Handwriting Text Recognition

Fakultät für Informatik und Mathematik
Goethe Universität Frankfurt

Abschlussprojekt

Praktikum Pattern Analysis and Machine Intelligence

vorgelegt von

Martin Ludwig und Pascal Fischer

Abgabetermin	01.05.2020
Betreuer1:	Iullia Pliushch
Betreuer2:	Martin Mundt

Abstract

Im Rahmen dieser Arbeit wurden verschiedene Ansätze für die Erkennung von handschriftlichem Text in Bildern untersucht, wobei alle Ansätze der Methode folgen, das Bild in kleinere Teilbilder (Zeilen, Wörter, Buchstaben) zu zerlegen. Es werden die zwei besten der in dieser Arbeit entwickelten Ansätze erläutert und deren Ergebnisse präsentiert. Zudem wird eine aktuelle Methode zur Erkennung von handschriftlichem Text und eine Methode für die Object Erkennung in Form von Object Detection vorgestellt. Zusätzlich wird ein Problem von aktuellen Datensätzen bzw. Methoden aufgezeigt und ein Ausblick für die Verbesserung von diesen gegeben.

Inhaltsverzeichnis

1 Einleitung **1**

2 Stand der Technik **2**
2.1 Handwriting recognition and language modeling with MXNet Gluon . . 2
2.2 YOLOv1 2
2.2.1 Netzwerk Architektur 4
2.2.2 Loss Funktion 5
2.3 CNN-RNN Hybrid-Netzwerke 7
2.3.1 Architektur-Übersicht 7
2.3.2 Spatial Transformer Network (STN) 7
2.3.3 CNN-RNN-Hybrid 7
2.3.4 CTCLoss 7
2.3.5 Pre-Training 8
2.3.6 Egbenisse 8

3 Methoden **9**
3.1 Textlines Detection 9
3.1.1 Netzwerk Architektur 10
3.1.2 Loss Funktion 10
3.1.3 Training 11
3.1.4 Evaluierung 11
3.2 Word Detection 16
3.2.1 Netzwerk Architektur 17
3.2.2 Loss Funktion 17
3.2.3 Training 17
3.2.4 Evaluation 18
3.3 Handschrifterkennung 20
3.3.1 Netzwerk Architektur (Übersicht) 20
3.3.2 Vorverarbeitung 21
3.3.3 Feature Extraction via CNN 21
3.3.4 BiLSTM 22
3.3.5 Dekodierung (Greedy) 23
3.3.6 Loss-Funktion: CTC-Loss 24
3.3.7 Optimizer: RMS-Prop 25
3.3.8 Training 25
3.3.9 Validierung 26
3.3.10 Beispiel 27
3.3.11 Kombiniertes Ergebnis 27

4 Ausblick **28**
4.1 Ansätze zur Performanz-Steigerung 28
4.2 Mögliche Erweiterungen 29

5 Fazit **30**

Literaturverzeichnis **31**

Anhang **32**

1 Einleitung

Die Erkennung von handschriftlichem Text ist heutzutage sehr gefragt. Zum einen wird die Digitalisierung von handgeschriebenen Texten in Form von Mitschriften, Notizen, etc. für Private Personen, aber auch für Firmen in Form von Notizen erleichtert. Zum anderen gibt es viele alte Bücher wessen Inhalt noch nicht bekannt ist und dessen manuelle Durcharbeitung einen erheblichen Arbeitsaufwand darstellt. Für diese ist die maschinelle Übertragung in digitalen Text eine hervorragende Methode, um anschließend durch weitere Verfahren den Inhalt in bestimmte Kategorien einzuordnen, oder eventuell sogar zusammenzufassen.
In dieser Arbeit werden unterschiedliche Ansätze zur Erkennung von handschriftlichem Text untersucht. Die Unterschiede der Ansätze sind in Abb.1 dargestellt. Der

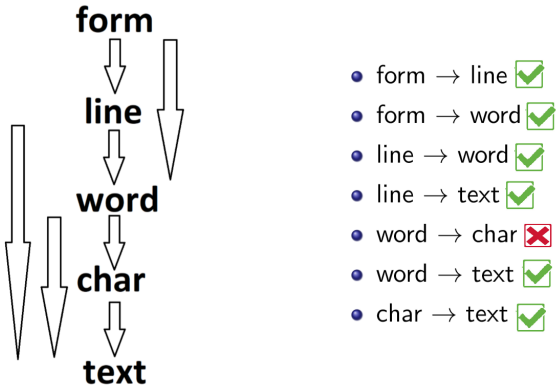


Abbildung 1: Ansätze für die Erkennung von handschriftlichem Text

Grundlegende Gedanke ist hierbei, dass das Eingabebild zuerst in kleinere Teilbilder (Textzeilen, Wörter, Buchstaben) zerlegt wird und anschließend aus diesen der Text extrahiert wird. Es wurden alle möglichen Ansätze ausprobiert (Abb.1 rechts), wobei es für die Erzeugung von Bildern für einzelne Buchstaben keine Datensätze gibt und deshalb dieser Ansatz nicht versucht werden konnte. Die besten Ergebnisse konnten für die Ansätze **Bild** → **Textzeilen** → **Text** und **Bild** → **Textzeilen** → **Wörter** → **Text** erreicht werden. Im weiteren Verlauf dieser Arbeit werden diese beiden Ansätze genauer erläutert und ihr Ergebnis präsentiert. Die zugrundeliegenden Daten stammen aus der *IAM Handwriting Database*[7].

2 Stand der Technik

2.1 Handwriting recognition and language modeling with MXNet Gluon

Der Ansatz von *Jonathan Chung* in *Handwriting recognition and language modeling with MXNet Gluon*[**?**] , unterteilt den Prozess in 3 Schritte: 1. Page segmentation, 2. Line segmentation, 3. Handwriting recognition. Zuerst werden Bereiche identifiziert, in denen sich handschriftlicherText befindet. Dies geschieht mit Hilfe von Objectdetection . Anschließend wird der verbleibende Bereich – ebenfalls unter Verwendung von Objectdetection - in Zeilen handschriftlichen Textes unterteilt. Schließlicht folgt die Erkennung/Übersetzung der Handschrift in digitale Zeichen. In *Page Segmentation with Gluon*[**?**] erläutert Jonathan Chung zwei verwendete Verfahren zur Segmentierung von Text und Zeilen in Bildern. Es werden einmal der MSERs algorithm verwendet, sowie ein deep-CNN-Ansatz zur Objektdetection. Bei beiden Ansätzen wird der Bereich der Bounding Box bestimmt, welche den handgeschriebenen Text, (bzw. die Textzeile) möglichst nah umgrenzen soll.

Für die Vorhersage von Text aus den Bildern der einzelnen Textzeilen, wird eine CNN-BiLSTM Architektur verwendet. Hierbei wird zuerst durch ein CNN (im Beispiel ResNet) eine Reihe an Featuremaps erstellt, welche anschließend direkt in eine BiLSTM Schicht und zusätzlich durch Downsampling erst verkleinert werden und dann in eine weitere BiLSTM-Schicht gegeben werden. Die Ausgaben der beiden Schichten werden konkateniert und anschließend von einem Decoder verarbeitet, welcher dann als Vorhersage eine Verteilung der möglichen Zeichen generiert. Der Text wird anschließend algorithmisch aus der Verteilung erzeugt. Für den letzten Teil sind 3 Algorithmen vorgeschlagen. Das beste Ergebnis erzielt eine Kombination aus Beam Search und Lexicon Search. Beam Search generiert aus der Verteilungsmatrix, welche vom Decoder ausgegeben wird, k Satzvorschläge. Lexicon Search evaluiert diese Vorschläge auf Basis eines Language Models hinsichtlich der „perplexity“ (=Verwirrung) und gibt den Vorschlag mit geringster perplexity aus. Damit wird laut Aussage des Autors eine durchschnittliche Fehlerrate von etwa 19 % erreicht.

2.2 YOLOv1

You Only Look Once[2] (YOLO) ist ein moderner Objekterkennungsansatz des *Facebook AI Reserarch Teams* (FAIR). Der Name kommt da her, dass das Neuronale Netzwerk das Bild nur ein mal betrachtet, um mehrere Objekte zu erkennen.
YOLO schlägt, im Gegensatz zu anderen Ansätzen, vor ein einziges Neuoronales Netzwerk zu verwenden, um die gesamte Ausgabe End-to-End zu generieren. Abb.2 veranschaulicht den Prozess, wie YOLO die Objekte detektiert. Das Eingabebild wird in

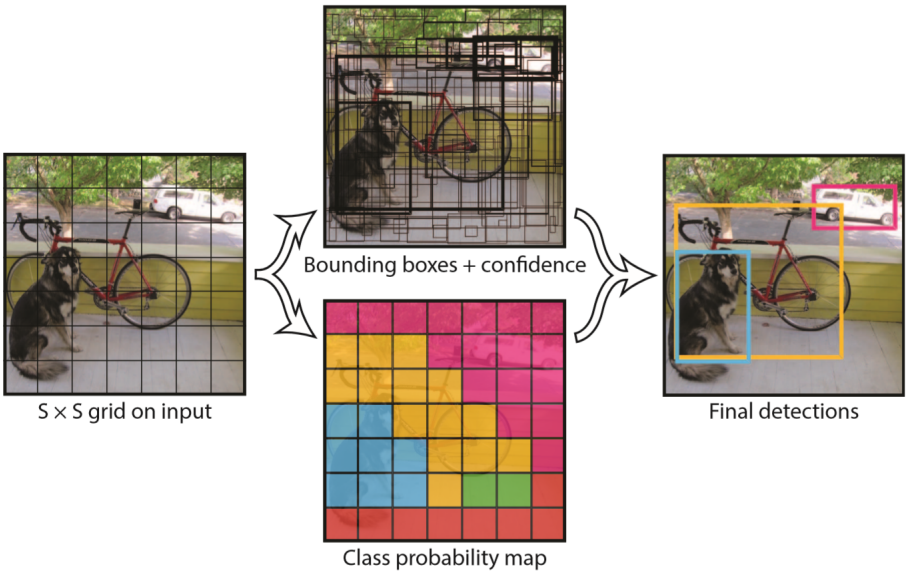


Abbildung 2: Erzeugung von Begrenzungsrahmen mit YOLOv1. Figure 2 in [2]

$S \times S$-Gitter ($S = 7$) unterteilt. Die Gitterzelle, in welcher das Zentrum eines Objekts liegt, ist zuständig für die Erkennung von diesem.
Jede Gitterzelle sagt B-Begrenzungsrahmen (*bounding boxes*, Bsp. $B = 2$) voraus, sowie einen Zuversichtlichkeitswert (*confidence*) zu jedem Begrenzungsrahmen, welcher als Maß für die Korrektheit des vorhergesagten Begrenzungsrahmens dient. Es werden also insgesamt $S \times S \times B$ Begrenzungsrahmen vorhergesagt.
Jeder Begrenzungsrahmen besteht aus 5 Vorhersagewerten: x, y, w, h und $confidence$.

- Die Koordinaten x und y repräsentieren das Zentrum des Begrenzungsrahmens relativ zu den Grenzen der Gitterzelle.
- Die Breite w und die Höhe h repräsentieren die Breite des Begrenzungsrahmens und werden relativ zur Breite bzw. Höhe des Bildes vorhergesagt. Ein Begrenzungsrahmen kann also durchaus größer als eine einzige Gitterzelle sein.
- $confidence$ repräsentiert die $IntersectionOverUnion$ (IOU) zwischen dem vorhergesagten Begrenzungsrahmen und dem Tatsächlichen.

