TECHNISCHE HOCHSCHULE LÜBECK

**Bachelorarbeit**

# Dokumenten und -Texterkennung anhand von Rechnungsbelegen


abgegeben von: Christoph Thomas

Abteilung: Elektrotechnik und Informatik
Studiengang: Informatik/Softwaretechnologie
Erster Gutachter: Prof. Dr. Ehlers



Anfangsdatum: 01.07.2019
Abgabedatum: 31.10.2019

**Arbeitsauftrag**

In der heutigen Software ist Machine Learning zu einem nicht mehr wegzudenkenden Teilgebiet geworden. Durch die Verwendung intelligenter Systeme können Ressourcen wie Zeit und Aufwand gespart werden. Unternehmen, die mit dieser Technologie arbeiten, haben einen ökonimischen Vorteilen gegenüber der Konkurrenz und heben sich von Diesen ab.

Die Automatisierung soll anhand einer künstlichen Intelligenz erfolgen.
Der Erkennungsprozess wird in drei Phasen untergliedert:

1. Mustererkennung

   Ein Klassifikator ist in der Lage Datenmuster in den Rechnungsbelegen zu erkennen und einem Belegtyp zuzuordnen.

2. Belegextraktion

   Auf Basis der Mustererkennung kann eine Extraktion der Informationen durchgeführt werden. Die Positionen der Daten werden durch die Musterkennung ermittelt.

3. Evaluation

   Eine Quantifizierung evaluiert die nachträglich ausgefüllten Felder, fehlerhafte Erkennungen und die durchschnittliche Zeitersparnis. Ergebnisse sollen visuell dargestellt werden.

TECHNISCHE
HOCHSCHULE
LÜBECK

Department of
Electrical Engineering and Computer Science
Dean's Office - Board of Examiners

**Erklärung zur Bachelorarbeit**

Ich versichere, dass ich die Arbeit selbst¨andig, ohne fremde Hilfe verfasst habe.

Bei der Abfassung der Arbeit sind nur die angegebenen Quellen benutzt worden.W¨ortlich oder dem Sinne nach entnommene Stellen sind als solche gekennzeichnet.

Ich bin damit einverstanden, dass meine Arbeit ver¨o↵entlicht wird, insbesondere dass die Arbeit Dritten zur Einsichtnahme vorgelegt oder Kopien der Arbeit zurWeitergabe an Dritte angefertigt werden.

Lübeck, den September 20, 2019

. . . . . .

*(signature)*

**Zusammenfassung der Arbeit / Abstract of Thesis**

| | |
|---|---|
| **Fachbereich:** | Elektrotechnik und Informatik |
| **Studiengang:** | Informatik/Softwaretechnologie B.Sc. |
| **Thema:** | Dokumenten und- Texterkennung von Dokumenten |
| **Zusammenfassung:** | Um die Automatisierung firmeninterner Prozesse zu ermöglichen soll anhand von Machine learning eine Klassifikation von Dokumenten stattfinden. Eine Texterkennung soll letzendlich den Nutzer das Ausfüllen eines Formulars abnehmen. Im Laufe dieser Arbeit sollen Probleme analysiert und geeignete Architekturmuster zur Problemlösung verwendet werden. Anhand von Trainingsdaten wird ein für dieses Problem geeigneter Klassifikator trainiert und im Betrieb verwendet. Letztendlich werden Präzisionsergebnisse, die über eine Schnittstelle gesammelt werden, evaluiert. |
| **Autor** | Christoph Thomas |
| **Betreuender Professor:** | Prof. Dr. Jens Ehlers |
| **WS / SS:** | WS 2019/20 |

# Contents

# 1 Einleitung

## 1.1 Motivation

Bei firmeninternen Schadensabwicklungen werden täglich eine Vielzahl von Bildern von Rechnungsbelegen hochgeladen, die auf einen Archiv-Server gespeichert werden. Darüber hinaus werden Formulare ausgefüllt, um Schäden zu protokollieren. Da die in die Formularfelder eingetragenen Informationen aber in Textform auf den Rechnungsbelegen vorliegen, besteht die Motivation daraus das Ausfüllen des Formulars zu automatisieren.

Machine Learning gewinnt immer mehr an Bedeutung, was sich an diesem Anwendungsfall zeigt. Aus Sicht des Kunden ist die Automatisierung der Schadensabwicklung ein Ersparniss an Zeit und Aufwand. Ökonomisch gesehen erzielt die KI also einen Vorteil für den Kunden. Eine Automatisierung über Bilderkennung würde den Kunden viel Zeit ersparen und somit dem Unternehmen ein enormen Wettbewerbsvorteil verschaffen.

Durch Datenbestände in kann eine künstliche Intelligenz trainiert werden Muster zu erkennen und eigenständig Lösungen für Probleme zu finden. Durch das Füttern der Datenbestände an den Klassifikator wird eine Trainingsphase durchlaufen.
Auf dem Archivserver ist eine Vielzahl von Bildern vorhanden, welche als Trainingsdaten genutzt werden um den Klassifikator auf zukünftige Prognosen vorzubereiten.

Es gibt eine Vielzahl von Klassifikatoren, die jeweils Anwendungsgebiete haben, in denen sie besonders gut Probleme Prognostizieren können. Im Bereich der Bild- und Texterkennung sind Neuronale Netze sehr effektiv.

## 1.2 Problemstellung

Beim Ausfüllen des Formulares geht Zeit und Aufwand verloren. Durch Machine Learning soll dieser Prozess automatisiert werden. Die Adaption bezüglich Machine Learning schafft

gegenüber nicht adaptierenden Unternehmen einen Wettbewerbsvorteil und sichert dem Unternehmen einen hohen Markstellenwert.
Der hochgeladene Beleg soll durch ein Klassifikator kategorisiert und somit der Belegtyp festgestellt werden.
Der bei der Schadensabwicklung hochgeladene Beleg beinhaltet Daten, die in die Formularfelder eingetragen werden müssen. Das Ausfüllen wird durch eine künstliche Intelligenz automatisiert.

## 1.3 Zielsetzung

Durch eine sinnvolle Implementierung zweier Klassifikatoren soll das Ausfüllen eines Schadensabwicklungsformular durch einen Bildupload automatisiert werden. Die Auswahl Der Klassifkationsmodelle sollen für die Aufgaben passend ausgewählt werden. Es soll eine möglichst hohe Präzisionsrate für Probleme erzielt werden. Weiterhin soll eine Schnittstelle die Ergebnisse der Klassifikation abfangen und auf einem Dashboard sammeln um diese dann zu visualisieren.
Es soll eine Stufenweise Klassifikation stattfinden:

1. Klassifikation der Belege
2. Extraktion der Informationen auf Basis der Belegart durch OCR (optical character recognition)

Die Trainingsphase soll den Klassifikator effektiv auf bevorstehende Mustererkennungen vorbereiten. Die vorhandenen Trainingsdaten sollen aufbereitet werden und dem Klassifikationsmodell übergeben werden.

