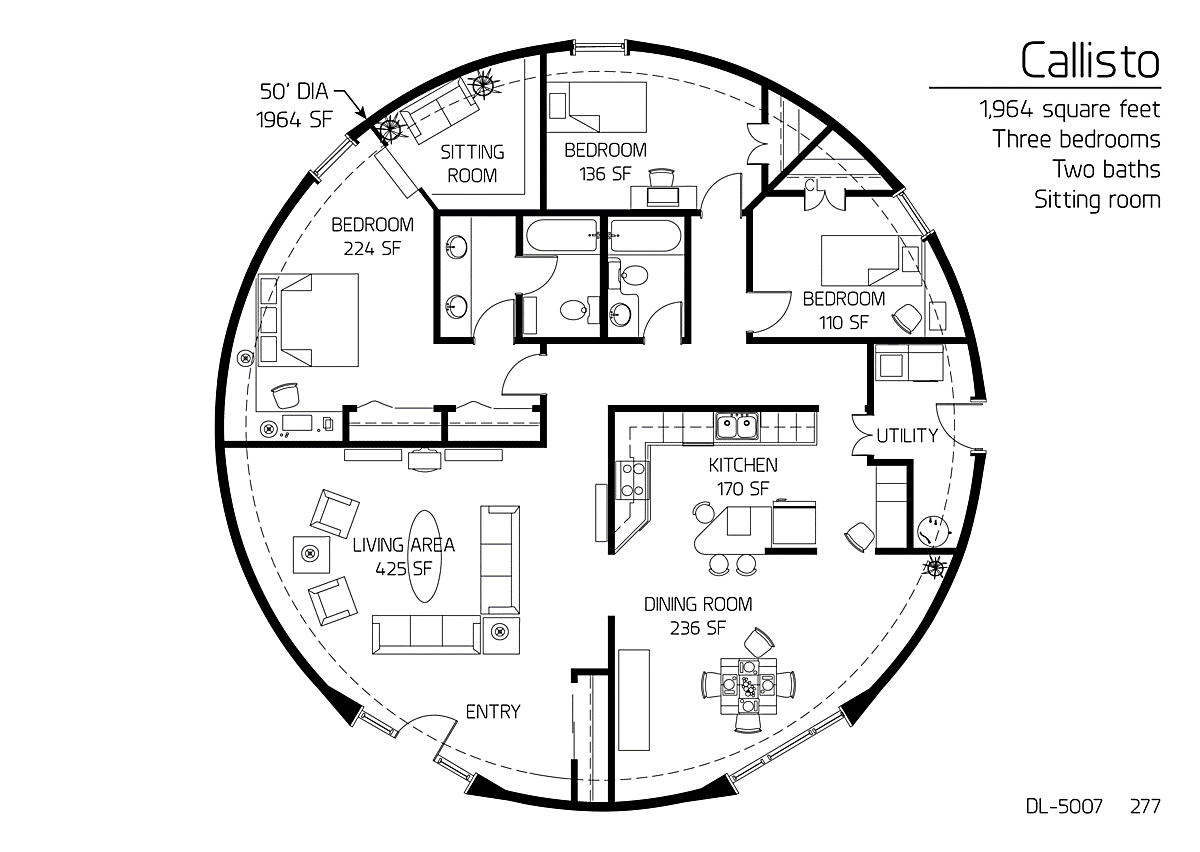




**Deep Learning zur**

**Objekterkennung in Grundriss-Bildern**





Abschlussbericht der Kooperationsphase 2019/20

Durchgeführt am Deutschen Forschungszentrum für Künstliche Intelligenz

Betreuer: Dr. Stefan Agne, Christoph Balada

Betreuer am Hector-Seminar: Dr. Joachim Götz, Ingmar Oehme

Annika Nassal, PF14

nassalan@hector-seminar.de

Inhaltsverzeichnis

[Inhaltsverzeichnis 2](#_Toc44750549)

[Abbildungsverzeichnis 3](#_Toc44750550)

[Tabellenverzeichnis 4](#_Toc44750551)

[1 Einleitung 1](#_Toc44750552)

[2 Durchführung 2](#_Toc44750553)

[2.1 Ansatz 2](#_Toc44750554)

[2.2 Datenvorverarbeitung und -vervielfältigung 2](#_Toc44750555)

[2.2.1 Aufbereiten des Datensatzes 2](#_Toc44750556)

[2.2.2 Zuschneiden und Vervielfältigen der Trainingssymbole 3](#_Toc44750557)

[2.3 Aufbau und Training des Netzes 4](#_Toc44750558)

[2.4 Optimieren des Netzes mithilfe von Metriken 4](#_Toc44750559)

[2.5 Vor- und Nachverarbeitung von Anwendungsdaten 5](#_Toc44750560)

[3 Ergebnisse 7](#_Toc44750561)

