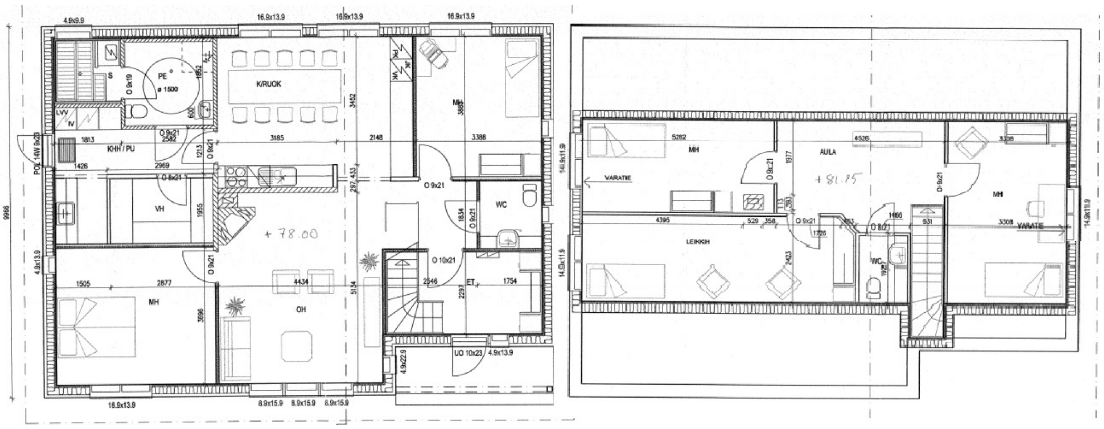

Abbildung 4.9: Sehr hochwertiger Grundrissplan aus dem CubiCasa5k Datensatz. Hochwertigkeit besticht durch die umfangreiche Vielzahl von Detail-Informationen wie Möbel, Textsegmente als auch Hilfslinien. Name: CC5K-4752

Ein überschaubarer Grundrissplan (s. Abbildung 4.10) verzichtet auf viele semantische Merkmale. Überwiegend besteht ein solcher Plan aus Wandsegmenten, Türen und Fenster. Komplett rausgelassen werden Möbel, Hilfslinienen, Legenden und äußere Ansichten wie Straßen oder Gärten. Ein solcher Plan dient dazu, dem Leser einen einfachen und schnellen Überblick der Räumlichkeiten zu verschaffen.

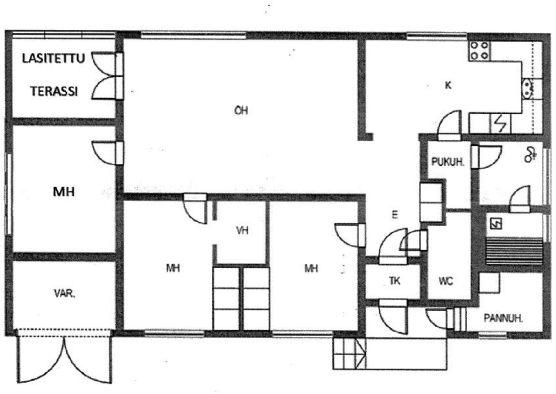

Abbildung 4.10: Hochwertiger Grundrissplan aus dem CubiCasa5k Datensatz. Plan beinhaltet wenige Details, wodurch es überschaubarer wird. Name: CC5K-631

Da es sich bei rasterbasierten Grundrissplänen größtenteils um eingescannte Pläne handelt, können sie auch verschiedene weitere Elemente und Farbgebungen enthalten. So ist in Abbildung 4.11 beispielsweise ein Grundrissplan zu sehen, welcher händisch eingezeichnete blaue Linien aufweist. Der fehlende Kontext macht es schwieriger zu urteilen, welchen Zweck diese haben. Des weiteren können Räume ebenfalls mit einem Farbton versehen sein, um den Raumtypen besonders auszuzeichnen. Ein weitere Herausforderung für das Modell, stellt die Unebenheit des Grundrissplan da, welches der mangelhaften Digitalisierung zu zuordnen ist.

Abbildung 4.11: Kolorierter Grundrissplan aus dem CubiCasa5k Datensatz. Farbgebungen untermalen den Raum. Eingezeichnete Linien liegen in blauer Farbe vor. Name: CC5K-2002

4.2.3 PLAN4 Datensatz

Das Unternehmen PLAN4 Software GmbH stellte für diese Arbeit insgesamt 786 Grundrisspläne aus ihrer Sammlung zur Verfügung. Da das Annotieren der Grundrisspläne besonders zeitintensiv ist, wurden selektiv Pläne nach Kriterien, wie der Raum-Komplexität und der Anzahl an möglichen Störquellen, ausgewählt. Angestrebt wird zusätzlich eine Ergänzung der frei zugänglichen Datensätze. Somit beinhaltet der PLAN4 Datensatz insgesamt 35 annontierte Grundrisspläne. In Abbildung 4.12 (s. Abbildung 4.13 mitsamt der Annotation) wird ein Grundrissplan aus dieser Sammlung dargestellt. Zu beobachten ist,

dass die Schnittmenge an Merkmalen gegenüber den frei zugänglichen Datensätzen eine große Gemeinsamkeit aufweist. So sind äußere Wandsegmente mit schraffierten Linien und innere als schwarze dicke Linien abgebildet. Türen werden mit einem Öffnungsschlag und Fenster durch Lücken in den Wänden repräsentiert. Darüber hinaus beinhalten Räume Textsegmente, welche in Kombination mit Möbeln den Raumtypen spezifizieren, als auch die Raumgröße angegeben. Die Besonderheit bei diesem Plan besticht jedoch durch die unterschiedlichen Farbgebungen. So werden einigen wenigen Räumen Farbtöne zur Untermalung zugewiesen. Hilfslinien, welche überwiegend an den äußeren Wänden eingezeichnet waren, werden beispielsweise durch rote oder gelbe Linien durch die Räume gezogen. Die Bedeutung dieser Linien ist nicht spezifiziert. Außerdem befinden sich händisch eingezeichnet Elemente in dem Plan, welche das Erkennen zusätzlich erschweren.

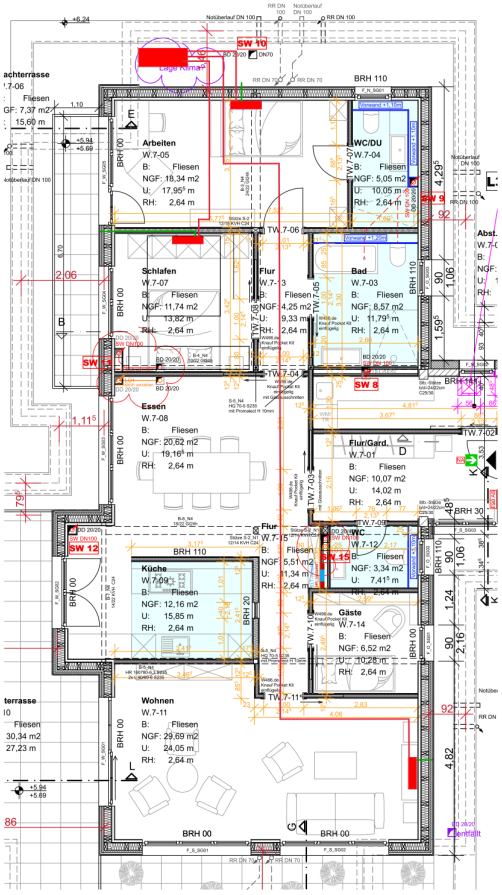

Abbildung 4.12: Besonders komplexer Grundrissplan aus dem PLAN4 Datensatz. Komplexität besticht neben der Vielzahl an Details, auch durch Hilfslinien, welche in unterschiedlichen Stärken und Farben eingezeichnet sind. Name: PLAN-226

