Hochschule Fulda
University of Applied Sciences

Digitale Tabellenerfassung – Tabellenextraktion durch Bildverarbeitung und künstliche Intelligenz

Christian Baltzer

Bachelorarbeit am Fachbereich AI der Hochschule Fulda

Matrikelnummer: 244860
Betreuer: Prof. Dr. Alexander Gepperth
Zweitgutachter: Diana Peters

Eingereicht am 20.09.2020

Abstract

Im Rahmen dieser Bachelorarbeit wurde – in Zusammenarbeit mit dem Deutschen Zentrum für Luft und Raumfahrt – eine Methode gefunden, Tabellen in digitalisierten Dokumenten zu finden und in ein maschinenlesbares Format zu bringen. Da hierfür bei Recherchen weder eine Lösung in Form von klassischen Algorithmen – wie der Bildverarbeitung – noch eine nutzbare Lösung in Form von neuronalen Netzen gefunden werden konnte, wurde eine neue Methode entwickelt.

Hierfür wurde die Aufgabe in verschiedene Probleme – wie die Tabellen- und Schrifterkennung – aufgeteilt. Um diese zu lösen, wurden Methoden der künstlichen Intelligenz in Form des YOLO Netzes, bestehende OCR Lösungen und eigene Algorithmen genutzt.

Dabei konnten alle Probleme gelöst werden und das Ergebnis wurde in einem funktionstüchtigen Prototyp umgesetzt. [1]

[1] Eine Demonstration ist hier zu finden: https://www.youtube.com/watch?v=8bUTVrqB2QY

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Kontext

Getrieben von Trends wie Industrie 4.0 und einer weiter zunehmenden Digitalisierung werden immer größere Datenmengen elektronisch gespeichert und verarbeitet. Viele Informationen liegen allerdings immer noch in Papierform, als gescannte Dokumente oder als PDF ohne maschineninterpretierbare Struktur vor. Somit werden Systeme benötigt, welche die enthaltenen Daten zuverlässig extrahieren können. Das Deutsche Zentrum für Luft und Raumfahrt entwickelt für die Interpretation technischer Dokumente derzeit ein System, welches diese Aufgabe erfüllen soll. Dabei gestaltet sich das Einlesen von Tabellen als besonders schwierig, da diese oft unabhängig vom restlichen Dokument betrachtet werden müssen und eine andere Vorgehensweise benötigen als normaler Text. In dieser Bachelorarbeit sollte hierfür eine Lösung gefunden werden.

Ein aktuelles Beispiel für die Notwendigkeit eines solchen Systems ist die Herausforderung der deutschen Testzentren für Corona bei der Erfassung von Laborbefunden: Eigentlich sollen dabei alle Befunde elektronisch in das DEMIS, dem Deutschen Elektronischen Melde- und Informationssystem für den Infektionsschutz, des Robert Koch Instituts übertragen werden, um dort zusammengeführt und ausgewertet zu werden. Nach Angaben von Harald Michels, dem Leiter des Gesundheitsamtes Trier, funktioniert dies jedoch nur bedingt. Da DEMIS noch in der Testphase ist, kommen die meisten Befunde per Fax an und werden dabei mit dem gleichen Namen versehen. Sie müssen daher manuell geöffnet und mit dem Namen des Getesteten abgespeichert werden.[1]

Laut Aussage von Herrn Michels gegenüber dem ZDF sei das Amt dabei mit der Arbeitsbelastung durch die manuelle Bearbeitung der Dokumente an der personellen Belastungsgrenze. Ein System, das die Formulare automatisch einlesen und die entsprechenden Daten nutzen kann, könnte hier helfen, die Mitarbeiter wesentlich zu entlasten.

1.2 Problemstellung und Ziele

Aufgrund der breiten Varianz im Aussehen und Aufbau von Tabellen ist es auch mit modernen Algorithmen schwer, eine als Bild gespeicherte Tabelle zu erkennen und ihre Daten zu extrahieren. Es gibt dafür auch keine einfach und universell einsetzbaren Lösungen am Markt. Deswegen ist das Hauptziel des Projektes die Realisierung eines Systems zur automatischen Erfassung von Tabellen und ihrer Inhalte mit Hilfe maschinellen Lernens.

Ziele für das System sind:

1. Die Realisierung des Einlesens von Dokumenten im Bildformat
2. Das Erkennen aller Tabellen in einem Dokument
3. Das Erkennen von 90% aller gegebenen Tabellen
4. Das Erkennen von 85% der zugehörigen Zellen
5. Die Ausgabe der Zelleninhalte in maschinenlesbarer Form
6. Eine Durchlaufzeit von unter einer Minute pro Dokument

1.3 Vorgehen und Aufbau

Zur Strukturierung der grundlegenden Aufgabe wurde ein Kanban Board erstellt, in der potentielle Verarbeitungsschritte des Zielsystems in den Phasen Analyse, Entwickelung und Bewertung geplant und nachverfolgt werden.
Nach einer Betrachtung der bestehenden Forschung in Kapitel 1.5 werden in Kapitel 2 die technischen Grundlagen definiert. In Kapitel 3 werden die Verarbeitungsschritte detailliert, welche in Kapitel 4 im Zusammenhang einer prototypischen Umsetzung implementiert werden. Die Ergebnisse der Forschung und der Implementierung werden in Kapitel 5 diskutiert. Abschließend gibt Kapitel 6 einen Ausblick auf mögliche zukünftige Verbesserungen.

Die gesamte Entwicklung wurde nach agilen Prinzipien durchgeführt. Dabei wurden die potentiellen Lösungsansätze in je einen Sprint umgesetzt. Am Ende eines jeden Sprints wurde das Projekt auf Grundlage der bisherigen Ergebnisse überplant.

Das Kanban Board wurde ständig an den aktuellen, sich durch neue Erkenntnisse und Überprüfungen und Anpassungen verändernden Planungsstand angepasst.

1.4 Vorstellung des Auftraggebers DLR

Das Projekt wurde in Zusammenarbeit mit dem Deutschen Zentrum für Luft- und Raumfahrt durchgeführt. Das Deutsches Zentrum für Luft- und Raumfahrt (DLR) ist ein Forschungszentrum der Bundesrepublik Deutschland. Es umfasst 9.000 Mitarbeiter, verteilt auf 29 Standorte in Deutschland, sowie mehrere Test- und Betriebsanlagen. Es werden außerdem vier Standorte in Brüssel, Paris, Tokyo und Washington DC unterhalten. Das DLR ist in 51 Institute und Einrichtungen, sowie 150 Großforschungsanlagen aufgeteilt, welche je einen eigenen Schwerpunkte setzten.

Das DLR forscht in mehreren Bereichen, darunter Raumfahrt, Luftfahrt sowie Verkehr und Energie. Ein weiterer Bereich ist Digitalisierung / künstliche Intelligenz. Dabei arbeitet das DLR eng mit internationalen Partnern zusammen und ist zum Beispiel verantwortlich für die Überwachung und Planung des Raumlabor Columbus, einem Multifunktionslabor der ISS (International Space Station), einer bemannten Raumstation im Erdorbit.

Abbildung 1: Standorte des DLR in Deutschland

Am Standort Jena ist das Institut für Datenwissenschaften untergebracht. Hier werden Lösungen für neue Herausforderungen der Digitalisierungsära erforscht. Neben den Bereichen Datenmanagement, IT-Sicherheit und Smart Systems wird hier auch auf dem Gebiet der Bürgerwissenschaften geforscht, welche sich zum Ziel gesetzt hat, Wissenschaft besser in gesellschaftliche Prozesse einzufügen.[2]

1.5 Related Work

Im Folgenden wird der derzeitige Stand der Forschungen und Technik zum Thema der Tabellenerkennung näher beleuchtet. Dabei existieren zum einen Ansätze, welche sich speziell mit dem Format PDF befassen, sowie andere, welche einen offeneren Ansatz für unterschiedliche Dokumentenformate thematisieren.

1.5.1 Tabellenerkennung an Hand von Metadaten

Tools wie Camelot (PDF Table Extraction for Humans) [3] nutzen Metadaten und Code eines PDF Dokumentes, um daraus Tabellen zu extrahieren.

Dabei kann entweder nach entsprechenden Strukturen im Code der PDF Dateien gesucht werden oder nach Markierungen, welche bei der Erstellung der Datei erstellt wurden. Diese Methode funktioniert jedoch nur, wenn entsprechende Strukturen vorhanden und korrekt sind.

Wie in Kapitel 2.1.2 beschrieben, kann ein PDF unterschiedlich zusammengesetzt werden, weshalb ein einfacher Standard-Algorithmus nicht verlässlich ist. Des weiteren enthalten nicht alle PDF Dokumente die erforderlichen Daten. Gerade Scans werden meist ohne inhaltliche Erkennung erstellt, wodurch der Einsatz dieser Methode ausgeschlossen wird.

1.5.2 Tabellenerkennung mit Tablenet

Tablenet ist ein Ansatz zur Tabellenerkennung der Firma “Tata Cosultancy Services Limited”. Dabei wird ein Deep Learning Model genutzt. [4]

Das Modell erhält hier als Input ein Ursprungsdokument und erstellt daraus zwei Masken. Die erste markiert dabei den Bereich der Tabelle, wohingegen die zweite Maske die Spalte markiert. Mit diesen Masken werden nun alle Texte herausgefiltert, welche außerhalb des Maskenbereichs liegen. Nach Aussage der Autoren erreicht das Netz dabei eine Genauigkeit von ca. 95%.

Das Originalmodell konnte bei den Recherchen dieser Arbeit nicht gefunden werden, wodurch dieses Ergebnis nicht reproduziert werden kann.

1.5.3 Tabellenerkennung mit DeepDeSRT

Das DeepDeSRT Netz ist ein Ansatz des „German Research Center for Artificial Intelligence“ in Kaiserslautern in Zusammenarbeit mit der „Mannheim University of Applied Sciences“ und der „Kaiserslautern University of Technology“. [5]
Der Ansatz nutzt dabei Deep Learning in Form der Objekterkennung zum Erkennen von Tabellen. Nach Angaben der Autoren erreichte der Ansatz dabei eine Präzision von 94%.

Leider wurde auch dieses Modell nicht veröffentlicht, wodurch das Ergebnis nicht validiert werden konnte.

1.6 Eigenleistung

1. Einrichten einer geeigneten Forschungsumgebung
   - Auswählen geeigneter Tools, Frameworks und Programmiersprachen
2. Entwicklung eines Lösungsansatzes für die Detektion von Tabellen
   - Evaluierung klassischer Bildverarbeitung und Kantendetektion
   - Evaluierung der Objektdetektion mit neuronalen Netzen
   - Trainieren des YOLO Netzes zum Erkennen von Komponenten eines Dokumentes
3. Evaluierung geeigneter OCR Lösungen
   - Technische und juristische Betrachtung der Lösungen
4. Entwickeln einer Kontextdetektion
   - Vergleich von neuronalen Netzen und klassischen Algorithmen
5. Umsetzung der Ergebnisse in prototypischer Form
6. Betrachtung des zukünftigen Einsatzes von Quantencomputern im Bereich der neuronalen Netze

2 Systemzusammenstellung und technische Grundlagen

Im Folgenden wird ein Einblick in die technischen Details der neuronalen Netze gegeben, sowie das Dateiformat PDF beleuchtet.

Zur Durchführung der Entwicklung muss eine Entwicklungsumgebung geschaffen werden, die den praktischen Teil dieser Arbeit in allen Bereichen unterstützt. Dazu muss sie durch jeweils auszuwählende Teilsysteme verschiedene Anforderungen erfüllen, welche anschließend tabellarisch dargestellt und weiter vertieft werden.

2.1 Verwendete Methoden

2.1.1 Neuronale Netze

Ein künstliches neuronales Netz (KNN) ist eine Methode zur Informationsverarbeitung. Dabei wird, nahe dem Vorbild des Gehirns, ein Netz aus Neuronen erzeugt. Ein Neuron ist dabei ein Objekt, welches aus einem Input einen Output generiert. Der Output wird dabei durch eine Aktivierungsfunktion und den Aufbau des Neurons bestimmt.

Das Netz besteht aus mehreren Schichten. Jede Schicht besteht aus mehreren Neuronen, welche mit den Neuronen der nächsten Schicht verbunden sind.

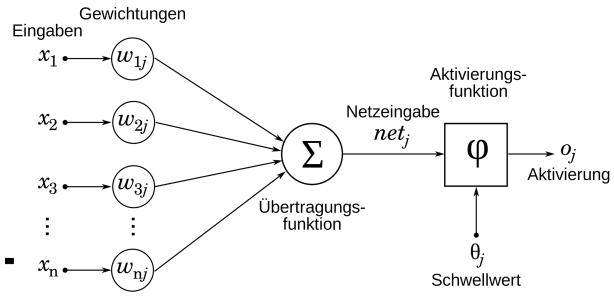


Abbildung 2: Abbildung eines Neuron - [6]

Jede Verknüpfung zwischen den Neuronen, sowie die Neuronen selbst haben ein Bias, welches eine Gewichtung des entsprechenden Objektes darstellt.
Damit das Netz eine Aufgabe erfüllen kann, wird es trainiert. Es kann dabei lernen, indem es die Verbindungen zwischen den Neuronen verändert, das Bias anpasst oder, in bestimmten Netzen, neue Neuronen erstellt. Im Folgenden wird dies anhand eines Trainingsprozesses betrachtet, welcher sich überwachtes Lernen nennt: Hierbei werden zunächst Testdaten in das entsprechende Netz gegeben und das Ergebnis mit der gegebenen Lösung verglichen. Auf dieser Grundlage werden die Bias des Netzes angepasst. Es ist hierbei nicht vorgesehen, den grundlegenden Aufbau des Netzes zu verändern.