[3.1 Hyperparameter 7](#_Toc44750562)

[3.2 Metriken 7](#_Toc44750563)

[4 Fehleranalyse und Deutung der Ergebnisse 8](#_Toc44750564)

[5 Zusammenfassung und Ausblick 9](#_Toc44750565)

[5.1 Erweiterte Fragestellungen 9](#_Toc44750566)

[5.2 Anwendung und Nutzen des Projekts 9](#_Toc44750567)

[6 Danksagung 10](#_Toc44750568)

[7 Quellenverzeichnis 11](#_Toc44750569)

[8 Selbständigkeitserklärung 12](#_Toc44750570)

[9 9 Anhang 13](#_Toc44750571)

Abbildungsverzeichnis

[Abb. 2: Beispiel aus dem Validierungsdatensatz 2](file:///D:\Users\anni\Documents\GitHub\CoopDFKI\Formal\Kooperationsphase2019_ProtokollAnnikaNassal.docx#_Toc44853751)

Tabellenverzeichnis

[Tab. 1: Beispiel einer Tabelle 3](#_Toc21882774)

# Einleitung

In unserem heutigen Alltag sind Computer allgegenwärtig und sie werden immer intelligenter. Sie lernen, Sprache zu verarbeiten und Antworten zu geben und können uns bei der Navigation ohne große Schwierigkeiten den schnellsten Weg zu unserem Ziel unter Einbezug der Verkehrsdaten zeigen. So übernehmen sie immer öfter Aufgaben, die sonst Menschen übernehmen müssten, welche dafür deutlich länger brauchen. Doch es gibt eine Aufgabe, in deren Bewältigung Computer noch weit hinter uns Menschen liegen – die Interpretation visueller Daten. Während Menschen ohne Probleme Informationen aus einem Bild extrahieren können und wissen, was dargestellt ist, sind Bilder für den Computer zunächst einfach nur Daten in Form von Zahlen. Dies wird zu einem Problem, wenn Computer Aufgaben für uns durchführen sollen, die auf Bilddokumenten beruhen. Ein bekanntes Beispiel hierfür ist die Klassifikation von Bildern anhand des dargestellten Tiers – Katze oder Hund. Dieses Problem wurde mit einem neuen, doch immer häufiger erfolgreich eingesetzten informatischen Konzept gelöst – dem neuronalen Netz. Neuronale Netze gehören zum Bereich der künstlichen Intelligenz und simulieren ein virtuelles Netz aus Neuronen ähnlich dem menschlichen Gehirn. Im hier vorgestellten Projekt wird mithilfe eines solchen Neuronalen Netzes konkret das Problem der Analyse von Grundrissplänen mit eingezeichneter Möblierung bearbeitet. Dabei sollen die Pläne eingelesen und danach mit entsprechenden Objektbezeichnungen an allen Möbelstücken wieder ausgegeben werden. Schwierigkeiten, die hierbei auftreten können, sind beispielsweise eine sehr variierende Symbolwahl in Plänen unterschiedlichen Ursprungs und die Feststellung der Position der Objekte auf dem Plan. Für die Lösung der Aufgabe soll ein passendes neuronales Netz trainiert werden, bis es die höchstmögliche Effizienz erreicht hat. Für das Training wird ein Datensatz aus einem Wissenschaftswettbewerb zum Einsatz kommen.

**Fragestellung:**

Wie muss ein neuronales Netz trainiert und in Vor- und Nachverarbeitung der Daten eingebettet werden, um die höchstmögliche Genauigkeit bei der Objekterkennung in Grundrissplänen zu erreichen?

# Durchführung

## Ansatz

Um das Ziel zu erreichen, auf einem Grundrissbild alle Möbelstücke und Einrichtungsobjekte zu erkennen und deren Position zu bestimmen, wurde ein mehrteiliger Ansatz gewählt. Dieser beinhaltet für die Objektklassifizierung ein Faltungsnetz, das zwischen den verschiedenen Möbelkategorien unterscheiden kann. Da ein solches Netz eine feste Eingabegröße hat und zur Positionsbestimmung nicht geeignet ist, muss vorher und nachher ein Datenverarbeitungsschritt eingefügt werden. Bevor die Daten zur Klassifizierung dem Netz übergeben werden, ist es daher notwendig, den Plan in Ausschnitte fester Größe zu teilen. Um dabei zu gewährleisten, dass kein Objekt nur am Rand eines Ausschnitts zu finden ist, wird das Bild in überlappende Quadrate geteilt. Dabei weicht jedes Bild vom vorherigen nur genau eine Pixelreihe oder eine Pixelspalte ab. Aufgrund einer Größe der Teilstücke von 100\*100 Pixeln ist jeder Pixel außer den ersten 99 Randpixeln an allen Rändern 10.000 Mal in der Netzeingabe zu finden. Dabei befindet er sich auf jedem Teilstück an einer anderen Position. Das Netz verarbeitet dann nacheinander alle Teilstücke und gibt für jedes von diesen einen Vektor mit einer Wahrscheinlichkeit für die verschiedenen Möbelklassen aus. Diese Vektoren werden zugeordnet zu dem Pixel des Grundrissplanes, der dem mittleren Pixeln des Eingabequadrats (X: 50; Y: 50) entspricht, abgespeichert und am Ende werden mithilfe von Schwellwerten aus den Wahrscheinlichkeiten die tatsächlichen Positionen und Objekttypen extrahiert.