## 1.4 Gliederung der Arbeit

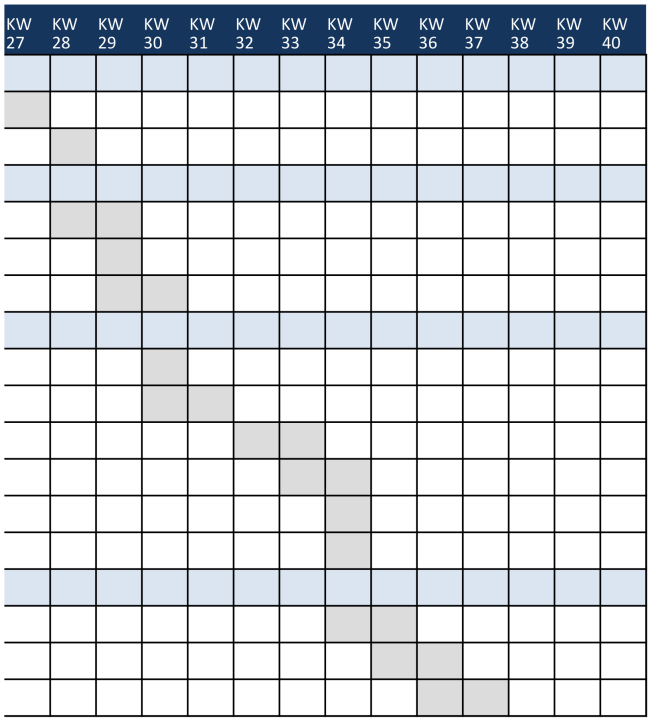
Die Arbeit unterteilt sich in momentan 3 Kernbereiche:

- Dokumentenerkennung (Document Image Analysis)
- Bilderkennung mit einem neuronalen Netz
- OCR mit Tesseract

## 1.5 Meilensteinplan

| | |
|---|---|
| **Startdatum** | 01.07.19 |
| **Enddatum** | 01.10.19 |
| **Fortschritt** | 5,00% |

| Aufgaben | Start | Ende | Tage | Status | |
|---|---|---|---|---|---|
| **Vorarbeiten** | | | | | |
| Aufgabenanalyse | 1.7 | 5.7 | 4 | 75,00% | |
| Informationsbeschaffung | 5.7 | 10.7 | 5 | 25,00% | |
| **Modellierung** | | | | | |
| Klassifikationsmodell | 10.7 | 15.7 | 5 | 25,00% | |
| Architektur | 15.7 | 20.7 | 5 | Nicht begonnen | |
| Trainingsumgebung | 20.7 | 22.7 | 2 | Nicht begonnen | |
| **Entwicklung** | | | | | |
| Abhängigkeiten, Umgebung | 22.7 | 24.7 | 2 | Nicht begonnen | |
| Architektur umsetzen | 24.7 | 4.8 | 11 | Nicht begonnen | |
| Trainingsphase | 4.8 | 12.8 | 8 | Nicht begonnen | |
| Evaluation der Ergebnisse | 12.8 | 19.8 | 7 | Nicht begonnen | |
| Optimierung | 19.8 | 22.8 | 3 | Nicht begonnen | |
| Kommentierung | 22.8 | 23.8 | 1 | Nicht begonnen | |
| **Evaluation** | | | | | |
| Pipeline definieren | 23.8 | 29.8 | 6 | Nicht begonnen | |
| Ergebnisse visualisieren | 29.8 | 6.9 | 8 | Nicht begonnen | |
| Auswertung | 6.9 | 11.9 | 5 | Nicht begonnen | |



# 2 Anforderungsanalyse

Im Laufe dieser Arbeit soll eine Softwarelösung mithilfe von Machine Learning entwickelt werden, die in der Lage ist Belege zu erkennen und Informationen zu extrahieren. Die Daten sollen automatisch in Formularfelder eingetragen werden, wobei der Benutzer die Möglichkeit hat Korrekturen vorzunehmen.

Die Präzisionsrate Der Belegextrahierung und die durchschnittliche Zeitersparnis beim Ausfüllen des Formulars sollen quantitativ evaluiert und schließlich visualisiert werden.

## 2.1 Anwendungsfall

Im 2.1 wird der automatisierte Prozess veranschaulicht. Beim Hochladen eines Nachweises in Bildform prognostiziert ein Klassifikator, der darauf trainiert ist Belege zu erkennen, ob es sich um einen Rechnungsbeleg handelt. Sollte die Prognose eintreffen, werden Informationen von einem Klassifikationsmodell extrahiert, das auf Texterkennung trainiert ist. Das System soll mit den extrahierten Informationen das Formular ausfüllen und bei Falschinformationen den Benutzer Korrekturen vornehmen lassen.

Für die Schadensfallabwicklung ist das Hochladen von Nachweisen optional. Der Benutzer ist weiterhin in der Lage die Formularfelder manuell auszufüllen.

Aus den Kontext des Anwendungsfall ergeben sich mehrere funktionale Anforderungen die an die Klassifikatoren und an das System gestellt werden.

## 2.2 Funktionale Anforderungen

Das Anwendungsszenario leitet funktionelle Anforderungen, die in der Tabelle 2.1 aufgelistet sind, für eine Softwarelösung ab, um eine Prozessautomatisierung zu realisieren. Im Automatiserungskontext werden sowohl Anforderungen an das System als auch an die Klassifikatoren gestellt um die Problemstellung zu lösen.