Zudem sagt jede Gitterzelle bedingte Wahrscheinlichkeiten $P(Klasse \mid Objekt)$ voraus, welche Vorhersagen wie hoch die Wahrscheinlichkeit eines Objekts einer Gitterzelle ist, ein Objekt einer bestimmten Klasse zu sein.

In Abb.3 ist die Ausgabe von YOLOv1 dargestellt. Die Ausgabe ist 3-dimensional und hat eine Ausgabegröße von $7 \times 7 \times (2 \times 5 + 20) = 1470$ im Beispiel, wenn die Anzahl verschiedener Klassen 20 ist.

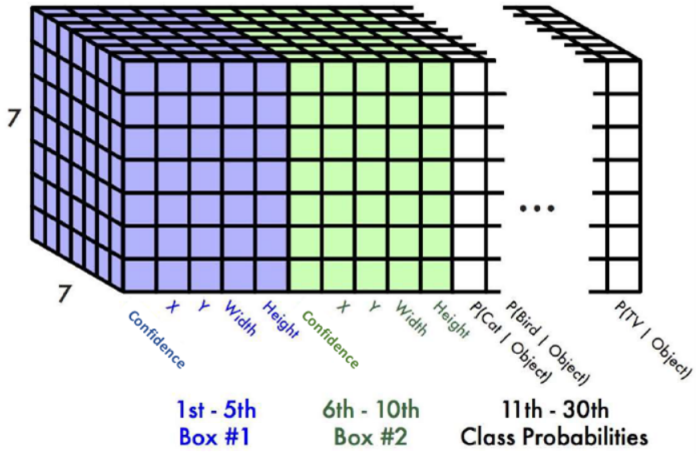


Abbildung 3: YOLOv1 Output Cube. Figure 2b in [3]

2.2.1 Netzwerk Architektur

Abb.4 zeigt die Netzwerk Architektur von YOLOv1. Das Netzwerk besteht aus 24 *Convolution Layers*, sowie 4 *Pooling Layers*, gefolgt von 2 *Fully Connected Layers*. Ab-

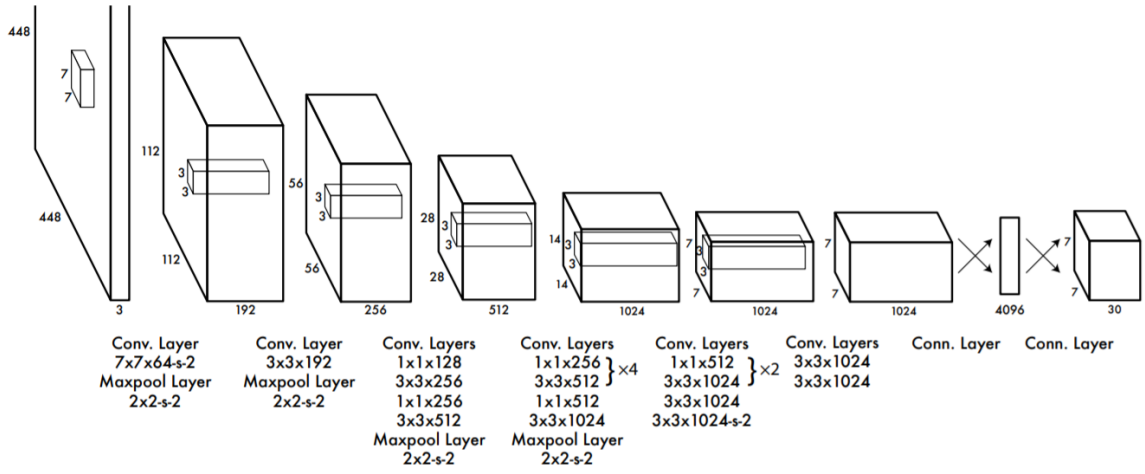


Abbildung 4: YOLOv1 Architektur, Figure 3 in [2]

wechselnde 1×1-Faltungsschichten werden verwendet, um die Anzahl der *Featuremaps* aus der vorherigen Schicht zu reduzieren. Zudem wird auf allen Schichten außer der letzten *ReLu* als Aktivierungsfunktion verwendet und auf das erste Fully Connected Layer *Dropout* angewendet.

Fast YOLO[4] ist eine weitere Architektur mit nur 9 Convolution Layers anstatt 24. Diese Variante gewinnt deutlich an Geschwindigkeit gegenüber der normalen Architektur, verliert im Gegenzug allerdings an Genauigkeit bei der Vorhersage.

$$
\begin{aligned}
&\lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \Big[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \Big] \\
&+ \lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\mathbf{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{1}
$$

Abbildung 5: YOLOv1 Loss Funktion[2]

2.2.2 Loss Funktion

Abb. 5 zeigt die Loss Funktion von YOLOv1. Die 5 Terme der Funktion sind wie folgt zu verstehen:

1. Term: Die x und y Koordinaten des Zentrums eines Begrenzungsrahmens sind relativ zur Gitterzelle, in der sie liegen, gesetzt und normiert (liegen im Intervall $\Big[0, 1\Big]$). Die Indikatorvariable $\mathbb{1}_{ij}^{obj}$ wird genau dann auf 1 gesetzt, wenn sich an Position (i, j) ein Objekt befindet. Die Summe des $quadratischenFehlers$ wird also nur für Begrenzungsrahmen berechnet, an denen sich ein Objekt befindet. Es gilt $\lambda_{\mathbf{coord}} = 5$ um den Fehler des Begrenzungsrahmens höher zu priorisieren.

2. Term: Die Höhe h und Breite w eines Begrenzungsrahmens wird relativ zur Höhe/Breite des gesamten Bildes angegeben und ebenfalls normiert. Die Berechnung des Fehlers von h und w, erfolgt analog zur Berechnung des Fehlers der x und y Koordinaten mit der Ausnahme das hierbei nur die Wurzel von w und w verwendet werden. So kann dem Problem, dass kleine Abweichungen bei großen Begrenzungsrahmen weniger wichtig sind als bei kleinen Begrenzungsrahmen, etwas entgegengewirkt werden.

3./4. Term: Der IOU eines vorhergesagten Begrenzungsrahmens mit dem Tatsächlichen wird in 2 Terme aufgeteilt. der 3.Term präsentiert den Fehler wie gewohnt, für die Positionen, an denen sich ein Objekt befindet. Der 4. Term berechnet das dazugehörige Gegenstück. Die Indikatorvariable $\lambda_{\mathbf{noobj}}$ ist also genau dann 1, wenn sich kein Objekt an dieser Position befindet. Bei Gitterzellen die kein Objekt enthalten, gilt $C_i = 0$. Da allerdings viele Gitterzellen kein Objekt besitzen, wird

der Fehler durch $\mathbb{1}_{ij}^{noobj} = 0.5$ verringert um eine Übersteuerung des Gradienten, der Zellen die Objekte enthalten, zu verhindern.

5. Term: Der Fehler der Klassenwahrscheinlichkeiten wird, an den Stellen wo sich ein Objekt befindet, für alle Klassen berechnet.

2.3 CNN-RNN Hybrid-Netzwerke

In [5] (2015) wurde Handschrifterkennung erstmals mit einem CNN-RNN-Hybrid-Netzwerk durchgeführt und erzielte im Vergleich zu bisherigen Ansätzen sehr gute Ergebnisse. In [1] wurde die vorgeschlagene Architektur verfeinert und um einige Methoden zur Verbesserung des Ergebnisses erweitert.

Mit Greedy-Decoding erzielten die Autoren eine Wort-Fehler-Rate (WER=Word-Error-Rate) von 17.82% und eine Zeichenfehlerrate (CER=Character-Error-Rate) von 5.7%.

2.3.1 Architektur-Übersicht

Eine Übersicht der verwendeten Architektur befindet sich in Abbildung 6.

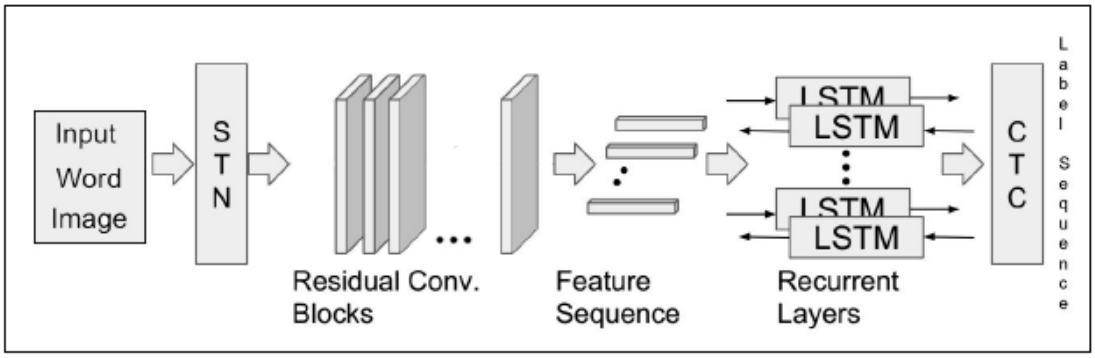


Abbildung 6: Architektur-Übersicht des State-of-the-Art HWR-Systems nach [1]

2.3.2 Spatial Transformer Network (STN)

Das Eingabebild wird an ein Spacial Transformer Network (STN) gegeben, welches trainiert wurde, um geometrische Transformationen auszuführen, die Verzerrungen (z.B. Neigung), die durch die Natur der Handschrift gegeben sind, zu korrigieren.

2.3.3 CNN-RNN-Hybrid

Die Convolutional Layers von ResNet18 wurden verwendet, um Features aus dem Bild zu extrahieren, welche an die bidirektionalen LSTMs weiter geleitet werden.

2.3.4 CTCLoss

Trainiert wurde mittels CTCLoss, welches in weiteren Verlauf der Ausarbeitung genauer beschrieben wird.

2.3.5 Pre-Training

Die verwendete Netzwerkarchitektur enthält anpassbare Parameter im Millionenbereich, wodurch eine große Menge an Daten nötig sind, um eine gute Generalisierung zu erreichen. [1] haben ihr Netzwerk auf synthetischen Daten vortrainiert. Diese wurden auf Basis des IIIT-HWS-Datanesatzes [6] und einem Vokabular von 90.000 Wörtern generiert.

2.3.6 Ergebnisse

Die Ergebnisse, mit einem Vergleich zu anderen Arbeiten, sind in Tabelle 1 dargestellt.

Tabelle 1: Ergebnisse aus [1]

Method	Seg.	Decoding	WER	CER
Krishnan et al. [35]	Word	Unconstrained	16.19	6.34
Wigington et al. [18]			19.07	6.07
Sueiras et al. [14]			23.8	8.8
This Work			**12.61**	**4.88**
Sun et al. [15]		Full-Lexicon	11.51	-
Wigington et al. [18]			5.71	3.03
Stuner et al. [25]			5.93	2.78
Poznanski et al. [20]			6.45	3.44
This Work			**4.80**	**2.52**
Sueiras et al. [14]		Test-Lexicon	12.7	6.2
Wigington et al. [18]			4.97	2.82
Krishnan et al. [21]			6.69	3.72
Krishnan et al. [35]			5.10	2.66
This Work			**4.07**	**2.17**
Pham et al. [16]	Line	Unconstrained	35.1	10.8
Puigcerver et al. [19]			18.4	5.8
Chen et al. [17]			34.55	11.15
Krishnan et al. [35]			32.89	9.78
This Work			**17.82**	**5.7**

Diese Ergebnisse werden als Grundlage für Vergleiche herangezogen. Die Vergleichbarkeit ist nicht zu 100% gegeben, da wir neben der Handschrifterkennung auch die Objekterkennung der Zeilen bzw. Wörter mit dabei haben. Allerdings liefert dies einen guten Anhaltspunkt, um das Ergebnis ohne die Objekterkennung zu vergleichen.