## Datenvorverarbeitung und -vervielfältigung

Der Grundrissplan-Datensatz des Wettbewerbes „ICDAR2019-ORF“ der „International Conference on Document Analysis and Recognition 2019“ (engl.: Internationale Konferenz zur Dokumentenanalyse und -erkennung 2019; zweijährlich stattfindende wissenschaftliche Konferenz) beinhaltet knapp 250 Pläne, auf denen mehr als 7000 Objekte aus 12 Kategorien zu finden sind. Diese Kategorien sind Toilette, Dusche, Badewanne, Waschbecken, Bidet, Tisch, Stuhl, Sofa, Sessel, Nachttisch, Bett und Herd, wobei diese Kategorien auf unterschiedlichen Plänen unterschiedlich dargestellt werden. Die Bilder stehen als PNG-Dateien zur Verfügung und die Metadaten sind in einer JSON-Datei festgehalten.

### Aufbereiten des Datensatzes

Schon früh während der Arbeit mit dem Datensatz konnten Fehler und Probleme mit den Daten festgestellt werden, weshalb dieser aufbereitet werden musste. Zunächst fiel auf, dass die Metadaten Informationen zu Grundrissbildern enthielten, die im Download nicht mitgeliefert wurden. Um diesen Fehler zu beheben, wurde ein Programm geschrieben, das die Informationen, die keinem Bild zugeordnet werden konnten, aus den Metadaten entfernt. Das selbe Programm entfernte auch Fehlinformationen, die in den Metadaten enthalten waren. So waren beispielsweise Angaben zu Positionen von Symbolen außerhalb der Größe des Grundrissplans, was nicht möglich ist. Des Weiteren hatten die Symbole auf den Grundrissplänen sehr variierende Größen, also unterschiedliche Seitenlängen in Pixeln, was ein Problem darstellte, da die Eingabegröße des Neuronalen Netzes, das zur Klassifizierung verwendet werden sollte, fest ist. Die Größe des größten Symbols war dafür nicht geeignet, da auf anderen Grundrissbildern die Symbole sehr klein waren. Um einen Überblick zu erhalten, wie viele Symbole mit welcher Größe vorhanden sind, wurden die Metadaten mit einem Programm ausgewertet, das diese Information als Histogramm darstellt. Darauf ist zu erkennen, dass viele Symbole zwischen 100\*100 und 10\*10 Pixel groß sind. Also wurde das Projekt eingegrenzt auf die Erkennung von Symbolen die in ein Feld von 100\*100 Pixeln passen. Diejenigen Symbole, die zu groß oder zu klein sind, wurden aus den Metadaten gelöscht, sodass das Netz beim Training im Abgleich mit den korrekten Lösungen keine verfälschten Werte ausgibt.

### Zuschneiden und Vervielfältigen der Trainingssymbole

Um nun für das Training des Netzes zur Klassifizierung der Objektsymbole diese 100\*100 Pixel großen Symbolbildchen zu erhalten, mussten diese aus den Grundrissplänen ausgeschnitten werden.

Dafür wird zunächst die aufbereiteten Metadaten eingelesen und die Grundrisspläne als Numpy Arrays importiert und in einer Liste gespeichert. Daraufhin müssen die Pläne den richtigen Metadaten zugeordnet werden, da die Liste keine Dateinamen speichert, sondern jeder Plan nur einen Listenindex hat. Um dies zu erreichen, wurde ein Dictionary eingeführt, das beim Einlesen der Bilder die Namen den Listenindices zuordnet. Da später aber nicht der Name, sondern die in den Metadaten verwendete Bild-ID zum Auffinden dieses Bildes notwendig ist, wurde ein zweites Dictionary verwendet. Dieses ordnet für jedes in den Metadaten genannten Bild, sofern es importiert wurde und damit vorhanden ist, der ID in den Metadaten den Index in der Bildliste zu. So kann anhand der ID der Index des Bildes ermittelt und dann über diesen auf das Bild-Array in der Liste zugegriffen werden.