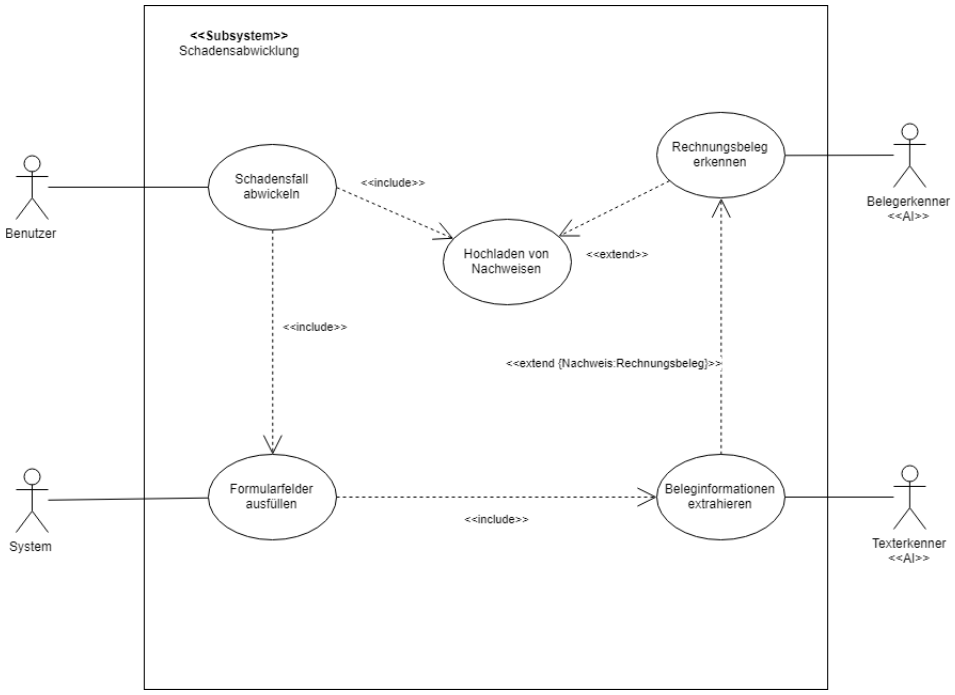



Figure 2.1: Chapter 1 - Anwendungsfalldiagramm

| ID | Beschreibung |
|---|---|
| TR010 | Weitergabe der hochgeladenen Nachweisen an Belegerkennung |
| TR020 | Entgegennahme der übermittelten Nachweise |
| TR030 | Erkennung der Belege ( Wahrscheinlichkeit Prognose > 90% ) |
| TR040 | Interpretation des prognostizierten Merkmals |
| TR050 | Erfassen der extrahierten Informationen |
| TR060 | Gliedern der Informationen in Sinnabschnitte |
| TR070 | Validation der erfassten Daten |
| TR080 | Entgegennahme der übermittelten Belege |
| TR090 | Extrahieren der auf Beleg abgebildeten Informationen |
| TR100 | Sinnvolles Ausfüllen der Formularfelder |
| TR110 | Messung der für die Schadensabwicklung aufgewendete Zeit |
| TR120 | Bewertung des Korrekturverhalten des Nutzers |

Table 2.1: Funktionale Anforderungen

# 3 Technische Anforderungen

Dieser Chapter fügt sich in den Bereich der Document Image Analysis (DIA).

## 3.1 Forschungsstand

DIA bezeichnet allgemein eine große Gruppe von Techniken, die visuelle Informationen charakterisieren können. Dia gehört zu den ältesten Bereichen der Informatik. 1913 Erfand Edmund Fournier das Optophon, mit dem dunkle Schriftzeichen auf einem Stück Papier erkannt werden können und als Töne interpretiert wurden.
Heutzutage durchlaufen moderne DIA-Algorithmen Rasterbilder, die mit Scannern, Kameras oder anderen Digitalisierungsgeräten erzeugt wurden.

## 3.2 Probleme

Durch die stetige Weiterentwicklung von DIA ist die Vielfalt der Probleme gewachsen [Ten19]:

- Binarisierung - Klassifizieren welche Pixel im Vordergrund un welche im Hintergrund ist
- Korrektur - Korrigieren der perspektivischen Verzerrung. Dies passiert wenn die Kamera nicht orthogonal zum Physikalischen Dokument positioniert ist
- Seitensegmentierung - Unterteilen einer Seite in homogene Komponenten wie Text, Bilder und Grafiken
- Textsegmentierung - Unterteilen eines Textabschnitts in Zeilen
- Dokumentklassifizierung - Klassifizierung des Dokumenttyps
- Schrift und - Spracherkennung - Klassifizieren mit welcher Schrift geschrieben wurde und welche Sprache benutzt wird
- Zeichensegmentierung - Segmentiert ein Wortbild in individuelle Charaktere
- Grafische Erkennung - Erkennung von graphischen Komponenten
- Tabellenlayout-Struktur Erkennung - Wiederherstellen der Zeilen und- Spaltenstruktur von einem Bild einer Tabelle

- Öffentlich genutzte Text und Grafiken erkennen - In der Öffentlichkeit genutzen Text und Grapfiken klassifizieren, wie beispielsweise Nummerschilder und Straßenzeichen
- Identifikation des Verfassers - Das bestimmen von indivuellen Charakteristiken des Verfassers des Textes, wie beispielsweise Geschlecht oder Alter
- Messung der Dokumentenqualität
- Handgeschriebenen Text erkennen - Texterkennung bei handgeschrieben Text von individuellen Verfassern

## 3.3 Vorgehensweise in DIA

Die typische Bildverarbeitungspipeline für Dokumente besteht aus drei allgemeinen Schritten.

1. Vorverarbeitung - Das preprocessing der Dokumente beinhaltet das Entfernen von Rauschen und Unschärfe, Korrektur, Entzerrung und Binarisierung.
2. Layout-Analyse - Verstehen der Dokumentstruktur um Regionen die im Interesse (Regions of Interest) liegen zu identifizieren.
3. Erkennung - Extrahieren von den Informationen aus jeder Rol.

## 3.4 Optical Charakter Recognition (OCR)

Eine Technik von DIA ist die optische Zeichenerkennung (OCR), die auf das Erkennen gedruckter Zeichen ausgelegt ist. Die Technologie findet derzeit bei der automatisierten Weiterleitung von E-Mails statt, indem Postleitzahlen von Briefumschlägen geparst werden. Generell kann OCR als möglicherweise gelöstest Problem für maschinell gedruckte und gescannte Bürodokumente in Englischer Schrift betrachtet werden. Es werden Genauigkeiten von mehr als 99% erzielt. Jedoch gibt es über 3000 geschriebene Sprachen und bei der optischen Zeichenerkennung ist bei der Mehrheit der Sprachen ein ungelöstes Problem, welches sich in laufender Forschung befindet.

