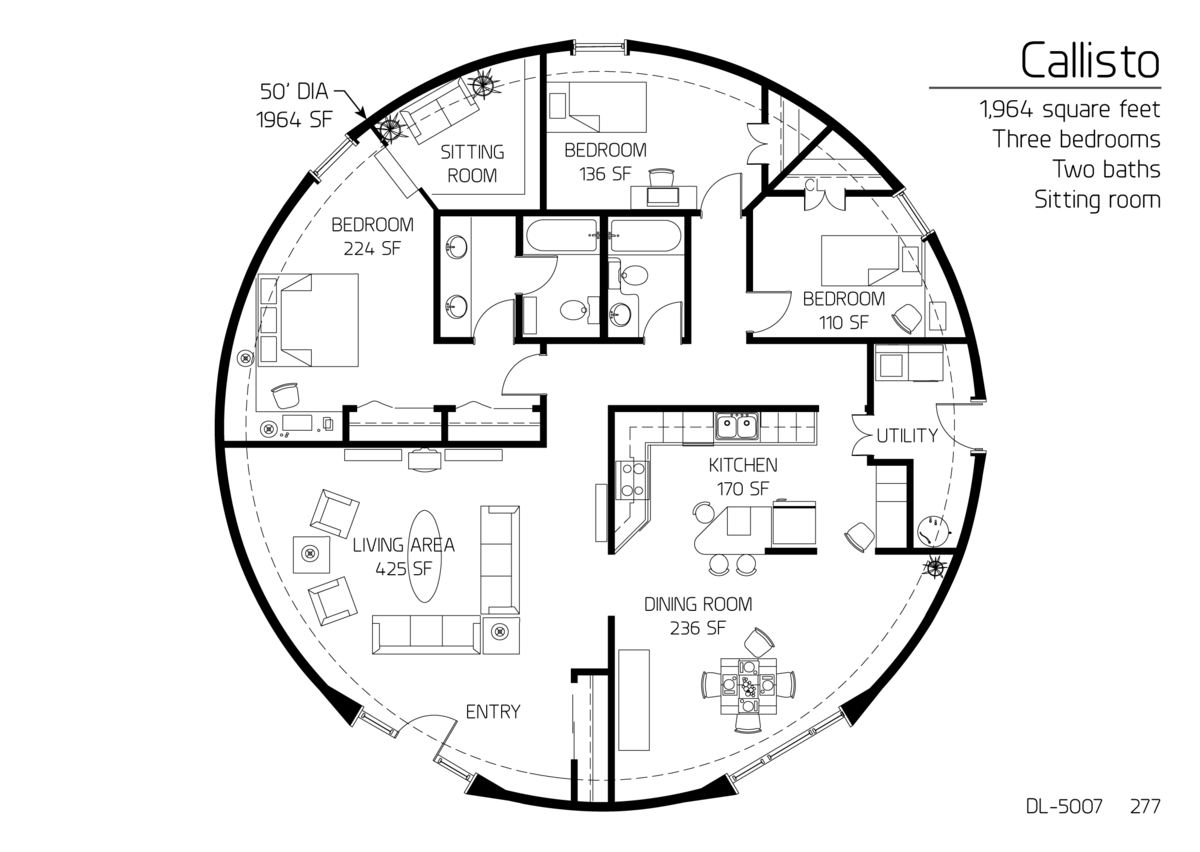
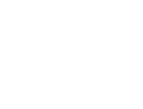




**Deep Learning zur**

**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

[1 Einleitung 1](#_Toc42697633)

[2 Durchführung 2](#_Toc42697634)

[2.1 Ansatz 2](#_Toc42697635)

[2.2 Datenvorverarbeitung und -vervielfältigung 2](#_Toc42697636)

[2.3 Grundaufbau des Netzes 2](#_Toc42697637)

[2.4 Optimierung 2](#_Toc42697638)

[3 Ergebnisse 3](#_Toc42697639)

[3.1 Variation in der Anzahl der Layers 3](#_Toc42697640)

[4 Fehleranalyse und Deutung der Ergebnisse 4](#_Toc42697641)

[5 Zusammenfassung und Ausblick 5](#_Toc42697642)

[5.1 Erweiterte Fragestellungen 5](#_Toc42697643)

[5.2 Anwendung und Nutzen des Projekts 5](#_Toc42697644)

[6 Danksagung 6](#_Toc42697645)

[7 Quellenverzeichnis 7](#_Toc42697646)

[8 Selbständigkeitserklärung 8](#_Toc42697647)

[9 Anhang 9](#_Toc42697648)

[10 weitere Hinweise 10](#_Toc42697649)

Abbildungsverzeichnis

[Abb. 1: Beispiel einer Abbildung 1](#_Toc21887860)

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 relativ 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 den entsprechenden Pixeln 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 Index hat. Um dies zu erreichen, wurde ein Dictionary eingeführt, das beim Einlesen der Bilder der Namen dem Listenindex 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.

## 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. Von Keras wurden das Netz InceptionV3 ausgewählt. Dies wurde dann mit einem als Trainingsdatensatz deklarierten Teil der Symbolbilder trainiert. Der Trainingsdatensatz wurde so gewählt, dass nur gerade Wandkanten, die rechtwinklig aufeinander stehen, beinhaltet sind. Dies dient dem Zweck, dass später in einem Test mit anderen Wandkantenformen gut zu erkennen ist, ob das Netz die gelernten Inhalte verallgemeinern kann. 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 dem Problem 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. 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.

## Vor- und Nachverarbeitung von Anwendungsdaten

# Ergebnisse

## Hyperparameter

## Metriken

# Fehleranalyse und Deutung der Ergebnisse

# Zusammenfassung und Ausblick

## Erweiterte Fragestellungen

## Anwendung und Nutzen des Projekts

# Danksagung

# 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

# 9 Anhang