Nachdem alle Daten importiert und sortiert sind, wird nun eine for-Schleife genutzt, um alle Annotationen, wie die Informationen für jedes Symbol in den Metadaten bezeichnet sind, nacheinander durchzugehen. Innerhalb der Schleife wird dann die sogenannte Bounding Box, die Positionsangabe des aktuellen Symbols auf dem Grundrissplan, und die Bild-ID verwendet, um das Symbol auf dem entsprechenden Plan zu finden. Ein Quadrat der Kantenlänge 100\*100 mit dem Symbol im Zentrum wird aus den Bilddaten herauskopiert und in einer Liste gespeichert. Daraufhin wird in einer zweiten Liste an der entsprechenden Stelle die Nummer der Objektkategorie, die auf dem Quadrat dargestellt ist, abgespeichert. Bei der Auswahl des Quadrats gibt es einen Ausnahmefall, in dem das Symbol nicht im Zentrum positioniert werden kann. Findet sich der Mittelpunkt des Symbols näher als 50 Pixel am Rand des Grundrissbildes, so muss das Quadrat in dieser Richtung am Rand ausgerichtet werden. Anderenfalls würde das Quadrat leere Pixel kopieren, was zu einem Fehler bei der Verarbeitung durch das Netz führen würde. Für die Lösung dieses Problems wurde eine Funktion geschrieben, die die Koordinaten des linken oberen Ecks des Quadrats von 100\*100 Pixeln für beide Dimensionen innerhalb des Intervalls [0; Ausdehnung des Bildes -100] hält. Nachdem dieses Standardsymbolbild abgespeichert wurde, wird diese nun vervielfältigt. Dies steuert der geringen Datenmenge entgegen, die mit dem Datensatz zur Verfügung steht. Die Vervielfältigung beinhaltet immer eine Abwandlung des Standardbildes, da nur so die Verallgemeinerung in der Klassifizierung durch das Netz gewährleistet werden kann. So ist beispielsweise ein Verdrehen des Bildes eine Möglichkeit, dem Netz die Fähigkeit zu geben, unbekannte Symbole beliebiger Orientierung zu bestimmen. Für die Rotation wird die Bibliothek Imutils verwendet. Neben der Rotation werden zudem eine Skalierung mithilfe der Bibliothek OpenCV und eine handgeschriebene Verschiebung des Symbols innerhalb des Quadrates von 100\*100 Pixeln genutzt. Die Werte der Rotation, Verschiebung und Skalierung werden für jeden Aufruf zufällig neu bestimmt und auch die Verteilung der verschiedenen Vervielfältigungen pro Symbol ist zufällig. So soll gewährleistet werden, dass möglichst viele verschiedene Versionen jedes Symbols entstehen. Die Anzahl der Vervielfältigungen pro Symbolbildchen wurde zunächst fest auf 20 gelegt. Es stellte sich jedoch heraus, dass manche Objektklassen deutlich häufiger vertreten waren als andere, was für die Auswertung des Netzes unvorteilhaft ist. Daher wurde der konstante Wert durch eine umgekehrte Skalierung mit Toleranzbereich ersetzt. Dafür wurde für jede Kategorie die Anzahl der verfügbaren Symbolbildchen bestimmt. Als nächstes wurde der Mittelwerte berechnet. Alle Kategorien, deren Symbolanzahl im Toleranzbereich um den Mittelwert lag, bekamen den Standardwert von 20. Für Symbolanzahlen über dem Toleranzbereich wurde die Obergrenze des Toleranzbereiches durch diese dividiert und der entstehende Wert, der immer unter 1 liegt, mit 20 multipliziert, sodass die Vervielfältigung bei dieser häufig vorkommenden Klasse pro Symbol etwas geringer ausfällt. Entsprechend wurde für Werte unter dem Toleranzbereich die Untergrenze diesen durch die Symbolanzahl geteilt und der immer über 1 liegende Wert mit 20 multipliziert, wodurch die Vervielfältigung dieser eher seltenen Klasse höher ausfällt. Die mathematische Formulierung der Skalierung ist in Tab. 1 dargestellt und wird an einem Beispiel gezeigt. Die Ergebnisse des Beispiels sind in Abb. 1 visualisiert. Es ist zu erkennen, dass für sehr kleine Bildanzahlen sehr groß wird und umgekehrt und es einen konstanten Bereich gibt.

|  |  |
| --- | --- |
| **Allgemeine Theorie**  Ursprüngliche Anzahl von Bildern pro Klasse:  Durchschnitt der Anzahlen:  Vervielfältigungsfaktor für jedes Bild pro Klasse:  Für jedes  Falls    Falls    Falls | **Beispiel**  Gegeben:  Anzahl von Bildern pro Klasse  {185;67;70;318;9;161;381;46;77;145;63;78}Durchschnitt der Anzahlen  Gesucht:  Vervielfältigungsfaktor für jedes Bild pro Klasse  Ergebnis:  {20,00; 21,79; 20,86; 12,14; 162,22; 20,00; 10,13; 31,74; 20,00; 20,00; 23,17; 20,00} |

Tab. : Umgekehrte Skalierung mit einem Toleranzbereich allgemein und an einem Beispiel