## 3.5 Funktionsweise von OCR

In 2.1 ist die seqentuelle Workflow von einem Texterkennungsprozesses abgebildet. Das Bild wird als Dokument hochgeladen und mit preprocessing wird das Bild fur den Klassierungsprozess aufbereitet. Abhangig von den bestimmten Schwellwert wird das Bild binarisiert. Texte werden als weie Pixel dargestellt, wahrend alle darumliegende Pixel zu schwarz konvertiert

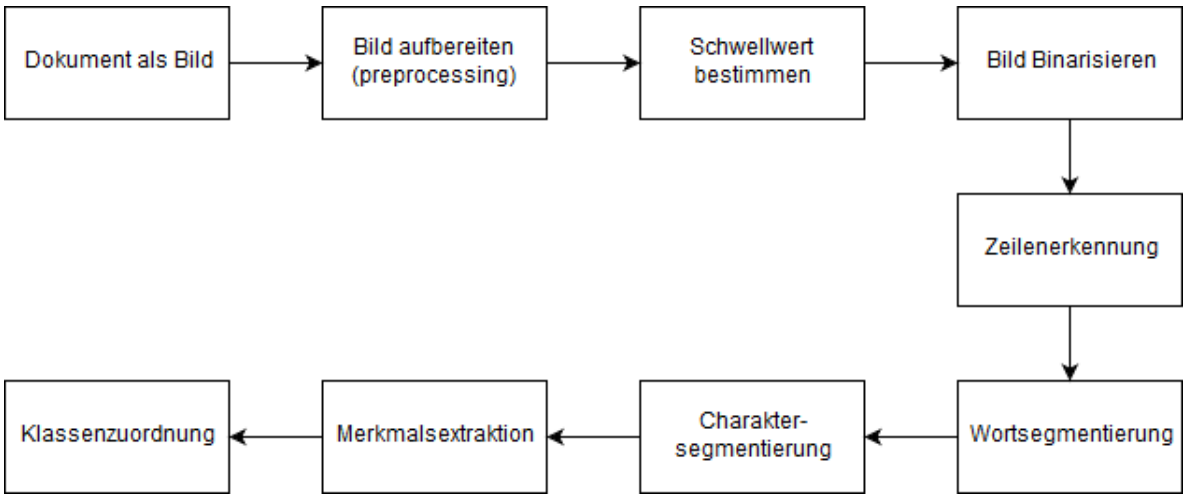



Figure 3.1: Chapter 2 - Wasserfalldiagramm OCR

werden. Somit hebt sich der Text komplett vom Hintergrund ab. Nun konnen Zeilenweise die Worter segmentiert werden, um diese wiederum in Charakter zu segmentieren. Die Charaktere werden als einzelnde als Merkmale extrahiert und klassiziert.

# 4 Grundlagen

Die Lösungsansätze im Automatisierungskontext setzen Kenntnisse über das vorhandene System und Techniken im Bereich des Machine Learning vorraus.

## 4.1 Systembeschreibung

Grundlegend stellt das System eine Schnittstelle zwischen Versicherungsunternehmen und Logististikunternehmen dar. Es existiert als Webanwendung unter einer Domäne und setzt eine Registration des Kunden vorraus, um es zu gebrauchen. Benutzer oder Mandanten verfügen im Kontext der Webanwendung über bestimmte administrative Rechte, die mittels einer Rollenvergabe organisiert werden. Auch die Unterscheidung zwischen Versicherer oder Spediteur findet über die Rollenidentifikation statt.

Bei Erfassen eines Schadens seitens des Spediteurs wird für ihm eine Schadensakte angelegt. Eine Funktion in der Schadensakte beinhaltet das Hochladen von Nachweisen für die Schadensdokumentation, darunter Rechnungsbelege, die es zu erkennen gilt.

### 4.1.1 Technische Details

Die Webanwendung ist mit der Programmiersprache Python 2.7 geschrieben wurden, wobei Schnittstellen bereits mit Python 3 arbeiten. Die Datenübertragung zwischen Frontend und Backend findet über HTTP-Requests statt. TODO: Erklärung HTTP Requests, Abfangen im Backend genauer beschreiben

## 4.2 Künstliche Intelligenz

[Stu16]

## 4.3 Machine Learning

Machine Learning beruht auf den Gebiet der Mustererkennung, die sich mit der automatischen Erkennung von Regelmäßigkeiten in Daten unter Verwendung von Algorithmen befasst. Durch das Erkennen von Regelmäßigkeiten werden Maßnahmen zur Klassifierung der Daten in verschiedene Kategorien ergriffen. [Bis06]

Machine Learning ist die Wissenschaft Computer so zu programmieren, sodass sie in der Lage sind von Daten zu lernen und ist bereits eine weiterforschte Technologie, die Verwendung in hochtechnologischen Produkten findet. Sie ist veranwortlich für die Realiserung von unter anderem die Spracherkennung in Smartphones, das Empfehlen von Videos auf Videoportale und das Ranking von Suchergebnissen im Internet-Suchmaschinen. [Ger17]

### 4.3.1 Spamfilter

Eine der ersten populärsten Anwendungsfälle von Machine Learning ist der Filter um Spam-Mails zu erkennen. Der Spamfilter ist dazu in der Lage seriösen Mails von Spam-Mails zu unterscheiden und als solche zu kennzeichnen. Eine dementsprechende Intelligenz setzt voraus, dass der Software beigebracht werden muss wie es eine Spam-Mail als solche erkennt.

Spam-Mails weisen Regelmäßigkeiten auf, die sie von seriösen Mails unterscheiden. Ein ausgelegter Filter überprüft unter anderem auf Regelmäßigkeiten wie das Fehlen der E-Mail-Adresse im *An: Feld* oder *Cc: Feld*, invalide Angaben und auf eine Verdächtige ID der Nachricht. Ein sehr prägnantes Merkmale einer Spammail ist der Inhalt der Nachricht. Wörter wie *kostenlos, schnelles Geld, reich werden, risikofrei* und *hier klicken* [Spy03] sind Indize dafür, dass es sich um eine Spam-Mail handelt.

Damit ein intelligenter Filter in der Lage ist Spam-Mails zu erkennen, muss er die korrespondierenden Regelmäßigkeiten kennen. Das notwendige Wissen eignet sich die Software über ein Verfahren im Maschinellen Lernen an: Es wird eine Vielzahl von Daten benötigt, die zum Trainieren der Software dient, damit sie Charakteristiken von Spam-Mails erkennen kann. Dieser Datensatz nennt sich *Trainingssatz* und wird zum Einstellen von Parametern eines adaptiven Modells benötigt. Im Fall desSpamfilters gibt es zwei Zielmerkmale:

1. True: Es handelt sich um eine Spam-Mail
2. False: Es handelt sich um keine Spam-Mail

Das Modell adaptiert Erkennungsmuster anhand von den Trainingssatz und kann bei neuen Daten anhand der Charakteristiken das Zielmerkmal prognostizieren. Um Trainingsdaten zu generieren wurden Spam-Mails von Benutzern gekennzeichnet, gesammelt und als Trainingsdaten verwendet.

### 4.3.2 Überführung auf die Belegerkennung

Es soll eine Bilderkennung auf die Hochgeladenen Nachweise praktiziert werden, wobei Rechnungsbelege anhand ihrer Datenmuster zu erkennen sind.

## 4.4 Neuronale Netzwerke

Neuron -> Etwas, was eine Nummer hält. Eine Nummer zwischen 0 und 1. Die Nummer wird als Aktivierung bezeichnet.

Neuron -> Funktion, die alle Gewichte von den den Neuronen der vorherigen schicht als Input nimmt und eine Nummer zwischen 0 und 1 zurückgibt.

