technische universität
dortmund

End-to-End-Transkription und Konfidenzbewertung handschriftlicher Datumsangaben in historischen Dokumenten

Thi Dung Pham
8. Januar 2026

Supervisors:
Prof. Dr.-Ing. Gernot A. Fink
Arthur Matei, M.Sc.

Fakultät für Informatik
Technische Universität Dortmund
http://www.cs.uni-dortmund.de

INHALTSVERZEICHNIS

1 EINLEITUNG

Viele Verwaltungsabläufe im 19. und 20. Jahrhundert basierten auf Formularen und Karteikarten. Heute liegen solche Dokumente in großer Zahl in Archiven und enthalten wertvolle Informationen, die bislang nur eingeschränkt zugänglich sind. Besonders für Historiker, Sozialwissenschaftler und Genealogen sind die personenbezogenen Angaben in diesen Formularen jedoch von großem Forschungsinteresse. Datenschutzbestimmungen erschweren häufig ihre Veröffentlichung. Nach Bundesarchivgesetz in § 11 Abs. 2[1] gelten Informationen in der Regel zehn Jahre nach dem Tod einer Person oder hundert Jahre nach ihrer Geburt als nicht mehr schutzbedürftig. Durch die Bestimmung des Geburtsdatums der betroffenen Personen lässt sich daher prüfen, ob personenbezogene Daten noch dem Datenschutz unterliegen oder bereits freigegeben werden können.

Die manuelle Transkription dieser Daten ist allerdings zeitaufwendig, da die Dokumente in alter Schrift verfasst, oft durch Alterungsprozesse beschädigt und zudem in großer Zahl vorhanden sind. Um diesem Aufwand zu begegnen, bietet sich die automatisierte Informationsextraktion als anspruchsvolle Lösung an. Da den Ergebnissen eines automatisierten Prozesses jedoch nicht uneingeschränkt vertraut werden kann, ist eine Konfidenzbestimmung hilfreich, um die Verlässlichkeit der erkannten Informationen einzuschätzen.

Ziel dieser Arbeit ist es zu untersuchen, wie effektiv ein End-to-End-Modell zur Extraktion historischer Geburtsdaten eingesetzt werden kann. Dabei wird ein zweistufiges Fine-Tuning durchgeführt, gefolgt von einer Bewertung der Konfidenz der Transkriptionen. Es ist zu beachten, dass eine automatisierte Transkription grundsätzlich keine hundertprozentige Genauigkeit erreichen kann. Da Archive sensible personenbezogene Daten verwalten, ist eine alleinige Abhängigkeit von der automatisierten Transkription nicht möglich. Deshalb wird neben der Erkennungsgenauigkeit auch die Konfidenz der Modellvorhersagen bewertet, um die Verlässlichkeit der extrahierten Informationen besser einschätzen zu können.

Die Aufgabe, Informationen aus historischen Dokumenten zu extrahieren, hat unter anderem durch den Wettbewerb „Information Extraction in Historical Handwritten Records (IEHHR)“ im Jahr 2017 [BRM+20] bereits Interesse geweckt. Dieser basierte

1 www.bundesarchiv.de/das-bundesarchiv/rechtsgrundlagen/bundesarchivgesetz

auf der Esposalles Datenbank [RFS+13], einer historischen Sammlung von Heiratsurkunden. Ziel des Wettbewerbs war es, relevante Informationen über Braut und Bräutigam sowie deren Eltern zu extrahieren.

Um die Herausforderung der Transkription handschriftlicher Daten aus historischen Dokumenten anzugehen, setze ich in diese Arbeit den Document Understanding Transformer (DONUT) [KHY+22] ein, ein Modell, das sich laut aktuellem Stand der Forschungen als besonders leistungsfähig für generische Transkriptionsaufgaben erwiesen hat. DONUT ist ein OCR-freies Transformer-basiertes Modell für Dokumentenverständnis, der entwickelt wurde, um Herausforderungen bei der Verarbeitung von vollständigen Dokumentbildern zu bewältigen, einschließlich Texterkennung und Dokumentenverständnis. Das Modell zeichnet sich durch eine End-to-End-Architektur und ein Vortrainingsziel aus, wodurch es in verschiedenen Aufgaben der visuellen Dokumentenanalyse zuverlässig gute Ergebnisse in Bezug auf Genauigkeit und Geschwindigkeit erzielt.

Eine Strategie zum Finetuning des DONUT-Modells wird in dieser Arbeit entwickelt, um die Extraktion von Datumseingaben aus historischen Formularen gezielt zu verbessern. Die Methode basiert auf einem zweistufigen Fine-Tuning-Prozess: In der ersten Phase wird das vortrainierte DONUT-Modell gezielt mit ausgeschnittenen Bildausschnitten trainiert, die Datumsangaben enthalten. Diese stammen aus dem Datensatz DC-2 der DARE-Datenbank (Database of Abstracts of Reviews of Effects) [DJS+22] und ermöglichen dem Modell, die visuelle Struktur sowie das typische Textmuster von Datumsformaten zu erlernen. In der zweiten Phase erfolgt ein weiteres Fine-Tuning mit vollständigen Dokumentseiten aus dem CM1-COVER-Datensatz. Dadurch lernt das Modell, die zuvor erkannten Datumsmuster auch im komplexen Seitenkontext präzise zu identifizieren und zu extrahieren.

Diese Arbeit besteht neben diesem Kapitel aus fünf weiteren Kapiteln. Kapitel 2 vermittelt das theoretische Grundlagenwissen, das für das Verständnis des eingesetzten Dokumentenextraktionsverfahrens erforderlich ist. In Kapitel 3 werden verwandte Ansätze vorgestellt, die die Entwicklung der in Kapitel 4 beschriebenen Donut-Architekturen und Konfidenzmaß maßgeblich beeinflusst haben. Kapitel 5 erläutert das Evaluationsprotokoll, die verwendeten Metriken sowie die eingesetzten Datensätze, die als Basis für die Bewertung und den Vergleich verschiedener Dokumentenextraktionsverfahren dienen. Zudem werden hier die Ergebnisse der durchgeführten Experimente präsentiert. Kapitel 6 bietet eine abschließende Bewertung der Resultate.

2 GRUNDLAGEN

In diesem Kapitel werden grundlegende Konzepte und Begriffe vorgestellt, die für das Verständnis der in dieser Arbeit angewendeten Methodik von Bedeutung sind.

2.1 PROBLEMDEFINITION

Die Extraktion relevanter Informationen aus historischen handschriftlichen Dokumentensammlungen ermöglicht, um diese Quellen systematisch zugänglich zu machen. In diesem Zusammenhang reicht eine reine Handschriftenerkennung [FB14] nicht aus. Das Ziel liegt mehr darin, ein umfassenderes Dokumentenverständnis zu entwickeln, das in der Lage ist, gezielt semantisch bedeutsame Inhalte zu extrahieren [FRB+17].