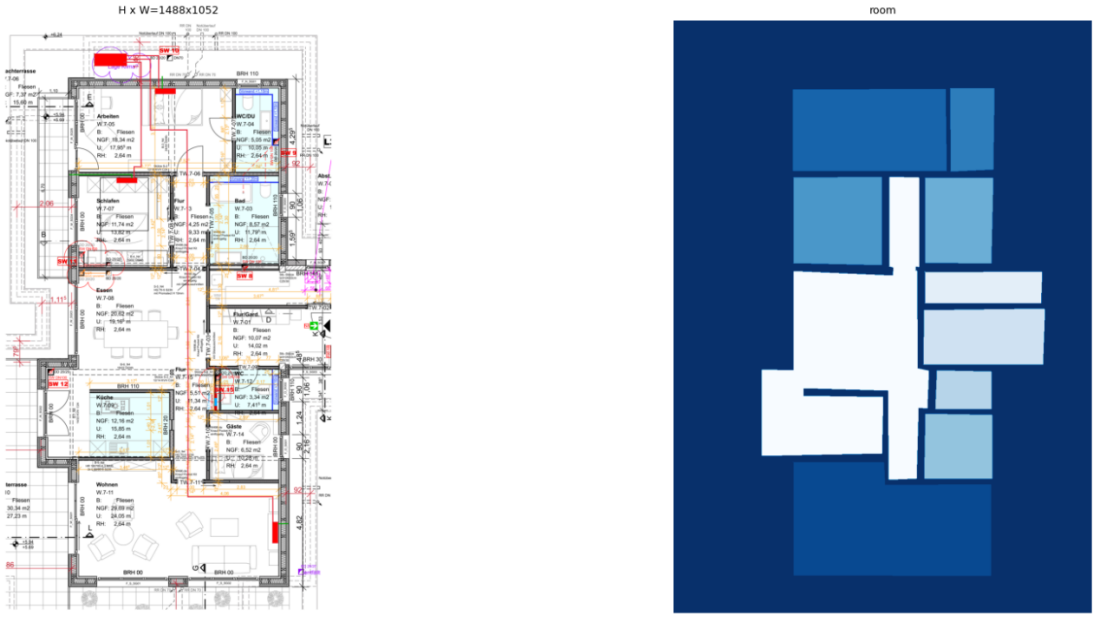

Abbildung 4.13: Besonders komplexer Grundrissplan mit Annotation aus dem PLAN4 Datensatz. Instanzen von Räumen werden durch unterschiedliche Farben repräsentiert. Name: PLAN-226

4.2.4 Data Augmentation

Für diese Arbeit wurde für das Training des Modells eine Augmentation mit Hilfe des Imaug-Paket[3] definiert, welches nach einem zufallsbasierten Prinzip agiert. Dabei werden sämtliche Grundrisspläne vor Beginn einer Epoche dynamisch augmentiert. In Listing 4.5 ist ein Ausschnitt aus dem Quellcode abgebildet. So wird für jeden Grundrissplan mit einer Wahrscheinlichkeit von 83% entschieden, ob vor der Verarbeitung eine Modifizierung vorgenommen werden soll oder nicht. Fällt die Zufallsentscheidung auf eine Modifizierung, so wird mittels mathematischer Operationen der Grundrissplan horizontal oder vertikal geflippt. Darüber hinaus kann ebenso eine Affine-Transformation vorgenommen werden, bei welchem der Grundrissplan unterschiedlich rotiert oder skaliert wird. Parallel dazu werden die Annotationen bzw. Ground-Truth Werte mit angepasst. Ein Auszug von Augmentation auf Grundrissplänen, wird in Abbildung 4.14 abgebildet.

[3] https://imgaug.readthedocs.io/en/latest/source/api$_i mgaug.html - Zugriffsdatum$: 15.09.2022

Listing 4.5: Dynamsiche Augmentation des Trainings-Datensatz

```
import imgaug
augmentation = imgaug.augmenters.Sometimes(5 / 6, imgaug.augmenters.OneOf(
    [
        imgaug.augmenters.Fliplr(1),
        imgaug.augmenters.Flipud(1),
        imgaug.augmenters.Affine(rotate=(-45, 45)),
        imgaug.augmenters.Affine(rotate=(-90, 90)),
        imgaug.augmenters.Affine(scale=(0.5, 1.5))
    ])

# Data generators
train_generator = data_generator(train_dataset, self.config, shuffle=True,
                                 augmentation=augmentation,
                                 batch_size=self.config.BATCH_SIZE,
                                )
val_generator = data_generator(val_dataset, self.config, shuffle=True,
                               batch_size=self.config.BATCH_SIZE)
```

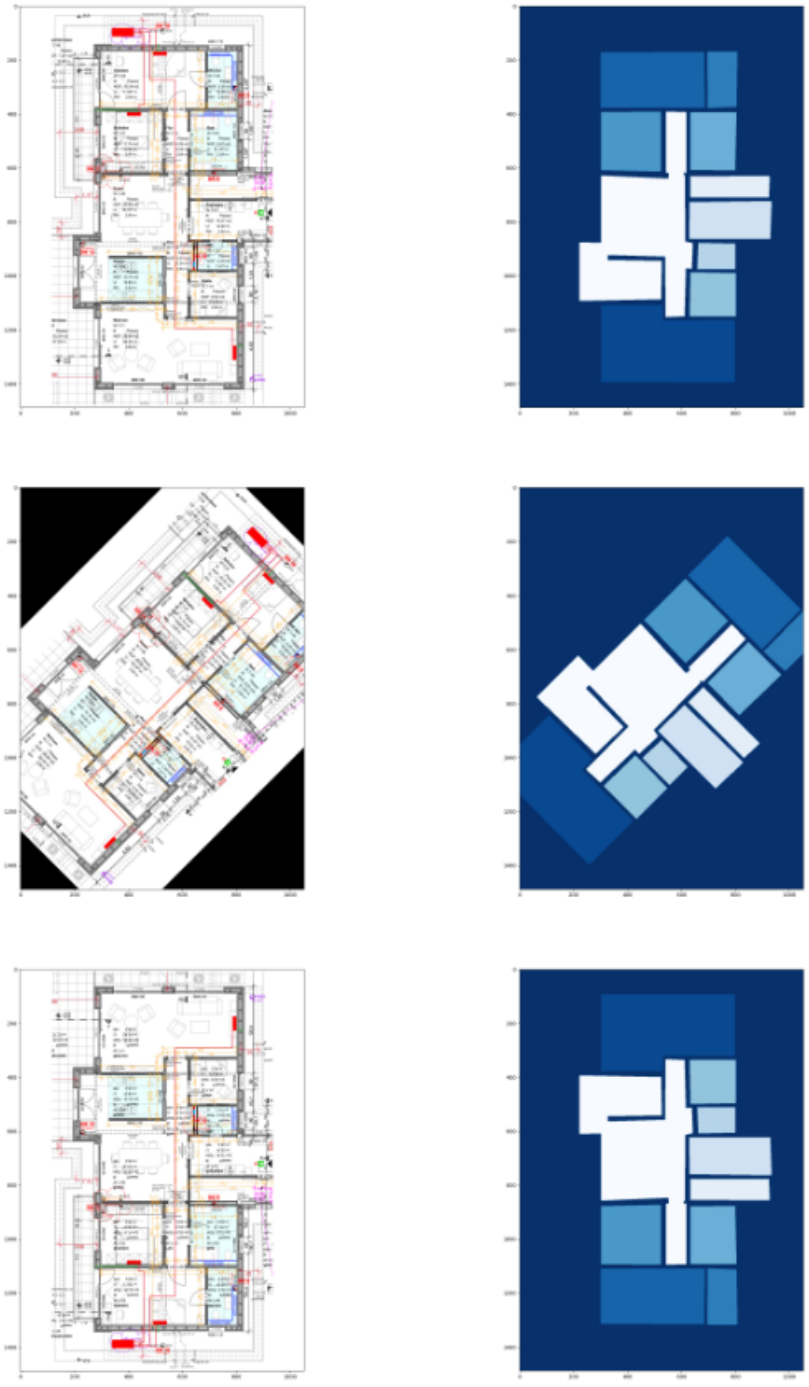

Abbildung 4.14: Data Augmentation eines Grundrissplan. Links: Grundrissplan, Rechts: Annotationen. Oben befindet sich der Grundrissplan ohne Augmentation. Mitte augmentiert um 45°. Unten vertikal geflippt. Name: PLAN-226

4.2.5 Trainingsvorbereitung

In dieser Sektion werden für das Training relevante Informationen zusammengefasst. Dazu gehört insbesondere die Aufteilung der Datensätze in Trainings,- Validierungs,- und Testdaten, als auch die verwendete Hardware für die Rechenleistung.