3 Methoden

3.1 Textlines Detection

Im ersten Schritt um Text aus Bildern zu erkennen, muss erst einmal der Text in den Bildern detektiert werden. Abb. 7 zeigt eine Lokalisierung von handgeschriebenen Satz-

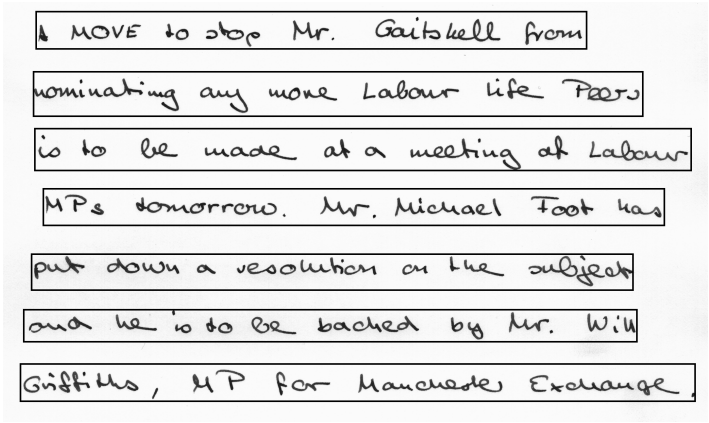


Abbildung 7: Beispielausgabe Bild mit Begrenzungsrahmen für handgeschriebene Textzeilen

zeilen in einem Bild durch Begrenzungsrahmen.
Das Eingabebild wird bei der Vorhersage in S-Bildzeilen ($S = 50$) unterteilt. Die Bildzeile, in welcher das Zentrum einer Textzeile liegt, ist bei der Erzeugung des Begrenzungsrahmens, für die Textzeile zuständig.
Jede Bildzeile sagt genau einen Begrenzungsrahmen voraus, sowie einen Zuversichtlichkeitswert (C), welcher ein Maß für die Korrektheit von diesem darstellt. Es werden also genau S-Begrenzungsrahmen vorhergesagt. Jeder Begrenzungsrahmen besteht aus

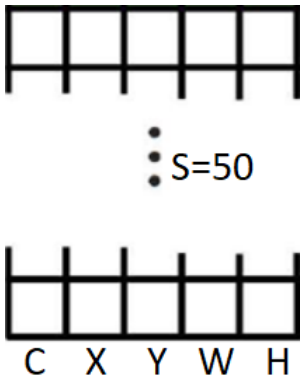


Abbildung 8: Line Detection Output

5 Vorhersagewerten: x, y, w, h und c, die Bedeutung ist identisch wie bei YOLOv1[2]. Der große Unterschied liegt darin, dass es keine bedingten Klassenwahrscheinlichkeiten gibt, da nur Textzeilen vorhergesagt werden und keine anderen Objekte. Zudem Wird über die gesamte Breite nach Textzeilen gesucht und das Bild nur vertikal unterteilt, was zu einer 2-dimensionalen Ausgabe mit einer Ausgabegröße von $50 \times 5 = 150$ führt.

3.1.1 Netzwerk Architektur

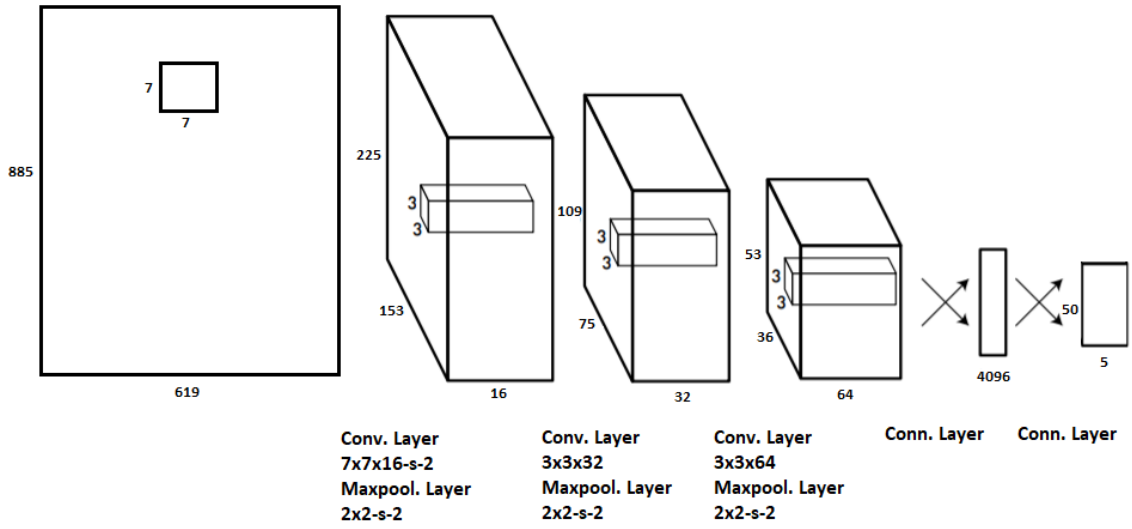


Abbildung 9: Line Detection Netzwerk Architektur

Abb.9 zeigt die entwickelte Netzwerk Architektur für die Erzeugung der Begrenzungsrahmen von Textzeilen eines Bildes. Das Netzwerk besteht aus 3 *Convolution Layers*, sowie 3 *Pooling Layers*, gefolgt von 2 *Fully Connected Layers*. Auf allen Schichten wird *ReLu* als Aktivierungsfunktion angewendet, mit Ausnahme der letzten Schicht bei der *Sigmoid* verwendet wird. Das Eingabebild wird anfänglich auf eine Größe von 885×619 Pixeln skaliert. Auf das erste Fully Connected Layer wird *Dropout* angewendet, um beim Trainingsvorgang einer möglichen Überanpassung entgegenzuwirken.

3.1.2 Loss Funktion

$$
\begin{aligned}
&\lambda_{\mathbf{coord}} \sum_{i=0}^{S} \mathbb{1}_i^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&\quad + \lambda_{\mathbf{coord}} \sum_{i=0}^{S} \mathbb{1}_i^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&\qquad + \sum_{i=0}^{S} \mathbb{1}_i^{obj} \left(C_i - \hat{C}_i \right)^2 \\
&\qquad + \lambda_{\mathbf{noobj}} \sum_{i=0}^{S} \mathbb{1}_i^{noobj} \left(C_i - \hat{C}_i \right)^2
\end{aligned}
\tag{2}
$$

Abbildung 10: Line Detection Loss Funktion

Die *Loss Funktion* (Abb.10), welche an die Loss Funktion von YOLOv1[2] angelehnt wurde, besteht aus 4 Termen. Der wesentliche Unterschied besteht darin, dass nicht über die unterschiedlichen Begrenzungsrahmen summiert wird, da pro Textzeile nur ein einziger Begrenzungsrahmen vorhergesagt wird. Zudem fällt der 5. Term komplett weg, da keine Klassenwahrscheinlichkeiten vorhanden sind. Der Parameter C_i ist also in diesem Fall mehr ein Maß für die Wahrscheinlichkeit, dass sich in Bildzeile i das Zentrum einer Textzeile befindet, als ein Maß für den IOU einer vorhergesagten Textzeile mit einer Tatsächlichen. Anzumerken ist das die x und y Koordinaten zwar relativ zur Bildzeile normiert angegeben werden, da aber die Breite der Bildzeile gleich der Breite des Gesamtbildes ist, wird die x Koordinate auch gleichzeitig relativ zur Breite des Bildes angegeben. Zudem gilt $\lambda_{\mathbf{coord}} = 5$, sowie $\lambda_{\mathbf{coord}} = 0.5$ weiterhin.

3.1.3 Training

Das Künstliche Neuronale Netz wird 100 Epochen mit einer Lernrate von $1e{-}4$ (0.0001) trainiert. Verwendet werden hierbei die Handschriftlichen Dokumente aus der *IAM ON-Line Handwriting Database*[7]. Die Bilder wurden vorher in Trainingsdatensatz (1319 Bilder) und Validierungsdatensatz (220 Bilder) aufgeteilt. Als Backpropagation Algorithmus wird *Stochastic Gradient Descent* benutzt.

3.1.4 Evaluierung

Das Ergebnis des Trainings wird durch 4 Funktionen getestet, welche die Funktionalität des Netzes aus mehreren Blickwinkeln betrachtet, um einen guten Gesamteindruck der über die Leistungsfähigkeit von diesem zu gewinnen. Einerseits muss das Neuronale Netz möglichst genau das Zentrum eines Begrenzungsrahmens sowie seine Breite und Höhe vorhersagen. Dazu wird der *Intersection over Union* (IoU) verwendet. Andererseits müssen die richtigen Begrenzungsrahmen gewählt werden. Hierfür helfen *Accuracy*, *Precision* und *Recall*. Abb.11 verdeutlicht was der *IoU* zweier Begrenzungsrahmen be-

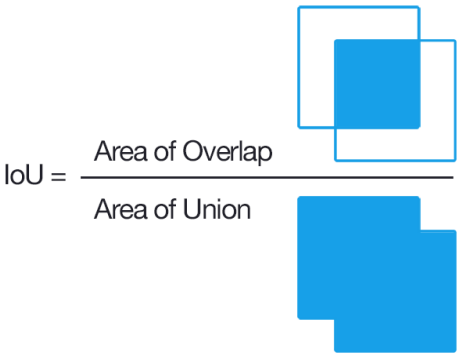


Abbildung 11: Optische Darstellung der Berechnung des IoU, Figure2 in [8]

deutet und hilft ein intuitives, geometrisches Verständnis zu entwickeln. Mathematisch

kann der IoU für zwei Begrenzungsrahmen B_1, B_2 in Form von Punktmengen definiert werden, um eine sauberere mathematische Definition zu erhalten:

$$\textbf{IoU} = \frac{B_1 \cap B_2}{B_1 \cup B_2} \tag{3}$$

Er gibt also an, wie groß der Anteil der sich überlappenden Fläche zur Gesamtfläche zweier Begrenzungsrahmen ist und bietet somit ein gutes Maß für die Genauigkeit des vorhergesagten Begrenzungsrahmens.