Die Aufgabe der Informationsextraktion besteht darin, semantisch relevante Informationen aus Dokumenten zu identifizieren und in strukturierter Form bereitzustellen [FRB+17]. Herausfordernd ist dies bei handschriftlichen Datumsangaben aus historischen Dokumenten. Zum einen sind sie visuell nicht einfach zu erkennen, da sie durch stark variierende Schriftarten, verblasste Tinte oder historische Schreibweisen geprägt sind. Zum anderen entstehen semantische Ambiguitäten, beispielsweise durch die Vielfalt an Datumsformaten wie „12/03/1895“ oder „12. März 1895“. Erschwerend kommt hinzu, dass Geburtsdaten in vollständigen Dokumentseiten meist nicht isoliert vorliegen, sondern in komplexe, mehrzeilige Registereinträge innerhalb von Formularen eingebettet sind.

Klassische Ansätze [KKY+14, ACM+15, gQL+19] basieren häufig auf OCR-Systemen, die zunächst Textsegmente erkennen und anschließend die relevanten Informationen extrahieren. Hierbei ist jedoch häufig eine zusätzliche Segmentierung notwendig, um einzelne Textblöcke voneinander zu trennen. Solche Verfahren stoßen schnell an ihre Grenzen. OCR arbeitet in der Regel zuverlässig, solange die Dokumentbilder nur wenige Störungen enthalten und die Texte nicht stark beschädigt sind. Sobald jedoch die Bildqualität erheblich abnimmt, wie es insbesondere bei historischen Dokumenten häufig der Fall ist, verschlechtert sich die Erkennungsleistung erheblich [DSS08]. Um diese Limitierungen zu überwinden, werden End-to-End-Ansätze untersucht, die ohne klassische OCR-Zwischenschritte auskommen. Die Bausteine des Systems, insbesondere die Verarbeitung der visuellen Eingaben und die Sequenzmodellierung der extrahier-

ten Informationen, können durch neuronale Netze und Transformer-Architekturen realisiert werden. In dieser Arbeit wird ein transformerbasierter End-to-End-Ansatz eingesetzt, der ein Dokumentenbild als Eingabe erhält und die darin enthaltenen Datumsangaben direkt extrahiert.

Eine reine Bewertung anhand der Transkriptionsgenauigkeit reicht nicht aus, um die Verlässlichkeit der Ergebnisse zu beurteilen. Deshalb ist es notwendig, die vom Modell generierten Konfidenzwerte systematisch zu untersuchen und deren Aussagekraft im Hinblick auf die Qualität der Transkriptionen zu bewerten.

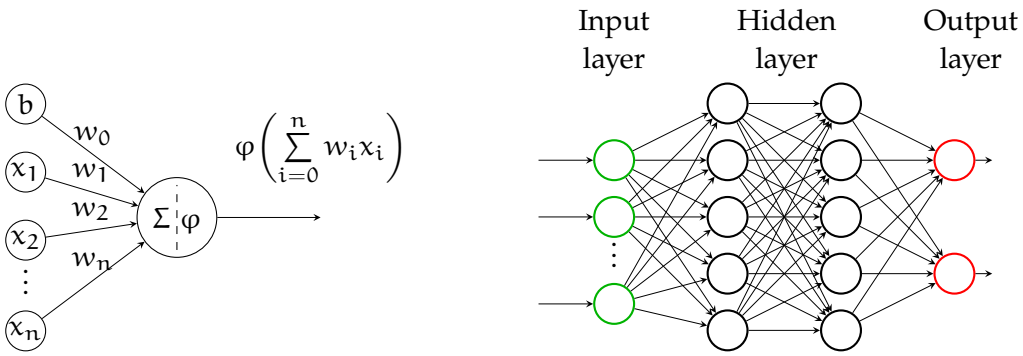
2.2 NEURONALE NETZE

Künstliche Neuronale Netze sind aus schichtweise organisierten künstlichen Neuronen aufgebaut, deren Struktur von biologischen Neuronalen Netzen in dem menschlichen Gehirn inspiriert ist. Jedes Neuron kann Eingangssignale empfangen, diese verarbeiten und ein Ausgangssignal erzeugen. Neuronen sind dabei mit mindestens einem anderen Neuron verbunden, und jede Verbindung besitzt einen reellen Wert, den sogenannten Gewichtungskoeffizienten, der die Bedeutung der Verbindung innerhalb des Netzwerks beschreibt [SKP97].

Dieses Kapitel bietet eine grundlegende Einführung in künstliche neuronale Netze, die als theoretische Grundlage für die vorliegende Arbeit dient. Zunächst wird in Abschnitt 2.2.1 das Feedforward-Netzwerk beschrieben. Danach werden verschiedene Aktivierungsfunktionen vorgestellt und die Layer-Normalisierung wird anschließend näher erläutert. In dem nächsten Abschnitt 2.2.2 werden Convolutional Neural Networks, die eine spezielle Form von neuronalen Netzen für Daten mit Gittertopologie sind, erläutert. Schließlich handelt sich im Abschnitt 2.2.3 um die Optimierung der vorher beschriebenen neuronalen Netzen.

2.2.1 *Feedforward-Netzwerk*

Ein Feedforward-Netzwerk (FFN) ist einfaches neuronales Netzwerk, das ausschließlich Informationen von einer Schicht zur nächsthöheren weitergibt. Die Bezeichnung Feed-Forward bedeutet, dass es keine Rückverbindungen gibt und somit keine Ausgaben oder Zwischenwerte zurück in das Modell geführt werden [GBC16, S. 164]. Ein FFN kann als gerichteter Graph wie in der Abbildung 2.2.1 interpretiert werden, in dem die Knoten Neuronen repräsentieren und die gerichteten Kanten die gewichteten Verbindungen zwischen den Neuronen darstellen. Jedes Neuron empfängt Eingaben,



(a) (b)

Abbildung 2.2.1: (a) Berechnung innerhalb eines einzelnen Neurons mit gewichteten Eingaben x_i, Gewichtungsfaktoren w_i, Bias-Term w_0 sowie Aktivierungsfunktion σ nach [Agg23, S. 5]. (b) Aufbau eines Multi-Layer FNN mit einem Input Layer, zwei Hidden Layer und einem Output Layer nach [Agg23, S. 14].

verrechnet diese mit den zugehörigen Gewichtungen sowie einem Bias-Term und leitet das Ergebnis nach Anwendung einer Aktivierungsfunktion weiter.