Abb. : Graphische Darstellung der Ergebnisse des Beispiels

## Aufbau und Training des Netzes

Für die Symbolerkennung auf den Quadraten von 100\*100 Pixeln wurde ein neuronales Netz vom Typ eines Faltungsnetztes verwendet. Die anfänglichen Gewichte im Netz wurden von Tensorflow Keras bezogen und sind auf dem ImageNet-Datensatz vortrainiert. Dies bietet den Vorteil, dass ein spezifisches Training für die Aufgabe schneller und erfolgreicher abläuft, da die Gewichte bereits an die Erkennung von Strukturen in Bildern angepasst sind. Es wurden die Netze InceptionNetV3 und ResNet50 von Keras ausgewählt. Diese wurden dann mit einem Teil der Bilder mit den Objektsymbolen trainiert. Um eine möglichst hohe Genauigkeit des Netzes zu erreichen, wurden in mehreren Trainingsdurchläufen die Hyperparameter Epochen, Batch Size, Lernrate und Optimizer variiert und die Ergebnisse festgehalten und verglichen.

## Optimieren des Netzes mithilfe von Metriken

Die Genauigkeit des Netzes auf den Daten, mit denen es trainiert, ist aufgrund des sogenannten „Overfitting“ (dt.: Überanpassung) nicht repräsentativ. Das Overfitting beschreibt hierbei die Überanpassung des Netzes an die Trainingsdaten und damit eine fehlende Verallgemeinerung. Man könnte sagen, das Netz hat die Trainingsdaten „auswendiggelernt“. Die Folge davon ist, dass das Netz zwar auf diesen Daten sehr genau ist, auf anderen Daten aber nur eine geringe Verlässlichkeit bei den Ergebnissen zeigt. Um dieses Problem frühzeitig zu erkennen und ihm entgegenzuwirken, wird vor dem Training ein Teil des Datensatzes aussortiert. Dabei wird darauf geachtet, dass dieser Teil auch Eigenschaften enthält, die im Trainingsanteil nicht vorhanden sind, um die Fähigkeit des Netzes zur Verallgemeinerung zu überprüfen. Für die Aufteilung der Grundrisspläne in diesem Projekt wurde als Kriterium festgelegt, dass nur Pläne, die ausschließlich rechtwinklig aufeinander stehende Wände zeigen, für das Training verwendet werden. Der aussortierte Teil wurde wieder in zwei Kategorien unterteilt. Der sogenannte Validierungsdatensatz enthält nur Pläne mit geraden, aber auch diagonal und nicht rechtwinklig verlaufenden Wände und der Testdatensatz enthält schließlich die Pläne, auf denen auch gekrümmte Wände und runde Gebäude gezeigt werden. Je ein Beispiel für jeden Teil des Datensatzes ist in Abb. 1 zu sehen.

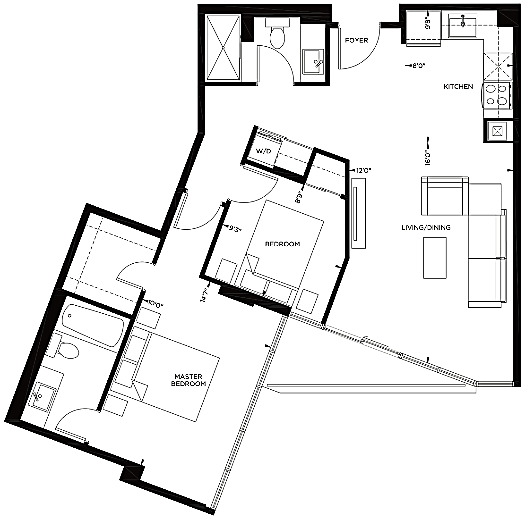
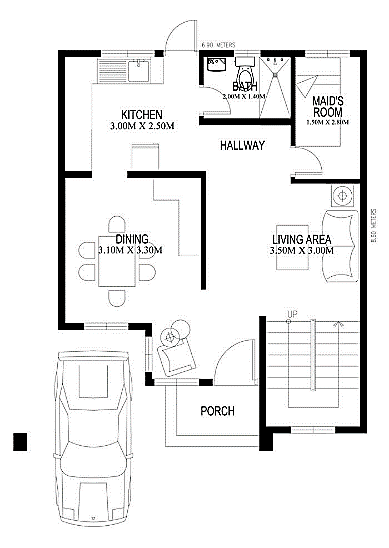
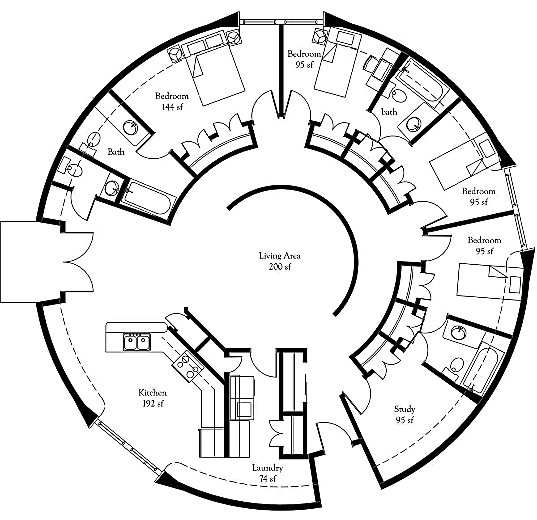