Mathematisch lässt sich ein Netz als Gleichung darstellen:

$y_1 + b_1 * y_2 + b_2 * y_3 + ... + b_{n-1} * y_n = Output$

y = Neuron
b = Bias

Dabei gilt für alle Neuronen: $y_n = d_n * f(x)$

d = Gewichtung der Zelle
f(x) = Aktivierungsfunktion der Zelle

Jeder Wert der Gleichung stellt dabei eine Matrix mit den einzelnen Werten der jeweiligen Schicht dar. Die Ausgabe wird mit dem erwarteten Ergebnis verglichen. Aus der entsprechenden Abweichung lassen sich dabei die sogenannten Kosten der Funktion berechnen.

$Kosten = Lösung - Output$

Da die Kosten den Abstand zum gewünschten Ergebnis darstellen ist es Ziel dieser Optimierung, die Kosten zu minimieren. Dafür können Gewichtungen und Bias angepasst werden, wodurch ein Optimierungsproblem mit 2^{n-1} Unbekannten entsteht. Diese Unbekannten werden nun mit jedem Training besser an das gewünschte Ziel angepasst.

Für diesen Lernprozess wird ein Algorithmus namens Backpropagation genutzt. Dabei wird, vom Output aus, jedes Neuron einer jeden Schicht mit einer Gewichtung bewertet. Diese beschreibt dabei die Richtung, in die sich der Output entwickeln soll. Diese Richtung wird, je nach Abstand zum Ziel, verstärkt oder verringert, so dass sich das System optimal anpasst und nicht übers Ziel hinaus schwingt.

Die Anpassung kann dabei offline oder online geschehen. Bei der online Anpassung werden mehrere Beispiele gleichzeitig in ein Netz gegeben und das Ergebnis aufkumuliert. Somit erlernt das System alle Trainingsbeispiele gleichzeitig. Beim offline Lernen werden alle Trainingsbeispiele nacheinander genutzt.

Zu Beginn werden alle Gewichtungen und Bias zufällig gesetzt.

Wird dieses Training nun über alle Testdaten hinweg durchgeführt, so erhält man eine Funktion bzw. ein Modell, welches die Testdaten möglichst gut lösen kann. Wie sich dieses Netz optimiert ist größtenteils vom Zufall bestimmt, die Neuronen können jedoch mit unterschiedlichsten Zelltypen und Aktivierungsfunktionen angepasst werden, so dass sie das gewünschte Ziel bestmöglich erreichen können.

Ein Problem dieses Ansatzes ist, dass nicht unbedingt das perfekte Minimum erreicht wird. Es kann sich immer auch um ein lokales Minimum handeln.

Ein allgemeines Problem dieser Lernmethode ist das Overfitting. Dabei spezialisiert sich das Netz zu stark auf die Trainingsdaten, wodurch das Gelernte nicht mehr allgemein genutzt werden kann.

Um die Optimierung des Netzes weiter zu unterstützen, können bestimmte Optimierungsalgorithmen eingesetzt werden. Diese nutzen dabei verschiedene Methoden, um den Trainingserfolg zu vereinfachen. Ein Beispiel hierfür wäre Adam.

Adam Adam wurde von Diederik P. Kingma der University of Amsterdam und Jimmy Lei Ba der University Toronto entwickelt. In ihrem Paper [7] bezeichnen sie diesen Algorithmus als sehr effizient, skalierbar und gut für Probleme geeignet, in welchen viele Parameter genutzt werden. Dabei soll Adam wenig Speicher verbrauchen. Auf Grundlage dieser Eigenschaften wurde Adam im Rahmen dieses Projektes für einige Tests eingesetzt.

CNN - Convolutional Neural Network Ein Convolutional Neural Network ist eine Netzarchitektur, welches entsprechend seines Entwicklungszieles besonders gut für die Bilderkennung verwendet werden kann.

Dabei werden Bilder als Matrizen ins Netz geladen und analysiert. Hierfür werden Convolutional Layer eingesetzt, welche einen Kernel mit vorher festgelegter Größe über die Matrix schieben und die Werte per Skalarprodukt miteinander verrechnen. Durch das Skalarprodukt reagieren die einzelnen Zellen auf ihre Nachbarn und es wird schrittweise eine neue Matrix erstellt.

Beim Training des Netzes wird hier der Filter festgelegt, welcher zum Schluss über die Matrix wandert. Die meisten CNN besitzen zur Vereinfachung des Problems noch Pooling Layer. Diese fassen Schichten des Netzes zusammen und entfernen somit überflüssige Informationen.

2.1.2 Details des Dokumentenformates PDF

PDF ist ein plattformunabhängiges Dateiformat und wurden 1993 von Adobe Inc. entwickelt und veröffentlicht. Es ist dabei Teil des offenen Standards nach ISO 32.000.[8]

In PDF Dokumenten werden alle Dateien als Objekte abgespeichert. Diesen können Eigenschaften zugewiesen werden, zum Beispiel das Sperren von gewissen Operationen wie dem Drucken. Objekte können dabei auch Formulare oder JavaScript Code enthalten.
Das PDF-Format bestehen aus vier Bestandteilen:

1. Im Header wird die Versionsnummer des PDF gespeichert.
2. Ihm folgt der Body, welcher alle Objekte des Dokuments enthält.
3. Auf diese Objekte wird in der folgenden Xref Sektion verwiesen und es wird eine Reihenfolge festgelegt.
4. Das Dokument endet mit dem Trailer, welcher das Root Objekt des Dokumentes darstellt. Hier wird wiederum auf die Xref Sektion verwiesen.

Dabei ist zu beachten, dass PDFs nicht von oben nach unten, sondern von unten nach oben geladen und interpretiert werden.

Moderne PDF Reader können zusätzlich Notizen in PDFs einfügen. Dabei werden neue Objekte an das bestehende PDF angesetzt und neue Trailer und Xref Sektionen hinzugefügt. Die alten bleiben dabei bestehen, nur die Verweise werden geändert. Dabei können Objekte auch neu definiert oder ersetzt werden. Dies nennt man inkrementelles Update. Über diesen Weg lassen sich PDFs auch signieren. Dabei wird ein neues Objekt eingefügt, welches die Prüfsumme enthält. Die Prüfsumme wird dabei aus dem ganzen Dokument gebildet. Es können jedoch auch nach der Signierung inkrementelle Updates durchgeführt werden, die Signatur wird dabei nicht beeinträchtigt.

Texte innerhalb eines PDF können auch verschlüsselt werden. Dabei werden jedoch nur die Textbausteine verschlüsselt. Die entsprechenden Objekte können immer noch neu angeordnet oder verändert werden.

2.2 Zusammenstellung der Toolumgebung

Die Auswahl der Tools der Entwicklungsumgebung erfolgte auf Basis entsprechender Anforderungen:

Anforderung	**gewählt**	**Alternative**	**Begründung**
Tool zur Analyse größerer Datenmengen	Jupyter Notebook	RStudio	Einfache Analyse größerer Datenbestände mit der Möglichkeit, Python Libraries einzubinden.
Tägliches Deployment	Git	/	Einfache Einbindung in die Plattform Github
Planung der einzelnen Sprints	Jira	Trello	Marktführer für entsprechende Planungstools
Programmiersprache zum schnellen Erstellen einfacher Skripte	Python	JavaScript	Einfache Syntax mit der Möglichkeit, zusätzliche Libraries leicht einzubinden
Programmiersprache zur Erstellung eines Prototyps	Swift	Java, C	Persönliche Präferenz des Entwicklers, hohe Performance.
Tool für die Bildverarbeitung	Framework OpenCV	Gimp, Photoshop	Einfache Nutzung von Bildbearbeitungs-Algorithmen
Tool für das Erstellen von neuronalen Netzen	Framework Tensorflow	Framework Pytorch	Tensorflow bietet die Möglichkeit zum Bau hoch komplexer Machine Learning Modelle zum Lösen von realen Problemen. Dabei kann eine einfache Syntax genutzt werden, ohne die Umsetzungsmöglichkeit einzuschränken.
Tool zum Erstellen einer Schnittstelle zwischen einem Server und einer App für das iOS System	Framework Flask	Django	Einfache Erstellung einer API
IDE für die Entwicklung	Visual Studio Code	Vim, Emacs	Vom Entwickler bevorzugte Entwicklungsumgebung, welche sich durch eine hohe Übersichtlichkeit und der Möglichkeit der individuellen Anpassung auszeichnet.
IDE für die Entwicklung der App für das iOS System	XCode	AppCode	Auf iOS zugeschnittene Entwicklungsumgebung mit zusätzlichen Features

2.3 Entwicklungsumgebung

2.3.1 Jupyter Notebook

Jupyter Notebook ist eine web-basierter IDE. Im Unterschied zu normalen Texteditoren wie Emacs oder Vim, kann der Code dabei wie bei einem Dokument verwaltet werden, wobei Code in Blöcke getrennt werden kann. Dadurch können Zwischenergebnisse direkt eingesehen, Blöcke mehrfach berechnet und die Zwischenergebnisse analysiert werden, wodurch die Analyse von größeren Datenbeständen stark vereinfacht wird. Dabei kann Code in Python, C oder Swift genutzt werden. [9]

Als Entwicklungsumgebung für Python wurde der Dienst COLAB von Google genutzt. Hier werden einzelne virtuelle Umgebungen erstellt, welche je ein Notebook beinhalten. Die virtuellen Umgebungen können dabei je nach Bedarf mehr oder weniger Rechenleistung bekommen und es können auch eigene Programme neben dem Notebook installiert werden. Diese Umgebung hat dabei Zugriff auf leistungsstarke Graphik und Tensor Prozessing Units (TPU und GPU) welche für den Einsatz mit neuronalen Netzen optimiert sind. [10]

2.4 Projektsteuerung

2.4.1 Jira

Jira ist eine Anwendung der Firma Attlassian. Häufig in Form einer Webanwendung eingesetzt, kann es für das operative Projektmanagment genutzt werden. [11]

Hierzu enthält das Tool ein eigenes Ticketsystem welches mit Dashboards, Statistiken und einigen Möglichkeiten der Automation an die eigenen Bedürfnisse angepasst werden kann.
Die Tickets selbst können dabei durch Dokumentation, weiteren Verknüpfungen oder neuen Eigenschaften ergänzt werden.

2.5 Genutzte Programmiersprachen

2.5.1 Python

Python ist eine interpretierte Programmiersprache, und wird meistens als Skriptsprache verwendet. Sie hat eine sehr leicht lesbare Syntax und kann sehr einfach Fremdcode laden, womit das Schreiben eines Skriptes stark vereinfacht und beschleunigt wird.
Diese Faktoren machen Python zur optimalen Sprache zum Testen von Ideen, jedoch ist die Sprache nur bedingt für größere Projekte ausgelegt. [12]

2.5.2 Swift

Swift ist eine multiparadigmatische Compilersprache, welche auf C basiert. Aufgrund dessen ist sie bei größeren Projekten deutlich performanter als Python, kann jedoch trotzdem Python Librarys einbinden und nutzen. Es existiert außerdem ein breites Spektrum an Libraries, welches die Entwicklung von Lösungen stark vereinfacht. [13]

2.6 Eingesetzte Frameworks

Da bereits bekannte Techniken nicht von Grund auf neu entwickelt werden sollten, kamen neben eigenständigen Tools verschiedene Librarys und Frameworks zum Einsatz, welche nach verschiedenen Kriterien gewählt wurden.
Dabei müssen sie entweder den neusten Stand der Technik darstellen oder sich zumindest in ihrem Spezialgebiet bewährt haben.

2.6.1 Tensorflow

Tensorflow ist ein Framework für maschinelles Lernen und wird zur Entwicklung von neuronalen Netzen genutzt. Es ist in C++ geschrieben und kann in verschiedensten Sprachen integriert werden. Entwickelt von Google, wird das Framework dort in Forschung und Produktivbetrieb in zahlreichen Projekten verwendet. [14]

2.6.2 Keras

Keras ist eine Deep Learning Library. Sie wurde zunächst von François Chollet initiiert und später als Kommunityprojekt weitergeführt. Sie bietet dabei viele nützliche Funktionen um ein neuronales Netz zu entwickeln. Seit Tensorflow 1.4 ist Keras auch Teil des Tensorflow Frameworks und vereinfacht dessen Nutzung. [15]

2.6.3 OpenCV

OpenCV ist eine von Willow Garage entwickelte Library, welche inzwischen von Intel gepflegt wird. Geschrieben in Java, Python, C und C++ stellt die Library dabei Algorithmen für die Bildverarbeitung und Computer Vision zur Verfügung. [16]

2.6.4 Flask

Flask ist ein in Python geschriebenes Webframework. Entwickelt von Armin Ronacher bietet es Funktionen zur Erstellung von Schnittstellen zwischen Webservern und Webframeworks. [17]

2.7 Test- und Trainingsdaten

Um die bereits bestehenden Tools, Programme und Systeme zu testen oder neu einzurichten werden Daten benötigt. Diese müssen zur Verwendung mit gewünschten Ergebnissen versehen werden, damit entsprechende Resultate verglichen werden können. Bei der Nutzung im Bereich der neuronalen Netze wird das gewünschte Ergebnis als Teaching Input bezeichnet.

Für diese Arbeit wurden unterschiedliche Dokumente benötigt. Hierfür stehen einige Datenpools zu Verfügung, welche sich in Umfang und Aufbau unterscheiden. Dabei wurden Datenblätter von Herstellern von Raumfahrtkomponenten für DLR und ESA sowie ein Datensatz der IBM Australia namens PubLayNet genutzt, welcher aus mehr als 300.000 Dokumente besteht. [18]

Als Beispiel für den Teaching Input wurde hier die Position der Tabellen genutzt, welche beim PubLayNet bereits von IBM hinzugefügt wurden, weswegen es sofort eingesetzt werden kann.