Ein FFN besteht aus Schichten l_k, $k = 0, \ldots, L$, die jeweils $M^{(k)}$ Neuronen enthalten. Die Eingabeschicht (Input Layer) l_0 stellt den Eingabevektor $x \in \mathbb{R}^d$ bereit, die Zwischenschichten l_k, $k = 1, \ldots, L-1$ werden als versteckte Schichten (Hidden Layers) bezeichnet, die Ausgabeschicht (Output Layer) l_L enthält die finalen Ergebnisse der Berechnung $\hat{y} \in \mathbb{R}^{M^{(L)}}$ (siehe Abbildung 2.2.1 (b)). In den Hidden-Layern werden Berechnungen auf Basis der Ausgaben der Neuronen im Vorgänger-Layer durchgeführt und anschließend durch die Ausgabeschicht ausgegeben. Die Anzahl der Schichten wird als Tiefe bezeichnet. Dabei wird die Eingabeschicht nicht mitgezählt, da sie lediglich die Daten weiterleitet und selbst keine Berechnungen durchführt.

In einem FFN berechnet das j-te Neuron der k-ten Schicht zunächst eine gewichtete Summe der Ausgaben der vorherigen Schicht. Dabei wird ein Bias-Term addiert, bevor das Ergebnis durch eine nicht-lineare Aktivierungsfunktion transformiert wird (siehe Abbildung 2.2.1 (a)). Entsprechend der Definition in [Bis06, S. 227-230] ergibt sich die folgende mathematische Darstellung:

$$y_j^{(k)} = \varphi^{(k)}\left(a_j^{(k)}\right) \quad \text{mit} \quad a_j^{(k)} = \sum_{i=1}^{M^{(k-1)}} w_{ji}^{(k)} y_i^{(k-1)} + w_{j0}^{(k)}, \tag{2.2.1}$$

Hierbei bezeichnet $y^{(k-1)} \in \mathbb{R}^{M^{(k-1)}}$ die Eingaben des Neurons, also die Ausgaben der Neuronen aus der $(k-1)$-ten Schicht, $w_j^{(k)} \in \mathbb{R}^{M^{(k-1)}}$ den Gewichtungsvektor für

das j-te Neuron der k-ten Schicht und $w_{j0}^{(k)} \in \mathbb{R}$ den zugehörigen Bias. Auf die Summe $a_j^{(k)}$ wird anschließend eine nicht-lineare Aktivierungsfunktion $\varphi^{(k)}$ angewendet, damit die Berechnung nicht durch eine einzige Matrixmultiplikation ersetzt werden kann und das Netzwerk auch nicht lineare Funktionen approximieren kann.

Die Berechnung innerhalb eines FFNs lässt sich effizient in vektorisierter Form darstellen. Dabei wird jede Schicht als eine Kombination aus einer linearen Transformation und einer nicht-linearen Aktivierungsfunktion (vgl. Abschnitt 2.2.1) modelliert. Um die Darstellung zu vereinfachen, kann der Bias in den Gewichtungsvektor integriert werden. Dies geschieht, indem ein sogenanntes Bias-Neuron eingeführt wird, das stets den festen Wert 1 weitergibt. Das Gewicht der Kante zwischen Bias-Neuron und Ausgabeneuron übernimmt dann der Bias. Dadurch kann Gleichung 2.2.1 umgeschrieben werden zu:

$$a^{(k)} = W^{(k)} y^{(k-1)} \tag{2.2.2}$$

Dabei umfasst der Eingabevektor $y^{(k-1)} \in \mathbb{R}^{M^{(k-1)}+1}$ neben den Ausgaben der vorherigen Schicht zusätzlich einen konstanten Eintrag $y_0^{(k-1)} = 1$. Die Gewichtsmatrix $W^{(k)} \in \mathbb{R}^{M^{(k)} \times (M^{(k-1)}+1)}$ enthält in jeder Zeile die Gewichte (inklusive Bias) eines Neurons der k-ten Schicht.

Somit lässt sich die vom FFN approximierte Funktion $f(x; W)$ als Komposition der durch die einzelnen Schichten realisierten Teilfunktionen $f^{(k)}(\cdot; W^{(k)})$ darstellen:

$$\hat{y} = f(x; W) = \left(f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}\right)(x; W), \tag{2.2.3}$$

Dabei bezeichnet $W = \{W^{(1)}, \ldots, (W^{(L)})\}$ die Parameter des gesamten Netzwerks. Jede Teilfunktion $f^{(k)}$ berechnet die nichtlineare Transformation einer Schichteingabe $y^{(k-1)}$:

$$f^{(k)}(y^{(k-1)}; W^{(k)}) = \varphi^{(k)}\left(W^{(k)} y^{(k-1)}\right), \tag{2.2.4}$$

wobei $\varphi^{(k)}$ eine (nichtlineare) Aktivierungsfunktion darstellt.

Aktivierungsfunktionen

Aktivierungsfunktionen spielen eine zentrale Rolle in neuronalen Netzen, da sie es ermöglichen, nichtlineare Zusammenhänge in den Daten abzubilden. Dies erhöht die Flexibilität und Leistungsfähigkeit neuronaler Netze bei der Modellierung komplexer

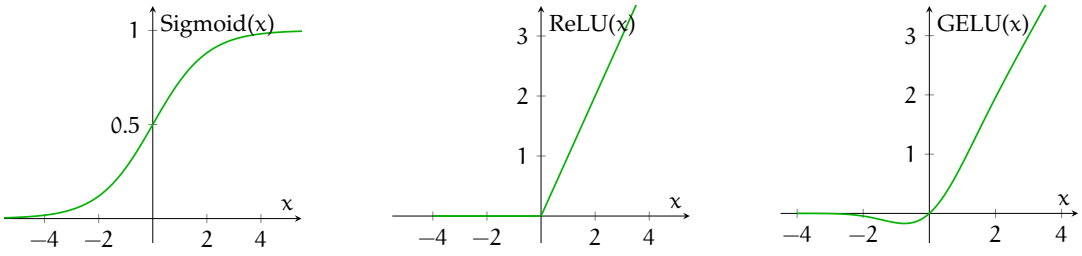


Abbildung 2.2.2: Drei Aktivierungsfunktionen (Sigmoid, ReLU, GELU) nach [Agg23, S. 13] im Vergleich. Identische x-Skala (4 bis 4).

und differenzierter Daten [RAS20]. Ein neuronales Netz, das ausschließlich lineare Aktivierungsfunktionen nutzt, wäre nicht in der Lage, die Trainingsdaten korrekt zu klassifizieren, da viele Datenpunkte nicht linear trennbar sind [Agg23]. Die Auswahl einer geeigneten Aktivierungsfunktion hängt dabei von der jeweiligen Modellarchitektur ab. Abbildung 2.2.2 zeigt drei der in dieser Arbeit relevanten Aktivierungsfunktionen: Sigmoid, die Rectified Linear Unit (ReLU) und die Gaussian Error Linear Unit (GELU).