Gesamte Netz als funktion mit (x0..., x738) Pixel als input und Vek[y0....y9] als output

Neuronale Netze bestehen aus Neuronen die schichtenweise in *Layern* angeordnet sind. Jedes Neuron einer Schicht ist immer mit allen Neuronen der darauffolgenden Schicht vernetzt.

Gewicht ->

Neuronale Netze sind eine leistungsstarke Technologie zur Klassifizierung visueller Eingaben von Dokumenten. in den frühen 1990 waren Neuronale Netze im Verruf, da sie im Gegensatz zu anderen Klassifikationsstrategien wie "Support Vector Machines" und Bayische Netzwerke schlechter abschnitten[Pat03]. Sie arbeiten mit Vektoren ohne Kenntnisse über die Eingabetopologie

[Her91]

### 4.4.1 Mehrschichtige Netzwerke

Mehrschichtige Netzwerke kennzeichnet sich durch zusätzliche eingebettete verborgenen Schichten, wobei die Anzahl sehr variiert.

Das Erlernen von Charakteristiken tiefer neuronaler Netze arbeiten ähnlich wie das visuelle System von Primaten. Die Wahrnehmung von Objekten verläuft anhand einer Abfolge von Verarbeitungsstufen: Das Erkennen von einzelnen Kanten bis hin zu komplexeren Polygonen [Xav11]. Diese Eigenschaft werden durch verborgene Schichten realisiert.

<bild gemäß https://www.youtube.com/watch?v=aircAruvnKk>

In höheren Schichten werden zunehmend faktorenunabhängig gegenüber Variation wie Kamerabewegungen und sind dadurch flexibler.

Sobald einem Neuronalen Netz mindestens eine Verborgene Schicht hinzugeügt wurden ist, werden sie als Mehrschichtige Netzwerke bezeichnet. Sie kategorisieren sich inherhalb von Deep learning.

#### 4.4.1.1 Aktivierungsfunktion

Die Aktivierungsfunktion definiert den Aktivierungszustand eines Neurons, abhängig von den Eingaben aller anderen verbundenen Neuronen. Ein Neuron kann abhänging von der Aktivierungsfunktion in einem aktiven oder inaktiven Zustand versetzt werden.

Mit der Aktivierungsfunktion wird die Aktivierungsstufe eines Neuronen in ein Ausgangssignal umgewandelt. Es gibt eine Vielzahl von Aktivierungsfunktionen die bei künstlichen

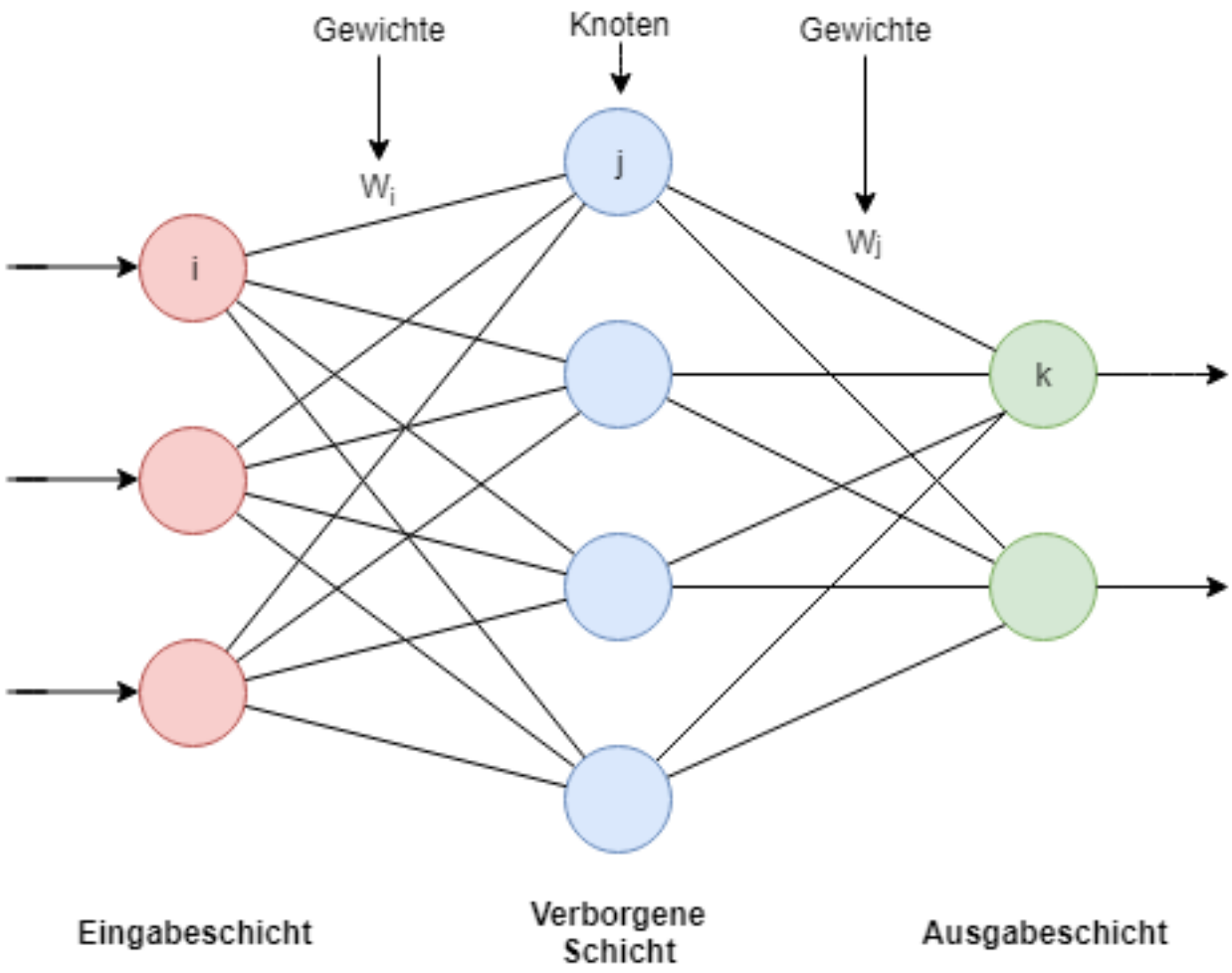



Figure 4.1: Chapter 3 - Beispiel eines tiefen Neuronalen Netzes (Quelle: in Anlehnung an[San17])

neuronalen Netzen angewandt werden. Generell finden Aktivierungsfunktionen viel Anwendung in künstlichen neuronalen Netzen und gelten als populär, da sie als Transferfunktionen verwendet werden. Vorteile bieten Aktivierungsfunktionen durch ihre schnelle Berechenbarkeit der Funktion und ihrer Ableitung.[Bek10]

Jeder Eingabe x i ist ein Gewicht w i zugeordnet. Die Summe aller gewichteten Eingaben x i w i wird dann durch eine nichtlineare Aktivierungsfunktion f geleitet, um den Voraktivierungspegel des Neurons in eine Ausgabe y j umzuwandeln. Der Einfachheit halber wurden die Verzerrungsterme weggelassen. Die Ausgabe y j dient als Eingabe für einen Knoten in der nächsten Schicht. Es stehen verschiedene Aktivierungsfunktionen zur Verfügung, die sich darin unterscheiden, wie sie eine Voraktiverungsstufe einem Ausgabewert zuordnen. Die meistgenutzeste Aktivierungsfunktion ist die Gleichrichterfunktion (ReLu), bei denen Neuronen als gleichgerichtete Lineareinheit bezeichnet wird. In der Ausgabefunktion wird am

häufigsten die Softmaxfunktion verwendet, da hiermit die Wahrscheinlichkeit von Beschriftungen mit mehreren Klassen berechnet werden kann.