Es gibt 4 Klassen in die ein vorhergesagtes Element (Bsp. Begrenzungsrahmen) bei

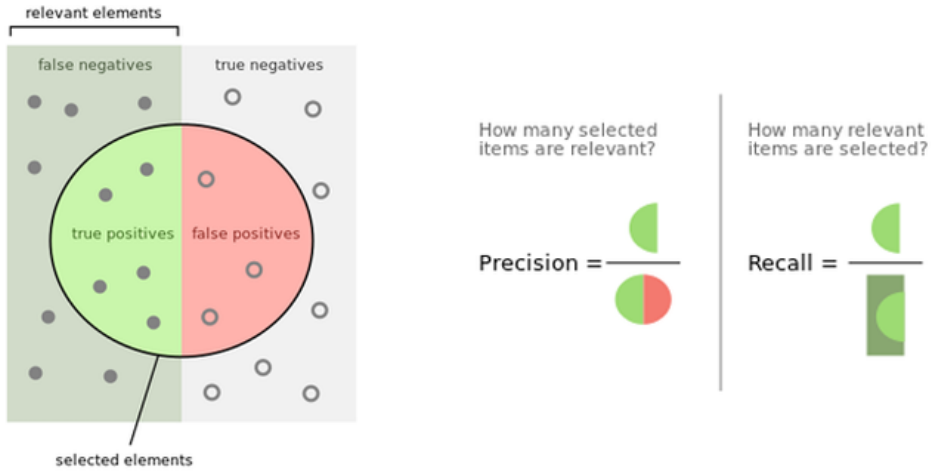


Abbildung 12: Precision und Recall [9]

der Evaluierung fallen kann (Abb.12 links). Im weiteren Verlauf werden aus Gründen der besseren Übersichtlichkeit folgende Abkürzungen für diese verwendet:

- **TP** = true positives (Es wurde ein Begrenzungsrahmen vorhergesagt, wo sich tatsächlich einer befindet)
- **FP** = false positives (Es wurde ein Begrenzungsrahmen vorhergesagt, wo sich keiner befindet)
- **TN** = true negatives (Es wurde kein Begrenzungsrahmen vorhergesagt, wo sich keiner befindet)
- **FN** = false negatives (Es wurde kein Begrenzungsrahmen vorhergesagt, wo sich tatsächlich einer befindet)

Die *Accuracy* (Genauigkeit) ist der Anteil an richtig vorhergesagten Elementen (Begrenzungsrahmen bzw. kein Begrenzungsrahmen) und wird über die vorher definierten

Abkürzungen wie folgt definiert:

$$\textbf{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

Sie bietet also ein Maß für die allgemeine Richtigkeit der vorhergesagten Begrenzungsrahmen im Bezug auf die Bildzeilen. Da allerdings überwiegend Bildzeilen vorhanden sind, welche kein Zentrum eines Begrenzungsrahmens enthalten, reicht sie nicht aus um einen guten Überblick über die Richtigkeit der tatsächlichen Begrenzungsrahmen zu erhalten.
Deshalb werden zusätzlich *Precision* und *Recall* verwendet. Die *Precision* liefert einen Guten Überblick über den Anteil der vorhergesagten Begrenzungsrahmen, welcher tatsächlich ein Begrenzungsrahmen ist:

$$\textbf{Precision} = \frac{TP}{TP + FP} \tag{5}$$

Der *Recall* hingegen zeigt wie viele tatsächliche Begrenzungsrahmen auch als solche vorhergesagt wurden:

$$\textbf{Recall} = \frac{TP}{TP + FN} \tag{6}$$

Durch die Verwendung von *Precision* und *Recall* wird ein sehr guter Gesamteindruck über die Fähigkeit des Neuronalen Netzes, tatsächliche Begrenzungsrahmen richtig vorherzusagen, ermöglicht.

Abb. zeigt die ersten Ergebnisse des Trainings im Bezug auf den Loss und Abb. im Bezug auf IoU, Accuracy, Precision und Recall für alle Epochen. Das Neuronale Netz

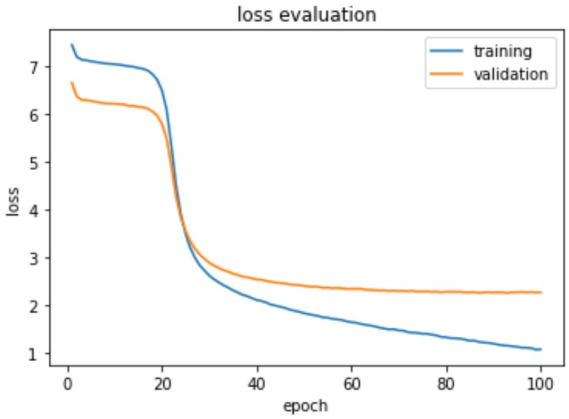
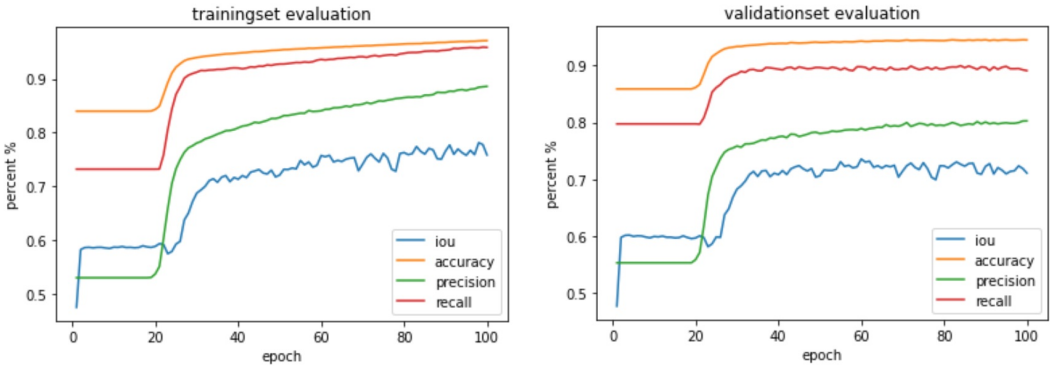


Abbildung 13: Loss Evaluation

lernt in den ersten Epochen gut, befindet sich dann aber für ca. 20 Epochen auf einem

Plateau. Anschließend lernt das Netz in ca. 5 Epochen rasant schnell, flacht dann wieder radikal ab und landet in einem lokalen Minimum. Der Trainingsdatensatz wird im Vergleich zum Testdatensatz aber noch weiter gelernt.

Das beste Ergebnis des Validierungsdatensatzes im Bezug auf den IoU wird bei Epo-



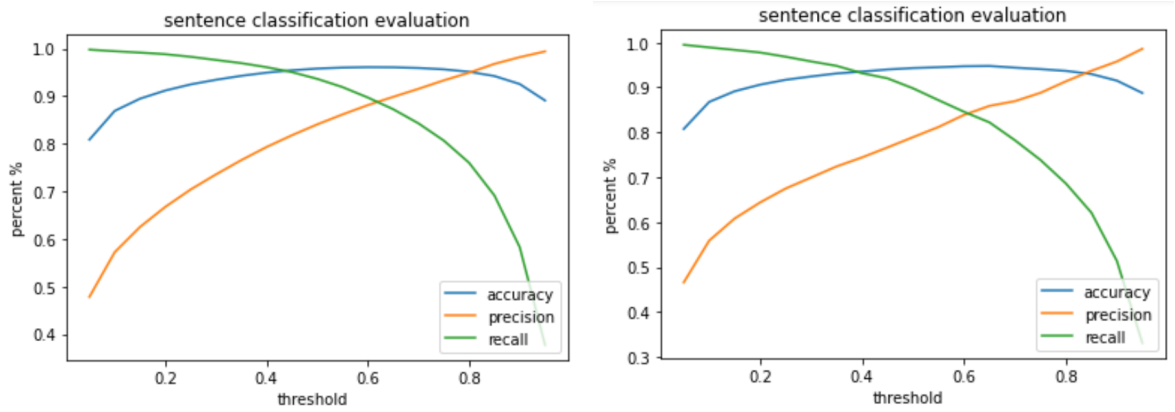
(a) Trainingsdatensatz (b) Validierungsdatensatz

Abbildung 14: Trainings Evaluation

che 60 gefunden. Die Gewichte von Epoche 60 wurden gespeichert und im weiteren Verlauf verwendet: Um zu einem besseren Ergebnis zu kommen wird der Schwellen-

Daten	IoU	Accuracy	Precision	Recall
Training	0.79	0.96	0.84	0.94
Validierung	0.77	0.94	0.79	0.9

wert (*Threshold*) für C, bei dem ein Begrenzungsrahmen vorhergesagt wird analysiert. Im bisherigen Verlauf wurde C einfach so interpretiert, dass bei einem Schwellenwert $C \geq 0.5$ sich ein Begrenzungsrahmen an der Bildzeile befindet und bei $C < 0.5$ keiner. Die Vorhersage wurde für unterschiedliche Schwellenwert in $[0, 1]$ getestet:



(a) Trainingsdatensatz (b) Validierungsdatensatz

Abbildung 15: Evaluation Threshold für Begrenzungsrahmen Vorhersage

Der Punkt an dem sich die Precision und Recall beim Validierungsdatensatz schneiden, ist ca. bei Threshold 0.6 und wird als Optimum gewählt:

Daten	IoU	Accuracy	Precision	Recall
Training	0.79	0.96	0.88	0.90
Validierung	0.77	0.95	0.84	0.85

Zum Schluss wird ein anderer Algorithmus zur Auswahl der Begrenzungsrahmen verwendet, welcher das folgende Problem lösen soll:

Abbildung 16: Problemdarstellung mehrere ähnliche Boxen für ein Objekt

Für ein Objekt (Textzeile) werden mehrere ähnliche Begrenzungsrahmen vorhergesagt. Eine Anlehnung an den *Non-maximum Suppresion Algorithmus* wird verwendet, um möglichst nur den besten vorhergesagten Begrenzungsrahmen zu übernehmen.

Algorithm 1 Calculate Bounding Boxes

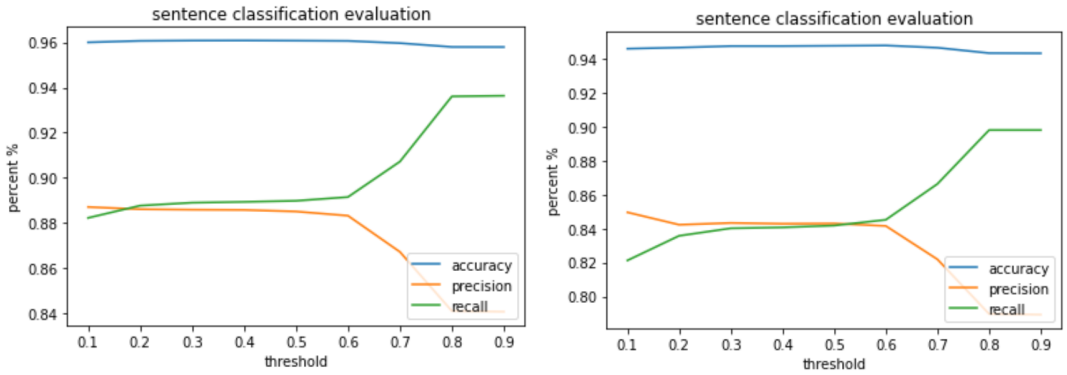
Require: List A with predicted bounding boxes $b = [c, x, y, w, h]$, $E = [\ ]$, thresholds $s_1\ ,\ s_2 \in [0, 1]$

Ensure: right bounding boxes

```
while ∃ b ∈ A, with b_c ≥ s_1 do
    best = b ∈ A, with b_c ≥ c_c ∀ c ∈ A
    E.append(best)
    A.remove(best)
    for each: b ∈ A do
        if IoU(best, b) ≥ s_2 then
            A.remove(b)
        end if
    end for
end while
return E
```

Der Algorithmus liefert für $s_1 = s_2 = 0.5$ das beste Ergebnis. In Abb. 17 gilt $threshold = s_2$, sowie $s_1 = 0.5$, der Schnittpunkt von Precision und Recall im Ergebnis des Validierungsdatensatz wurde wieder als bestes Ergebnis gewählt.