In der Arbeit wird das Symbol $\varphi(\cdot)$ zur Bezeichnung der Aktivierungsfunktion verwendet. Der Wert, der vor der Anwendung der Aktivierungsfunktion $\varphi(\cdot)$ berechnet wird, wird als Pre-Activation (Voraktivierung) bezeichnet. Die Pre-Activation spielt eine wichtige Rolle bei der Berechnung im Rahmen des Backpropagation (vgl. Abschnitt 2.2.3), der später noch genauer behandelt wird. Sobald die Aktivierungsfunktion auf diesen Wert angewandt wurde, spricht man vom Post-Activation, welcher letztlich die Ausgabe des Neurons darstellt [Agg23, S. 12].

Eine klassische Aktivierungsfunktion ist die Sigmoid- oder logistische Aktivierungsfunktion, die mathematisch nach [RAS20] wie folgt definiert ist:

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{2.2.5}$$

Sie transformiert den Eingabebereich von $(-\infty, +\infty)$ in den Bereich $[0, 1]$. Damit ermöglicht sie sowohl die Erzeugung von Wahrscheinlichkeitsausgaben als auch die Formulierung von Verlustfunktionen, die auf Maximum-Likelihood-Modellen basieren [Agg23, S. 12].

Die ReLU

$$\varphi(x) = \max(0, x) \tag{2.2.6}$$

ist derzeit die am häufigsten verwendete Aktivierungsfunktion [RZL17]. Die ReLU-Funktion hat für $x \geqslant 0$ den Wert x und 0 andernfalls. Ihre Ableitung ergibt sich

den Wert 0 für $x < 0$, den Wert 1 für $x > 0$ und ist für $x = 0$ nicht differenzierbar. Ein Vorteil der ReLU [Agg23, S. 28] liegt darin, dass sie im positiven Bereich stets eine konstante Ableitung von 1 besitzt und somit weniger anfällig für das Vanishing-Gradient-Problem [BSF94] ist. Ein Nachteil besteht jedoch darin, dass während des Trainings die Parameter so angepasst werden können, dass ein Neuron ausschließlich Werte kleiner oder gleich 0 produziert und dadurch dauerhaft inaktiv bleibt [CMB23, S. 185].

Die Gaussian Error Linear Unit (GELU) [HG23] lässt sich durch folgende Näherungsformel beschreiben:

$$\varphi(x) \approx 0.5x \left(1 + \tanh\left[\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right]\right) \tag{2.2.7}$$

Die GELU ist eine nichtlineare Aktivierungsfunktion, die in vielen Transformer-basierten Modellen, einschließlich BERT [DCLT19] und Swin [LLC+21] Transformer, standardmäßig verwendet wird. Im Gegensatz zu ReLU führt GELU eine stochastisch motivierte Glättung ein, indem sie die Eingabe gewichtet basierend auf deren Wahrscheinlichkeit, positiv zu sein. Sie kombiniert dabei Merkmale linearer und nichtlinearer Aktivierungen.

Layer-Normalisierung

Die Layer-Normalisierung [BKH16] ist eine von vielen Normalisierungstechniken, die das Training tiefer neuronaler Netze verbessern, indem sie die Ausgabewerte einer versteckten Schicht auf eine standardisierte Verteilung bringt.

Beim Training tiefer Netze hängen die Gradienten einer Schicht stark von den Ausgaben der vorherigen Schicht ab. Dieses Problem lässt sich verringern, indem die summierten Eingaben (Voraktivierungen) jeder Schicht auf einen festen Mittelwert und eine feste Varianz normalisiert werden [BKH16]. Gegeben sei der Vektor der Voraktivierungen der k-ten Schicht $a^{(k)} \in \mathbb{R}^{M^{(k)}}$. Das arithmetische Mittel und die Standardabweichung werden berechnet als:

$$\mu^{(k)} = \frac{1}{M^{(k)}} \sum_{i=1}^{M^{(k)}} a_i^{(k)}, \qquad \sigma^{(k)} = \sqrt{\frac{1}{M^{(k)}} \sum_{i=1}^{M^{(k)}} \left(a_i^{(k)} - \mu^{(k)}\right)^2} \tag{2.2.8}$$

Anschließend wird jeder Wert des Vektors normiert, indem der Mittelwert subtrahiert und durch die Standardabweichung geteilt wird:

$$\hat{a}^{(k)} = \frac{a^{(k)} - \mu^{(k)}}{\sigma^{(k)}} \tag{2.2.9}$$

Um der Normalisierung Flexibilität zu verleihen, werden zwei lernbare Parameter eingeführt: ein Skalierungsfaktor $\gamma \in \mathbb{R}^{M^{(k)}}$ und ein Bias-Term $\beta \in \mathbb{R}^{M^{(k)}}$. Diese Parameter dienen dazu, die normalisierten Werte in lernbarer Weise neu zu skalieren und zu verschieben. Die endgültige Ausgabe der Layer-Normalisierung ergibt sich somit zu:

$$a^{(k)}_{\text{normalized}} = \gamma \odot \hat{a}^{(k)} + \beta \tag{2.2.10}$$

wobei $\odot$ das elementweise Produkt bezeichnet. Layer-Normalisierung wird typischerweise nach der linearen Transformation, vor der Aktivierungsfunktion angewendet. Nachteile sind der zusätzliche Rechenaufwand und eine eingeschränkte Wirksamkeit bei Schichten mit wenigen Neuronen [BKH16].

2.2.2 *Convolutional Neural Networks*

Convolutional Neural Networks (CNNs) [LB98] unterscheiden sich von klassischen neuronalen Netzen dadurch, dass sie besonders gut für Daten mit einer Gitterstruktur geeignet sind, wie beispielsweise Zeitreihen oder Bilddaten [GBC16, S. 326] [ON15].

Dies liegt daran, dass CNNs die räumliche oder zeitliche Nachbarschaft der Daten gezielt ausnutzen können. Die Intuition hinter Convolutional Neural Networks besteht darin, dass die Filter während des Trainings selbstständig lernen, Merkmale wie Kanten, Formen oder Texturen zu erkennen. Dadurch müssen die relevanten Eigenschaften nicht manuell vorgegeben werden, sondern werden direkt aus den Trainingsdaten gelernt. Durch das anschließende Pooling wird zudem Ortsinvarianz erreicht, sodass ein erkanntes Merkmal unabhängig von seiner genauen Position im Bild berücksichtigt wird [Ket17, S. 77–78].

Ein wesentlicher Unterschied von CNNs zu klassischen Neural Networks besteht darin, dass ihre Neuronen in drei Dimensionen Höhe, Breite und Tiefe organisiert sind. Die Tiefe bezeichnet dabei nicht die Anzahl der Netzwerkschichten, sondern die dritte Dimension des Aktivierungsvolumens. Zudem sind die Neuronen einer CNN-Schicht nicht mit allen Neuronen der vorherigen Schicht verbunden, sondern nur mit einem kleinen lokalen Bereich, was eine effiziente Verarbeitung räumlicher Strukturen ermöglicht [ON15].