3 Evaluierung ausgewählter Lösungsansätze

Eine Umsetzung der Interpretation codierter PDF Dokumente wurde nicht weiter verfolgt, da die hier zu entwickelnde Methodik auch mit Dokumenten funktionieren soll, welche in reinen Bildformaten vorliegen. Daher werden alle Dokumente bereits vor Beginn des Prozesses in standardisierte Bilddateien umgewandelt, was zu einer universellen Einsetzbarkeit, unabhängig vom Ursprungsformat, führt.

3.1 Tabellenerkennung

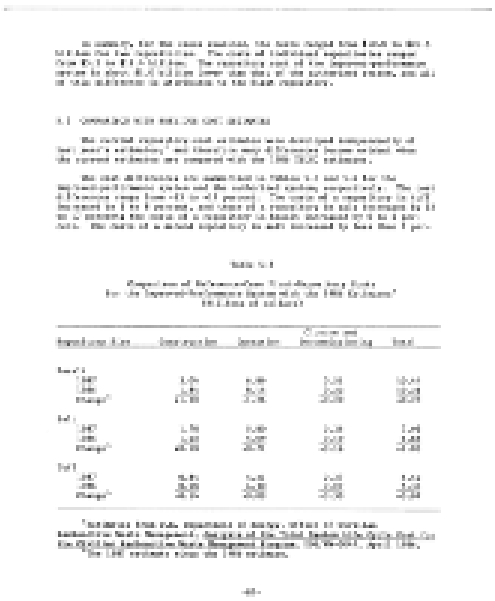
Das Abbild eines Dokumentes kann in mehrere Bestandteile bzw. Objekte wie Textbausteine, Überschriften oder Tabellen aufgeteilt werden. Die zu einem Objekt gehörenden Elemente zeichnen sich durch Gemeinsamkeiten, wie gleiche Spaltenbreite, gleiche Berandung, regelmäßig wiederkehrende Elemente o.Ä. aus.
Diese sind innerhalb eines Objektes konsistent und lassen so eine Abgrenzung von anderen Objekten zu. Kennt man die inhaltliche Bedeutung der Objekte nicht, ist diese Struktur die einzige Möglichkeit, das Dokument in Objekte zu zerlegen.

3.1.1 Klassische Bildverarbeitung

Die klassische Bildverarbeitung kennt viele Wege, um einheitliche Strukturen – wie z.B. Muster – in Bildern zu erkennen. Dabei werden unterschiedliche Algorithmen, wie der Canny-Algorithmus, eingesetzt, um einzelne Strukturen des Bildes zu verstärken oder Kanten zu erkennen.

Die Library OpenCV bietet diese Funktionen. Um die Funktionalität zu testen wurden mehrere Beispieldokumente mit verschiedenen Algorithmen und Filtern, wie z.B. der Kantendetektion bearbeitet.

Es zeigte sich jedoch, dass die einzelnen Zeichen, also Buchstaben, Zahlen oder zugehörige Satzzeichen, in ihrer Struktur nicht deutlich genug sind. Zwar werden die einzelnen Zeilen in ihrer Summe noch erkannt, die einzelnen Zellen der Tabelle sind im Beispiel jedoch uneindeutig identifiziert, so dass eine Zuordnung zur gesuchten Tabelle nicht eindeutig möglich ist.



(a) Beispieldokument

(b) Ergebnis der Kantenerkennung in Y-Richtung

Ein weiteres Problem stellt die Größe der Dokumente dar.
Da ein Dokument der Größe DIN A4 bei einer Auflösung von 300 dpi bereits aus 2.480 * 3.508 Pixeln besteht, benötigt ein Algorithmus entsprechend Zeit, um diese zu verarbeiten. Ein erster Test wurde nach acht Stunden abgebrochen, wobei in dieser Zeit nur 800 Dokumente verarbeitet werden konnten. Daher wurde in weiteren Tests mit jeweils zehn Beispieldokumenten gearbeitet.

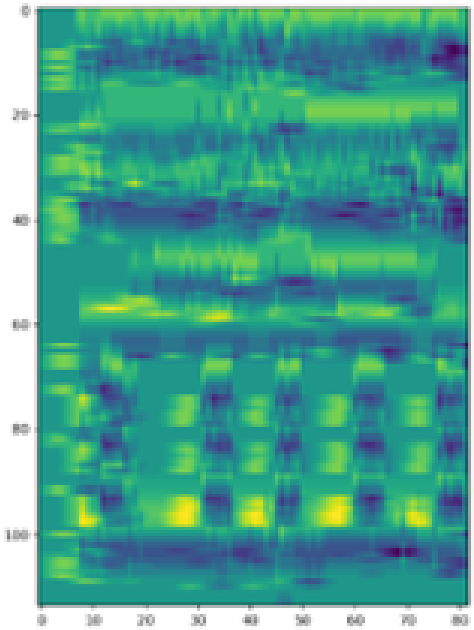
Um die Durchlaufzeiten zu verringern wurden die Dokumente zunächst unschärfer gerechnet und komprimiert: Hierzu wird eine festzulegende Anzahl benachbarter Pixel zusammengefasst und auf einen Durchschnittswert gesetzt. Dadurch sind die einzelnen Buchstaben nicht mehr erkennbar, was für die Erkennung von Tabellen auch nicht notwendig ist. Das Ergebnis muss am Ende wieder auf das Ausgangsformat zurückskaliert werden.

Für die Erkennung der tabellentypischen Strukturen wurde ein Korrelationsfilter eingesetzt: Da Tabellen eine einheitliche Struktur (regelmäßiger Wechsel von hellen und dunklen Bereichen) haben, kann mit diesem Filter direkt nach ihnen gesucht werden. Der Grauwertverlauf der Struktur eines unscharf gerechneten Abbildes einer Tabelle wird dabei näherungsweise als Sinuskurve angenommen.

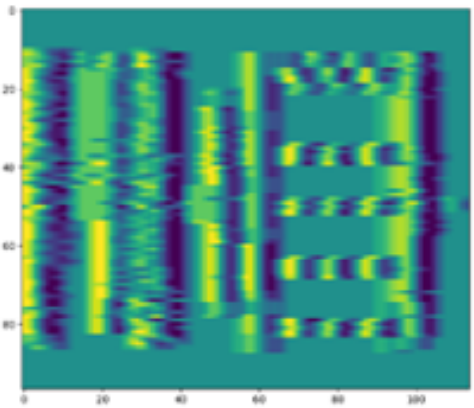
Der gleiche angenommene Grauwertverlauf wird als Korrelationsfilter genutzt:

$f(x) = \sin x/b * 2\pi$

Dabei repräsentiert b die Länge des Filters, bzw. die Länge der gesuchten Spalte. Als Längen werden dabei nacheinander Bruchteile der Seitenbreite genutzt, z.B. die Hälfte, ein Drittel oder ein Viertel. Der entsprechenden Filter wird dann in X- und Y-Richtung über das Bild geschoben und ein summierter Zwischenwert von Filter und betrachteter Spalte oder Zeile berechnet. Daraus ergeben sich zwei Heatmaps, welche wieder kombiniert wurden.



(c) Ergebnis des Filters in X-Richtung



(d) Ergebnis des Filters in Y-Richtung

Die Stellen, an denen beide Heatmaps einen Ausschlag markieren, wird nun in einer Kombination ebenfalls markiert, siehe Abbildung 3.

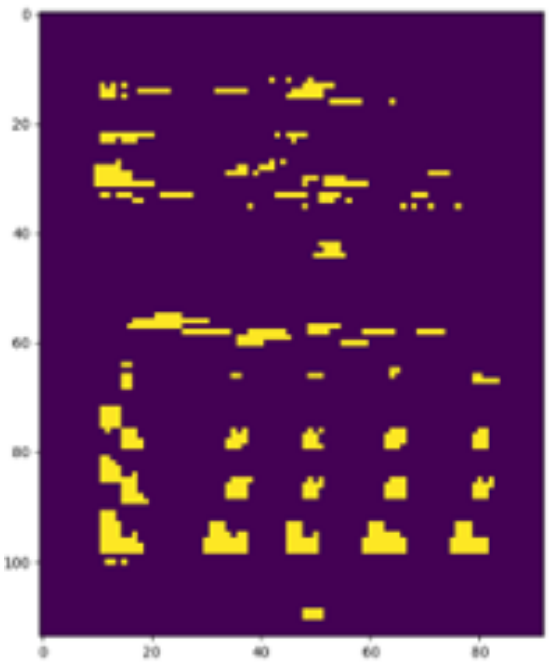


Abbildung 3: Ergebnis des Filters - Kombination

Im Ergebnis ist die Tabelle noch deutlich erkennbar. Leider weisen jedoch auch Teile des Textes die gleichen Muster wie eine Tabelle auf, weswegen sie im Bild auch vertreten sind. Es kommt normalerweise nicht zu einer vollkommenen Markierung der Tabelle und einer vollständigen Auslöschung des Textes. Die Ursachen sind u.A., dass die Tabelle normalerweise nicht ganz regelmäßig ist – das Mapping mit der Sinusfunktion also nicht 100% erfolgreich ist – und dass sich auch im Text zufälligerweise Frequenzanteile der Prüffrequenz befinden. Zudem dürfte die Frequenz des Testfilters nur in kleinen Schritten erhöht werden, um das Optimum nicht zu verpassen. Dieses würde aber zu unverhältnismäßig vielen Rechenoperationen führen.

Schlussendlich ist das Ergebnis nicht optimal, es könnte jedoch für weitere Analysen genutzt werden, da es die Position der Tabelle bereits leichter erkennen lässt als das Ausgangsbild. Dafür wurden drei Ansätze identifiziert:

Verbessern der Filter Der erste Ansatz greift beim Erstellen der Filter. Dabei wird ein neuronales Netz trainiert, welches die Frequenzanalyse der unterschiedlichen Filter als Input nimmt, und dann auf Basis der Änderungen entscheidet, welcher Filter am besten geeignet ist. Das Netz muss dabei mehrere Bilder als Input nutzen können und eine Zahl ausgeben, welche entweder direkt die Länge der Filter beschreibt oder auf einen der vorgegebenen Filter verweist.

Analyse der Heatmap In einem zweiten Ansatz wird das entsprechende Bild zunächst mittels Frequenzanalyse in eine Heatmap umgewandelt. Auf Basis dieser wird dann ein neuronales Netz trainiert, welches auf Basis der Heatmap die Tabelle erkennen kann.

Eine Kombination Der dritte Ansatz setzt auf den gleichen Daten auf wie der erste, soll jedoch direkt die Position der Tabelle ausgeben, anstatt die Filter zu optimieren. Das Netz soll also die Änderungen der Kombinationen analysieren und so erkennen, wo sich die Tabelle befindet.

3.1.2 Einsatz von neuronalen Netzen

Aufgrund der zeitlichen Beschränkung wurde nur der zweite Ansatz umgesetzt.

Für das Training werden dabei Test- und Trainingsdaten benötigt. Um die Machbarkeit des Prinzips festzustellen wurden zunächst synthetische Daten erzeugt. Dabei wurde das gewünschte Muster mit einer zufälligen Sinuskurve erstellt und der Hintergrund mit einem Rauschen belegt. Das Netz sollte dabei lernen, nur die Tabelle zu erkennen und das Rauschen nicht zu beachten. Als Aufbau wurde hier ein CNN verwendet.

Die Testdaten sind dabei zunächst deutlich kleiner im Umfang, als die originalen Dokumente.

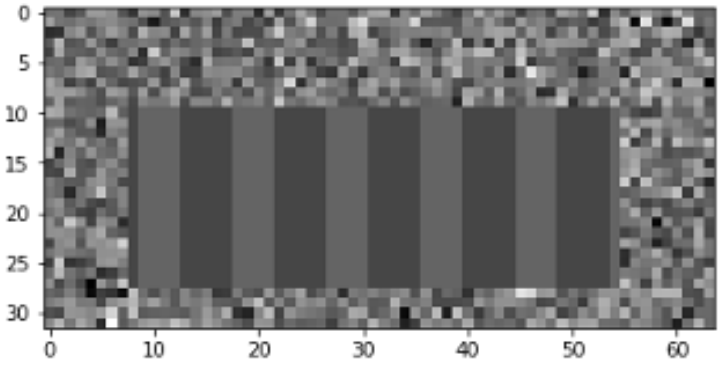


Abbildung 4: Beispiel der synthetischen Datenerzeugung

Zunächst wurde das Netz trainiert, um eine Maske zu erstellen, welche die Zielfläche markieren sollte. Die ersten Tests zeigten jedoch, dass das Netz bei den gegebenen Beispielen als Ergebnis nur eine einfarbige Fläche ausgibt. Das hier gezeigte Problem wurde zunächst als Overfitting eingestuft. Gegen diese Einschätzung spricht jedoch, dass das bestehende Netz die gegebenen Beispiele nie gelöst hat, sondern nur eine Minimallösung für das Problem erbrachte.

Um diesen Effekt aufzuheben wurden mehrere Tests durchgeführt. Dabei wurden verschiedene Variablen wie die Lernrate oder die Anzahl an zusätzlichen Schichten angepasst, was jedoch nur bedingt zu einer Verbesserung des Lernverhaltens führte. Gleichzeitig zeigte sich in allen Modellen ein gleicher Trainingsverlauf. Dabei stieg die Genauigkeit während des Trainings an, jedoch gleichzeitig auch der Loss-Wert. Dieser beschreibt das Ergebnis der Kostenfunktion und sollte im Verlauf der Optimierung eigentlich sinken.