4.2.5.1 Aufteilung der Daten

Im Kontext des Überwachten Lernens, werden die Daten in Trainings-, Validierungs- und Testdatensätze aufgeteilt. Dabei tragen Trainingsdaten dazu bei, die Gewichte eines Modells in Hinsicht der Prognose zu optimieren. Im Gegensatz dazu, bilden Validierungsdaten eine Kontrolleinheit, welche nach jeder Epoche durchlauft werden. Dabei wird unter anderem festgestellt, ob sich das Modell dem Overfitting nähert, bei welchem es deutlich schlechtere Ergebnisse gegenüber Validierungsdaten aufweist. Darüber hinaus kann auch erkannt werden, ob ein frühes Abbrechen des Trainings sinnvoll wäre. Testdaten werden nach dem durchlaufen des kompletten Trainings verwendet, um Aussagen gegenüber Generalisierbarkeit und Performance zu treffen. In Tabelle 4.1 wird die Gesamtanzahl der verwendeten Datensätze dargestellt. Es ergeben sich insgesamt 5157 annontierte Grundrisspläne.

Datensatz	Gesamtzahl	Gesamtzahl mit Annotation
CVC-FP	122	122
CubiCasa5k	5000	5000
PLAN4	786	35
=	**5908**	**5157**

Tabelle 4.1: Überblick der Datensammlung von Grundrissplänen. Gesamtzahl beläuft sich auf 5157 annontierte Grundrisspläne.

Für diese Arbeit wurden die Datensätze in 80% Trainingdaten und jeweils 10% Validierungs- und Testdaten aufgeteilt (Tabelle 4.2). Insgesamt werden somit 3600 Grundrisspläne, bestehend aus 3000 für das Training und 600 für die Validierung, in jeder Epoche verarbeitet.

Datensatz	Training	Validierung	Test
CVC-FP	98	12	12
CubiCasa5k	4000	500	500
PLAN4	29	3	3
=	**4127**	**515**	**515**

Tabelle 4.2: Aufteilung der Daten in Training, Validierung und Test.

4.2.5.2 Hardware

Im Rahmen dieser Arbeit, wurde für das Training die Cloud-Infrastruktur der HAW verwendet. Nachfolgend werden Komponenten der Hardware aufgelistet.

- GPU: NVIDIA V100s 8GB HBMS
- RAM: 16GB
- CPU: Intel Xeon Processor

5 Experimente

In diesem Kapitel wird das Mask R-CNN anhand der vorgestellten Metriken quantitativ evaluiert. Die Grundlage bilden dabei Experimente, welche eine Verbesserung der Leistung anstreben. Analysiert werden Optimierungen im Bezug Hyperparameter als auch Auswirkungen von Data Augmentation. Darüber hinaus wird das optimierte Modell im Hinblick auf eine verlängerte Trainingsphase untersucht.

5.1 Backbone

Die Auswahl für das Backbone beschränkt sich für diese Arbeit auf das ResNet-101 und ResNet-50. Der ausschließliche Unterschied besteht in der Anzahl an Schichten für die Feature-Extraktion. So beinhaltet das ResNet-101 101-Schichten und ResNet-50 50-Schichten. In Tabelle 5.1 werden die erzielten Ergebnisse des Mask R-CNN mit den jeweiligen Backbone dargestellt.

	Test-Datensatz			
ResNet	mAP	Precision	Recall	IoU
ResNet-101	81,7	85,2	52,2	58,2
ResNet-50	78,9	83,3	50,3	56,7

Tabelle 5.1: Backbone Verlgeich: ResNet-101 vs ResNet-50. Metriken zeigen die Überlegenheit von ResNet-101 im Bezug auf Feature-Extraktion.

Das ResNet-101 erzielt eine Precision von $85,2\%$, somit befinden sich unter den Klassifizierungen, überwiegend richtige Pixel. Der Recall fällt auf $52,2\%$. Es lässt sich darauf schließen, dass von der Gesamtmenge der Ground-Truh-Pixel mehr als die Hälfte korrekt klassifizert werden. Des weiteren deutet der IoU mit einem Wert von $58,2\%$, eine gute Balance zwischen den erzeugten Masken und den Ground-Truth-Pixeln. Das ResNet-50 schneidet im Vergleich dazu schlechter ab. Dabei liegen sämtliche Metriken

mit durchschnittlich 2% unterhalb des ResNet-101. Dem entsprechend fällt die mAP beim ResNet-101 mit $81,7\%$ und für ResNet-50 $78,9\%$ aus. Das Backbone bildet somit die essentielle Grundlage des Modells, dabei zeigt sich insbesondere, das große DNN zur Feature-Extraktion besonders gut geeignet sind.

5.2 Verbesserung mit Data Augmentation

In der Tabelle 5.2 werden die Auswirkungen einer Augmentation gegenüber ohne Augmentation gestellt. Die Grundlage bildet das ResNet-101 als Backbone.

	Test-Datensatz			
Augmentation	mAP	Precision	Recall	IoU
Ohne Data Augmentation	81,7	85,2	52,2	58,2
Data Augmentation	83,7	85,8	52,7	59,1

Tabelle 5.2: Mask R-CNN mit und ohne Data Augmentation. Modell mit Augmentation erzielt Verbesserungen.

Der Einsatz von Data Augmentation bewirkt eine Gesamtverbesserung des Modells. So weist die Precision mit $85,5\%$ eine Steigerung von $0,6\%$ auf. Ein ähnliche Verbesserung gilt auch für den Recall, welche mit $52,7\%$ eine Steigerung von $0,5\%$ erzielt. Der IoU verbessert sich dagegen mit $0,9\%$ auf insgesamt $59,1\%$. Dem entsprechend erreicht die mAP ein Gesamtergebnis von $83,7\%$, im Verhältnis zum Modell ohne Data Augmentation ist dies eine Steigerung von 2%. Data Augmentation stellt somit eine einfache Methode da, um die Performance zu verbessern.

5.3 Anchor-Boxen, Momentum und Learning-Rate

Das Mask R-CNN kann neben der Anpassung des Backbones oder das Trainieren mit oder ohne Augmentation, noch weitere Parameter unterschiedlich konfigurieren. Dazu gehören ins besondere verschiedene Skalierungen von Anchor-Boxen des RPN. Des weiteren kann ebenso die Learning-Rate und das Momentum optimiert werden.

In den nachfolgenden Experimenten wird für das Mask R-CNN als Backbone ResNet-101 definiert und trainiert wurde mitsamt Data Augmentation.

5.3.1 Skalierungen von Anchor-Boxen

In Abschnitt 4.1.2.1 werden die Skalierungen von den Autoren des Mask R-CNN standardmäßig mit $32^2, 64^2, 128^2, 256^2, 512^2$ für das RPN spezifiziert. Die Annahme beruht darauf, große Objekte erfassen zu können. Da jedoch die abgebildeten Räume in Grundrissplänen, in überwiegenden Plänen als kleine Objekte ausfallen, ist die Evaluation der optimalen Skalierungsgröße von Interesse. Experimentiert wurde dabei mit den folgenden Skalierungsgrößen, welche nachfolgend als Kleine-Skalierungen, Mittlere-Skalierungen und Große-Skalierungen abgekürzt werden.

1. Kleine-Skalierungen: $8^2, 16^2, 32^2, 64^2, 128^2$
2. Mittlere-Skalierungen: $16^2, 32^2, 64^2, 128^2, 256^2$
3. Große-Skalierungen: $32^2, 64^2, 128^2, 256^2, 512^2$

In Abbildung 5.1 werden vorgestellte Skalierungen von Anchor-Boxen für einen zentralen Pixel illustriert.

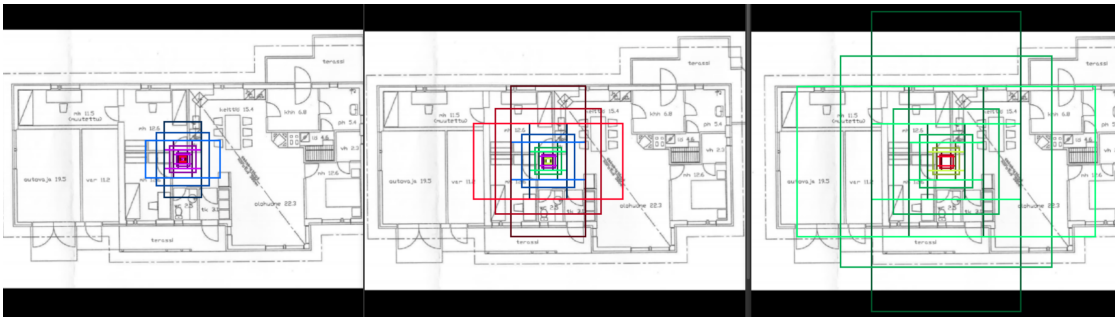

Abbildung 5.1: Grundrissplan mit Anchor-Boxen aus dem Test-Datensatz. Links Kleine-Skalierungen, in der Mitte die Mittleren-Skalierungen und rechts die Großen-Skalierungen. Name: CC5K-4972

Zu beobachten ist, dass Kleine-Skalierungen den Raum nicht komplett erfassen, wodurch relevante Pixel verloren gehen können. Mittlere-Skalierungen hingegen bieten eine optimale Abdeckung des Raumes an. Die Großen-Skalierungen dagegen überschatten den gesamten Grundrissplan mit Anchor-Boxen, wodurch größere Mengen an nicht relevanten Pixel in die Berechnung mit berücksichtigt werden. Die evaluierten Ergebnisse werden in Tabelle 5.3 abgebildet.