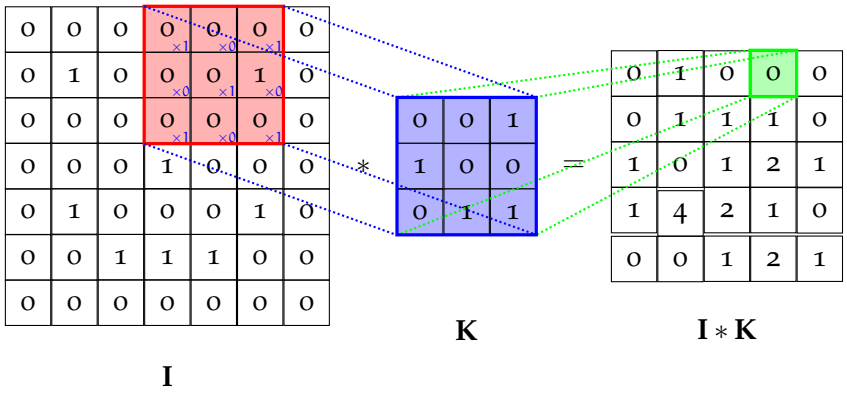


Abbildung 2.2.3: Abbildung nach [AAS20] zeigt die Faltung eines Eingabebildes I mit einem Kernel K. Der Kernel K wird über das Eingabebild I verschoben. An jeder Position werden die überdeckten Pixel mit den Kernel-Gewichten multipliziert und summiert, um einen Wert in der Feature Map $I * K$ zu erzeugen.

Im Folgenden wird zunächst erläutert, was unter einer Convolution zu verstehen ist. Anschließend wird das Verfahren des Poolings beschrieben, das in nahezu allen Convolutional Networks eingesetzt wird. Die in diesem Kapitel verwendete Terminologie orientiert sich an [CMB23, S. 290–298] und [AAS20].

Convolution

Das zentrale Konzept der Faltung basiert auf dem sogenannten „Rezeptiven Feld“. Darunter versteht man einen kleinen Ausschnitt des Bildes, in dem ein Neuron Merkmale erkennt. In diesem Feld lernen die Gewichte des Modells, einfache Strukturen wie Kanten, Linien oder andere visuelle Muster zu identifizieren.

Für die anschauliche Erklärung kann man sich die Eingabebilder zunächst auf Graustufenbilder beschränken. Der dafür verwendete Kernel ist eine kleine Matrix von Gewichtswerten, die dieselbe Tiefe wie die Eingabeebene, jedoch eine geringere räumliche Ausdehnung besitzt. Dieser Kernel wird über das Bild verschoben, wobei an jeder Position eine gewichtete Summe der überdeckten Pixelwerte berechnet wird. Durch die wiederholte Anwendung derselben Gewichtswerte auf alle Positionen des Bildes entsteht die sogenannte Faltung. Dieses Teilen der Gewichte ist ein zentrales Merkmal der CNN-Architektur, da es die Anzahl der zu lernenden Parameter im Vergleich zu einem vollständig verbundenen Netzwerk drastisch reduziert [GBC16,

S. 331–333]. Die Ergebnisse dieser Berechnungen werden in einer neuen Ausgabematrix gespeichert, die als Feature Map bezeichnet wird (siehe Abbildung 2.2.3).

Die Faltung kann nach mathematisch formalisiert werden. Für ein Bild mit Intensitäten (Helligkeitswert) I und einen Kernel K ergibt sich die Aktivierung der Feature-Map S an der Position (i, j) nach [GBC16, S. 328] zu

$$S(i,j) = (K * I)(ij) = \sum_{n} \sum_{m} I(i+n,\, j+m)\, K(n,m). \tag{2.2.11}$$

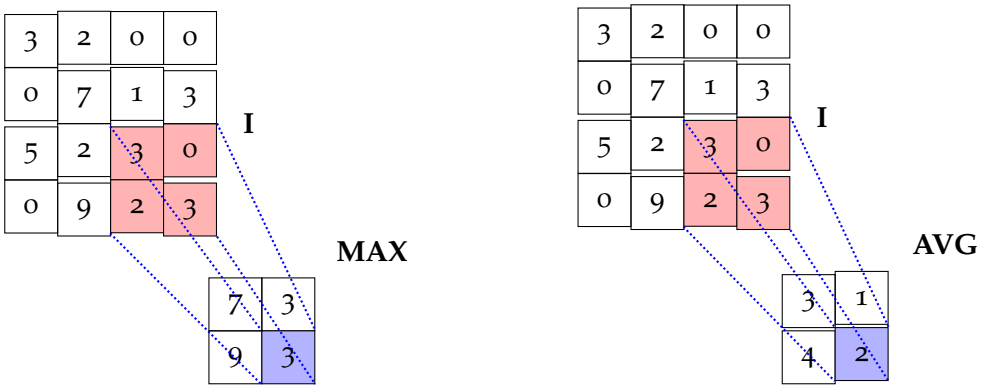
Genau genommen handelt es sich hierbei um eine *Cross-Correlation*, in der Literatur zum maschinellen Lernen wird dieser Vorgang jedoch üblicherweise als Faltung bezeichnet. Durch Hinzufügen eines Bias-Terms sowie der Anwendung einer Aktivierungsfunktion werden die endgültigen Werte der Feature Map bestimmt. Traditionell wurde in neuronalen Netzen häufig die Sigmoid-Funktion als Aktivierungsfunktion (vgl. Abschnitt 2.2.1) eingesetzt, da sie eine glatte, stetige Form besitzt und sich leicht differenzieren lässt.

Pooling

Eine typische Schicht in einem CNN besteht aus drei Phasen [GBC16, S. 355]. Zunächst werden mehrere Faltungen parallel durchgeführt, um lineare Aktivierungen zu erzeugen. Im nächsten Schritt werden diese linearen Aktivierungen durch eine nichtlineare Aktivierungsfunktion transformiert. Schließlich kommt Pooling zum Einsatz, die die Ausgaben weiter zusammenfasst und damit die Dimensionen reduziert, ohne die relevanten Informationen zu verlieren.

Pooling-Schichten verfolgen das Ziel, die Dimensionalität der Repräsentationen schrittweise zu verringern. Dadurch sinkt nicht nur die Anzahl der Parameter, sondern auch die rechnerische Komplexität des Modells [ON15, AAS20]. Ein weiterer Ziel dabei ist die Translationsinvarianz. Das bedeutet, dass die Ausgabe des Netzes auch dann weitgehend stabil bleibt, wenn das Eingabebild leicht verschoben wird. Auf diese Weise kann das Modell erkennen, dass ein bestimmtes Merkmal vorhanden ist, unabhängig davon, ob es sich im Bild ein wenig nach links, rechts, oben oder unten bewegt hat [AAS20].