Das Netz wurde daher so umgebaut, dass nun direkt Zielkoordinaten ausgegeben werden. Es wurde außerdem der grundlegende Aufbau einfacher umgesetzt. Im neuen Netz werden die Bilder zunächst in eine eindimensionale Form gebracht und dann durch zwei Schichten an Neuronen geschleust. Dieser Ansatz brachte deutlich bessere Ergebnisse.

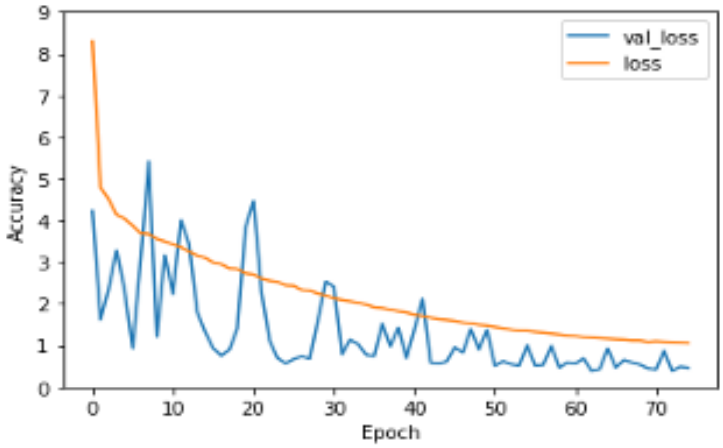


Abbildung 5: Trainingsverlauf

Im folgenden Beispiel wurde ein ähnliches Dokument wie in Abbildung 4 getestet. Dabei stellt die gelbe Fläche den gesuchten Bereich dar und das Ergebnis des Netzes wird in Rot dargestellt.

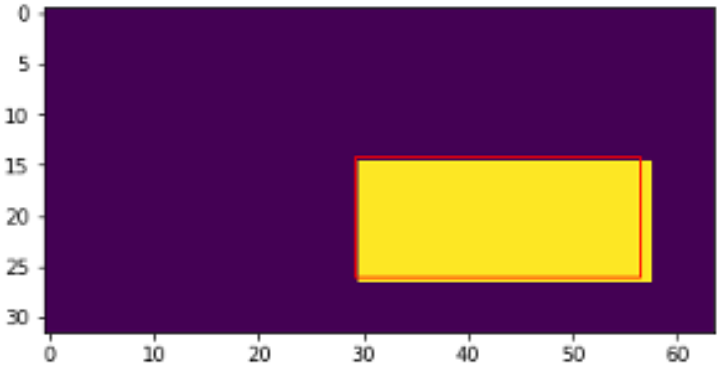


Abbildung 6: Prognose des neuronalen Netzes

Das Ergebnis ist nicht perfekt, zeigt jedoch, dass der Ansatz funktioniert.

Beim Training stellten sich jedoch auch einige Einschränkungen heraus: Für eine Nutzung mit den Originaldokumenten muss das Netz in seinem Umfang deutlich vergrößert werden. Dadurch müssen deutlich mehr Dokumente für das Training genutzt werden, um gleiche Ergebnisse zu erzielen. Hierdurch erhöht sich der zeitliche Aufwand den das Netz zum Trainieren benötigt massiv.

Das Netz ist außerdem darauf ausgelegt, eine Tabelle zu erkennen. Sollten mehrere Tabellen in einem Dokument vorhanden sein, funktioniert das Netz nicht mehr wie vorgesehen.

Ein weiteres Problem ist die Zeit welche das System benötigt, um ein Dokument zu verarbeiten. Durch die vorherige Berechnung, wie die Unschärfe und Komprimierung, benötigt das Netz häufig bis zu einer Minute pro Dokument.

Um diese Probleme zu lösen wurde das Netz neu geplant. Dabei wurde unter anderem die Aufgabe des Netzes neu bedacht.

3.1.3 Bildsegmentierung

Aufbauend auf den Erfolgen des DeepDeSRT wurde das Netz neu entworfen. Hierfür wird das Originalbild zunächst in Bildausschnitte aufgeteilt. Das Netz soll diese nun klassifizieren und diese so als Tabelle, Text oder ähnlichem erkennen.

Um die Bildausschnitte zu wählen gibt es verschiedene Methoden: Zum einen kann das gesamte Bild in rein geometrisch gewählte Ausschnitte unterteilt werden, um alles zu klassifizieren. Durch die Anzahl an möglichen Ausschnitten dauert diese Methode jedoch entsprechend lang und es müssen mehrere Klassifizierungen pro Bild durchgeführt werden.

Dies kann verbessert werden, indem man zunächst Bereiche des Bildes mit ähnlichen Konturen, Farben oder Intensität zusammenfasst und diese dann klassifiziert. Dies beschleunigt den Vorgang deutlich. Einen neuen, schnelleren Ansatz bietet das YOLO Netz.

3.1.4 YOLO - You only look once

YOLO wurde von Joseph Redmon und Ali Farhadi der University of Washington entwickelt und ist ein völlig neuer Ansatz der Objekterkennung. Dabei wird das Bild nicht mehr in einzelne Ausschnitte zerlegt, sondern durch ein einziges Netz in einem Durchgang verarbeitet. Das Netz unterteilt das Bild dann selbst in Boxen, welches es klassifiziert. Durch diese Methode wird zum einen nur noch ein Netz pro Bild und Durchgang benötigt, zum anderen ist es laut Aussage der Entwickler tausendmal schneller als die herkömmlichen Methoden. [19]

Auf Grund dieser Prozessbeschleunigung wurde YOLO als Netz für diese Arbeit gewählt. Das Netz wurde dabei mit dem PubLayNet Trainingsset trainiert.

Anders als beim DeepDeSRT wurde die Implementierung der Methode von mir nicht einzig auf Tabellen ausgelegt, sondern auf alle möglichen Anwendungsfälle.

Im Gegensatz zu den ersten Ansätzen dieser Arbeite wurde außerdem auf die Komprimierung der Bilder verzichtet, wodurch nun Dokumente in ihrer ursprünglichen Größe verarbeitet werden.

Es wurden folgende Klassifizierungen identifiziert:

1. Text
2. Überschrift
3. Liste
4. Tabelle
5. Grafik

3.1.5 Ergebnis

Nach dem Training wurden mehrere Tests mit Beispieldokumenten durchgeführt. Dabei zeigte sich, dass das Netz alle Objekte erkennt, jedoch die zugehörigen Bereiche nicht perfekt identifizieren kann.

Abbildung 7: Beispiel Prognose des YOLO-Netzes

Bei der Nutzung des Netzes fielen noch weitere Effekte auf:
Füttert man das Netz mit Dokumenten größerer Pixeldichte, so erkennt das Netz keine Objekte. Eine Theorie hierfür wäre, dass das Netz pixelgenau arbeitet. Die entsprechen Objekte sind bei einem Dokument von größere Pixeldichte jedoch deutlich zu groß, um erkannt zu werden. Dieser Umstand kann jedoch umgangen werden, indem das Dokument auf die Größe der Testdokumente verkleinert wird.

Die Abweichung der Boxen könnte durch einen Filter korrigiert werden. Dieser würde die Kanten der einzelnen Objekte erkennen und so die Boxen anpassen. Aufgrund der zeitlichen Einschränkung der Thesis wurde diese Optimierung nicht durchgeführt.

Nachdem die Tabellen nun gefunden und ihre Positionen ausgegeben werden, wird ein weiterer Mechanismus benötigt, um die Felder innerhalb der Tabellen zu finden. Da Tabellen in ihrem Aussehen und Aufbau sehr unterschiedlich sind, reicht ein klassischer Algorithmus hier nicht aus.

Als Lösung für dieses Problem wurde zunächst geplant das YOLO Netz erneut einzusetzen, diesmal allerdings darauf trainiert, Blöcke von Zeichen zu finden. Eine Alternative dazu ist die Schrifterkennung.

3.2 OCR

Die Schrifterkennung soll aus der gefundenen Tabelle die Wörter, Zahlen und Buchstaben extrahieren. Die entsprechende Technik wurde bereits 1958 zum ersten Mal vorgestellt. Dabei erstellten Frank Rosenblatt und Charles Wightman in Zusammenarbeit mit dem MIT und dem United States Office of Naval Research das Mark 1 Perceptron. Damals noch eher in Form eines physischen Computers als einer Software, konnte dieses System bereits einfache Ziffern in Bildern der Maße 20 x 20 Pixeln erkennen. Das zugrundeliegende Verfahren wurde seitdem immer weiter verbessert und ist heute ein gut erforschtes Forschungsgebiet mit vielen verschiedenen Lösungen. [20]

Für dieses Projekt wurden mehrere freie Tools für die Schrifterkennung getestet. Besonders bewährte sich das Tool Tesseract [21] und die Vision Library des IOS Systems [22]. Beide sind Offline-Lösungen.

Tesseract ist eine von Google entwickelte freie Texterkennungssoftware. Sie kann Textzeichen und Textzeilen erkennen, sowie Layout-Analysen durchführen. Dabei beherrscht das Programm mehr als 100 Sprachen, sowie mehrere verschiedene Schriften wie Chinesisch, Arabisch oder Griechisch. Nach Aussage eines Tests der Zeitschrift c't nutzt Google das Tool selbst für eigene Produkte wie Google Books. Dabei soll das Tool an ähnliche Erkennungsraten und Verarbeitungsgeschwindigkeiten wie kostenpflichtige Tools herankommen. Auf welcher Grundlage diese Aussagen getroffen wurden oder auf welche Tools sich bezogen wird, wird in diesem Test leider nicht genannt. Für die Nutzung in diesem Projekt sollte die Leistung dabei jedoch trotzdem ausreichend sein. [23]

In ersten Tests zeigte sich aber auch Einschränkungen des Tools. Zeichen wie das Dollar Zeichen ($) werden beispielsweise häufig als Ziffer 5 oder gar nicht erkannt.

Das Vision Framework des iOS-Systems bietet verschiedene Funktionen zur Bildanalyse, unter anderem eine eigene Texterkennung. Die Implementierung ist dabei im iOS System standardmäßig vorhanden und für den Einsatz optimiert.

Beide Tools können neben den erkannten Zeichen auch deren Positionen ausgeben. Mit Hilfe eines einfachen Algorithmus kann daraus wieder die Ursprungstabelle erstellt werden.

Im Bereich der Schrifterkennung gibt es auch einige online Tools, wie z.b. Amazon Rekognition oder Googles OCR. Diese sind in der Regel deutlich treffsicherer als freie Tools, haben jedoch durch die zum Anbieter ausgelagerte Datenverarbeitung auch entscheidende Nachteile.

3.2.1 Rechtliche und technische Betrachtung von Online-Lösungen

Die zuletzt betrachteten Tools sind Online Services. Dies bedeutet, dass alle zu verarbeitenden Dokumente zunächst zu einem Server hochgeladen werden müssen und auf den Servern des Anbieters verarbeitet werden. Dies bedeutet, dass alle Tools auf eine Online Nutzung ausgelegt werden müssen und nur mit Verbindung zum Server funktionieren. Dies schränkt eine Nutzung auf mobilen Geräten oder in abgeschotteten Intranets ein.

Zusätzlich müssen bei der Verarbeitung von Daten in Deutschland, abhängig von Art und Schutzbedarf, gewisse Regelungen zur Informationssicherheit und zum Datenschutz beachtet werden. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) hat hierfür Standards erstellt, mit denen eine Risikoanalyse durchgeführt werden kann. Anforderungen und Bewertungen werden dabei in Reihe 27.00-x der ISO Normen beschrieben. [24]

In Deutschland gilt zudem die Datenschutz-Grundverordnung (kurz DSGVO). Diese Europäische Richtlinie vereinheitlicht die Regelungen zur Verarbeitung von personenbezogenen Daten in Europa und beschreibt unter anderem die Pflichten des Verantwortlichen. Dabei gilt nach Artikel 25 DSGVO, dass derjenige, der die Datenverarbeitung verantwortet, den Datenschutz entsprechend dem Stand der Technik umsetzen muss. [25]

Bei der hier angesprochenen Nutzung von Cloud-Diensten werden diese Regelungen noch erweitert. Dazu fordert Artikel 44 DSGVO, dass die Übertragung von Daten an ein Drittland oder eine internationale Organisation nur zulässig ist, wenn sichergestellt werden kann, dass die Anforderungen an den Datenschutz eingehalten werden. [26]

Das BSI hat hierfür den „Mindeststandart des BSI zur Nutzung externer Cloud-Dienste“ herausgegeben.[27] Dieser richtet sich nach § 8 Absatz 1 Satz 1 BSIG (Gesetzes über das Bundesamt für Sicherheit in der Informationstechnik). Dabei müssen alle Daten in vorgegebene Kategorien einsortiert werden:

1. Kategorie 1: Privat- und Dienstgeheimnisse
2. Kategorie 2: personenbezogene Daten
3. Kategorie 3: Verschlusssachen gemäß allgemeiner Verwaltungsvorschrift
4. Kategorie 4: sonstige Daten, welche nicht in die vorherigen Kategorien einsortiert werden konnten.

Je nach Kategorisierung der Daten müssen die Cloud-Dienste dann die Vorgaben des Anforderungskatalog Cloud Computing des BSI (kurz C5) erfüllen. Darin werden Auswahlprozesse für den Cloudanbieter, sowie empfohlene vertragliche Regelungen festgelegt. [28]

Nutzt man nun die Dienste von Amazon oder Google, so werden die übertragenen Daten dort gespeichert und zur Verbesserung der Dienste genutzt. Da es sich bei beiden um Firmen mit Sitz in den USA handelt, gelten dabei auch deren Gesetzte. In diesem Fall gilt z.B. der CLOUD Act, der „Clarifying Lawful Overseas Use of Data Act“, unter welchem Firmen mit Sitz in den USA US-Behörden Zugriff auf gespeicherte Daten gewährleisten müssen, selbst wenn die Speicherung nicht in den USA erfolgt. Der Datenschutz kann hier also nicht ohne weitere Maßnahmen gewährleistet werden. [29]

3.3 Kontextdetektion

Um die Nutzung der Daten zu vereinfachen wurde eine Kontext-Detektion entwickelt. Diese soll die Art der erkannten Daten erkennen, damit diese leichter von einem weiteren System, wir z.B. Microsoft Excel, verarbeitet werden können.