	Test-Datensatz			
Skalierung	mAP	Precision	Recall	IoU
Kleine-Skalierungen	82,5	84,5	53,1	57,9
Mittlere-Skalierungen	85,1	87,1	54,1	57,8
Große-Skalierungen	83,7	85,8	52,7	59,1

Tabelle 5.3: Mask R-CNN mit unterschiedlichen Skalierungen für die Anchor-Boxen, welche vom RPN erzeugt werden. Mittlere-Skalierungen erreichen die besten Resultate.

Im Gesamtvergleich schneiden die Anchor-Boxen, welche mit Kleinen-Skalierungen erzeugt werden, am schlechtesten ab. Die mAP erreicht einen Wert von $82,5\%$ mit einer Precision von $84,5\%$. Der Recall erzielt mit $53,1\%$, im Gegensatz zu Großen-Skalierungen, eine Steigerung von $0,4\%$. Große-Skalierungen erreichen in der Precision und IoU bessere Ergebnisse, wodurch die mAP insgesamt stärker ausfällt. Die besten Resultate erzielt das Mask R-CNN mit Anchor-Boxen der Mittleren-Skalierung. Besonders auffallend ist dabei die Verbesserung der Precision und des Recalls, mit $87,1\%$ und $54,1\%$. Der IoU fällt jedoch zu Großen-Skalierungen, um $1,3\%$ schwächer aus. Es ergibt sich eine mAP von $85,1\%$, damit lässt sich begründen, dass Skalierungen einen besonderen Einfluss auf die Performance ausüben. Für Grundrisspläne eigenen sich am effektivsten Mittlere-Skalierungen von Anchor-Boxen.

5.3.2 Momentum und Learning-Rate

Nachfolgend werden Auswirkungen der Parameter Momentum und Learning-Rate auf das Training und die Validierung des Modells untersucht. Standardmäßig definieren He et al. [4] Momentum mit $\gamma = 0.09$ und Learning-Rate mit $\eta = 0.002$ (s. Abbildung 5.2). Experimentiert wurde mit $\gamma = 0.05$ (s. Abbildung 5.3) und $\eta = 0.004$ (s. Abbildung 5.4). Die Grafik links in den nachfolgenden Abbildungen repräsentiert die Kostenfunktion für den Trainings-Datensatz und rechts für den Validierungs-Datensatz. Die Y-Achse zeigt dabei den skalaren Verlust und X-Achse die jeweilige Epoche an.

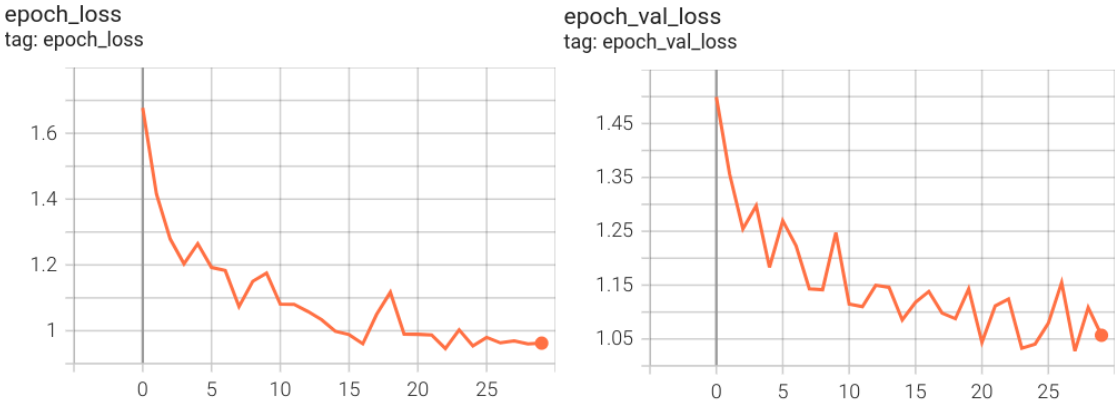

Abbildung 5.2: Momentum mit $\gamma = 0.09$ und Learning-Rate mit $\eta = 0.002$. Training (links) verläuft monoton fallend. Ab Epoche 15 werden kleine Verbesserungen erreicht, bis es anfängt zu stagnieren. Der tiefste Punkt wird in Epoche 23 mit einem Wert von 0.95 erreicht. Validierung (rechts) verläuft fallend, jedoch mit kleinen zick-zack Ausprägungen. Ab Epoche 20 ist ein Plateau zu beobachten.

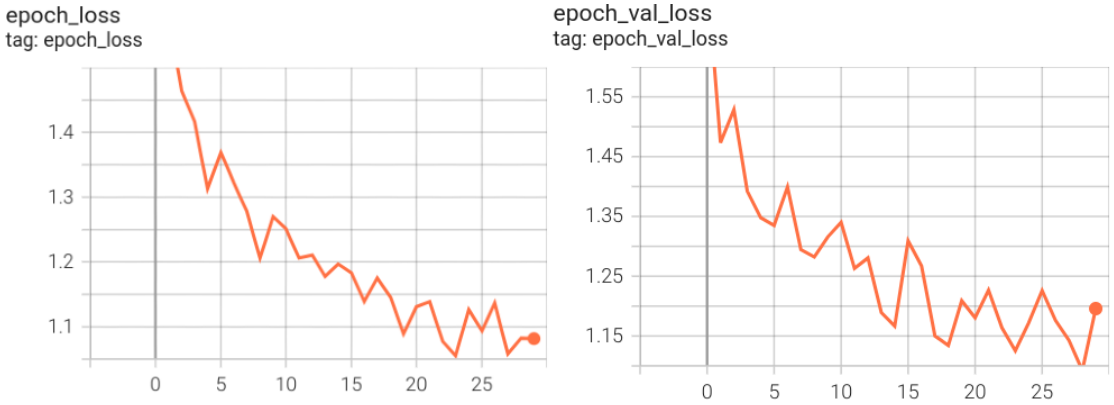

Abbildung 5.3: Momentum mit $\gamma = 0.05$ und Learning-Rate mit $\eta = 0.002$. Training (links) verläuft zick-zackig und erreicht einen Plateau ab Epoche 23. Der tiefste Punkt wird in Epoche 23 mit einen Wert von 1.05 erreicht. Validierung (rechts) bildet große zick-zackige Schwankungen und Verbesserungen bleiben ab Epoche 20 aus.

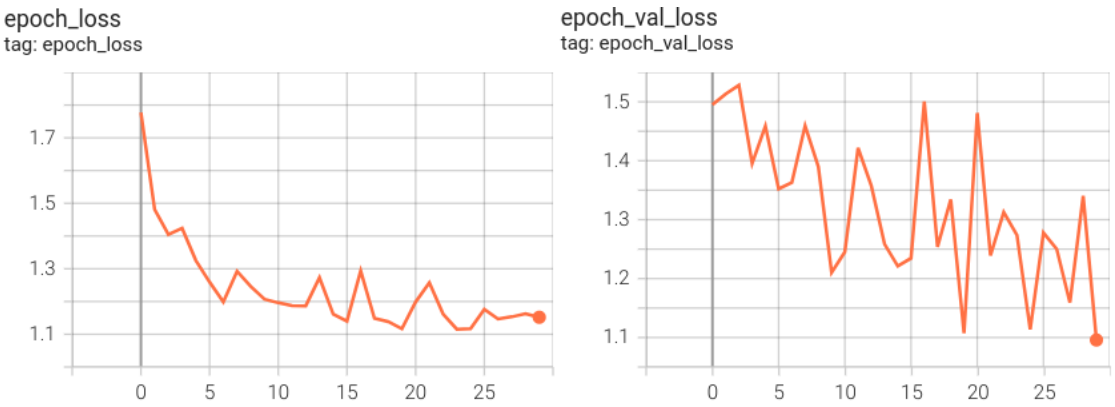

Abbildung 5.4: Momentum mit $\gamma = 0.09$ und Learning-Rate mit $\eta = 0.004$. Training (links) verläuft bis Epoche 5 monoton fallend, anschließend wird ein Plateu erreicht. Der tiefste Punkt wird in Epoche 18 mit dem Wert von 1.13 erreicht. Die Validierung (rechts) oszilliert besonders stark.