Das Grundprinzip von Pooling besteht darin, die Ausgaben einer bestimmten Region in der Feature Map durch eine statistische Kenngröße zu ersetzen. Eine verbreitete Methode ist das sogenannte Max-Pooling [ZC88] (siehe Abbildung 2.2.4 (a)), das in Convolutional Neural Networks häufig eingesetzt wird, da dabei keine zusätzlichen Parameter gelernt oder angepasst werden müssen. Hierbei wird innerhalb jeder



(a) (b)

Abbildung 2.2.4: Beispiel für Pooling-Operationen in CNNs: (a) Max-Pooling und (b) Average-Pooling.

Pooling-Region der höchste Wert ausgewählt und an die nächste Schicht weitergegeben. Auf diese Weise bleibt erhalten, ob ein Merkmal in der betrachteten Region vorhanden ist und wie stark es ausgeprägt ist, während Details über die exakte Position teilweise verloren gehen.

Neben Max-Pooling gibt es auch andere Varianten. So kann die Ausgabe beispielsweise als Durchschnittswert aller Werte innerhalb des Bereichs berechnet werden (siehe Abbildung 2.2.4 (b)), oder man verwendet die L2-Norm zur Zusammenfassung. Ebenso sind gewichtete Mittelwerte möglich, bei denen Werte, die näher am Zentrum des Rezeptiven Feldes liegen, stärker berücksichtigt werden.

2.2.3 *Optimierung*

In den vorangegangenen Abschnitten wurde der grundlegende Aufbau neuronaler Netze beschrieben. Dieses Kapitel beschäftigt sich mit der Optimierung dieser Netze. Unter der Optimierung eines neuronalen Netzes versteht man die Anpassung seiner Parameter W (also der Gewichte und Bias) mit dem Ziel, dass die Ausgaben $\hat{y}_i = f(x_i; W)$ für Eingaben (x_i, y_i) aus einer Stichprobe $X = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ möglichst genau mit den tatsächlichen Zielwerten y_i übereinstimmen [Agg23, S. 6–7].

Im Folgenden werden zunächst verschiedene Verlustfunktionen vorgestellt, die messen, wie groß die Abweichung zwischen Vorhersage und Zielwert ist. Darauf aufbauend wird das Verfahren des Gradientenabstiegs (Gradient Descent) erläutert, ein Algorithmus zur schrittweisen Minimierung der Verlustfunktion. Schließlich wird

die Backpropagation als algorithmisches Verfahren zur effizienten Berechnung dieser Gradienten vorgestellt.

Verlustfunktion

Im Kontext neuronaler Netze beschreibt eine Verlustfunktion (Loss Function, Cost Function, oder Error Function), wie stark die geschätzte Ausgabe eines Modells von der gewünschten Zielausgabe abweicht. Um die Sicherheit einer Klassenvorhersage für ein Eingabedatum x_i zu bewerten, wird eine Verlustfunktion definiert, die einen hohen Wert annimmt, wenn die vorhergesagte Klasse $\hat{y}_i$ nicht mit der tatsächlichen Klasse y_i übereinstimmt.

Für Klassifikationsprobleme wird häufig die Cross-Entropy-Loss (CE) zwischen den Trainingsdaten und den Modellvorhersagen verwendet. Dabei kommt üblicherweise die Softmax-Funktion in der Ausgabeschicht des Klassifikators zum Einsatz, um die k reellwertigen Scores bzw. Klassen-Scores $v_1, \ldots, v_k$ in Wahrscheinlichkeiten $o_1, \ldots, o_k$ über k verschiedene Klassen zu transformieren [CMB23, S. 419].

Seien $y_1, \ldots, y_k \in \{0, 1\}$ die One-Hot-codierten Labels der Trainingsdaten für die k Klassen. Dabei nimmt die Komponente der korrekten Klasse den Wert 1 an, während alle anderen Komponenten 0 sind [SF17]. Für ein einzelnes Trainingsbeispiel $i \in \{0, \ldots, k\}$ ist der Cross-Entropy-Loss nach [Agg23, S. 118] definiert als:

$$L_i(w) = -\sum_{j=1}^{k} y_j \log(o_j), \qquad o_j = \frac{e^{v_j}}{\sum_{l=1}^{k} e^{v_l}} \tag{2.2.12}$$

In den meisten Optimierungsproblemen lässt sich die Verlustfunktion als Summe der Verluste über alle Trainingsbeispiele darstellen. Dann ergibt sich der Gesamtverlust für eine Trainingsmenge mit n Trainingsbeispiele nach [CMB23, S. 234][GBC16, S. 122] zu:

$$L(w) = \sum_{i=1}^{n} L_i(w). \tag{2.2.13}$$

wobei L_i der Verlust für ein einzelnes Trainingsbeispiel i ist.

Gradient Descent

Eine grundlegende Methode zur Minimierung der Verlustfunktion $L(w)$ ist das Gradient Descent [Lem12]. Die zugrunde liegende Idee von Gradient Descent besteht

darin, von einem zufällig gewählten Startpunkt aus schrittweise in die Richtung der negativen Steigung der Verlustfunktion zu gehen bis ein (lokales) Minimum erreicht ist. In hochdimensionalen Räumen ist es jedoch schwierig, das globale Minimum zu erreichen, weshalb in der Praxis meist eine Lösung akzeptiert wird, deren Funktionswert zwar nicht minimal, aber dennoch hinreichend klein ist [GBC16, S. 81–82]. Um eine ausreichend gute Lösung zu erhalten, kann es notwendig sein, den Algorithmus mehrfach mit unterschiedlichen Startpunkten auszuführen und die Ergebnisse anschließend auf einer unabhängigen Validierungsmenge zu vergleichen [CMB23, S. 213]. Abbildung 2.2.5 veranschaulicht das Verfahren des Gradientenabstiegs.

Der Gradient $\nabla_w L\left(w^{(t)}\right)$ ist ein Vektor und enthält als Komponenten die partiellen Ableitungen der Funktion L nach den Gewichten w. Die Parameter lassen sich dabei als Vektor w auffassen, der nach jedem Schritt iterativ angepasst wird. Formal ergibt sich das Aktualisierungsschema [CMB23, S. 214] zu:

$$w^{(t+1)} = w^{(t)} - \epsilon \nabla_{w^{(t)}} L\left(w^{(t)}\right), \tag{2.2.14}$$

wobei ϵ die Lernrate (learning rate) ist, ein positiver Skalar, der die Größe der einzelnen Schritte bestimmt. Nach jeder Aktualisierung wird der Gradient für den neuen Parametervektor $w^{(t+1)}$ erneut berechnet und der Prozess wiederholt.