Dafür wurden acht Datenklassen definiert:

1. Zahlen
2. Text
3. Kommazahl
4. Angaben in Prozent
5. Physikalische Angaben
6. Adressen
7. Datum
8. Mix aus mehreren Klassen

Um diese Daten zu klassifizieren werden zunächst Eigenschaften der Daten identifiziert, z.B. Länge des Datensatzes oder die Anwesenheit bestimmter Zeichen, wie Punkte oder Zahlen.

Auf Basis der Systematik dieser Eigenschaften lässt sich dann ein Entscheidungsbaum erstellen, welcher die Klassifizierung durchführt. Dieser kann von einem Entwickler in relativ kurzer Zeit in Code umgesetzt werden.

Algorithm 1: Beispielumsetzung

```
Data: Eingabetext
Result: Kategorie des Inhalts
if Enthält Zahlen then
    Analyse der nächsten Eigenschaft;
else
    Rückgabe der Kategorie Text;
end if
```

Eine weitere Möglichkeit der Umsetzung ist der Einsatz eines neuronalen Netzes. Dieses würde darauf trainiert werden, die Eigenschaften als Input zu nutzen und daraus eine Klassifizierung zu berechnen. Beide Methoden haben Vor- und Nachteile:

Da ein klassischer Entscheidungsbaum mit Hintergrundwissen des Entwicklers erstellt wird, kann mit sehr wenigen Beispieldaten gearbeitet werden, wohingegen ein neuronale Netz viele Daten benötigt, um daraus zu lernen. Dabei nutzt das Netz allerdings alle Daten und kann auch Verbindungen ziehen, die ein Mensch nicht in Betracht ziehen würde. Das Netz kann außerdem deutlich mehr Beispiele in kürzerer Zeit in Betracht ziehen und durch die interne Gewichtung besser optimieren.

In dieser speziellen Anwendung ist die Generierung von Testdaten kein Problem. Zahlen jeglicher Art lassen sich durch Zufallsfunktionen erstellen und Texte aus Wörtern zusammensetzen. Dabei kann eine Wörterliste des Duden als Grundlage genutzt werden, so dass alle Wörter an sich auch Sinn ergeben.

Adressen können unterschiedlich zusammengesetzt werden, da z.B. Firmen im Detail ein anderes Anschriftenformat haben, als Privatpersonen. Hier können allerdings auch Echtdaten aus dem Telefonbuch oder von Google Maps zum Training verwendet werden.

3.3.1 Abschätzung des zeitlichen Aufwandes

Um die beiden Ansätze vergleichen zu können werden sie im Folgenden tabellarisch gegenübergestellt. Dabei wird von einem Code mit vielen If-Abfragen ausgegangen, welcher pro Abfrage 30 Sekunden an Entwicklungszeit benötigt.

	Klassische Entwicklung	Neuronales Netz
Komplexität	Bei 7 Eigenschaften gibt es $2^7 = 128$ verschiedene Kombinationen	Wird durch das Netz erstellt
Vorbereitung	ca. 60 min zur Erstellung des Ereignisbaums	15 min zur Erstellung des Netzes
Benötigte Zeit der eigentlichen Entwicklung	$128 * 30sec = 3.840sec = 64min$	1 min für 50.000 Datensätze
Präzision	ca. 100%	ca. 93%
Gesamtzeit	124 min	16 min

Der Vergleich wird hierbei stark durch einige Rahmenparameter beeinflusst. So werden hier Kennwerte genutzt, welche vorher von einem Menschen festgelegt wurden, was bereits eine Vorauswahl darstellt. Außerdem ist das Training des neuronalen Netzes auf einem speziell dafür ausgelegten Server ausgeführt worden, wodurch die Trainingszeit natürlich deutlich optimiert ist.

4 Experimentelle prototypische Umsetzung

Um die Möglichkeit der Implementierung als funktionsfähiges System zu testen, wurden alle Lösungen aus Kapitel 3 in einem Prototyp umgesetzt. Dieser wurde in Form einer iPhone-App ausgearbeitet, um einer realen Anwendung möglichst nahe zu kommen.

Da sich einige Methoden nicht ohne größeren Entwicklungsaufwand auf ein mobiles Endgerät portieren lassen, wurde die Anwendung als Client- / Serveranwendung entwickelt, indem neben dem Test-iPhone noch ein MacBook Air als Server verwendet wurde.
In der App kann der Nutzer ein Bild auswählen und analysieren lassen. Das Ergebnis wird dem Nutzer angezeigt.

4.1 Kommunikation zwischen den Geräten

Die Server Applikation wurde in Python geschrieben. Für die Kommunikation zwischen iPhone und Server wurde eine API mit dem Framework Flask erstellt, welche als Schnittstelle fungiert und von der App in Form von http-Requests genutzt werden kann. Erhält die API eine spezifische Nachricht, so führt die Schnittstelle die entsprechende Logik aus und sendet das Ergebnis in Form eines Bildes oder einer Zeichenkette zurück.

4.2 Einlesen von Bildern

Das Modul „Camera Kontroller“ stellt ein Interface zum Einlesen von Bildern zur Verfügung. Dabei wird die iOS Foto API angesprochen, mit der die Kamera gesteuert oder Bilder aus der Foto Library geladen werden können. Dem Nutzer wird dabei ein UI im Stil der klassischen iOS Kamera zur Verfügung gestellt.

4.3 Module

Die Methoden aus Kapitel 3 wurden in einzelnen Modulen bzw. Modulgruppen umgesetzt, welche im folgenden weiter betrachtet werden. Eine Übersicht, sowie die Planungsdiagramme befinden sich in Anhang A.

4.3.1 Tabellenerkennung

Interen Bezeichnung des Moduls	Object Detektion mit OpenCV
Funktionale Beschreibung in Kapitel	3.1
Input	Dokumente in Form einer Bilddatei
Output	Objekt mit Bounding Box und Wahrscheinlichkeit

Die Aufgabe der Tabellenerkennung wird mit Hilfe des Frameworks OpenCV ausgeführt. Dabei kann OpenCV das YOLO Netz mit gegebener Gewichtung nutzen und so das Dokument in Bestandteile kategorisieren. Da die Funktion von OpenCV nicht ohne großen Portierungsaufwand auf mobilen Geräten einsetzbar ist, wurde diese Funktion auf der Serverseite umgesetzt.

4.3.2 OCR

Interen Bezeichnung des Moduls	Schrifterkennungs Logik
Funktionale Beschreibung in Kapitel	3.2
Input	Dokument in Form einer Bilddatei
Output	Struktur aus Zellen mit Informationen wie Position und Inhalt

Das OCR Modul verwendet das Vision Framework zu Erkennung von Schrift in einem gegebenen Bild. Dabei existiert ein Modul „Schrifterkennungs Logik“ welches die API vom Vision Framework anspricht. Vision fast die gefundenen Schriftzeichen in Gruppen zusammen und gibt Daten wie die Position, Höhe und Breite der Gruppen zurück. Diese werden vom Modul in eine interne Struktur umgewandelt und in neue Gruppen zusammengefasst, welche wieder in eine Tabelle umgewandelt werden können.
Da das Vision Framework standardmäßig auf iOS Systemen vorhanden ist, wurde dieses Modul auf App Seite umgesetzt.

4.3.3 Kontextdetektion

Interen Bezeichnung des Moduls	Kontext Detektion
Funktionale Beschreibung in Kapitel	3.3
Input	String mit Inhalt
Output	Zahl, welche die Klasse darstellt.

Die Kontextdetektion kann einen gegebenen String in festgelegte Klassen einordnen. Hierfür wurde von mir sowohl ein klassischer Algorithmus als auch ein neuronales Netz implementiert, welche zwar in ihrem Aufbau sehr unterschiedlich sind, jedoch in ihrer Funktionsweise gleich genutzt werden können. Beide Funktionen wurden serverseitig umgesetzt.

4.3.4 Unterstützende Funktionen

Um die einzelnen Komponenten weiter zu unterstützen, wurden einige Funktionen entwickelt, mit denen ein gegebenes Dokument aufbereitet werden kann. Sollte das Dokument z.B. lediglich Teil des gegebenen Bildes sein, d.h. beispielsweise auf einem Tisch liegen, so kann eine Kantendetektion genutzt werden, um das Bild entsprechend zuzuschneiden und dem Netz damit das Erkennen zu erleichtern. Die entsprechende Methodik wurde serverseitig implementiert.

Um die Aufbereitung weiter zu unterstützen wurden zusätzliche Methoden geplant, mit denen Störfaktoren – wie Falten im Dokument – entfernt werden können. Hierfür wurden zusätzlich erste Tests mit einem Netz durchgeführt, welches ein zerknittertes Dokument ohne Falten neu erstellen kann. Dieses wurde nur testweise trainiert und nicht in den Prototypen integriert.

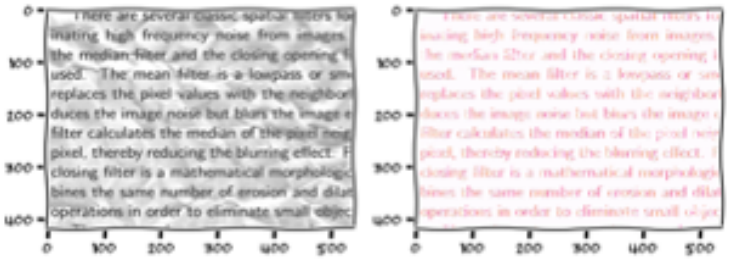


Abbildung 8: Beispielumsetzung der Aufbereitung

5 Diskussion

Um die Ziele aus Kapitel 1.2 zu erreichen, wurden mehrere Probleme identifiziert und in Kapitel 3 gelöst.
Im folgenden werden die entsprechenden Ergebnisse weiter beleuchtet.

5.1 Ergebnis der Recherche

Bei Recherchen in Kapitel 1.5 zu verfügbaren Lösungen konnte keines der betrachteten Netze überprüft bzw. nachimplementiert werden, sodass die Ergebnisse der beschriebenen Forschungsarbeiten nicht direkt nutzbar sind. Daher habe ich mich entschlossen, eine eigene Methode zu entwickeln.
Eine Interpretation von in PDF codierten Tabellen wurde ausgeschlossen, da die hier entwickelte Methodik auch mit Dokumenten funktionieren soll, welche z.B. aus Bildern oder Scans entstanden sind.

5.2 Ergebnis der eigenen Entwicklung

Die Forschung wurde in die Bereiche Tabellenerkennung, Schrifterkennung und Kontextdetektion unterteilt.

In Kapitel 3.1 wurde das Problem der Tabellendetektion erforscht. Hierfür wurden zunächst Ansätze der klassischen Bilderkennung genutzt, welche jedoch kein weiterverarbeitbares Ergebnis erbrachten. Zur Lösung der Problematik wurde daher der moderne Ansatz der künstlichen neuronalen Netze getestet, welcher in Form der Objektdetektion seine Stärken beweisen konnte. Dabei wurde auf moderne Forschungsergebnisse wie das YOLO Netz zurückgegriffen, um das gewünschte Ergebnis zu erreichen.
In weiteren Tests konnte hierdurch das Ziel der Erkennung von 90% aller gegebenen Tabellen erreicht werden. Es zeigte sich jedoch auch, dass die eingesetzte Technik Grenzen hat. So wird eine Tabelle nicht erkannt, wenn sie z.B. deutlich hochauflösender gescannt wurde, als die Trainingsbeispiele. Dies konnte jedoch mit Kompressionsalgorithmen teilweise gelöst werden.

Im Kapitel 3.2 wurde die Schrifterkennung untersucht. Es zeigte sich, dass manche Probleme bereits mit dem Einsatz von KNNs gelöst wurden, jedoch auch unter gesellschaftlichen Aspekten, wie der Forderung nach Datenschutz, betrachtet werden müssen. So wurde für die Schrifterkennung der Einsatz von Cloud-Diensten in technischer und juristischer Hinsicht nach aktuellen Vorgaben geprüft und bewertet. Es stellte sich heraus, dass entsprechende Tools gerade im Hinblick auf den Datenschutz nicht alle Voraussetzungen erfüllen können. Es wurden jedoch auch On-Premise Tools gefunden, welche für den Einsatz unbedenklich sind. Auf Grundlage dieser Regelungen und Anforderungen an ein System, dass auch zur Verarbeitung personenbezogener Daten geeignet sein soll, wurde deswegen entschieden, die genannten Offline-Tools zu nutzen.

Im Bereich der Kontextdetektion in Kapitel 3.3 wurde der Einsatz von künstlichen neuronalen Netzen mit klassischem Code verglichen. Dabei zeigte sich die große Stärke der KNN in der schnellen und präzisen Klassifizierung großer Datenbestände, veranschaulichte jedoch auch die Grenzen dieser Technik, wodurch hier ein klassisches System besser für das gegebene Problem geeignet ist. Das neuronale Netz kann zwar eine große Hilfe bei der Lösung komplexer Probleme sein, ist jedoch für diesen einfachen Fall nicht das richtige Tool.

5.3 Ergebnis der Integration

Die Ergebnisse der Forschung konnten in einen funktionstüchtigen Prototyp integriert werden. Dieser wurde als Client-Server Applikation für die Plattform iOS entwickelt.