Daten	IoU	Accuracy	Precision	Recall
Training	0.79	0.96	0.89	0.89
Validierung	0.77	0.95	0.84	0.84



(a) Trainingsdatensatz (b) Validierungsdatensatz

Abbildung 17: Evaluierung non-maximum suppression

3.2 Word Detection

Im zweiten Schritt werden aus den Bildern der Textzeilen einzelne Wörter detektiert. Abb. 18 zeigt eine Lokalisierung von handgeschriebenen Wörtern in einem Bild einer

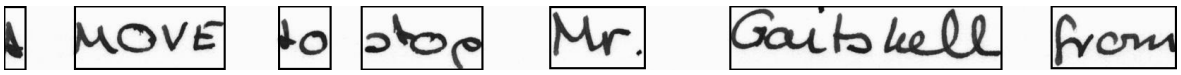


Abbildung 18: Beispielausgabe Bild von einer Textzeile mit Begrenzungsrahmen für handgeschriebene Wörter

Textzeile durch Begrenzungsrahmen.

Das Eingabebild wird bei der Vorhersage in S-Bildspalten ($S = 30$) eingeteilt. Die Spalte, in welcher die linke horizontale Linie des Begrenzungsrahmens liegt, ist bei der Erzeugung des Begrenzungsrahmens, für das Wort zuständig.
Jede Spalte sagt auch hier genau einen Begrenzungsrahmen voraus, sowie einen Zuversichtlichkeitswert (C), welcher wieder ein Maß für die Korrektheit von diesem darstellt. Die Anzahl der vorhergesagten Begrenzungsrahmen beträgt also ebenfalls S. Die Be-

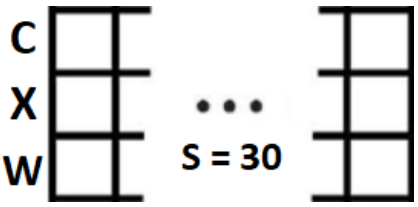


Abbildung 19: Word Detection Output

grenzungsrahmen bestehen hierbei aus 3 Vorhersagewerten c, x und w, wobei x hierbei die horizontale Position der linken horizontalen Linie des Begrenzungsrahmens bezeichnet und w die Breite. Als fester Punkt in vertikaler Richtung wird der untere Rand (also Stelle 0) gesetzt. Die x Koordinate kann also direkt als die linke untere Ecke des

Begrenzungsrahmen definiert werden. Zudem wird die Höhe des Begrenzungsrahmens mit der Höhe des Bildes der Textzeile gleichgesetzt. Das Bild wird bei der Vorhersage also in Spalten aufgeteilt, was zu einer 2-dimensionalen Ausgabe mit einer Größe von $30 \times 3 = 90$ führt.

3.2.1 Netzwerk Architektur

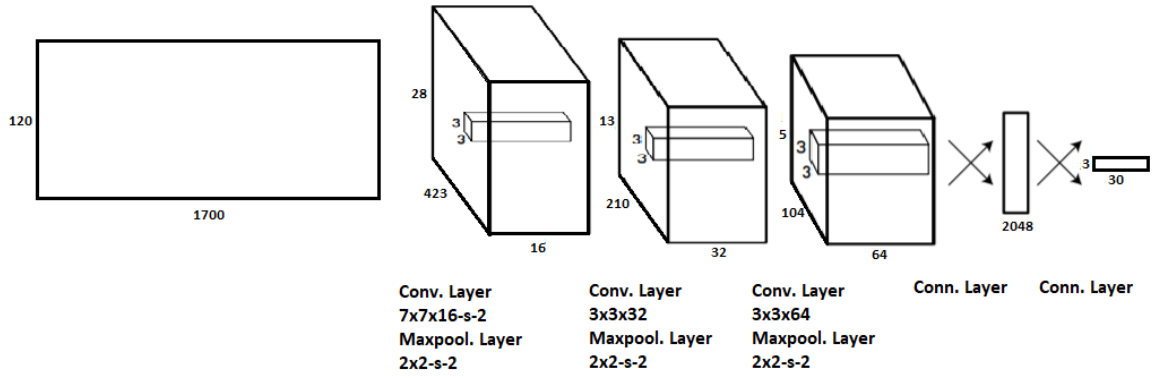


Abbildung 20: Word Detection Netzwerk Architektur

Die verwendete Netzwerk Architektur für die Lokalisierung der Wörter, ist in Abb. 20 abgebildet. Die Architektur ähnelt, der die bei der Textzeilenerkennung verwendet wurde. Das Netzwerk besteht aus 3 *Convolution Layers*, sowie 3 *Pooling Layers*, gefolgt von 2 *Fully Connected Layers*. Auf allen Schichten wird *ReLu* als Aktivierungsfunktion angewendet, mit Ausnahme der letzten Schicht bei der *Sigmoid* verwendet wird. Das Eingabebild wird anfänglich auf eine Größe von 200×1700 Pixeln skaliert, diese Größe wurde aus einer Analyse des Durchschnittswerts aller beim vorhandenen Bilder im Datensatz ermittelt. Auf das erste Fully Connected Layer wird wie gewohnt *Dropout* angewendet, um einer möglichen Überanpassung entgegenzuwirken.

3.2.2 Loss Funktion

Es handelt sich bei der Loss Funktion (Abb.21) für die Worterkennung um die Loss Funktion für die Textzeilenerkennung, wobei im ersten Term der Fehler für die y Koordinate und im zweiten Term der für die Höhe (h) des Bildes entfernt wurde. Die Werte für die Parameter $\lambda_{\mathbf{coord}}$ (5) und $\lambda_{\mathbf{noobj}}$ (0.5) bleiben ebenfalls erhalten.

3.2.3 Training

Das Künstliche Neuronale Netz wird 30 Epochen mit einer Lernrate von $1e-3$ (0.001) trainiert. Verwendet werden hierbei die Handschriftlichen Textzeilen aus der *IAM ON-Line Handwriting Database*[7]. Die Bilder der Textzeilen wurden vorher in Trainingsdatensatz (11507 Bilder) und Validierungsdatensatz (1846 Bilder) aufgeteilt. Als Backpro-

$$\begin{aligned}
&\lambda_{\mathbf{coord}} \sum_{i=0}^{S} \mathbb{1}_i^{obj} \left(x_i - \hat{x}_i \right)^2 \\
&+ \lambda_{\mathbf{coord}} \sum_{i=0}^{S} \mathbb{1}_i^{obj} \left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 \\
&+ \sum_{i=0}^{S} \mathbb{1}_i^{obj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\mathbf{noobj}} \sum_{i=0}^{S} \mathbb{1}_i^{noobj} \left(C_i - \hat{C}_i \right)^2
\end{aligned} \tag{7}$$

Abbildung 21: Word Detection Loss Funktion

pagation Algorithmus wird wieder *Stochastic Gradient Descent* benutzt. Zudem wurde für die Begrenzungsrahmen, aus den gegebenen .xml Dateien für die einzelne Bilder der handschriftlichen Texte, zu jeder Textzeile eigene .xml Dateien generiert.

3.2.4 Evaluation

Das Ergebnis des Trainings wird, wie bei Der Textzeilenerkennung evaluiert. Der *Loss* ist in Abb.23 dargestellt. Der Künstliche Neuronale Netz lernt den Validierungsdaten-

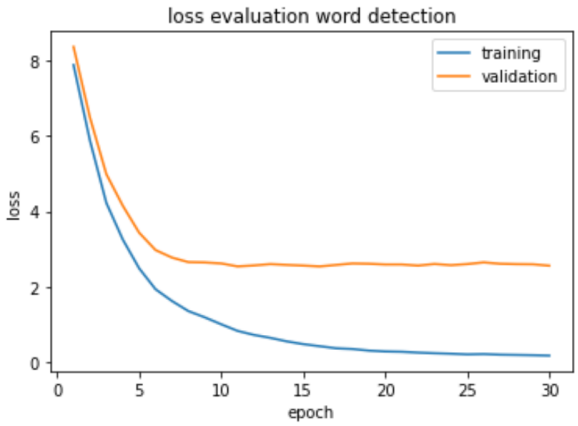
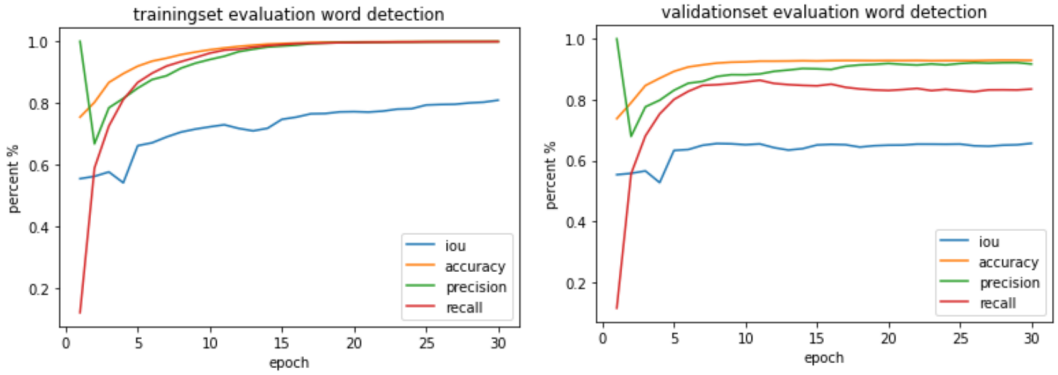


Abbildung 22: Loss Evaluation

satz in den ersten Epochen sehr gut, der Lernerfolg flacht dann aber allmählich ab und stagniert. Der Trainingsdatensatz lernt wie gewohnt noch etwas länger und stagniert dann ebenfalls. Das beste Ergebnis des Validierungsdatensatzes im Bezug auf den IoU wird bei Epoche 30 gefunden. Die Gewichte von Epoche 30 wurden gespeichert und werden für die weitere Evaluierung weiter verwendet:

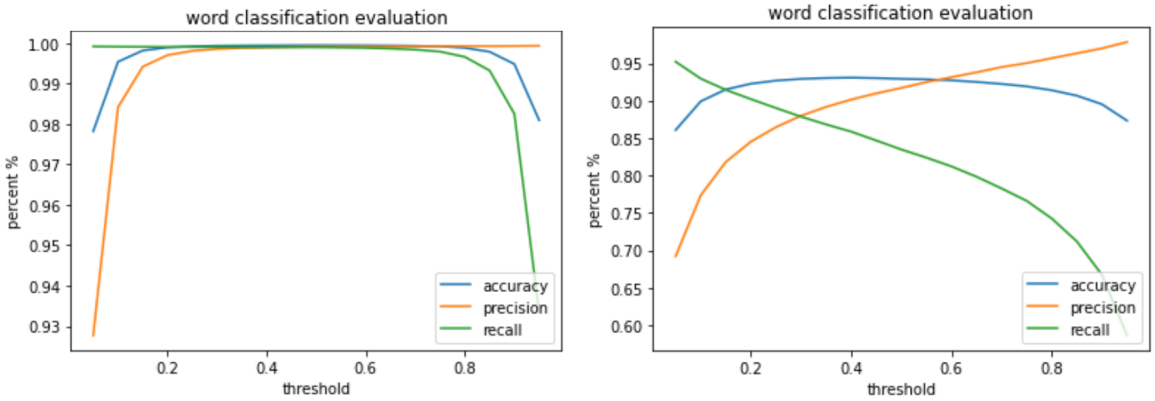
Daten	IoU	Accuracy	Precision	Recall
Training	0.81	1.00	1.00	1.00
Validierung	0.68	0.93	0.92	0.83