Abb. : Beispiele aus Trainings-, Validierungs- und Testdatensatz

Nach jeder Epoche des Trainings wird nun ein sogenanntes Callback aufgerufen. Darin wird das Netz auf die Validierungsdaten angewandt. Die Ergebnisse des Netzes werden dann mit den Musterlösungen verglichen und das Netz wird mithilfe von daraus berechneten Metriken bewertet. Die genutzten Metriken sind die Konfusionsmatrix, die Precision-Recall-Kurve, der Average-Precision-Score und der F1-Score.

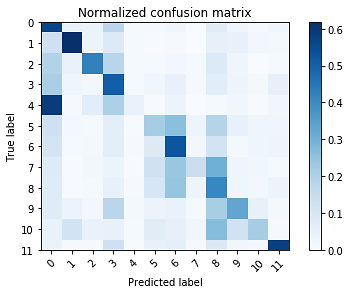
Die Konfusionsmatrix ist die einzige Metrik, die für alle Klassen gemeinsam berechnet wird. Sie wird als tabellenartiges Diagramm dargestellt. Die x-Achse gibt dabei die vom Netz vorhergesagte Klasse an und die y-Achse die tatsächliche Klasse. Das entstehende Raster ist über eine Farbskala gefüllt, die die Häufigkeitsverteilung der vom Netz vorhergesagten Klasse für jede Klasse angibt. Optimal ist ein Netz dann, wenn bei der Vorhersage immer die vorhergesagte Klasse mit der tatsächlichen Klasse übereinstimmt. Die Konfusionsmatrix hat dann einen diagonalen, intensiv gefärbten Streifen und ist sonst vollkommen weiß. In Abb. 2 ist ein Beispiel aus dem Training eines ResNet 50 dargestellt. Darauf ist zu erkennen, dass die Klassen 1 und 11 bereits sehr gut trainiert sind, da sowohl in der Zeile, als auch in der Spalte der Klassen nur ein dunkles Feld zu sehen ist. Klasse 0 beispielsweise wird zwar oft korrekt erkannt, was daran zu erkennen ist, dass in der Zeile nur ein sehr dunkles Feld ist. Dafür wird die Klasse 4 aber oft als Klasse 0 erkannt, was im Diagramm durch das dunkle Feld auf Höhe der 4 in der Spalte 0 dargestellt ist. An der hellen Spalte von Klasse 4 ist zu erkennen, dass das Netz fast nie ein Bild als Objekt der Kategorie 4 klassifiziert.

Abb. : Beispielhafte Konfusionsmatrix aus Epoche 48 des Trainings eines ResNet 50 mit Batch Size 32 und Lernrate 0,001