### 4.4.2 Faltende Neuronale Netze

Eine der meistgenutzesten und akkuratesten Methode für Bilderkennungsprobleme ist die Anwendung eines faltenden neuronalen netzes. Sie eignen sich besser als vollständtig verbundene Neuronale Netze bei der Dokumentenerkennung. Neuronale Faltungsnetze können die modernste Perfomance und erfordert keine komplexen Methoden[Pat03].

### 4.4.3 Rekurrente Neuronale Netze

Rekurrente Neuronale Netze eignen sich besonders gut für bildbasierte Sequenzerkennung [Bao15]. Es wird eine Architektur neuronale Netzwerkes benötigt, die Sequenzmodellierung, Merksmalsextraktion und Transkription in ein einheitliches Framework integriert.

Es besitzt vier Eigenschaften:

1. Ende-zu-Ende trainierbar, sprich die Komponenten werden seperat trainiert und abgestimmt
2. Es behandelt natürlich Sequenzen in willkürlichen Längen, die keine Zeichen, Segementierung oder horizontale Normierung enthalten
3. Nicht beschränkt auf jedes vordefinierte Lexikon.
4. Es generiert ein viel kleineres Modell, das für reale Anwendungszenarien praktischer ist.

Anders wie bei Faltendenten Neuronalen Netzen, die nützlich bei der Erkennung von einzelnen Objekten, die keine Korrelation zueinander haben, sind Rekurrenter neuronale Netze auf das Erkennen von sequenzähnlichen Objekten nützlich. Das Klassifizieren von einer Reihe aus Objekten ist ein Problem, welches allgemein als Sequenzerkennungsproblem bezeichnet wird. Bei dem Erkennungsprozess können die Objekte von der Länge stark varieren. Im OCR-Kontext gibt es Wörter die wie “Ja”, die lediglich 2 Zeichen aufweisen, während “Allgemeinmedizin” 16 Zeichen aufweist. Die Anzahl der Kombinationen aus Zeichen zu Sequenzen von beispielsweise chinesischen Zeichen, Musiknoten und Wörter kann größer als 1 Million sein, weshalb ein faltendes neuronales Netz durch die hohe Anzahl der Klassen unbrauchbar wäre. Das rekurrente neuronale Netzwerk ist ein weiterer wichtiger Zweig der Familie von tiefen neuronalen Netzwerken und sind hauptsächlich für die Klassifizierung von Sequenzen konzipiert. Eine Eigenschaft von RNN’s ist es, dass es nicht die Position von jedem Element in einem Sequenzobjektbild benötigt beim Training und testen.

Gut für Spracherkennung, Texterkennung. Long short-term memory network

#### Tesseract

Seit Tesseract 4 ist die OCR-Engine auf Basis von neuronalen Netzen mit der LSTM-Methodik umgestiegen. Sie konzentriert sich auf Linienerkennung

## 4.5 Überwachtes Lernen

Überwachtes Lernen ist die häufigste Form des Maschinellen Lernens um ein klassifizierbares System aufzubauen. Im Gegensatz zum unüberwachten Lernen erfordert Überwachtes Lernen einen möglichst großen Datensatz zum Lernen.

Wie lernt ein Neuronales netz? Beim Initliasieren des Netzes ist das Bias und die Gewichte zufällig gewählt. Es wird eine Kostenfunktion zum Lernen des Netzwerkes benutzt. Die Funktion wird benutzt um die Kosten zu berechnen. Es wird von den einzelnen Neuronen in der Ausgabeschicht die Differenz von den tatsächlichen Ergebnissen der Kostenfunktion gebildet, exponeniert und die die Summe gebildet <abbildung>. Das wird die Kosten des einzelnen Trainingsbeispiel genannt. Die Kosten sind niedrig, wenn das Netzwerk das Bild richtig klassifiziert und hoch wenn das Netzwerk das Bild falsch klassifizert. Der Durchschnitt aus der Summe aller Kosten ist ein Messwert für die Klassifikationsrate des Netzwerkes. Um das Minimum der Kostenfunktion mit allen Gewichten und Biases für das Trainingsbeispiel zu finden, wird die Steigung berechnet und abhängig von der Lernrate wird der Input abhängig von der Steigung weiter nach rechts oder links verlagert wodurch nach mehreren Beispielen das lokale Minimum gefunden wird. Es wird der Gradient der Kostenfunktion berechnet um den “steilsten” Weg zum lokalen Minimum zu finden. Der Gradient ist ein Indikator für diesen “steilsten” Weg. Die Gewichte werden als Vektor dargestellt. Der negative Gradientenvektor wird auf den den Gewichtsvektor, der den schnellsten steilsten Abstieg anzeigt. so werden die Kosten niedrig. der Bias stellt in Form eines Neurons mit der Aktivierung “1” eine Konstante dar, die Schichten dazu bringt, dass wenn die Funktion durch den Ursprung verläuft, mehr Flexibiltät in den Verschiebungsprozess des Graphen. Der Bias ist besonders nützlich wenn der Eingabevektor nur nullen enthält, da keine Allgemeingültigkeit verloren geht.

### 4.5.1 Testphase

Die Daten sind mit der korrespondierenden Klasse kategorisiert, so dass das Klassifkationsmodell lernt Datenmuster der entsprechenden Klasse zuzuordnen. Mithilfe des Datensatzes berechnen wir eine objektive Funktion, die den Fehler zwischen den ausgegebenen Bewertungen und dem gewünschten Bewertungsmuster misst. Das Modell ändert dann ihre internen einstellbaren Parameter, um die Fehler zu reduzieren. Die Parameter, auch als Gewichte bezeichnet, werden anhand von Zahlen abgebildet, die die Eingabe/Ausgabe-Funktion des Modells definieren.