(a) Trainingsdatensatz (b) Validierungsdatensatz

Abbildung 23: Trainings Evaluation

Die Vorhersage wurde mit den Gewichten von *Epoche 30* für unterschiedliche Schwellenwerte in $[0, 1]$ getestet:



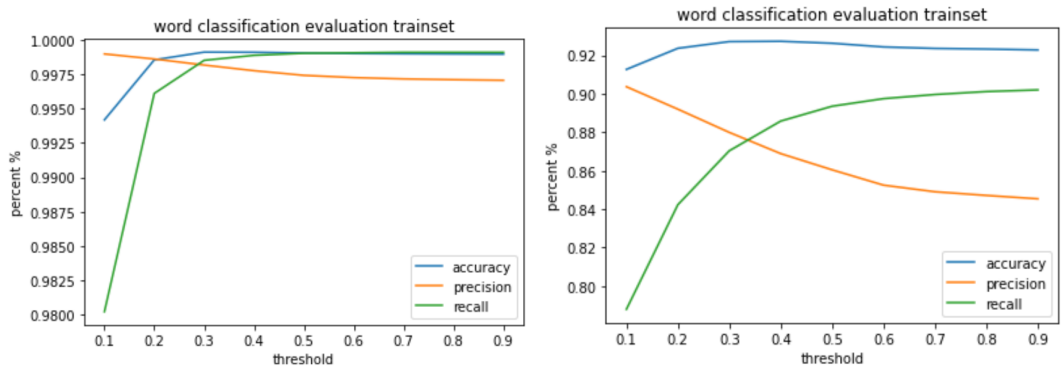
(a) Trainingsdatensatz (b) Validierungsdatensatz

Abbildung 24: Evaluation Threshold für Begrenzungsrahmen Vohersage

Bei der Schnittpunkt von Precision und Recall in der Auswertung des Validierungsdatensatzes wird bei $threshold = 0.3$ gefunden und als beste Vorhersagte gewählt:

Daten	IoU	Accuracy	Precision	Recall
Training	0.81	1.00	1.00	1.00
Validierung	0.68	0.93	0.88	0.88

Der an den *Non-Maximum Suppression Algorithmus* angelehnte Algorithmus liefert für $s_1 = 0.2,\ s_2 = 0.35$ das beste Ergebnis. In Abb.25 gilt $threshold = s_2$, der Schnittpunkt von Precision und Recall im Ergebnis für den Validierungsdatensatz, wurde wieder als bestes Ergebnis gewählt.



(a) Trainingsdatensatz (b) Validierungsdatensatz

Abbildung 25: Evaluierung non-maximum suppression

Daten	IoU	Accuracy	Precision	Recall
Training	0.81	1.00	1.00	1.00
Validierung	0.68	0.93	0.88	0.88

3.3 Handschrifterkennung

Wir haben verschiedene Ansätze zur Handschrifterkennung untersucht. Der ursprünglich favorisierte Ansatz, den Text auf Zeichenebene zu digitalisieren, schlug fehl, da der Datensatz einen zu großen Anteil falsch klassifizierter Zeichen beinhaltet. Sehr oft überspannten die vorgegebenen Begrenzungsrahmen mehrere Zeichen oder ganze Wörter, obwohl sie nur einzelne Zeichen beinhalten sollten.

Daraufhin wurde der Ansatz der Erkennung von gesamten Textzeilen verfolgt, da bei der Objekterkennung die Ergebnisse für Textzeilen besser waren, als für einzelne Wörter.

3.3.1 Netzwerk Architektur (Übersicht)

In Abbildung 26 ist eine Übersicht der von uns verwendeten Netzwerk-Architektur dargestellt.

Die Eingabe ist ein Bildausschnitt einer Seite mit handschriftlich geschriebenen Text. Dies kann entweder eine Zeile oder ein Wort sein.

Alle Eingaben werden zunächst in ein einheitliches Format von 512x64 Pixel gebracht.

Mittels eines CNN's werden nun lokale Features aus dem Bild generiert. Grundsätzlich könnte das nachfolgende Bi-LSTM auch direkt die Pixelspalten als Eingabe erhalten, doch mit extrahierten Features wurden bessere Ergebnisse erzielt. Pro Pixelspalte werden 64 Features als Ausgabe erzeugt.

Die Eingabe für das Bi-LSTM besteht somit aus 512 Zeitschritte zu je 64 Features.

Das BiLSTM erzeugt eine Ausgabe der gleichen Größe (512x64).

Die Ausgabe jedes Zeitschrittes wird durch ein „fully connected layer“ (FC-Layer) mit 79+1 Ausgabeneuronen geleitet. 79 ist die Anzahl der im Datensatz enthaltenen

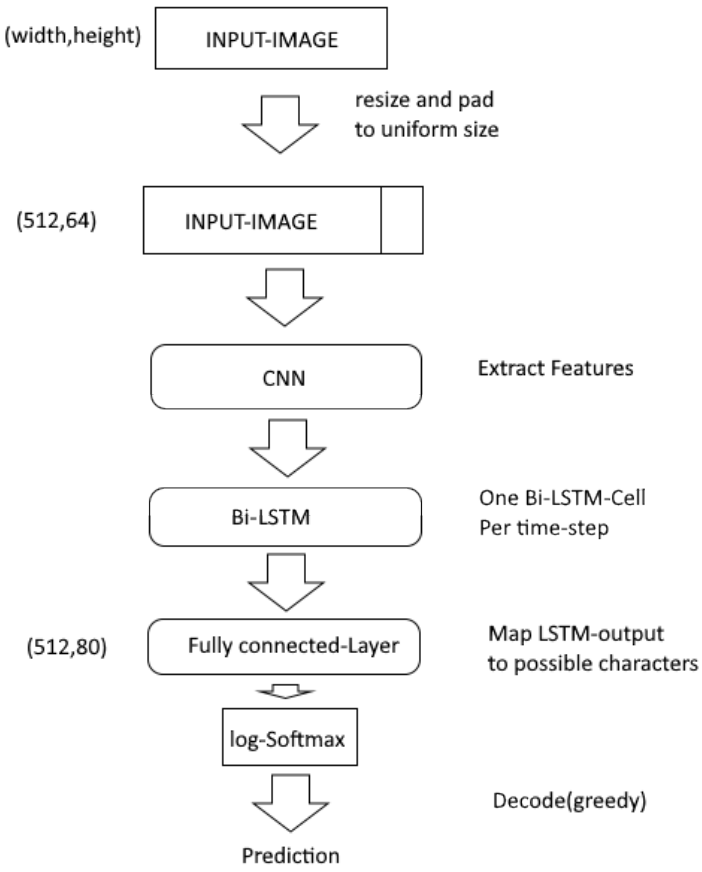


Abbildung 26: Übersicht der Verwendeten Architektur zur Handschrifterkennung

Zeichen (Kleinbuchstaben, Großbuchstaben, Zahlen, Satzzeichen). Eine weitere Ausgabe wird für das „Blank-Symbol“ benötigt.

Auf die Ausgabe des FC-Layers wird die Log-Softmax-Funktion angewendet, um log-Wahrscheinlichkeiten für jeden Zeitschritt zu erhalten.

Durch Greedy Dekodierung dieser Ausgabe erhält man die Vorhersage des Modells.

3.3.2 Vorverarbeitung

Das Ziel ist eine Größe (Breite, Höhe) von 512x64 Pixel zu erreichen. Da die Textzeilen sich kaum in der Größe unterscheiden, wurden diese lediglich direkt durch entsprechende Streckung/Stauchung auf die Zielgröße gebracht.

Da Wörter oft deutlich kürzer sind, wurde bei diesen die Größe ohne Verzerrung angepasst. Dabei wurde die Höhe auf die Zielhöhe gebracht. Die verbleibenden Pixel in der Breite wurden mit weiß aufgefüllt.

3.3.3 Feature Extraction via CNN

Die Architekur des CNN-Teils ist schematisch in Abbildung 27 dargestellt.

In den ersten beiden Schichten wird ein Convolutional Layer mit Kernelgröße 5x5 und Schrittweite 2x2 verwendet. In den anderen 4 Schichten wurde eine Kernelgröße von 3x3

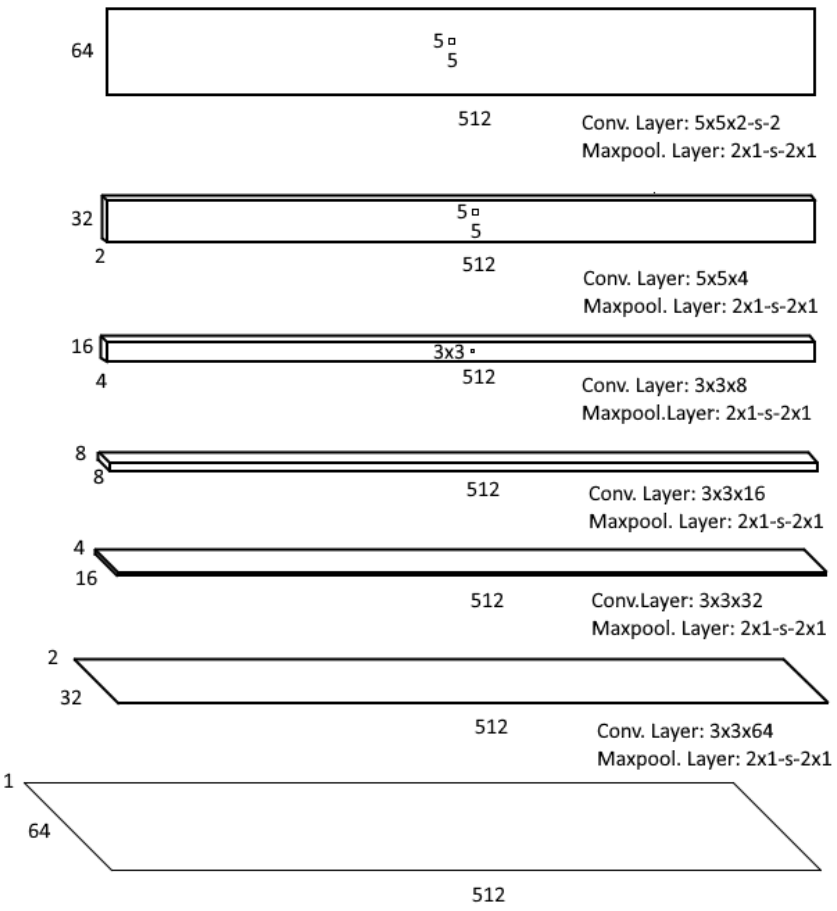


Abbildung 27: Schematische Darstellung der verwendeten CNN-Architektur

verwendet. Auf jedes Convolutional Layer folgt ein Maxpooling Layer mit Kernelgröße 2x1 und Schrittweite 2x1 (das Pooling geschieht also nur Vertikal). Dadurch wird die Höhe der Ausgabe jeweils halbiert. Zum Schluss ist die Ausgabedimension (Features x Zeitschritte) = 64x512.