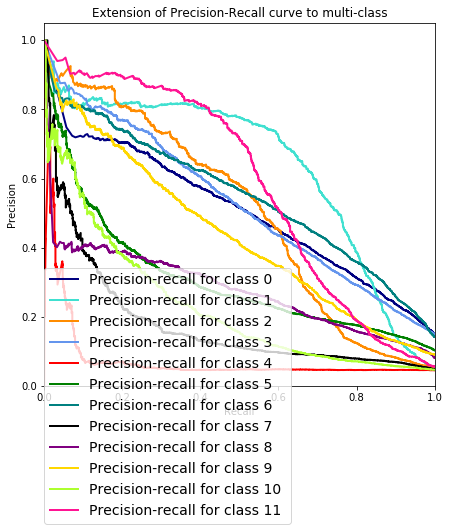
Die Precision-Recall-Kurve wird für jede Klasse einzeln berechnet und alle Kurven können am Ende in einem Diagramm gezeichnet werden. Für die Kurve werden je Klasse zunächst Precision und Recall berechnet. Die Precision wird dabei definiert als der Quotient aus den korrekt als diese Klasse vorhergesagten Bildern und allen als diese Klasse vorhergesagten Bildern. Der Recall ist festgelegt als das Verhältnis von Bildern dieser Klasse, die korrekt als Bilder dieser Klasse vorhergesagt wurden und allen Bildern dieser Klasse. Beide Berechnungen sind in Tab. 2 dargestellt. Um nun mehrere Precision-Recall-Paare für eine Kurve zu erhalten, wird ein sogenannter Threshold, also ein Schwellwert benutzt. Das Netz gibt für jedes Bild für jede Klasse eine Wahrscheinlichkeit aus, mit das Objekt auf dem Bild dieser Klasse zuzuordnen ist. Der Threshold gibt an, ab welcher Wahrscheinlichkeit das Objekt als erkannt gilt. Ein geringerer Threshold bedeutet also, dass das Netz sich weniger sicher sein muss, um das Bild der aktuellen Klasse zuzuordnen. Daraus resultiert dann ein höherer Recall-Wert, da mehr Bilder als die aktuelle Klasse erkannt werden, aber ein geringerer Precision-Wert, da das Netz sich weniger sicher war und sich so mehr Fehler einschleichen. Optimal ist ein Netz dann, wenn bei einem hohen Threshold für alle Klassen ein hoher Recall-Wert und ein hoher Precision-Wert festgestellt werden können. Je nach Anwendung ist dabei oft entweder die Precision oder der Recall von größerer Bedeutung. So soll ein Klassifikator, der Krebstumore auf Bildern erkennt, lieber zu oft Alarm geschlagen haben und dann falschliegen, als dass er zwar immer richtig liegt, wenn er einen Tumor erkennt, er aber viele übersieht. Der Recall hat hier also eine höhere Priorität als die Precision In einem Spamerkennungsprogramm wiederum wird angestrebt, wirklich nur Spam und auf keinen Fall wichtige Mails als Spam zu klassifizieren, während eine nicht als Spam erkannte, aber unerwünschte Mail im Posteingang das kleinere Übel darstellt. Im Beispiel in Abb. 4 wären die Klassen 1 und 11 am besten, da deren Kurven in ihrem Verlauf dem Punkt (1|1) am nächsten sind. Besonders schlecht ist die Klasse 4, da ihr Graph am nächsten an (0|0) verläuft.

Abb. : Beispielhafte Precision-Recall-Kurve für alle zwölf Klassen

|  |
| --- |
| (True Positive) = korrekt als aktuelle Klasse erkannt  (False Positive) = fälschlicherweise als aktuelle Klasse erkannt  (True Negative) = korrekt als nicht die aktuelle Klasse erkannt  (False Negative) = fälschlicherweise als nicht die aktuelle Klasse erkannt |

Tab. : Berechnung von Precision und Recall

Der Average Precision Score ist definiert als die Fläche unter dem Graphen der Precision-Recall-Kurve. Er beschreibt die Aufsummierung aller Precision-Werte, die jeweils mit der Differenz des aktuellen Recall-Wertes zum Recall-Wert bei dem vorangegangenen Threshold multipliziert und so gewichtet werden.

|  |
| --- |
|  |

Der F1-Score ist ein anderes Bewertungsmaß für neuronale Netze, für das Precision und Recall kombiniert werden. Um diese Metrik anzugeben, wird das harmonische Mittel der beiden Werte angegeben. Da sich beide Werte zwischen Null und Eins bewegen, liegt auch der F1-Score in diesem Bereich und erreicht sein Optimum bei Eins. Die Verwendung des harmonischen Mittels führt dazu, dass ein sehr kleiner Wert mehr ins Gewicht fällt und nicht von einem sehr großen Wert ausgeglichen wird.

Schließlich wurde eine Funktion implementiert, die das Training unter Variation der Hyperparameter automatisiert wiederholt. Die Ergebnisse der F1-Scores als Leitmetrik wurden abgespeichert und mithilfe einer PowerQuery-Abfrage in Excel geladen. Dort wurden das arithmetische und das harmonische Mittel der jeweils zwölf Werte gebildet und mithilfe von Farbskalen die besten Hyperparameter abgelesen. Diese wurden gesondert erneut trainiert und verglichen, um so die optimale Hyperparameter-Konfiguration für das Netz zu finden.

## Vor- und Nachverarbeitung von Anwendungsdaten

Ist das beste Netz gefunden und trainiert worden, können nun auch unbekannte Daten verarbeitet werden. Für die Klassifizierung von Objekten auf den Grundrissplänen muss allerdings zunächst die Iteration des 100\*100 Pixel großen Eingabequadrates über den zu bearbeitenden Plan implementiert werden. Hierfür wurde eine for-Schleife genutzt.

Wenn die Ergebnisse des Netzes für das Eingabequadrat vorhanden sind, werden sie zugeordnet zu dem Pixel auf dem Grundrissplan, der dem Pixel Y: 50; X: 50 im Eingabequadrat entspricht, abgespeichert.