### Gradientenverfahren

Um den Gewichtsvektor richtig einzustellen, berechnen der Lernalgorithmus einen Gradientenvektor, der für jedes Gewicht angibt, um welchen Betrag der Fehler sich erhöhen oder verringern würde. der Gewichtsvektor wird dann in entgesetzter Richtung zum Gradientenvektor eingestellt. Die über alle iterierte Trainingsbeispiele gemittelte Zielfunktion kann als eine Art Hügellandschaft im hochdimensionalen Raum von Gewichtsvektoren gesehen werden. der negative Gradientenvektor gibt die Richtung des steilsten Abstieg in dieser Landschaft an und geht dadurch näher an ein Minimum ran, wo der Ausgabefehler im durchschnitt niedrig ist.

In der Praxis wird meistens eine Prozedur angewandt, die sich Gradientenverfahren (engl. SGD) nennt. Das Verfahren besteht aus den Anzeigen des Eingabevektors für ein paar Beispiele aus dem Trainingsdatensatz, berechnen der Ausgaben und der Fehler, dem Berechnen der durchschnittlichen Gradienten und dem ensprechenden Anpassen der Gewichte. Dieser Vorgang wird für viele Beispiele aus dem Trainingsdatensatz wiederholt, bis der Durchschnitt von der Zielfunktionen nicht mehr kleiner wird. Die Prozedur wird Storchastisch genannt, da jeder Satz von den Trainingsdaten eine verrauschte Schätzung des durchschnittlichen Gradienten von jedem Beispiel gibt. Das Verfahren findet in der Regel relativ zu anderen Methoden überraschen schnell einen guten Satz von Gewichten.

Das Gradientenverfahren ist das mit Abstand am häuftigsten verwendete Optimierungsverfahren für die Gewichte-Parametisierung und dient zur Fehlerminimierung bei Prognosen.

Das Verfahren wird mit der Funktion $\nabla E(h(a))$ mathematisch dargestellt.

$$\nabla E(h(a)) = \left[\frac{\partial E(h(a))}{\partial a_1}, ..., \frac{\partial E(h(a))}{\partial a_n}\right]$$

Ziel des Verfahrens ist es Fehler $E$ (Error) zu minimieren. Es wird mit einer Hypothese $h$ mithilfe von allen Attributen $a$ eine Annahme getroffen, um welche Klasse es sich handelt. Attribute werden aus den Trainingsdaten entnommen.

Beispielsweise ist jeder Pixel eines Bildes Teilmenge der Attributmenge $a_1...a_n$, welches an Hypothese $h$ gegeben wird, um eine Vorhersage zu treffen, ob es sich bei dem Bild um eine bestimme Klasse handelt. eine Klasse kann ein Smartphone oder ein Laptop sein.

Die Fehlerfunktion E ergibt sich aus einer Funktionsmenge abhängig von der Attributmenge. Die Funktionen repräsentieren die Ableitungen der Attributmenge $a_1...a_n$ als Vektor.

Mit dem Wissen, was die Daten $a$ tatsächlich sind, berechnet Fehlerfunktion $E$ wie falsch die Hypothese ist. Aus der Fehlerfunktion wird der Gradient gebildet, indem die Fehlerfunktion abgeleitet wird.

$$\nabla a = -\mu * E(h(a))$$

Durch den berechneten Gradienten und der Lernrate $mu$ können die Gewichte entsprechend jedes Attribut der Attributmenge $a_1...a_n$ jeweils so angepasst werden, dass die Fehlerrate reduziert wird.

$$a = a + \nabla a$$

Der Vorgang wird so oft wiederholt, bis die Fehlerrate konvergiert. Lernrate $mu$ sollte so gesetzt sein, dass die Sprünge weder zu groß noch zu klein sind. Das lokale Mimimum könnte bei einer zu großen Lernrate übersprungen werden, sodass der Fehler niemals minimal wird.

#### Fehlerrückführung

Das eigentliche Training mehrschichtiger Architekturen finden anhand storchastischer fallenden Gradienten, die durch das Gradientenverfahren ermittelt wurden sind, statt. Das Fehlerrückführungsverfahren berechnet die Gradienten einer Zielfunktion mit der miteinbeziehung der Gewichte eines mehrschichtigen Modulstapels anhand der Kettenregel für Gradienten. Es kann der Gradient

### 4.5.2 Testphase

Nach dem Trainieren wird die Perfomance durch einem anderen Satz von Daten gemessen, welche Testdatensatz genannt wird. Dies dient zum Testen der Fähigkeit sinnvolle Anworten auf neue Daten Eingaben, die das Modell während des Trainings noch nie gesehen hat - die Verallgemeinerungsfähigkeit[Yan15]

## 4.6 Deep Learning

Der Begriff Deep Learning umsammelt Methoden der künstlichen Intelligenz, die die Funktionsweise des Menschlichen Gehirns bei der Verarbeitung von Daten und der Erstellung von Mustern zu der Entscheidungsfindung imitiert. Deep Learning Methoden werden auf eine Teilmenge von Klassifkatoren aus den Bereich des Maschinellen Lernens angewandt, um die Präzision des Modells zu verbessern.

[Yan15]

## 4.7 Bilderkennung

## 4.8 Daten

Daten existieren als Pixelmatritzen in Form von Bildern. Beim Füttern des Neuronales Netzes wird die Matrix in einen Vektor umgewandelnd, also von einem 2d-array zu einem

1d-array. Jedes Pixel wird jeweils ein Neuron in der Eingabeschicht zugewiesen. bei einem 28x28 Bild müssten dafür 784 Neuronen in der Eingabeschicht exisitieren.

Ein Neuron repräsentiert den Graustufenwert vom Entsprechenden Pixel eines Bildes. 0 = dunkel, 1 = hell.

Es existieren soviele Neuronen in der Ausgabeschicht, wie es zuweisbare Klassen gibt. Die Aktivierung in den Neuronen in der Ausgabeschicht sagt aus, wie hoch das Eingabebild mit der korrespondierenden Klasse übereintrifft.

Spärliche Daten: Die Daten enthalten viele Nullen in den Daten.

Dichte Daten: Viele Werte in den daten sind Nicht Null

## 4.9 Klassen

[**DL˙MS˙RS**]

# 5 Architektur und Training des Faltenden neuronalen Netzes

https://medium.com/datadriveninvestor/five-powerful-cnn-architectures-b939c9ddd57b

#### 5.0.0.1 Wahl der Aktivierungsfunktion

Der Stand der Technik der Nichtlinearität ist die Verwendung **gleichgerichtete Lineareinheiten* (engl. abgekürzt ReLu) zum Training eines neuronalen Netzes zu verwenden [Xav11]. ReLu hat gegen über der Sigmoid Funktion den Vorteil durch ihre simplizität perfomant zu sein. Des Weiteren erzielt sie eine bessere Leistung in voreinding tiefschichtigen Architekturen, da die spärliche Darstellung mit wahren Nullen für natürlich spärliche Daten passend erscheint. Das Training verläuft besser wenn künstliche Neuronen null, also ausgeschaltet sind.