3.3.4 BiLSTM

Im Allgemeinen haben sich rekurrente Neuronale Netze (RNN) und im Speziellen die LSTM Architektur als spezielle Form der RNN's bewährt, um sequenzielle Aufgaben durchzuführen. Insbesondere in den Bereichen der Sprachenmodelle [10] und der Sprachenerkennung [11]. Die Stärke eines RNN's liegt darin, dass es in der Lage ist Informationen aus vergangenen Zeitschritten zu „speichern “, wodurch die aktuelle Ausgabe, auf Basis von teilweise weit entfernten Informationen berechnet werden kann. Die grundlegende Struktur einer LSTM-Zelle ist in dargestellt.

Bei frühen Zeitschritten ist die Ausgabe größtenteils von der Eingabe abhängig, da noch kaum Information von den versteckten Schichten des Netzwerkes weiter gegeben werden. Im ersten Zeitschritt ist sogar nur Information von der Eingabe verfügbar. Um dieses Problem zu umgehen, wurden bidirektionale LSTMs (Bi-LSTM) entwickelt. Eine

LSTM-Zelle erhält Information von der Eingabe X_t, vom hidden Layer des vorherigen und des nachfolgenden Zeitschrittes h_{t-1}, h_{t+1} und generiert daraus eine Ausgabe Y_t sowie die versteckte Ausgabe h_t, welche je einmal an den nächsten bzw. den vorherigen Zeitschritt gegeben wird.

Eine Veranschaulichung der Bi-LSTM ist in Abbildung 28 dargestellt.

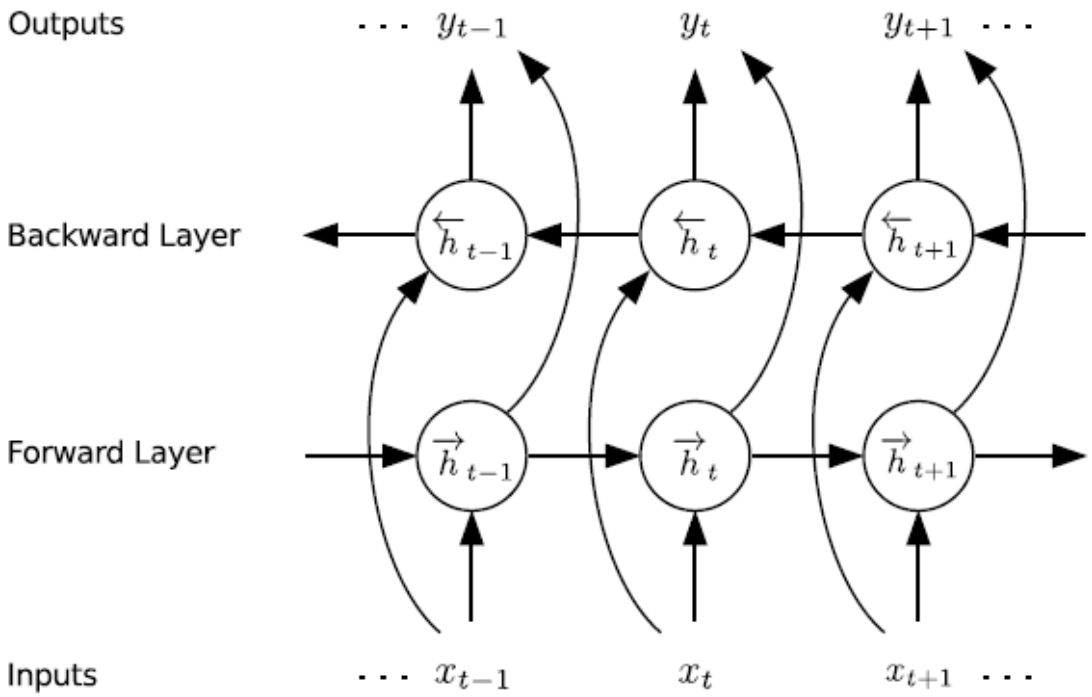


Abbildung 28: Schematische Darstellung eines Bi-LSTM's

Das BiLSTM erzeugt für jeden der 512 Zeitschritte aus den 64 Eingabewerten eine Ausgabe der gleichen Größe. Es wurden 3 Lagen BiLSTMs verwendet mit einer Größe der versteckten Schicht von 256 Neuronen. Die letzte Lage wurde während des Trainings mit einem Dropout von 50% versehen, um einem Overfitting entgegen zu wirken. Wie bereits zuvor beschrieben werden durch ein FC-Layer sowie durch Anwendung der log-Softmax Funktion log-Wahrscheinlichkeiten für jeden Zeitschritt und jeden Buchstaben bestimmt. Die Log-Wahrscheinlichkeiten werden der Loss-Funktion übergeben.

3.3.5 Dekodierung (Greedy)

Für jeden Zeitschritt wird der wahrscheinlichste Buchstabe bestimmt. Da der Logarithmus eine monotone Funktion ist, macht es keinen Unterschied, ob das Maximum über den Wahrscheinlichkeiten oder den log-Wahrscheinlichkeiten genommen wird.

Die Sequenz der wahrscheinlichsten Buchstaben wird in 2 Schritten zur finalen Ausgabe verarbeitet.

Schritt 1: Entferung von Dopplungen

Sind zwei aufeinander folgende Zeichen identisch, wird eines davon entfernt.

Schritt 2: Entfernung des Blank-Symbols

Alle verbleibenden Blank-Symbole werden entfernt.

Durch diese Reihenfolge ist es auch möglich, denselben Buchstaben mehrfach hintereinander zu erkennen.

3.3.6 Loss-Funktion: CTC-Loss

Der CTC-Loss (Connectionist Temporal Classification) wurde 2006 von A. Graves und Kollegen vorgestellt [12].

Die Idee besteht darin, eine Eingabesequenz in Zeitschritte zu unterteilen und für jeden Zeitschritt eine Vorhersage zu machen. Für das Beispiel der Handschrifterkennung

Sei L ein Alphabet und $L' = L \cup \{blank\}$, $\mathbf{x}$ eine Eingabesequenz der Länge T und $\mathcal{N}_w$ ein neuronales Netz mit den Gewichten w, welches in Abhängigkeit der Gewichte eine Funktion $\mathcal{N}_w : (R^m)^T \mapsto (\mathbb{R}^n)^T$ berechnet, wobei m die Anzahl der Eingabeneuronen und m die Anzahl der Ausgabeneuronen des Netzes $\mathcal{N}_w$ sind. Eine Ausgabe wird durch $\mathbf{y} = \mathcal{N}_w(\mathbf{x})$ beschrieben, wobei y_k^t die Aktivierung des k-ten Ausgabeneurons zum Zeitpunkt t darstellt. Es wird davon ausgegangen, dass die Ausgabe des Netzes z.B. durch Anwendung einer Softmax-Funktion in die Form einer Verteilung gebracht wird, sodass y_k^t als Wahrscheinlichkeit interpretiert werden kann, das der k-te Buchstaben des Alphabets L' zum Zeitpunkt t beobachtet wurde. Dadurch wiederum wird eine Verteilung über L'^T (die Menge der Sequenzen der Länge T über L') definiert:

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t$$

für jeden **Pfad** $\pi \in L'^T$.

Sei weiterhin $\mathcal{B}$ ein Many-to-one-mapping $\mathcal{B} : L'^t \mapsto L^{\leq T}$, bei dem eine beliebige Anzahl Zeichen des Eingabestrings auf jeweils höchstens ein Zeichen des Ausgabestrings gemappt wird. Für die Grundform des CTC-Loss entspricht dieses Mapping dem Greedy-Decoding, bei dem im ersten Schritt wiederholte Zeichen und im zweiten Schritt die Blank-Symbole entfernt werden. Schließlich lässt sich durch $\mathcal{B}$ sowie der Verteilung $p(\pi|\mathbf{x})$ die Wahrscheinlichkeit dafür berechnen, dass ein Ausgabestring $\mathbf{l} \in L^{\leq T}$ erzeugt wird, gegeben die Eingabe $\mathbf{x}$:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x})$$

Dies ist die Summe der Wahrscheinlichkeiten über alle Pfade $\pi \in L'^T$, welche durch die Dekodierung $\mathcal{B}$ auf den String $\mathbf{l} \in L^{\leq T}$ gemappt werden.

Den Loss erhält man aus der Wahrscheinlichkeit, indem der negative Logarithmus gezogen wird:

$$loss(\mathbf{l}|\mathbf{x}) = -\log(p(\mathbf{l}|\mathbf{x})) = -\log(\sum_{\pi \in \mathcal{B}^{-}1(\mathbf{l})} p(\pi|\mathbf{x})) = -\log(\sum_{\pi \in \mathcal{B}^{-}1(\mathbf{l})} \prod_{t=1}^{T} y_{\pi_t}^t)$$

Ist der Loss berechnet, wird dieser durch Backpropagation (Through-Time für RNNs) durch das Netz zurück propagiert.

3.3.7 Optimizer: RMS-Prop

Root Mean Square Propagation (RMSProp) ist ein Optimierungs-Algorithmus, der von Geoff Hinton vorgeschlagen wurde, bei dem ähnlich zu Adagrad die Lernrate während des Trainings für jedes Gewicht des Netzwerkes einzeln angepasst wird. Dabei wird die Robustheit von RProp mit der Effizienz des Mini-Batch-Lernens vereint.

Zunächst wird der "moving average"der quadrierten Gradienten definiert:

$$\mathbb{E}[g^2]_t = \beta \mathbb{E}[g^2]_{t-1} + (1-\beta)\left(\frac{\partial C}{\partial w}\right)^2$$

dabei ist $\mathbb{E}[g^2]_t$ der moving average der quadrierten Gradienten nach dem t-ten Mini-Batch, β ein Parameter (*moving average parameter*) und $\frac{\partial C}{\partial w}$ der Gradient der Lossfunktion C nach den Gewichten w des Netzwerkes. Die Aktualisierung der Gewichte ist dann wie folgt definiert

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{\mathbb{E}[g^2]_t}} \frac{\partial C}{\partial w}$$

wobei η die Lernrate ist.

Insbesondere für das Problem der Handschrifterkennung hat sich herausgestellt, dass RMSProp besonders gut geeignet ist, weswegen dieser Optimierer in unserer Implementierung verwendet wurde.

3.3.8 Training

Auf eine Hyperparametersuche wurde im Rahmen des Projektes verzichtet. Es ging zunächst nur um die Erstellung eines funktionierenden Prototypen.

Sowohl der Zeilen- als auch der Wortdatensatz wurde in Trainings-Set (95%) und Validierungs-Set (5%) aufgeteilt.

Somit wurde auf 12685 Zeilen trainiert.

Zunächst wurde versucht, direkt mit dem Training zu beginnen. Allerdings blieb sowohl bei der Validierung, als auch bei den Trainingszeilen die Ausgabe des Netzes „leer“ (es wurden nur Blanks ausgegeben, die in der Dekodierung wegfallen). Diese Problem wird in der Literatur beschrieben. Das Problem liegt darin, dass die Ausgabe des Blankssymbols für jeden Zeitschritt ein lokales Minimum darstellt. Hier ist der durch CTC verursachte Loss relativ gering, da die Editierdistanz zum Target-String

„nur“ der Länge des Target-Strings entspricht. Wandert die Ausgabe des Netzes zu einem bestimmten Buchstaben, wird dies im Mittel mehr Fehler verursachen, als richtige Buchstaben treffen.