Dieser erste Protoyp benötigt dabei einen Server, um Funktionen wie die Tabellendetektion nutzen zu können, was in der Praxis zu einigen Hindernissen, z.B. Bandbreitenproblemen, führen kann. Die Portierung der Funktionalität auf den Client wäre technisch möglich, ist allerdings mit zusätzlichem Aufwand verbunden. Dies wurde aus Zeitgründen nicht umgesetzt.

Anforderung	**Ergebnis des Prototyp (Durchschnittlicher Wert)**	**Wie geprüft?**
Die Realisierung des Einlesens von Dokumenten im Bildformat	funktioniert	Testdaten
Das Erkennen von einer oder mehreren Tabellen in einem Dokument	funktioniert	siehe Kapitel 3.1
Erkennen von 90% aller gegebenen Tabellen	95%	Tests mit 20 Beispieldokumenten
Das Erkennen von 85% der zugehörigen Zellen	ca. 90%	Test mit 20 Beispieldokumenten, mit je mindestens einer Tabelle
Die Ausgabe der Zelleninhalte in maschinenlesbarer Form	funktioniert	Testdaten
Eine Laufzeit von unter einer Minute pro Dokument	ca. 42 Sekunden	Test mit 20 Beispieldokumenten und je mindestens einer Tabelle

Tabelle 1: Vergleich der Anforderungen aus Kapitel 1.2 mit den Ergebnissen des Prototyps

Der Prototyp konnten alle Anforderungen aus Kapitel 1.2 erfüllen.

6 Zusammenfassung, Ausblick und Verbesserungen

6.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein System erschaffen, um Tabellen aus gegebenen Bilddokumenten zu extrahieren und in Daten umzuwandeln. Die Entwicklung des Systems wurde dabei in einzelne Bereiche wie die Tabellenerkennung, die Schrifterkennung oder die Kontextdetektion unterteilt, in welchen unterschiedliche Probleme gelöst werden mussten.

Im Bereich der Tabellenerkennung konnte der moderne Ansatz der künstlichen neuronalen Netze in Form der YOLO-Objektdetektion in einigen Tests seine Stärken beweisen.

Am Beispiel der Schrifterkennung zeigte sich, dass manche Probleme bereits mit dem Einsatz von KNNs gelöst wurden, jedoch auch von gesellschaftlichen Standpunkten, wie der Forderung nach Datenschutz, aus betrachtet werden müssen.
So wurden verschiedene Dienste in technischer und juristischer Hinsicht bewertet und Tools gewählt, welche für den Einsatz unbedenklich sind.

Im Bereich der Kontextdetektion wurde der Einsatz von künstlichen neuronalen Netzen mit klassischem Code verglichen. Dabei zeigte sich die große Stärke der KNN in der schnellen und präzisen Klassifizierung großer Datenbestände, allerdings führen optimierte klassische Ansätze zu gleichen bis besseren Ergebnissen.

Die Ergebnisse der Forschung konnten in einem funktionstüchtigen Prototyp genutzt werden. Dieser wurde als Client-Server Anwendung für die Plattform iOS entwickelt und zeigte die Möglichkeiten eines solchen Systems im Einsatz in der realen Welt.

Des weiteren wurde das YOLO Netz von Mitarbeitern des DLR in neue Projekte integriert. Es bildet dort einen Grundstein für eine neue Art der Datenerfassung und könnte in Zukunft vielen Institutionen und Unternehmen viel Arbeit bei der Digitalisierung ihrer Daten abnehmen.

6.2 Ausblick

Das hier erstellte System ist nicht perfekt. 95% als Erkennungsrate von Tabellen erscheint akzeptabel, bedeutet bei tausenden von Dokumenten aber ggf. viel manuelle Nacharbeit. Eine Möglichkeit der Optimierung wäre hier ein integrativeres Lernen, welche die Schritte Tabellenerkennung, OCR und Kontextdetektion nicht ausschließlich getrennt voneinander trainiert, sondern ein System mit einem Gesamtverständnis entwickelt.

Neben Optimierungsmöglichkeiten oder zusätzlichen Anwendungen, welche in der Arbeit bereits besprochen wurden, sollte auch an einer Optimierung der Performance gearbeitet werden. Eine Möglichkeit der Beschleunigung besteht hier mit dem Einsatz besser optimierter Hardware oder dem Einsatz neuer Technologien, wie beispielsweise dem Einsatz eines Quantencomputers:

Ein klassischer Computer rechnet auf Hardwareebene mit Bits. Dabei können diese einen Zustand von 0 oder 1 annehmen, so dass diese Zustände logisch verknüpft werden können. Quantencomputer nutzen Qubits für die Berechnung.
Ein Qubit nimmt dabei bis zum Zeitpunkt der eigentlichen Abfrage jeden möglichen Zustand ein, was Superposition genannt wird. Bei einer Abfrage fällt diese Superposition in sich zusammen und das Qubit nimmt einen Wert an.

Vorstellen lässt sich dies in Form einer kugelförmigen Sphäre mit einem Vektor im Mittelpunkt. Der Vektor zeigt dabei auf einen Punkt auf der Sphärenhülle und beschreibt damit die Tendenz des Qubit. Indem der Zeiger rotiert wird, verändert er seine Tendenz. Dieses kann mit Hilfe von Quantum Gates erreicht werden. Das Qubit selbst lässt sich also durch die entsprechende Rotation beschreiben, beziehungsweise mit vier verschiedenen Inputs. Werden nun zwei Qubits genutzt, so beeinflussen sich beide Qubits gegenseitig, was wiederum neue Effekte bei der Berechnung ergibt.

Diese neue Technologie kann nun zur Entwicklung neuer Netze eingesetzt werden, genannt Quanten Neuronale Netze oder QNN. Diese Idee wurde zum ersten Mal von Subhash Kak der Oklahoma State University beschrieben. [30] In aktuellen Forschungen lernt das Netz dabei nicht mehr über die Anpassung des Bias, sondern durch die Anpassung der Quantum Gates.

Es ist zu beachten, dass der Einsatz eines Quantencomputers nicht zwangsläufig bedeutet, dass das vorhandene, konventionelle Netz beschleunigt wird. Es ist eher eine völlig neue Methode zur Erstellung von Netzen. Um einen Vergleich der neuen Möglichkeiten zu testen, hat das Team hinter Tensorflow eine Zahlenerkennung in Form eines CNN und eines QNN erstellt wobei nur Bilder der Ziffern 3 und 6 genutzt wurden, um die Problematik weiter zu vereinfachen. [31]

Um diesen Test nachvollziehen zu können, wurde der Test für diese Arbeit nachgebaut. In der ursprünglichen Dokumentation wurde die genutzte Hardware nicht beschrieben. Da es sich hier jedoch um ein Forschungsteam von Google handelt, kann davon ausgegangen werden, dass für den Zweck optimierte Hardware eingesetzt wird.

IBM hat Forschern ein Netz von Quantencomputern zur Verfügung gestellt. Darauf kann jeder Forscher mit einem Account seine eigenen Experimente berechnen lassen. Der Zugang ist dabei jedoch beschränkt. [32]

Für Experimente lassen sich jedoch auch Simulationen nutzen. Diese sind zwar deutlich langsamer, berechnen jedoch das optimale Ergebnis und es treten keine Schmutzeffekte durch äußere Einflüsse auf, was bei der Nutzung echter Quantencomputer durchaus noch der Fall ist.

Für die Tests in dieser Arbeit wurden Simulatoren genutzt. Als Trainingsdatenset wurde der MNIST Datensatz verwendet, welcher Bilder von Zahlen zur Verfügung stellt. Aus den Ergebnissen des Tests lässt sich nun das CNN und das QNN gegenüberstellen. Dabei wurden zusätzlich zu den Ergebnissen des Tests noch die Ergebnisse des Tensorflow Teams betrachtet.

	CNN	QNN
Technische Voraussetzung	Lässt sich auf jeder CPU ausführen	Benötigt eine spezielle technische Ausstattung
Trainingszeit	Tensorflow Team: ca. 3 sec eigene Tests: ca. 33 sec	Tensorflow Team: ca. 20 min eigene Tests: ca. 4 Stunden und 9 Minuten
Präzision des Ergebnisses	Tensorflow Team: 99% eigene Tests: 99%	Tensorflow Team: 85% eigene Tests: 85%

Im direkten Vergleich stellen sich mehrere Unterschiede heraus. Zum einen lässt sich das QNN nicht ohne weiteres einsetzten, da eine spezielle technische Ausstattung benötigt wird, welche häufig nicht zur Verfügung steht. Zum anderen zeigen die teilweise deutlich besseren Trainingszeiten des Tensorflow Teams im Vergleich zum nachgestellten Test, wie stark der Einfluss optimierter Hardware in diesem Themengebiet ist.

Im direkten Vergleich des Netz-Ergebnisses ist das CNN dem QNN überlegen. Es wird nicht nur deutlich weniger Zeit benötigt um das Netz zu trainieren, das Ergebnis ist zusätzlich noch präziser.

Das grundsätzliche Prinzip des QNN ist jedoch spannend und es ist nicht auszuschließen, dass der Einsatz dieser Technik in Zukunft sehr erfolgreich sein wird und dabei völlig neue Netze ermöglicht.

Literatur

[1] *Corona-Testzentren: Auch Gesundheitsamt Trier meldet Pannen*. de. URL: https://www.zdf.de/uri/d9322f29-253b-4cf3-9f92-d329102c6277 (besucht am 19. 08. 2020).

[2] *DLR - Institut für Datenwissenschaften*. de. Library Catalog: www.dlr.de. URL: https://www.dlr.de/content/de/institutspraesentation/institut-fuer-datenwissenschaften.html (besucht am 14. 05. 2020).

[3] *atlanhq/camelot*. original-date: 2016-06-18T11:48:49Z. Sep. 2020. URL: https://github.com/atlanhq/camelot (besucht am 02. 09. 2020).

[4] Shubham Singh Paliwal u. a. „TableNet: Deep Learning Model for End-to-end Table Detection and Tabular Data Extraction from Scanned Document Images". en. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sydney, Australia: IEEE, Sep. 2019, S. 128–133. ISBN: 978-1-72813-014-9. DOI: 10.1109/ICDAR.2019.00029. URL: https://ieeexplore.ieee.org/document/8978013/ (besucht am 01. 04. 2020).

[5] Sebastian Schreiber u. a. „DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images". en. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Kyoto: IEEE, Nov. 2017, S. 1162–1167. ISBN: 978-1-5386-3586-5. DOI: 10.1109/ICDAR.2017.192. URL: http://ieeexplore.ieee.org/document/8270123/ (besucht am 01. 04. 2020).

[6] *Künstliches neuronales Netz*. de. Page Version ID: 197348429. März 2020. URL: https://de.wikipedia.org/w/index.php?title=K%C3%BCnstliches_neuronales_Netz&oldid=197348429 (besucht am 01. 04. 2020).

[7] Diederik P. Kingma und Jimmy Ba. „Adam: A Method for Stochastic Optimization". en. In: *arXiv:1412.6980 [cs]*. arXiv: 1412.6980. Jan. 2017. URL: http://arxiv.org/abs/1412.6980 (besucht am 21. 04. 2020).

[8] *PDF 32000-1:2008 - First Edition2008-7-1Document management — Portable document format — Part 1: PDF 1.7*. 2008.

[9] *jupyter/notebook*. original-date: 2015-04-09T06:58:03Z. Sep. 2020. URL: https://github.com/jupyter/notebook (besucht am 02. 09. 2020).

[10] *Google Colaboratory*. en. URL: https://colab.research.google.com/notebooks/intro.ipynb (besucht am 02. 09. 2020).

[11] Atlassian. *Jira | Issue & Project Tracking Software*. en. URL: https://www.atlassian.com/software/jira (besucht am 19. 08. 2020).

[12] *Welcome to Python.org*. en. URL: https://www.python.org/ (besucht am 02. 09. 2020).

[13] *Swift - Apple Developer*. en. URL: https://developer.apple.com/swift/ (besucht am 02. 09. 2020).

[14] *tensorflow/tensorflow*. original-date: 2015-11-07T01:19:20Z. Sep. 2020. URL: https://github.com/tensorflow/tensorflow (besucht am 02. 09. 2020).

[15] *keras-team/keras*. original-date: 2015-03-28T00:35:42Z. Sep. 2020. URL: https://github.com/keras-team/keras (besucht am 02. 09. 2020).

[16] *opencv/opencv*. original-date: 2012-07-19T09:40:17Z. Sep. 2020. URL: https://github.com/opencv/opencv (besucht am 02. 09. 2020).

[17] *pallets/flask*. original-date: 2010-04-06T11:11:59Z. Sep. 2020. URL: https://github.com/pallets/flask (besucht am 02. 09. 2020).

[18] Xu Zhong, Jianbin Tang und Antonio Jimeno Yepes. „PubLayNet: largest dataset ever for document layout analysis“. en. In: *arXiv:1908.07836 [cs]*. arXiv: 1908.07836. Aug. 2019. URL: http://arxiv.org/abs/1908.07836 (besucht am 18. 06. 2020).

[19] Joseph Redmon und Ali Farhadi. „YOLOv3: An Incremental Improvement“. en. In: S. 6.

[20] David Kriesel. *Ein kleiner Überblick über Neuronale Netze*. 2007. URL: erh%C3%A4ltlich%20auf%20http://www.dkriesel.com.

[21] *tesseract-ocr/tesseract*. original-date: 2014-08-12T18:04:59Z. Sep. 2020. URL: https://github.com/tesseract-ocr/tesseract (besucht am 02. 09. 2020).