Der Momentum-Parameter dient dazu, die Änderung des Gewichtes aus der vorherigen Iteration, mit in die neue Gewichtung einfließen zu lassen. Wird der Term geschwächt, wie in Abbildung 5.3 mit $\gamma = 0.05$, so ist ein unruhiges Verhalten zu beobachten. Die Kostenfunktion für den Trainings-Datensatz, bildet dabei eine zackige Ausprägung und erreicht ab der Epoche 23 einen Plateau. Der Validierungs-Datensatz schwankt besonders stark. Die Auswirkungen einer Steigerung der Learning-Rate wird in Abbildung 5.3 mit $\eta = 0.004$ verdeutlicht. Zu beobachten ist, dass der Verlauf des Trainings eine deutlich flachere Kurve widerspiegelt. Dabei liegen Verbesserungen nur in kleinen Maßen vor, stattdessen bildet sich nach anfänglichem bemühen ein Plateau ab Epoche 5 ab. Insbesondere oszilliert die Funktion für den Validierungs-Datensatz sehr stark, weshalb der Verlust konstant groß bleibt. Das Modell kann nicht generalisieren und passt sich dem Trainings-Datensatz an. Das beste Verhalten lässt sich in Abbildung 5.2 mit einem Momentum $\gamma = 0.09$ und Learning-Rate $\eta = 0.002$ beobachten. Der Verlust von der Funktion des Trainings-Datensatz nimmt dabei kontinuierlich ab. Dabei ist ein stetig ruhiger Verlauf, mit wenig bis keinen Ausschlagungen, zu beobachten. Ein ähnliches Verhalten lässt sich für den Validierungs-Datensatz feststellen. Der Verlust nimmt ebenfalls ab, mit kleinen Ausschwenkungen, jedoch bleibt dieser in der gleichen Höhe wie des Trainings-Verlaufs.

In Tabelle 5.4 werden die evaluierten Metriken für den Test-Datensatz dargestellt. Es lässt sich feststellen, dass eine Abweichung des Momentums und der Learning-Rate von den Standardwerten, keine Verbesserungen erzielen.

	Test-Datensatz			
γ, η	mAP	Precision	Recall	IoU
$\gamma = 0,09, \eta = 0,002$,	83,7	85,8	52,7	59,1
$\gamma = 0,05, \eta = 0,002$	80,5	85,1	51,1	56,6
$\gamma = 0,09, \eta = 0,004$	77,4	82,8	50,6	57,5

Tabelle 5.4: Mask R-CNN mit unterschiedlichen γ und η Konfigurationen. Basierend auf ResNet-101 + Data Augemntation + Große-Skalierungen. Empfohlene Konfigurationen mit $\gamma = 0,09, \eta = 0,002$ erreichen die besten Resultate.

5.4 Das Optimale Modell

Im Rahmen der Experimente, wurde das Modell quantitativ untersucht. Dabei stellten sich Konfigurationen der Parameter, als besonders effizient da. In Tabelle 5.5 wird untersucht, ob ein verlängertes Training mit den effizienten Parameter eine Steigerung der Performance erzielen kann. Das Mask R-CNN wurde dabei mit einem ResNet-101 als Backbone, Mittleren-Skalierungen für Anchor-Boxen und den standardisierten Werten für Momentum und Learning-Rate trainiert.

	Test-Datensatz			
Modell	mAP	Precision	Recall	IoU
Mask R-CNN (100 Epochen)	90,8	89,2	54,5	63,1
Mask R-CNN (30 Epochen)	85,1	87,1	54,1	57,8

Tabelle 5.5: Mask R-CNN basierend auf ResNet-101 + Data Augemntation + Mittlere-Skalierungen. Das Modell, mit einem Training von 100 Epochen, kann deutliche Verbesserungen aufweisen.

Es zeigt sich, dass eine Verlängerung des Trainings auf 100 Epochen, eine deutliche Verbesserung im Gegensatz zum Modell mit 30 Epochen aufweisen kann. Insbesondere verzeichnet die Precision und der IoU einen Aufschwung, während der Recall nahezu unverändert bleibt. Insgesamt ergibt sich eine mAP von $90,8\%$, also eine Steigerung um $5,7\%$. Es ist kein Overfitting zu beobachten, was unter anderem an der breiten Datengrundlage mit Hife von Data Augmentation zu verzeichnen ist. Der Verlauf des Trainings (s. Abbildung 5.5) bekräftigt diese Aussage. Dabei nimmt der Verlust kontinuierlich ab. Im Verlauf der Validierung zeichnen sich ab der Epoche 50 nur geringfügige Verbesserungen aus. Ein Trainieren über 100 Epochen hinaus, würde keine merklichen Verbesserung erzielen.

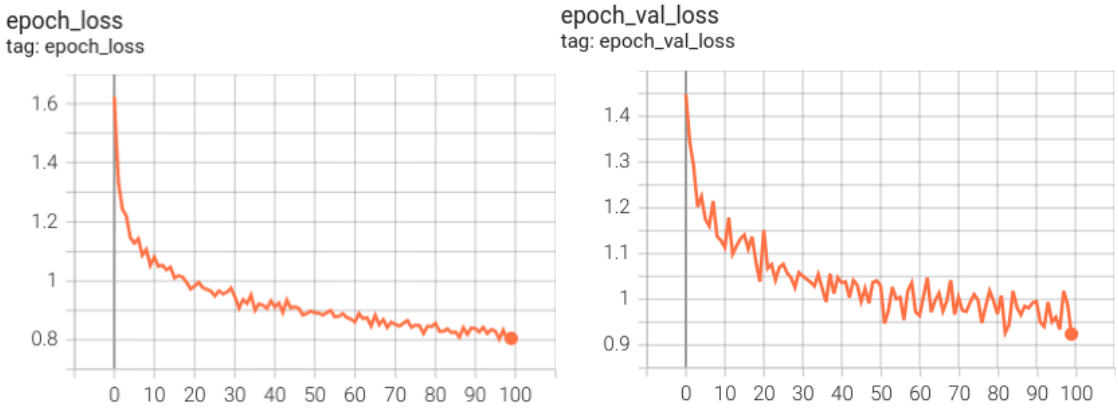

Abbildung 5.5: Mask R-CNN basierend auf ResNet-101 + Data Augemntation + Mittlere-Skalierungen und 100 Epochen. Die Grafik links repräsentiert die Kostenfunktion für den Trainings-Datensatz und rechts für den Validierungs-Datensatz. Die Y-Achse zeigt dabei den skaleren Verlust und X-Achse die jeweilige Epoche an. Der Verlauf des Trainings verläuft monoton fallend, ohne große Schwankungen aufzuweisen. Ab Epoche 30 werden die Verbesserungen geringfügiger. Es bahnt sich eine Konvergenz an. Der tiefste Punkt wird in Epoche 98 mit einem Wert von 0.80 erreicht. Die Validierung bildet ab Epoche 50 eine oszillierende Kurve. Demnach ist ein frühes Abbrechen, des Trainings, ab Epoche 50 empfehlenswert.

6 Bewertung

Um die Qualität der Ergebnisse auszuwerten, werden in diesem Kapitel zwei ausgewählte Grundrisspläne betrachtet. Entscheidend ist neben der Klassifikation, Detektion, und Segmentierug von Räumen, auch die Lokalisierung von Eckpunkten. Sämtliche Resultate wurden anhand des Mask R-CNN, welches in Abschnitt 5.4 ausformuliert wurde, errechnet. Repräsentativ wird ein einfacher als auch ein schwieriger Plan beleuchtet. Weitere Ergebnisse werden in Anlage A und Anlage B dargestellt.