Die Rektifizierende Aktivierungsfunktion wird mit der folgenden Funktion beschrieben:

$$f(x) = max(0, x)$$

Wenn der Input eines Neurons, welcher mit $x$ beschrieben ist, größer als null sein, berechnet der Neuron diesen Wert. Wenn der Input gleich oder kleiner null ist, dann wird das Neuron ausgeschaltet, sprich auf null gesetzt.

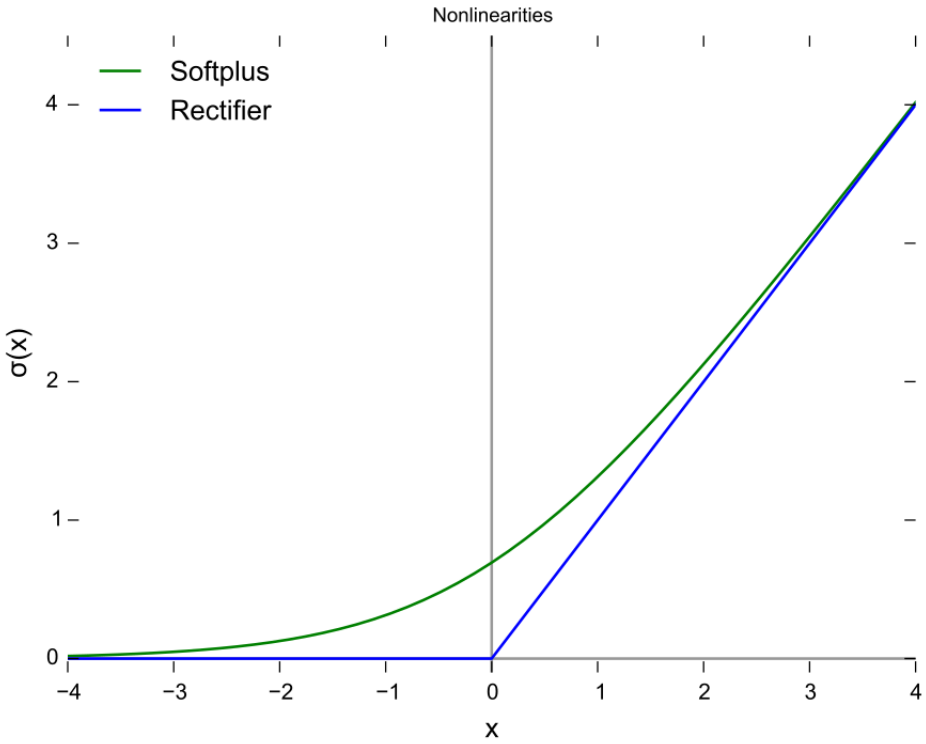



Figure 5.1: Chapter 4: ReLu und Softmax Funktionen im Graph (Quelle: https://de.wikipedia.org/wiki/Datei:Rectifier_and_softplus_functions.svg)

# 6 Implementierung und Architektur

Dieses Kapitel befasst sich mit der Umsetzung der Anforderungen an die Software. Der erste Abschnitt beschreibt den technischen Entwurf der Software. Zur Umsetzung der Anwendung wurden Werkzeuge und Frameworks verwendet, welche in Anhang B aufgelistet und kurz beschrieben sind.

## 6.1 REST-Schnittstelle

Sowohl das Faltende neuronale Netz als Belegerkennung als auch Tesseract als Textextrahierer werden als Servicekomponenten in einer REST-Schnittstelle zur Verfügung gestellt. Eine REST-Schnittstelle eignet sich gut als Architekturansatz, damit verteilte Systeme miteinander kommunizieren können [Fie00]. Die Klassifikatoren liegen hinter den Web-Services und können über HTTP/s-Requests angesprochen werden. Da beide Klassifikatoren ein Bild als Eingabe erwarten, muss der HTTP-Request ein Bild in Hexadezimalnotation in einem JSON mitgeliefert werden.

## 6.2 Schnittstellenbeschreibung

# List of Figures

# List of Tables

# Bibliography


[Bao15] Cong Yao Baoguang Shi Xiang Bai. *An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition*. https://www.researchgate.net/publication/280330424_An_End-to-End_Trainable_Neural_Network_for_Image-Based_Sequence_Recognition_and_Its_Application_to_Scene_Text_Recognition. 2015.

[Bek10] A Vehbi Olgac Bekir Karlik. *Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks*. http://www.cscjournals.org/manuscript/Journals/IJAE/Volume1/Issue4/IJAE-26.pdf. 2010.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. http://cds.cern.ch/record/998831. Feb. 2006.

[Fie00] Roy T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. https://www.ics.uci.edu/~fielding/pubs/dissertation. 2000.

[Ger17] Aurelien Geron. *Hands On Machine Learning with Scikit Learn and TensorFlow*. https://books.google.de/books?hl=de&lr=&id=khpYDgAAQBAJ&oi=fnd&pg=PP1&dq=Hands+On+Machine+Learning+with+Scikit+Learn+and+TensorFlow. 2017.

[Her91] John A. Hertz. *Introduction to the Theory of Neural Computation*. https://www.researchgate.net/publication/200033871_Introduction_To_The_Theory_Of_Neural_Computation. 1991.

[Pat03] John C. Platt Patrice Y. Simard Dave Steinkraus. *Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis*. http://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Fall.2016/pdfs/Simard.pdf. 2003.

[San17] Andrea Mechelli Sandra Vieira Walter Hugo Lopez Pinaya. *Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications*. https://www.researchgate.net/figure/a-The-building-block-of-deep-neural-networks-artificial-neuron-or-node-Each-input-x_fig1_312205163. 2017.

[Spy03] Mike Spykerman. *Typical spam characteristics*. https://www.spamhelp.org/articles/Spam-filter-article.pdf. 2003.

[Stu16] Peter Norvig Stuart J. Russell. *Artificial Intelligence, A Modern Approach*. http://thuvien.thanglong.edu.vn:8081/dspace/handle/DHTL_123456789/4010. 2016.

[Ten19] Christopher Alan Tensmeyer. *Deep Learning for Document Image Analysis*. https://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=8389&context=etd. Apr. 2019.

[Xav11] Yoshua Bengio Xavier Glorot Antoine Bordes. *Deep Sparse Rectifier Neural Networks*. http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf. 2011.

[Yan15] Geoffrey Hinton Yann LeCun Yoshua Bengio. *Deep learning*. https://www.researchgate.net/publication/277411157_Deep_Learning. May 2015.