[22] *Vision | Apple Developer Documentation*. URL: https://developer.apple.com/documentation/vision (besucht am 02. 09. 2020).

[23] David Wolski. „Toolbox: Texterkennung mit Tesseract OCR | c't Magazin“. In: (Aug. 2012). URL: https://www.heise.de/ct/artikel/Toolbox-Texterkennung-mit-Tesseract-OCR-1674881.html (besucht am 05. 05. 2020).

[24] *BSI - IT-Grundschutz - BSI-Standards*. URL: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzStandards/ITGrundschutzStandards_node.html (besucht am 02. 09. 2020).

[25] *Art. 25 DSGVO – Datenschutz durch Technikgestaltung und durch datenschutzfreundliche Voreinstellungen*. de-DE. URL: https://dsgvo-gesetz.de/art-25-dsgvo/ (besucht am 02. 09. 2020).

[26] *Art. 44 DSGVO – Allgemeine Grundsätze der Datenübermittlung*. de-DE. URL: https://dsgvo-gesetz.de/art-44-dsgvo/ (besucht am 02. 09. 2020).

[27] „Mindeststandard des BSI zur Nutzung externer Cloud-Dienste“. de. In: (), S. 12.

[28] *BSI - Kriterienkatalog C5*. URL: https://www.bsi.bund.de/DE/Themen/DigitaleGesellschaft/CloudComputing/Kriterienkatalog/Kriterienkatalog_node.html (besucht am 02. 09. 2020).

[29] Doug Collins. *Text - H.R.4943 - 115th Congress (2017-2018): CLOUD Act*. eng. webpage. Archive Location: 2017/2018. Juni 2018. URL: https://www.congress.gov/bill/115th-congress/house-bill/4943/text (besucht am 02. 09. 2020).

[30] Subhash Kak. „On Quantum Neural Computing." In: *Inf. Sci.* 83 (1995), S. 143–160. DOI: 10.1016/S1076-5670(08)70147-2.

[31] *MNIST classification | TensorFlow Quantum*. en. URL: https://www.tensorflow.org/quantum/tutorials/mnist (besucht am 02. 09. 2020).

[32] *IBM Quantum Experience*. en. URL: https://quantum-computing.ibm.com/ (besucht am 02. 09. 2020).

[33] *Swift (Programmiersprache)*. de. Page Version ID: 198272617. März 2020. URL: https://de.wikipedia.org/w/index.php?title=Swift_(Programmiersprache)&oldid=198272617 (besucht am 01. 04. 2020).

[34] *Python (Programmiersprache)*. de. Page Version ID: 198095335. März 2020. URL: https://de.wikipedia.org/w/index.php?title=Python_(Programmiersprache&oldid=198095335 (besucht am 01. 04. 2020).

[35] *Table Detection, Information Extraction and Structuring using Deep Learning*. en. Library Catalog: nanonets.com. Jan. 2020. URL: https://nanonets.com/blog/table-extraction-deep-learning/ (besucht am 01. 04. 2020).

[36] Andrew Tch. *The mostly complete chart of Neural Networks, explained*. URL: https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464 (besucht am 01. 04. 2020).

[37] Kenneth O. Stanley und Risto Miikkulainen. „Evolving Neural Networks through Augmenting Topologies". en. In: *Evolutionary Computation* 10.2 (Juni 2002), S. 99–127. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/106365602320169811. URL: http://www.mitpressjournals.org/doi/10.1162/106365602320169811 (besucht am 01. 04. 2020).

[38] *Neural networks*. en-GB. Library Catalog: www.youtube.com. URL: http://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi (besucht am 01. 04. 2020).

[39] *But what is a Neural Network? | Deep learning, chapter 1*. URL: https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=2&t=0s (besucht am 01. 04. 2020).

[40] *Gradient descent, how neural networks learn | Deep learning, chapter 2*. URL: https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=2 (besucht am 01. 04. 2020).

[41] *What is backpropagation really doing? | Deep learning, chapter 3*. URL: https://www.youtube.com/watch?v=Ilg3gGewQ5U&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=3 (besucht am 01. 04. 2020).

[42] *Backpropagation calculus | Deep learning, chapter 4*. URL: https://www.youtube.com/watch?v=tIeHLnjs5U8&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=4 (besucht am 01. 04. 2020).

[43] Nadja Geisler und Benjamin Hättasch. *Der Deep Learning Hype*. en. URL: /v/36c3-11006-der_deep_learning_hype (besucht am 01. 04. 2020).

[44] sven. *Thrust is not an Option: How to get to Mars really slow*. en. URL: /v/36c3-10918-thrust_is_not_an_option_how_to_get_to_mars_really_slow (besucht am 01. 04. 2020).

[45] *PDF Insecurity Website*. URL: https://www.pdf-insecurity.org/ (besucht am 01. 04. 2020).

[46] Fabian Ising und Vladislav Mladenov. *How to Break PDFs*. en. URL: /v/36c3-10832-how_to_break_pdfs (besucht am 01. 04. 2020).

[47] Minghao Li u. a. „TableBank: Table Benchmark for Image-based Table Detection and Recognition". en. In: *arXiv:1903.01949 [cs]*. arXiv: 1903.01949. März 2019. URL: http://arxiv.org/abs/1903.01949 (besucht am 01. 04. 2020).

[48] Clement Sage u. a. „Recurrent Neural Network Approach for Table Field Extraction in Business Documents". en. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sydney, Australia: IEEE, Sep. 2019, S. 1308–1313. ISBN: 978-1-72813-014-9. DOI: 10.1109/ICDAR.2019.00211. URL: https://ieeexplore.ieee.org/document/8978042/ (besucht am 01. 04. 2020).

[49] S. R. Qasim, H. Mahmood und F. Shafait. „Rethinking Table Recognition using Graph Neural Networks". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, S. 142–147.

[50] *Chaos Communication Congress: Glauben Sie nicht alles, was in einem PDF-Dokument steht | ZEIT ONLINE*. URL: https://www.zeit.de/digital/datenschutz/2019-12/chaos-communication-congress-ccc-2019-pdf-verschluesselung-signatur-sicherheit/komplettansicht (besucht am 01. 04. 2020).

[51] Lisa Hegemann. „Chaos Communication Congress: Glauben Sie nicht alles, was in einem PDF-Dokument steht". de-DE. In: *Die Zeit* (Dez. 2019). ISSN: 0044-2070. URL: https://www.zeit.de/digital/datenschutz/2019-12/chaos-communication-congress-ccc-2019-pdf-verschluesselung-signatur-sicherheit/komplettansicht (besucht am 01. 04. 2020).

[52] *Long short-term memory*. de. Page Version ID: 195196525. Dez. 2019. URL: https://de.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=195196525 (besucht am 01. 04. 2020).

[53] *Logistische Funktion*. de. Page Version ID: 198241993. März 2020. URL: https://de.wikipedia.org/w/index.php?title=Logistische_Funktion&oldid=198241993 (besucht am 01. 04. 2020).

[54] *Radiale Basisfunktion*. de. Page Version ID: 188657883. Mai 2019. URL: https://de.wikipedia.org/w/index.php?title=Radiale_Basisfunktion&oldid=188657883 (besucht am 01. 04. 2020).

[55] *Neuronales Netz*. de. Page Version ID: 198313029. März 2020. URL: https://de.wikipedia.org/w/index.php?title=Neuronales_Netz&oldid=198313029 (besucht am 01. 04. 2020).

[56] Sophia Thamm. „Einführung in Neuronale Netze“. de. In: (), S. 29.

[57] Martha O Perez-Arriaga, Trilce Estrada und Soraya Abad-Mota. „TAO: System for Table Detection and Extraction from PDF Documents“. en. In: S. 6.

[58] Heise Medien GmbH und Co Kg. „iX kompakt 2018 – Programmieren heute“. de. In: (2018), S. 156.

[59] „Table Extraction“. en. In: (), S. 7.

[60] Bernd Schneider. „Neuronale Netze für betriebliche Anwendungen: Anwendungspotentiale und existierende Systeme“. de. In: (), S. 38.

[61] Tom Hope, Yehezkel S. Resheff und Itay Lieder. *Einführung in TensorFlow: Deep-Learning-Systeme programmieren, trainieren, skalieren und deployen*. ger. Übers. von Kristian Rother und Thomas Lotze. 1. Auflage. OCLC: 1026995647. Heidelberg: O'Reilly, 2018. ISBN: 978-3-96010-181-9 978-3-96009-074-8 978-3-96010-180-2 978-3-96010-182-6.

[62] Jason Brownlee. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. en-US. Library Catalog: machinelearningmastery.com Section: Deep Learning Performance. Juli 2017. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (besucht am 21. 04. 2020).

[63] *Project Jupyter*. de. Page Version ID: 199136190. Apr. 2020. URL: https://de.wikipedia.org/w/index.php?title=Project_Jupyter&oldid=199136190 (besucht am 22. 04. 2020).

[64] *Backronym*. en. Library Catalog: www.infornopolitan.xyz. URL: https://www.infornopolitan.xyz/backronym (besucht am 24. 04. 2020).

[65] *Eine visuelle Einführung ins Maschinelle Lernen*. de. Library Catalog: www.r2d3.us. URL: http://www.r2d3.us/visuelle-einfuehrung-ins-maschinelle-lernen-teil-1/ (besucht am 24. 04. 2020).

[66] *U-Net: Convolutional Networks for Biomedical Image Segmentation*. URL: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/ (besucht am 28. 04. 2020).

[67] *Machine Learning Glossary*. en. Library Catalog: developers.google.com. URL: https://developers.google.com/machine-learning/glossary (besucht am 29. 04. 2020).

[68] *Review: MobileNetV2 — Light Weight Model (Image Classification)*. URL: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c (besucht am 29. 04. 2020).

[69] *Object detection with neural networks — a simple tutorial using keras*. URL: https://towardsdatascience.com/object-detection-with-neural-networks-a4e2c46b4491 (besucht am 30. 04. 2020).

[70] *Überanpassung*. de. Page Version ID: 198208931. März 2020. URL: https://de.wikipedia.org/w/index.php?title=%C3%9Cberanpassung&oldid=198208931 (besucht am 01. 05. 2020).

[71] *KISS-Prinzip*. de. Page Version ID: 194917272. Dez. 2019. URL: https://de.wikipedia.org/w/index.php?title=KISS-Prinzip&oldid=194917272 (besucht am 01. 05. 2020).

[72] *Camelot: PDF Table Extraction for Humans — Camelot 0.7.3 documentation*. URL: https://camelot-py.readthedocs.io/en/master/ (besucht am 03. 05. 2020).

[73] *Ocrad - GNU Project - Free Software Foundation (FSF)*. URL: https://www.gnu.org/software/ocrad/ (besucht am 06. 05. 2020).

[74] *Texterkennung*. de. Page Version ID: 199568576. Mai 2020. URL: https://de.wikipedia.org/w/index.php?title=Texterkennung&oldid=199568576 (besucht am 06. 05. 2020).

[75] *Datenschutz-Grundverordnung*. de. Page Version ID: 199908365. Mai 2020. URL: https://de.wikipedia.org/w/index.php?title=Datenschutz-Grundverordnung&oldid=199908365 (besucht am 14. 05. 2020).

[76] „VERORDNUNG (EU) 2016/ 679 DES EUROPÄISCHEN PARLAMENTS UND DES RATES - vom 27. April 2016 - zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/ 46/ EG (Datenschutz-Grundverordnung)“. de. In: (), S. 88.

[77] Edouard Belval. *pdf2image: A wrapper around the pdftoppm and pdftocairo command line tools to convert PDF to a PIL Image list.* URL: https://github.com/Belval/pdf2image (besucht am 14. 05. 2020).

[78] *Neuroevolution*. en. Page Version ID: 943638200. März 2020. URL: https://en.wikipedia.org/w/index.php?title=Neuroevolution&oldid=943638200 (besucht am 15. 05. 2020).

[79] Hunter Heidenreich. *NEAT: An Awesome Approach to NeuroEvolution*. en. Library Catalog: towardsdatascience.com. Jan. 2019. URL: https://towardsdatascience.com/neat-an-awesome-approach-to-neuroevolution-3eca5cc7930f (besucht am 15. 05. 2020).

[80] Kenneth O Stanley, David D'Ambrosio und Jason Gauci. „A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks“. en. In: (), S. 39.

[81] Joel Lehman und Risto Miikkulainen. „Neuroevolution“. en. In: *Scholarpedia* 8.6 (Juni 2013), S. 30977. ISSN: 1941-6016. DOI: 10.4249/scholarpedia.30977. URL: http://www.scholarpedia.org/article/Neuroevolution (besucht am 15. 05. 2020).

[82] *PubLayNet*. en-US. Library Catalog: developer.ibm.com. URL: https://developer.ibm.com/technologies/artificial-intelligence/data/publaynet/ (besucht am 17. 06. 2020).

[83] Neeraj Panse. *Neerajj9/Text-Detection-using-Yolo-Algorithm-in-keras-tensorflow*. original-date: 2019-05-07T12:22:24Z. Juni 2020. URL: https://github.com/Neerajj9/Text-Detection-using-Yolo-Algorithm-in-keras-tensorflow (besucht am 19. 06. 2020).

[84] *DeepMind - The Role of Multi-Agent Learning in Artificial Intelligence Research*. URL: https://www.youtube.com/watch?v=yE62Zwhmzi8&list=WL&index=20&t=670s (besucht am 19. 06. 2020).

[85] Mark Sandler u. a. „MobileNetV2: Inverted Residuals and Linear Bottlenecks“. In: *arXiv:1801.04381 [cs]*. arXiv: 1801.04381. März 2019. URL: http://arxiv.org/abs/1801.04381 (besucht am 24. 06. 2020).