6.1 Einfacher Grundrissplan

In Abbildung 6.1 ist die Auswertung eines einfachen Grundrissplan zu sehen. Dabei handelt es sich um einen Plan, mit wenigen Störquellen. Wandsegmente werden als schwarze dicke Linien abgebildet. Darüber hinaus beinhalten Räume keine Textsegmente, welche beispielsweise die Raumgröße zum Ausdruck bringen. Ebenso fehlen überwiegend Möbelsegmente. Die Türen werden nach klassischem Ansatz mit einem Öffnungsschlag gezeichnet. Es befinden sich insgesamt 16 Räume mit unterschiedlichen Raumformen. Dazu gehören quadratische, rechteckige als auch rundliche Formen. Darüber hinaus befinden sich an den Rändern Maßangaben, wie beispielsweise die Länge der Wände. Das Mask R-CNN erzeugt für jede Raum-Detektion eine Maske, welche visuell in einer unterschiedlichen Farbe abgebildet wird. In gestrichelten Linien werden dazugehörige ROIs dargestellt. Die roten Punkte repräsentieren Eckpunkte, welche anhand des Harris-Corner-Detector in der Nachbearbeitung evaluiert wurden sind. Zu beobachten ist, dass unterschiedliche Raum-Formen nahezu vollständig segmentiert werden. Äußere Störquellen werden als solche erfasst und als Hintergrundpixel klassifiziert. Die Qualität zeigt sich in der Generierung einer Maske, so ist eine Maske mit einheitlichen Dimension-Größen am effektivsten. Masken werden besonders gleichmäßig erzeugt, wenn ein Raum aus vier geschlossenen Wandsegmenten mit Fenster und einer Tür besteht. Mangelhafte Masken zeigen sich vor allem bei großen Räumen mit Durchgängen. Die Ursache liegt an den ROIs, welche

den Raum nicht ganz erfassen. Ein solcher Raum wird vom Mask R-CNN als n-Räume interpretiert, welche in Form von schwammigen Masken verdeutlicht werden. Dem entsprechend fällt das Ergebnis im Bezug auf die Eckenfindung mit dem Harris-Corner-Detector ab. Unsaubere Masken mit unterschiedlichen Krümmungen, weisen fehlerhafte Eckpunkte aus. Eine besondere Schwierigkeit entpuppt sich bei Räumen mit Trennwänden innerhalb des Raums. Das Mask R-CNN Modell kann diese teilweise erkennen und generiert Masken mit Segmentierungen von wenigen Pixel der Trennwände.

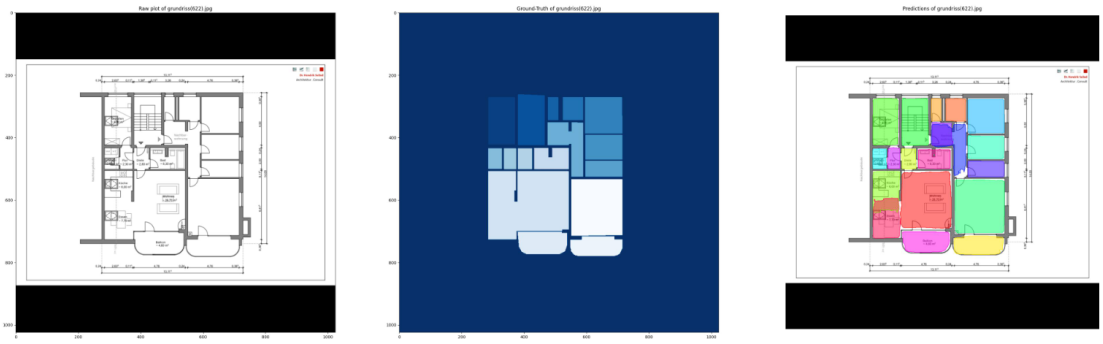

Abbildung 6.1: Links: Grundrisplan, Mitte: Ground-Truth-Pixeln, Rechts: Generierte Masken vom Mask R-CNN, Unten: ROIs in gestrichelten Linien + generierte Masken inklusive detektierte Eckpunkte mit Harris-Corner-Detector. Klassifikationen müssen eine Zuversicht von mind. 90% erfüllen. Mask R-CNN kann unterschiedliche Formen segmentieren. Räume mit Durchgängen erweisen sich als schwierig. Name: PLAN4-330

6.2 Schwieriger Grundrissplan

Ein schwieriger Grundrissplan kennzeichnet sich insbesondere durch die Vielfalt an Störquellen aus. In Abbildung 6.2 ist ein solcher Plan abgebildet. Der Plan beinhaltet 23 Räume, mit unterschiedlichen Formen. Darüber hinaus sind unter anderem handschriftlich eingravierte Linien, Kreise und Textsegmente, welche in einem roten Auszug im inneren als auch äußeren eines Raums vorliegen, eingezeichnet. An den äußeren Wänden befinden sich Maßangaben der Wände in Form von Hilfslinien. Zu dem beinhalten Räume maschinell erzeugte Textsegmente, welche die Größe des Raums angeben. Daneben äußern sich weitere Linien in den Räumen als Orientierungspunkte. Türen liegen mit einem Öffnungsschlag vor und Fenster werden durch weiße Lücken in den Wänden widergespiegelt.

Zu beobachten ist, dass das Mask R-CNN überwiegend Räumlichkeiten ausfindig machen kann. ROIs, welche in gestrichelten Linien vorliegen, definieren die Größe des Raums. Das Modell kann einige Räume nicht vollständig segmentieren, in welchen besonders viele Störquellen vorliegen. So beinhaltet beispielsweise der Flur, welcher einen langen Gang symbolisiert, handschriftliche Markierungen und maschinell erzeugte Segmente, weshalb das Modell den Raum als solches nur teilweise segmentieren kann. Ein weitere Schwäche zeigt sich, dass ein Raum mit vielen Textsegmenten in der Mitte, als Zwei-Räume interpretiert wird. Besonders qualitativ sind die erzeugten Masken, welche in überwiegend identisch großen Dimensionen vorliegen. Dadurch werden Räume wie beispielsweise mit 6-Ecken, richtig vom Harris-Corner-Detector erkannt.

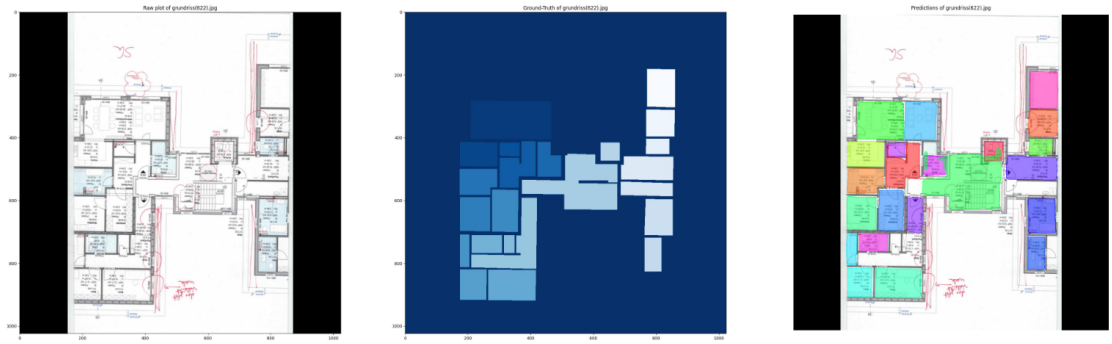

Abbildung 6.2: Links: Grundrissplan, Mitte: Ground-Truth-Pixeln, Rechts: Generierte Masken vom Mask R-CNN, Unten: ROIs in gestrichelten Linien + generierte Masken inklusive detektierte Eckpunkte mit Harris-Corner-Detector. Klassifikationen müssen eine Zuversicht von mind. 90% erfüllen. Überwiegend werden Räume vollständig segmentiert. Räume mit vielen Textsegmenten, werden als n-Räume interpretiert. Name: PLAN4-141

7 Fazit

Im Rahmen dieser Arbeit wurde die Anwendbarkeit von maschinellen Lernen auf rasterbasierte Grundrisspläne untersucht, um anschließend Eckpunkte von Räumen zu lokalisieren. Dabei wurde ein mehrstufiges System konzipiert, bei welchem das Mask R-CNN Modell die Segmentierung von Räumen in Form von Masken durchführt. Anschließend durchlaufen die generierten Masken den Harris-Corner-Detector, bei welchem Eckpunkte lokalisiert werden.

Die Grundlage der Daten bilden dabei frei zugängliche wie auch ein selbst annotierter Datensatz, welcher aus der bereitgestellten Datensammlung von PLAN4 erzeugt wurde. In der Analyse der Daten wurde festgestellt, dass ein Grundrissplan in einfache als auch komplexere Pläne unterteilt werden kann. Dabei bestehen einfache Grundrisspläne aus Wandsegmenten, Tür mit einem Öffnungsschlag, Fenster und wenigen Störquellen. Komplexere Grundrisspläne beinhalten neben unterschiedlichen Farbgebungen, auch viele weitere Störquellen wie beispielsweise Möbelsegmente, Hilfslinien oder handschriftliche Gravierungen. Ausgehend von der Ausgangslage, wurden ausschließlich die beiden Klassen Raum und Hintergrundpixel spezifiziert.