In wird beschrieben, dass diesem Problem durch Vortraining des Netzes auf wenigen, einfachen Daten entgegnet werden kann. Eine Möglichkeit dafür besteht darin, Bilder aus handschriftähnlichen Schriftarten zu generieren. Wir haben uns jedoch dazu entschieden, einen Teil (32 Zeilen) des Trainingsdatensatzes zu verwenden. Auf diesen wurde das Netz trainiert, bis kaum noch Änderungen stattfanden.

Das Training selbst wurde in Epochen unterteilt. In jeder Epoche wurde die Hälfte des Trainings-Datensatzes zufällig ausgewählt. Als Optimierer wurde RMSProp mit einer Startlernrate von 0.001 und weight-decay von 0.0001 gewählt.

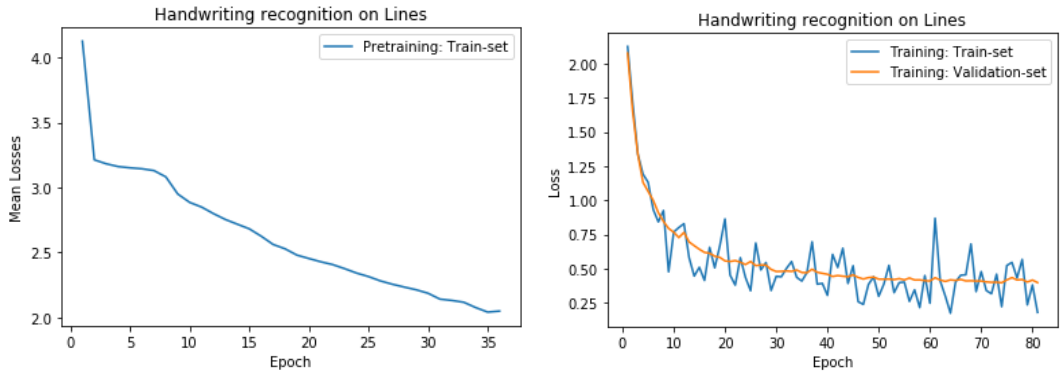
3.3.9 Validierung

Für die Validierung wurde die Ausgabe des Netzes nach dem Greedy-Verfahren dekodiert und die Editierdistanz zum Zielstring bestimmt. Die Zeichenfehlerrate (Character-Error-Rate CER) ist wie folgt definiert:

$$CER = \frac{\sum_{samples} EditDistance(Target, Prediction)}{\sum_{samples} Length(Target)}$$

Die Wortfehlerrate (Word-Error-Rate WER) ist das Verhältnis der korrekt vorhergesagten Wörter zur Gesamtzahl der in der Validierung präsentierten Wörter.

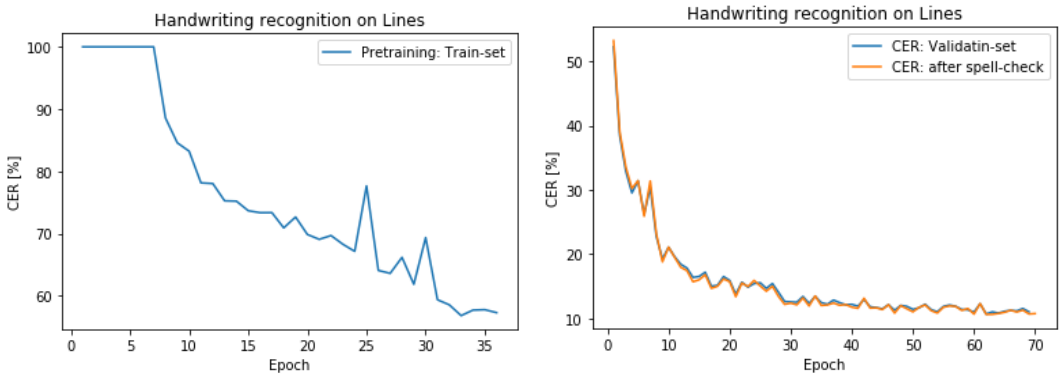
Während des Pretrainings bezieht sich sowohl der Loss, als auch die CER auf das Pretrain-set. Während des eigentlichen Trainings wird der Loss sowohl des Train-sets als auch des Validation-sets dargestellt.



(a) Pretraining (b) Main Training

Abbildung 29: Durchschnittlicher Loss im Trainingsverlauf

Abbildung 30 zeigt die CER's im Trainingsverlauf. Während des Haupttrainings wurde außerdem versucht, den Fehler durch einen Spellchecker, der wortweise angewendet wurde, zu verringern.



(a) Pretrain (b) Main Training

Abbildung 30: Character Error Rate im Trainingsverlauf

Das beste Ergebnis sowohl ohne Korrektur als auch mit Korrektur war nach der 62. Epoche mit einer CER von 10.71% ohne Korrektur und 10.64% mit Korrektur. Insgesamt waren die Unterschiede in der Fehlerrate durch die Korrektur eher gering.

3.3.10 Beispiel

Abbildung 31 zeigt ein Beispiel der Zeilendetektion. Der aus diesen Zeilen generierte Text durch die Handschrifterkennung lautet:

'A MovE to stop Mr. Gaibkell from
nominating any more Labour life Peers
io to be made at a meeting of Labon"
MPs tomorrow. Mr. Michael Foot ha
put down a resolution on the subjjeat
and he is to be bached by Mr. Will
Griftiths, MP for Mancheder Erchang

Es ist erkennbar, dass vor allem kurze Wörter in der Regel sehr gut erkannt werden. Längere Wörter und Eigennamen hingegen bereiten Schwierigkeiten.

3.3.11 Kombiniertes Ergebnis

Die Validierung auf den Zeilen, die durch die Zeilen-Detektion ermittelt wurden, ergab eine CER von 28.18%

Dieser starke Anstieg der Fehlerrate lag im Wesentlichen daran, dass fehlerhaft erkannte Zeilen sehr viele Fehler verursachen. Im schlimmsten Fall wird eine Zeile gar nicht erkannt und jedes Zeichen wird als fehlerhaft gewertet.

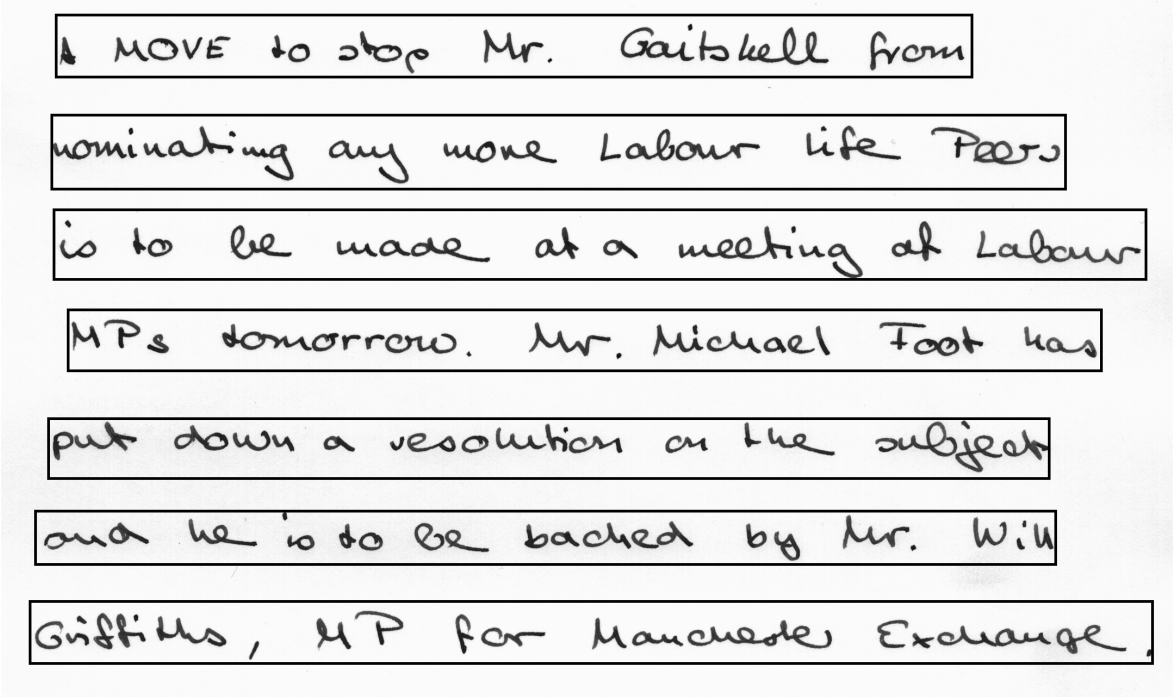


Abbildung 31: Beispiel-Ergebnis der Zeilendetektion

4 Ausblick

4.1 Ansätze zur Performanz-Steigerung

Um die Performanz unseres Systems zu verbessern, sehen wir noch folgende Verbesserungsmöglichkeiten:

- Pretraining mit synthetischen Daten, wie im state-of-the-Art-Ansatz [1] durchgeführt
- Rotationsinvariantes Lernen der Zeilenbegrenzungen
- Anstatt des Greedy-Decodings kann Lexikon-Beam-Search verwendet werden.
- Hyperparameter-Suche
- Data-Augmentation, um die Generalisierung zu verbessern
- Training der Handschrifterkennung auf den durch die Zeilen-Detektion bereitgestellten Zeilen
- Heranziehen eines expliziten Sprachenmodells zur Korrektur ganzer Sätze.

4.2 Mögliche Erweiterungen

Ferner könnte das System erweitert werden, indem zunächst eine beschriebene Seite in Bereiche kategorisiert wird: Handschriftlicher Text, mathematische Formeln, gedruckter Text, Abbildungen. Für jede dieser Kategorien könnte ein System zu Digitalisierung entwickelt werden. Durch Zusammensetzen der erkannten Einzelteile könnte somit ein komplettes handschriftliches Dokument, das nicht nur Text enthält, digitalisiert werden.

5 Fazit

Im Rahmen des Abschlussprojektes für das Praktikum haben wir neue Einblicke in die Methoden der Objekterkennung und Erkennung von Zeitsequenzen erhalten. Die beste erzielte Zeichenfehlerrate von 10.64% erreicht zwar nicht den State-of-the-Art-Ansatz [1] mit 5.7% (mit Greedy-Decoding), dennoch übersteigen unsere Ergebnisse die Erwartungen, die wir zu Beginn des Projektes hatten. Unter den in [1] zitierten Ansätzen befindet sich unser Ergebnis auf Rang 4.

Literaturverzeichnis

[1] *16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Falls, NY, USA, August 5-8, 2018*. IEEE Computer Society, 2018.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[3] mc.ai. Final layers and loss functions of single stage detectors (part 1), 2018.

[4] Mohammad Javad Shafiee, Brendan Chywl, Francis Li, and Alexander Wong. Fast yolo: A fast you only look once system for real-time embedded object detection in video. *arXiv preprint arXiv:1709.05943*, 2017.

[5] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *CoRR*, abs/1507.05717, 2015.

[6] Praveen Krishnan and C. V. Jawahar. Generating synthetic data for text recognition. *CoRR*, abs/1608.04224, 2016.

[7] Marcus Liwicki and Horst Bunke. Iam-ondb-an on-line english sentence database acquired from handwritten text on a whiteboard. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 956–961. IEEE, 2005.

[8] Adrian Rosebrock. Intersection over union(iou) for object detection, 2016.

[9] Christopher Riggio. What's the deal with accuracy, precision, recall and f1?, 2019.

[10] Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors. *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. ISCA, 2010.

[11] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[12] William W. Cohen and Andrew W. Moore, editors. *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*. ACM, 2006.