[86] Joseph Redmon u. a. „You Only Look Once: Unified, Real-Time Object Detection“. en. In: *arXiv:1506.02640 [cs]*. arXiv: 1506.02640. Mai 2016. URL: http://arxiv.org/abs/1506.02640 (besucht am 03. 07. 2020).

[87] *R (Programmiersprache)*. de. Page Version ID: 202586944. Aug. 2020. URL: https://de.wikipedia.org/w/index.php?title=R_(Programmiersprache)&oldid=202586944 (besucht am 18. 08. 2020).

[88] J Benzler, A Gilsdorf und G Kirchner. „DEMIS - Deutsches elektronisches Meldesystem für Infektionsschutz“. de. In: *Gesundheitswesen* 75.04 (Apr. 2013), s–0033–1337477. ISSN: 0941-3790, 1439-4421. DOI: 10.1055/s-0033-1337477. URL: http://www.thieme-connect.de/DOI/DOI?10.1055/s-0033-1337477 (besucht am 19. 08. 2020).

[89] *Jira (Software)*. de. Page Version ID: 196601839. Feb. 2020. URL: https://de.wikipedia.org/w/index.php?title=Jira_(Software)&oldid=196601839 (besucht am 19. 08. 2020).

[90] *RStudio*. de. Page Version ID: 202255725. Juli 2020. URL: https://de.wikipedia.org/w/index.php?title=RStudio&oldid=202255725 (besucht am 02. 09. 2020).

[91] Matthias Homeister. *Quantum Computing verstehen: Grundlagen – Anwendungen – Perspektiven*. ger. 5., aktualisierte und erweiterte Auflage. Computational Intelligence. OCLC: 1043117555. Wiesbaden: Springer Vieweg, 2018. ISBN: 978-3-658-22884-2 978-3-658-22883-5.

[92] Dr. rer. soc. oec. Georg Krempl. „Aktives maschinelles Lernen bei unvollständiger Information“. Habilitationsschrift. Magdeburg: Otto-von-Guericke-Universita t Magdeburg, Sep. 2016.

[93] *TensorFlow*. URL: https://www.tensorflow.org/ (besucht am 02. 09. 2020).

[94] *Quantum neural network*. en. Page Version ID: 973808233. Aug. 2020. URL: https://en.wikipedia.org/w/index.php?title=Quantum_neural_network&oldid=973808233 (besucht am 02. 09. 2020).

[95] *Datenschutz-Grundverordnung: DSGVO als übersichtliche Seite*. de-DE. URL: https://dsgvo-gesetz.de/ (besucht am 02. 09. 2020).

A Lösungsarchitektur

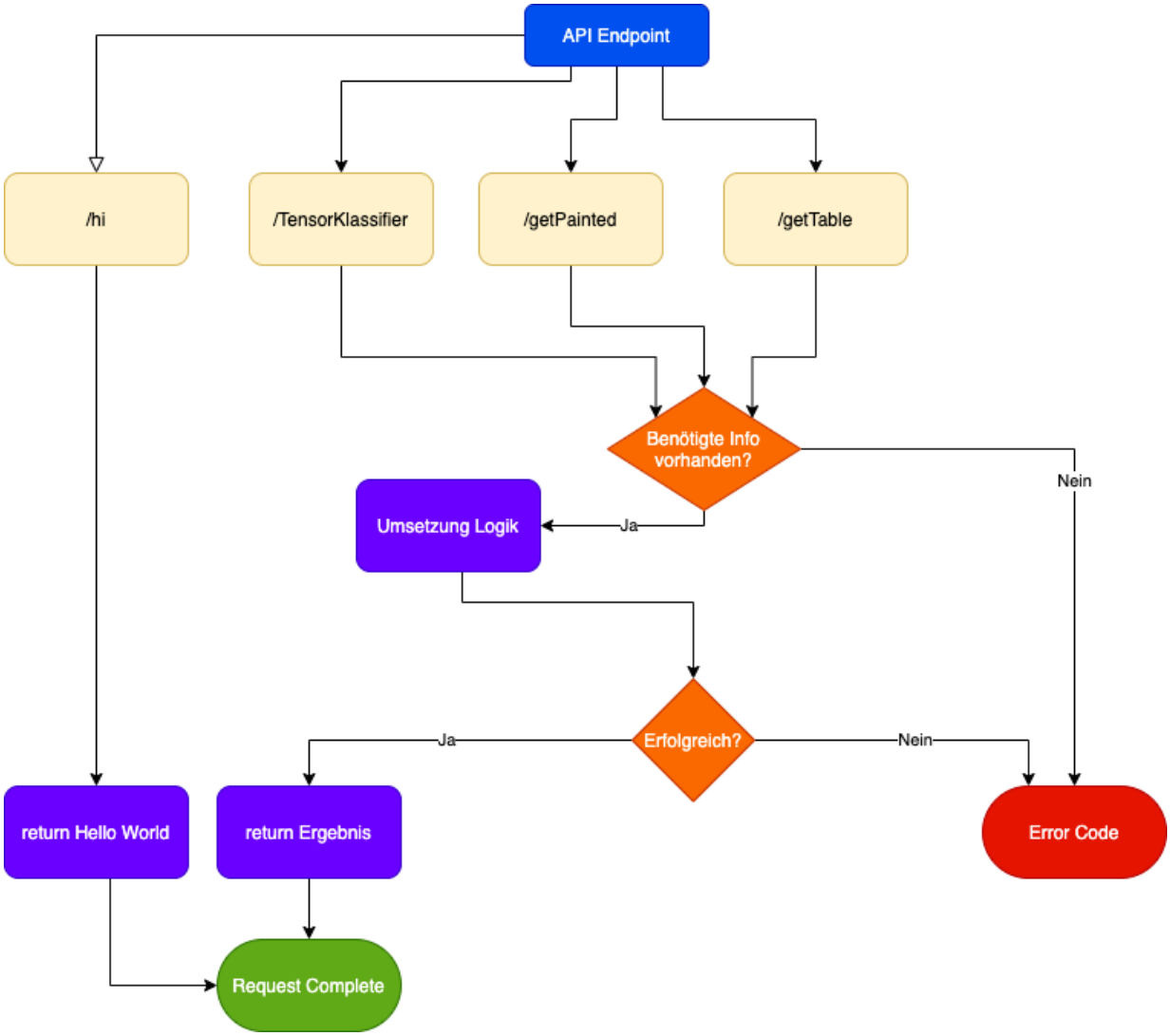


Abbildung 9: Flowchart der Prototyp - API

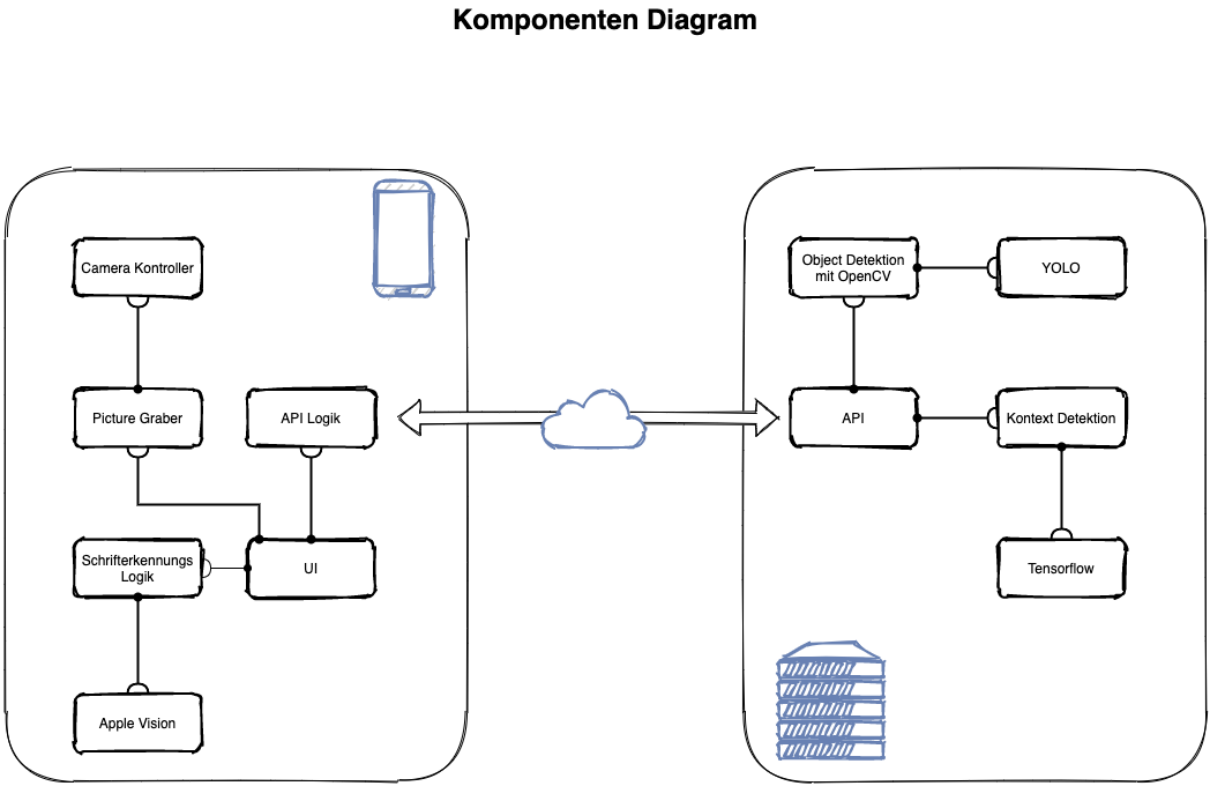


Abbildung 10: Komponentenplanung der Prototyp - APP

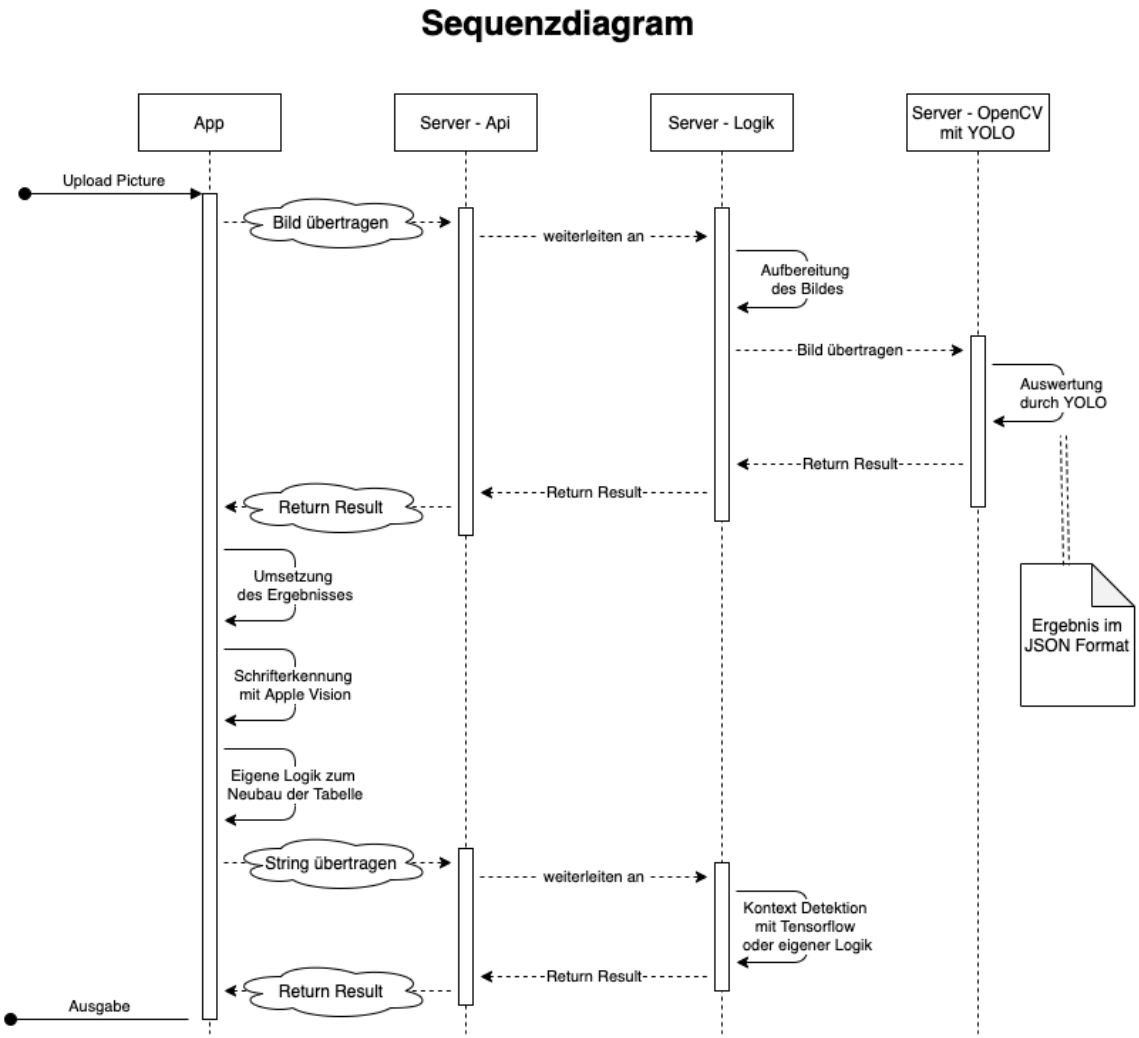


Abbildung 11: Sequenzdiagram der Prototyp - APP

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie die Zitate deutlich kenntlich gemacht zu haben.
Ich erkläre weiterhin, dass die vorliegende Arbeit in gleicher oder ähnlicher Form noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Fulda, den 20. September 2020 Christian Baltzer