In den darauf folgenden Experimenten wurde quantitativ das Mask R-CNN Modell untersucht. Unter anderem wurde das Backbone mit ResNet-50 und ResNet-101 evaluiert. Darüber hinaus wurden ebenso die Auswirkungen von Data Augmentation berücksichtigt, mit dem Ziel durch künstliche Vielfalt bessere Ergebnisse zu erzielen. Es wurden Modifikationen des RPN vorgenommen, in dem unterschiedliche Skalierungen von Anchor-Boxen durchgeführt wurden sind. Daneben sind unterschiedliche Parameter-Einstellungen, wie das Momentum und die Learning-Rate, im Hinblick auf das Verhalten des Trainings untersucht wurden. Es kristallisierte sich heraus, dass das Mask R-CNN mit ResNet-101, Data Augmentation und Mittlere-Skalierungen besonders performante Ergebnisse erzielen kann. Abweichungen von den standardisierten Parameter-Einstellungen für das Momentum und die Learning-Rate, erzielten mangelhafte Trainings-Verläufe.

Zum Schluss wurde das System qualitativ ausgewertet, bei welchem repräsentativ ein einfacher und ein schwieriger Plan untersucht wurden sind. Beim einfachen Plan konnte das Mask R-CNN nahezu vollständig Räume erfassen, während beim Schwierigen Plan Störquellen die Segmentierung erschwerten. ROIs bilden dabei die Grundlage für die Segmentierungen, dabei definieren sie die Größe des Raums. Erfasst ein ROI den Raum unvollständig, so ergibt sich dies in der Maske. Daneben stellte sich heraus, dass Masken mit identisch großen Dimensionen die Lokalisierung von Eckpunkten mit dem Harris-Corner-Detector vereinfachen. Bei Masken mit Krümmungen entstanden falsche Eckpunkte.

7.1 Design-Entscheidung

Das System verzichtet bei der Vorverarbeitung auf mathematische Filter, wie morphologische Filterung, um Störquellen (wie beispielsweise dünne Linien oder Textsegmente) zu entfernen. Gründe liegen dabei an der fehlenden Generalisierbarkeit auf die breite Diversität von Grundrissplänen, somit würden sie die Erkennung teilweise erschweren. Wie bereits Yin et al. [22] bei ihrer Forschung feststellten, können Filter ausschließlich auf ausgewählten Grundrissplänen angewandt werden, wodurch eine Vollautomatisierung ausgeschlossen wäre. Beispielsweise kann eine Erosion[1] auf Grundrissplänen nicht konsequent funktionieren, da das Erodieren von dünnen Linien zu fehlenden Semantiken führen könnte, unter anderem können Wände aus dünnen als auch dicken Linien bestehen. Stattdessen wird die Aufgabe Störquellen zu erfassen, an das maschinelle Lernen weitergereicht, bei welchem der Kerngedanke daraus besteht relevante Features zu erlernen

7.2 Ausblick

Das Mask R-CNN kann im Hinblick der Hyperpyarameter, weiter optimiert werden. Im Rahmen dieser Arbeit wurden ausgewählte Parameter untersucht.

In der Instanz-Segmentierung existieren weitere Modelle, welche Erweiterungen des Mask R-CNN darstellen. Dazu gehört beispielsweise das Mask Scoring R-CNN Modell [7], welches auf die Problemstellung der Arbeit untersucht werden könnte.

[1] Erosion ist eine Basisoperation der morphologischen Filterung.

Darüber hinaus können besonders etablierte semantische Segmentierungsnetze, wie die Unet-Architektur [17], in Kombination mit einem Objekt-Detektor eingesetzt werden. Räume können besser segmentiert und Störquellen gefiltert werden.

Literaturverzeichnis

[1] Abdulla, Waleed: *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. Github, 2017. – Zugriffsdatum: 26.08.2022

[2] Dodge, Samuel ; Xu, Jiu ; Stenger, Bjorn: *Parsing floor plan images*. IEEE, 2017. – URL https://ieeexplore.ieee.org/document/7986875. – Zugriffsdatum: 15.04.2022

[3] Harris, Christopher G. ; Stephens, M. J.: *A Combined Corner and Edge Detector*. The Plessey Company, 1988. – Zugriffsdatum: 21.08.2022

[4] He, Kaiming ; Gkioxari, Georgia ; Dollár, Piotr ; Girshick, Ross: *Mask R-CNN*. arXiv, 2017. – URL https://arxiv.org/abs/1703.06870. – Zugriffsdatum: 25.08.2022

[5] Heras, Llu'is-Pere de las ; Terrades, OriolRamos ; Robles, Sergi ; S'anchez, Gemma: *CVC-FP and SGT: a new database for structural floor plan analysis and its groundtruthing tool*. 2015. – Zugriffsdatum: 03.09.2022

[6] Hosang, Jan ; Benenson, Rodrigo ; Schiele, Bernt: *Learning non-maximum suppression*. arXiv, 2017. – URL https://arxiv.org/abs/1705.02950. – Zugriffsdatum: 26.08.2022

[7] Huang, Zhaojin ; Huang, Lichao ; Gong, Yongchao ; Huang, Chang ; Wang, Xinggang: *Mask Scoring R-CNN*. arXiv, 2019. – URL https://arxiv.org/abs/1903.00241. – Zugriffsdatum: 25.08.2022

[8] Kalervo, Ahti ; Ylioinas, Juha ; Häikiö, Markus ; Karhu, Antti ; Kannala, Juho: *CubiCasa5K: A Dataset and an Improved Multi-Task Model for Floorplan Image Analysis*. arXiv, 2019. – URL https://arxiv.org/abs/1904.01920. – Zugriffsdatum: 03.09.2022

[9] LeCun, Yann ; Bottou, Léon ; Bengio, Yoshua ; Haffner, Patrick: *Gradient-Based Learning Applied to Document Recognition.* Proceedings of the IEEE, 1998. – URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665. – Zugriffsdatum: 12.08.2022

[10] Lin, Tsung-Yi ; Dollár, Piotr ; Girshick, Ross ; He, Kaiming ; Hariharan, Bharath ; Belongie, Serge: *Feature Pyramid Networks for Object Detection.* arXiv, 2016. – URL https://arxiv.org/abs/1612.03144. – Zugriffsdatum: 16.08.2022

[11] Liu, Chen ; Wu, Jiajun ; Kohli, Pushmeet ; Furukawa, Yasutaka: *Raster-to-Vector: Revisiting Floorplan Transformation.* IEEE, 2017. – URL https://ieeexplore.ieee.org/abstract/document/8237503. – Zugriffsdatum: 13.04.2022

[12] Long, Jonathan ; Shelhamer, Evan ; Darrell, Trevor: *Fully Convolutional Networks for Semantic Segmentation.* arXiv, 2014. – URL https://arxiv.org/abs/1411.4038. – Zugriffsdatum: 14.08.2022

[13] Mivekannin, Maurice: *Grundriss - Definition, Tipps und Software.* appvizer, 2020. – URL https://www.appvizer.de/magazin/bauwesen/architektur/was-ist-ein-grundriss. – Zugriffsdatum: 10.08.2022

[14] Noh, Hyeonwoo ; Hong, Seunghoon ; Han, Bohyung: *Learning Deconvolution Network for Semantic Segmentation.* arXiv, 2015. – URL https://arxiv.org/abs/1505.04366. – Zugriffsdatum: 14.08.2022

[15] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross ; Sun, Jian: *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* arXiv, 2015. – URL https://arxiv.org/abs/1506.01497. – Zugriffsdatum: 16.08.2022

[16] Robbins, Herbert E.: *A Stochastic Approximation Method.* Annals of Mathematical Statistics, 2007. – Zugriffsdatum: 08.09.2022

[17] Ronneberger, Olaf ; Fischer, Philipp ; Brox, Thomas: *U-Net: Convolutional Networks for Biomedical Image Segmentation.* arXiv, 2015. – URL https://arxiv.org/abs/1505.04597. – Zugriffsdatum: 18.09.2022

[18] Rosenblatt, Frank: *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review, 1958. – URL https://citeseerx.ist.psu.edu/viewdoc/download. – Zugriffsdatum: 11.08.2022

[19] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael ; Berg, Alexander C. ; Fei-Fei, Li: *ImageNet Large Scale Visual Recognition Challenge*. arXiv, 2014. – URL https://arxiv.org/abs/1409.0575. – Zugriffsdatum: 09.08.2022