Wird der Verlust bei jeder Aktualisierung über alle Trainingsdaten hinweg berechnet, spricht man von *Batch Gradient Descent*, dem Standardverfahren des Gradient Descents. Dieses Verfahren ist insbesondere bei sehr großen Datensätzen rechenintensiv und führt zu langsamen Lernprozessen, da für jeden Schritt das gesamte Trainingsset berücksichtigt werden muss [Rud17].

Im Gegensatz dazu wird beim *Mini-Batch Gradient Descent* nur eine zufällig ausgewählte Teilmenge der Trainingsdaten verwendet. Größere Mini-Batches liefern genauere Schätzungen des Gradienten, erhöhen jedoch den Aufwand für jede einzelne Gradientenberechnung. Beim Mini-Batch Gradient Descent wird nur eine zufällig ausgewählte Teilmenge der Trainingsdaten verwendet. Dadurch reduziert sich der Rechenaufwand pro Schritt.

Eine weitere, häufig genutzte Alternative für große Datensätze ist das *Stochastic Gradient Descent* (SGD) [Bot10]. Dabei wird der Parametervektor auf Grundlage eines einzelnen zufällig ausgewählten Trainingsbeispiels aktualisiert:

$$w^{(t+1)} = w^{(t)} - \epsilon \nabla_{w^{(t)}} L_i\left(w^{(t)}\right). \tag{2.2.15}$$

Der Begriff „stochastisch“ weist darauf hin, dass der berechnete Gradient nur eine noisy Approximation des tatsächlichen Gradienten über das gesamte Trainingsset

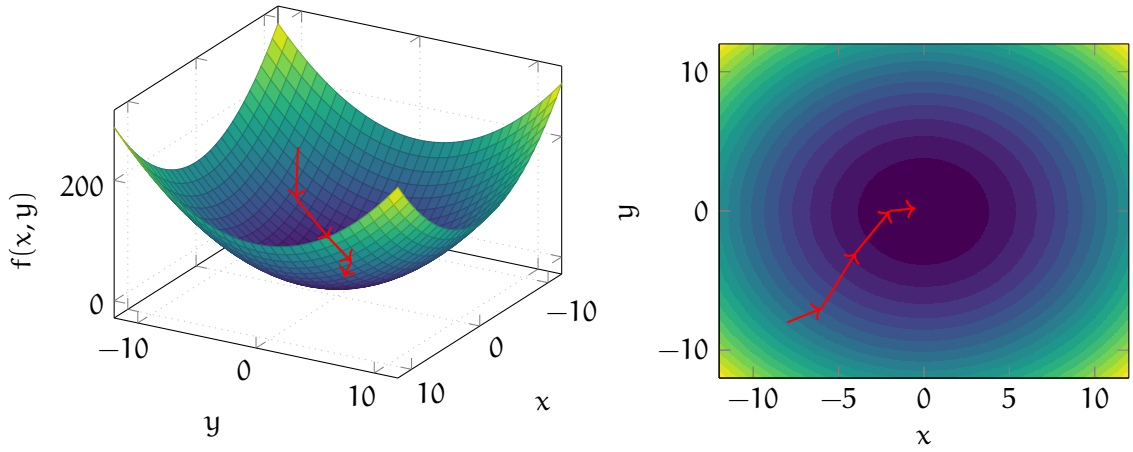


Abbildung 2.2.5: Visualisierung des Gradient Descent auf der Funktion $f(x, y) = x^2 + y^2$. Links ist die Zielfunktion als 3D-Oberfläche dargestellt. Der rote Pfeilpfad zeigt die sukzessiven Schritte des Gradientenverfahrens, die das Verfahren von einem Startpunkt in Richtung des globalen Minimums bei $(0, 0)$ führen. Rechts ist der Draufsicht mit Niveaulinien von $f(x, y)$.

darstellt [DFO20, S. 231]. Trotz dieser Verzerrung ist SGD effektiv für große Machine-Learning-Probleme, da es schnelle und regelmäßige Updates ermöglicht [BCN18].

Backpropagation

Die benötigten Ableitungen der Fehlerfunktion mit Bezug auf die einzelnen Parameter des Netzes können effizient mithilfe der *Backpropagation* [RHW86] berechnet werden. Dabei handelt es sich um ein Verfahren, das die Gradienten Schritt für Schritt rückwärts durch das Netzwerk propagiert. Die Berechnungen folgen dabei in umgekehrter Richtung dem Ablauf des Vorwärtsdurchlaufs, bei dem die Netzwerkausgabe bestimmt wird. Die folgende Darstellung orientiert sich an [CMB23, S. 233–238].

Der Backpropagation besteht aus zwei zwei Hauptphasen: *Forward-Pass* und *Backward-Pass*.

Im Forward-Pass werden die Eingabedaten durch das Netzwerk geleitet, und die Ausgabe wird berechnet. Dabei werden in jeder Schicht die Aktivierungen der Neuronen sowie Zwischenergebnisse (wie die Summeneingänge), die später für die Rückwärtsberechnung der Gradienten benötigt werden. Auf dieser Grundlage kann anschließend der Wert der Verlustfunktion $L(w)$ für die aktuellen Parameter w bestimmt werden. Für eine einzelne Eingabe (x_l, y_l) sei dabei die Fehlerfunktion L_l betrachtet.

Wie in Abschnitt 2.2.1 erläutert, berechnet jedes Neuron zunächst eine gewichtete Summe seiner Eingaben, die anschließend durch eine nichtlineare Aktivierungsfunktion $\varphi(\cdot)$ transformiert. Für das Neuron j gilt:

$$a_j = \sum_i w_{ji} z_i, \qquad z_j = \varphi(a_j), \tag{2.2.16}$$

wobei w_{ji} das Gewicht der Verbindung von Neuron i zu Neuron j. Die Werte z_i stammen entweder aus den Aktivierungen der vorherigen Schicht oder, im Falle der Eingabeschicht, direkt aus den Eingangsdaten des Netzes.

Im Backward-Pass werden die Gradienten der Verlustfunktion bezüglich der Gewichte mithilfe der Kettenregel berechnet. Zur Berechnung der Ableitung der Fehlerfunktion L_l bezüglich eines Gewichts w_{ji} nutzen wir die Kettenregel. Da L_l vom Gewicht w_{ji} nur über den Summeneingang a_j abhängt, ergibt sich:

$$\frac{\partial L_l}{\partial w_{ji}} = \frac{\partial L_l}{\partial a_j} \cdot \frac{\partial a_j}{\partial w_{ji}} = \delta_j \cdot z_i, \tag{2.2.17}$$

wobei δ_j den der Fehler des Neurons j bezeichnet und als die Ableitung der Verlustfunktion nach dem Summeneingang a_j definiert wird.

Aus Gleichung (2.2.17) folgt, um den Gradient für alle Gewichte im Netz zu berechnen, genügt es also, den Fehler δ_j eines Neurons mit der Aktivierung z_j seines Eingabe-Neurons zu multiplizieren.