Nachdem dann alle Eingabequadrate verarbeitet worden sind, werden die Informationen aus den Netzausgaben extrahiert. Hierfür wird für jeden Pixel festgestellt, welcher der Wahrscheinlichkeitswerte am höchsten ist und der Pixel dieser Klasse zugeordnet. Dann wird überprüft, ob die Wahrscheinlichkeit, die der Pixel für seine Klasse beinhaltet, den Schwellwert überschreitet, ab dem davon ausgegangen werden kann, dass tatsächlich ein Objekt rund um den Pixel vorhanden ist. Der Schwellwert wird experimentell und mithilfe der Precision-Recall-Kurve bestimmt. Schließlich müssen die Ergebnisse noch nach der Sinnhaftigkeit überprüft werden. Einzelne Pixel, die keine Nachbarn der selben Klasse können hier beispielsweise als Fehler gewertet werden und sollten daher gelöscht werden. Zum Schluss wird die Information aus den Ergebnissen in Textform gebracht und kann so an den Anwender übermittelt werden.

# Ergebnisse

## Hyperparameter

## Metriken

# Fehleranalyse und Deutung der Ergebnisse

# Zusammenfassung und Ausblick

## Erweiterte Fragestellungen

Die Erkennung von Objekten in Grundrissplänen ist nur ein Teil der Dokumentenanalyse zur Digitalisierung dieser. Das übergeordnete Ziel ist, alle Informationen, die der Mensch auf einem Grundrissplan erkennen kann, für den Computer und Programme nutzbar und so digital verarbeitbar zu erfassen. Dazu gehören bei den Grundrissplänen neben den Objekten auch die Bestimmung des Raumtyps anhand des Objektvorkommens und dafür auch die Einteilung in Räume durch die Detektion von Wänden und Türen. Des Weiteren können Maße und Größenverhältnisse im Plan erfasst werden. Auch auf dem Bild befindliche Texte sollten zudem eingelesen und in einem weitaus anspruchsvolleren Schritt nicht nur als Text, sondern als Information verarbeitet werden.

## Anwendung und Nutzen des Projekts

Die Bestimmung von Objekten und deren Positionen in Grundrissplänen kann vor allem im Bereich der Einrichtung und Vermietung von Wohnungen verwendet werden. So kann beispielsweise die Einrichtung einer Wohnung vereinfacht werden, indem die gewonnenen Informationen in eine Empfehlung für eine Zusammenstellung von Möbelstücken und Einrichtungsgegenständen des gewünschten Anbieters oder Herstellers entsprechend eines bestimmten Stils oder einer Preisvorgabe umgesetzt werden. Auf der Basis dieser Empfehlung können dann Anpassungen gemacht werden, bis der Kunde vollständig zufrieden ist. Dadurch würde die Dauer bis zur fertigen Einrichtung verkürzt, die Kundenzufriedenheit erhöht und die Arbeit eines Einrichtungsanbieters erleichtert werden.

Auch bei der Vermietung kann eine höhere Zufriedenheit des Mieters und gleichzeitig eine schnellere Verarbeitung der Wünsche erreicht werden. Wenn Vermieter Grundrisspläne ihrer möblierten Immobilien mithilfe des neuronalen Netzes analysieren und die Informationen bei einem Vermietungsportal einreichen, können Kunden schneller die perfekte Wohnung finden, da auch nach der Einrichtung gefiltert werden kann.

Wird die Objekterkennung als Teil einer detaillierteren Grundrissplananalyse eingesetzt, eröffnen sich noch weitere Anwendungsfelder. So könnte beispielsweise die Analyse von Grundrissen von Wohnhäusern, Bürogebäuden, Schulen und anderen öffentlichen Einrichtungen als Grundlage für eine computergestützte Bestimmung des besten Fluchtweges oder Zugangsweges für Rettungskräfte genutzt werden.

Allgemein Floorplan Detection:

* Computergestützte Bestimmung des besten Fluchtweges oder Zugangsweges für Rettungskräfte
* Computergestützte Optimierung von Brandschutzmaßnahmen
* Verlegung von Strom- und Wasserleitungen

## Fazit

# Danksagung

Ich möchten mich bei allen bedanken, die meine Arbeit am Projekt unterstützt haben. In erster Linie gilt mein Dank Christoph Balada und Dr. Stefan Agne vom Deutschen Forschungszentrum für künstliche Intelligenz in Kaiserslautern, die mir dieses Projekt und damit einzigartige Erfahrungen ermöglicht haben und mich inhaltlich bei allen Fragen unterstützt haben.

Dr. Hans-Werner und Josephine Hector danke ich für die langjährige Förderung im Hector-Seminar und die Möglichkeit, wissenschaftliche Arbeit hautnah zu erleben. Schlussendlich möchte ich mich bei meinen Kursleitern Herr Dr. Götz und Herr Oehme bedanken, die mir jederzeit bei Fragen und insbesondere beim Überarbeiten der Dokumentation beratend zur Seite standen.

# Quellenverzeichnis

# Selbständigkeitserklärung

Ich versichere, dass ich diese schriftliche Prüfungsarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass ich alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, durch Angabe der Quellen als Entlehnung kenntlich gemacht habe.

Königsbach-Stein, den [Datum]

Unterschrift

# Anhang