[20] Saha, Sumit: *A Comprehensive Guide to Convolutional Neural Network*. Towardsdatascience, 2018. – URL https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. – Zugriffsdatum: 14.08.2022

[21] Simonyan, Karen ; Zisserman, Andrew: *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv, 2014. – URL https://arxiv.org/abs/1409.1556. – Zugriffsdatum: 08.08.2022

[22] Yin, Xuetao ; Wonka, Peter ; Razdan, Anshuman: *Generating 3D Building Models from Architectural Drawings: A Survey*. IEEE Computer Graphics and Applications, 2009. – URL https://ieeexplore.ieee.org/document/4736453. – Zugriffsdatum: 19.04.2022

[23] Zeng, Zhiliang ; Li, Xianzhi ; Yu, Ying K. ; Fu, Chi-Wing: *Deep Floor Plan Recognition Using a Multi-Task Network with Room-Boundary-Guided Attention*. arXiv, 2019. – URL https://arxiv.org/abs/1908.11025. – Zugriffsdatum: 21.04.2022

[24] Zhang, Yiqing ; Chu, Jun ; Leng, Lu ; Miao, Jun: *Mask-Refined R-CNN: A Network for Refining Object Details in Instance Segmentation*. Sensors, 2020. – URL https://www.mdpi.com/1424-8220/20/4/1010. – Zugriffsdatum: 25.08.2022

A Anhang

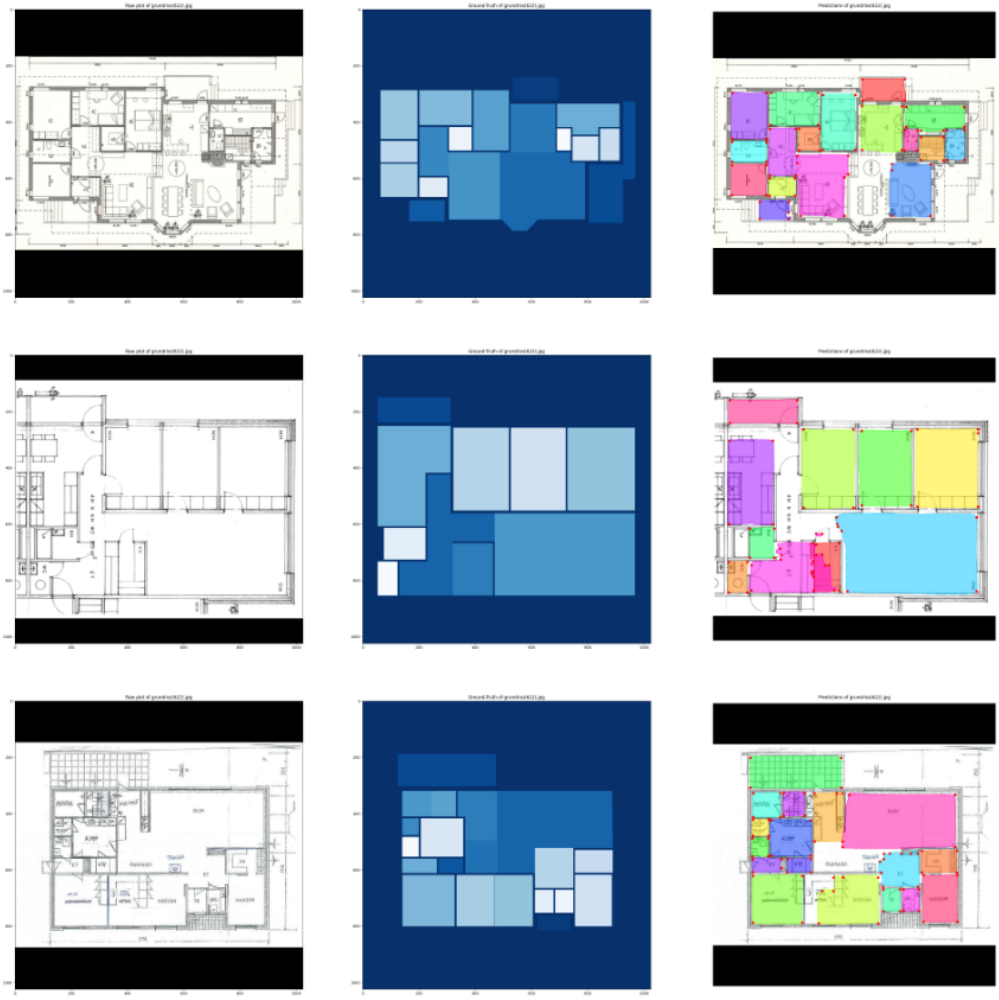

Abbildung A.1: Auszug aus dem Testdatensatz - Links: Grundrissplan, Mitte: Ground-Truth-Pixeln, Rechts: Generierte Masken vom Mask R-CNN mit detektierten Eckpunkten. Klassifikationen müssen eine Zuversicht von mind. 90% erfüllen.

B Anhang

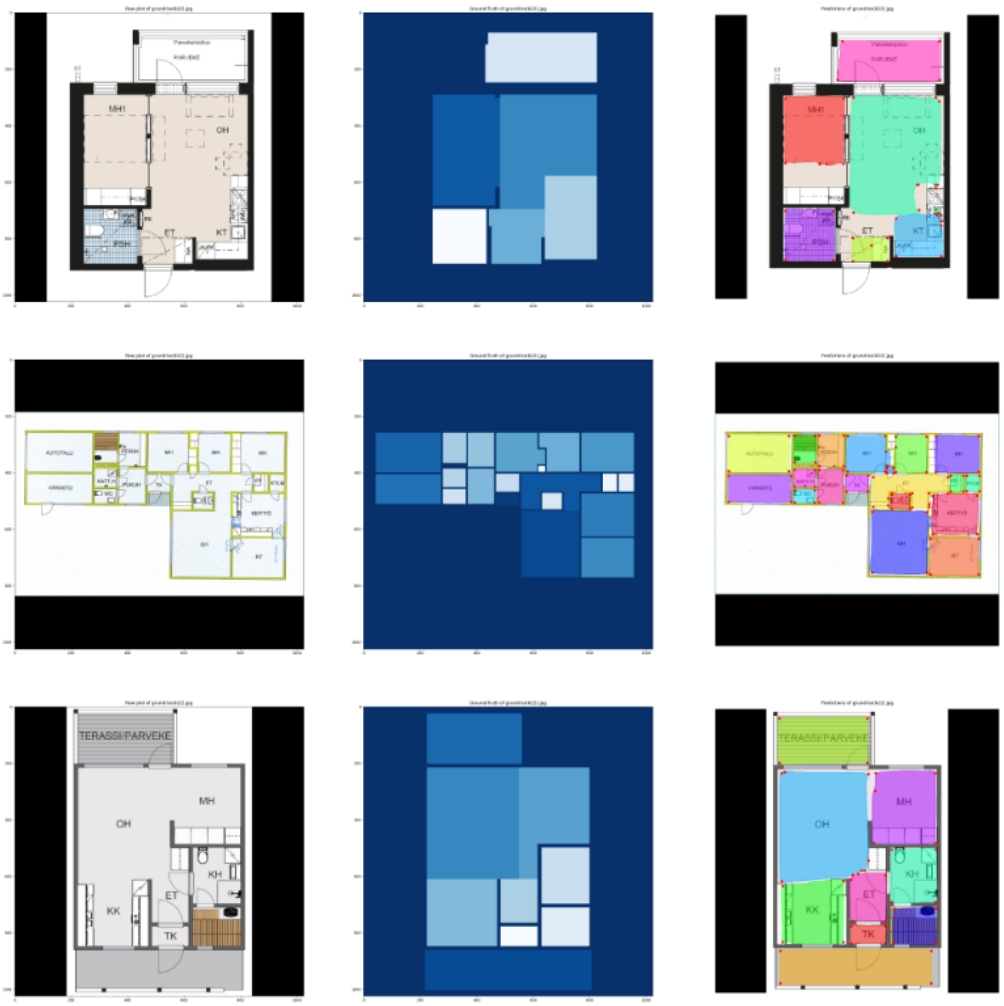

Abbildung B.1: Auszug aus dem Testdatensatz mit farblichen Grundrissplänen - Links: Grundrissplan, Mitte: Ground-Truth-Pixeln, Rechts: Generierte Masken vom Mask R-CNN mit detektierten Eckpunkten. Klassifikationen müssen eine Zuversicht von mind. 90% erfüllen.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original