Falls ein Neuron j im Output-Layer liegt, lässt sich der Fehler als

$$\delta_j = \frac{\partial L}{\partial a_j} = \frac{\partial L}{\partial \hat{y}_j} \cdot \varphi'(a_j) \tag{2.2.18}$$

wobei φ' die Ableitung der Aktivierungsfunktion ist.

Befindet sich ein Neuron j in einem Hidden Layer, so muss eine Summe über alle Neuronen k der nachfolgenden Schicht gebildet werden, da das Neuron j mit allen Neuronen im sukzessiven Layer verbunden ist. Die Aktivierung z_j geht somit in die Berechnungen der folgenden Schicht ein. Für den Fehler δ_j eines Neurons j im Hidden Layer gilt daher:

$$\delta_j = \frac{\partial L}{\partial a_j} = \sum_k \frac{\partial L}{\partial a_k} \cdot \frac{\partial a_k}{\partial a_j}. \tag{2.2.19}$$

Für zwei direkt miteinander verbundene Neuronen j (in der aktuellen Schicht) und k (in der nächsten Schicht) gilt $\frac{\partial a_k}{\partial a_j} = \frac{\partial a_k}{\partial z_j} \cdot \frac{\partial z_j}{\partial a_j} = w_{kj} \cdot \varphi'(a_j)$, wobei w_{kj} das

Gewicht der Verbindung von Neuron j zu Neuron k und $\varphi'(a_j)$ die Ableitung der Aktivierungsfunktion ist. Setzt man diesen Ausdruck in Gleichung (2.2.19) ein, so ergibt sich der Fehler für ein Neuron j im Hidden Layer:

$$\delta_j = \varphi'(a_j) \cdot \sum_k w_{kj}\, \delta_k. \tag{2.2.20}$$

Damit wird klar, dass der Fehler δ_j eines Hidden-Neurons durch Backpropagation der Fehlerwerte δ_k aus der nachfolgenden Schicht berechnet werden kann.

2.3 TRANSFORMERS

Der Transformer [VSP+23] stellt eine der bedeutendsten Entwicklungen im Bereich des Deep Learning dar [KNH+22] und findet inzwischen breite Anwendung in verschiedenen Feldern wie der natürlichen Sprachverarbeitung (NLP), der Computer Vision (CV) sowie der Sprachverarbeitung. Im Gegensatz zu CNNs benötigen Transformer weder rekurrente noch konvolutionale Strukturen zur Modellierung von Sequenzen, sondern setzen ausschließlich auf Attention-Mechanismen in Kombination mit Feed-Forward-Netzwerken. In diesem Kapitel werden die zentralen Bausteine von Transformer-Modellen vorgestellt und ihre Funktionsweise näher erläutert.

2.3.1 *Eingaberepräsentationen*

Die Eingaberepräsentation bildet die Grundlage für die Verarbeitung von Eingabetext durch Transformer-Modelle. Sie umfasst die Zerlegung von Texten in Tokens, die Abbildung dieser Tokens auf Vektoren mittels Embeddings sowie die Integration von Positionsinformationen über Positional Encoding. In den folgenden Abschnitten werden diese drei Komponenten im Detail beschrieben.

Tokenisierung

In praktischen Anwendungen des Deep Learning im Bereich Natural Language Processing (NLP) wird der Eingabetext zunächst in eine tokenisierte Präsentationen überführt. Die Tokenisierung ist ein Vorverarbeitungsschritt, bei dem ein Text in kleinere Einheiten, sogenannte Tokens, zerlegt wird. Tokens können einzelne Zeichen, ganze Wörter oder Teile von Wörtern darstellen [SN12].

Während es verschiedene Ansätze zur Tokenisierung gibt, wie etwa Zeichen- oder Wort-Tokenisierung, hat sich in der modernen Sprachverarbeitung die *Subword-*

Tokenisierung oft durchgesetzt [SHB16]. Das Grundprinzip besteht darin, seltene Wörter in kleinere Einheiten zu zerlegen, sodass das Modell auch mit unbekannten oder fehlerhaft geschriebenen Wörtern umgehen kann. Häufig vorkommende Wörter werden hingegen als ganze Tokens beibehalten, um die Länge der Eingabesequenz in einem handhabbaren Rahmen zu halten [TWW22, S. 33–34]. Der Tokenizer wird dabei auf einem großen Korpus während des Pre-Trainings gelernt, sodass er eine für das Modell geeignete Zerlegung des Textes erzeugen kann.

Es existieren verschiedene Ansätze zur Subword-Tokenisierung. Ein häufig eingesetztes Verfahren ist die *Byte Pair Encoding* (BPE) [SHB16], das ursprünglich aus der Datenkompression stammt und für die Text-Tokenisierung angepasst wurde. Die Grundidee besteht darin, häufig vorkommende Zeichenfolgen schrittweise zusammenzuführen. Zu Beginn wird die Tokenliste aus den einzelnen Zeichen des Alphabets sowie einem speziellen Wortend-Symbol '·' initialisiert. Im nächsten Schritt wird ein Textkorpus nach den am häufigsten auftretenden aufeinanderfolgenden Token-Paaren durchsucht. Das am häufigsten vorkommende Paar, beispielsweise ('A', 'B'), wird anschließend zu einem neuen Token ('AB') zusammengeführt. Dieser Vorgang wird iterativ wiederholt, sodass nach und nach längere und häufigere Zeichenfolgen als eigenständige Tokens entstehen. Um zu verhindern, dass Wörter über Wortgrenzen hinweg zusammengeführt werden, wird ein neues Token nicht gebildet, wenn das zweite Token mit einem Leerzeichen beginnt. Auf diese Weise entsteht eine Tokenliste, die sowohl häufige Wörter als Ganzes als auch seltenere Wörter in kleinere Einheiten zerlegt.

Token-Embeddings

Transformermodelle können keine Rohtexte in Form von Zeichenketten direkt als Eingabe verarbeiten. Token-Embeddings repräsentieren jedes Token durch einen Vektor in einem hochdimensionalen Raum mit typischerweise einigen hundert Dimensionen. Formal lässt sich dies nach [CMB23, S. 375] durch eine Embedding-Matrix $E \in \mathbb{R}^{d \times |V|}$ darstellen, wobei d die Dimension des Embedding-Raums und $|V|$ die Größe des Vokabulars bezeichnet. Für einen one-hot-kodierten Eingabevektor x_i lässt sich der zugehörige Embedding-Vektor berechnen als

$$v_i = Ex_i,$$

wobei v_i derjenigen Spalte von E entspricht, die mit dem Token x_n assoziiert ist. Die Embedding-Matrix E kann entweder zufällig initialisiert oder aus einer vortrainierten

Standard-Embedding-Matrix übernommen werden [CMB23, S. 376]. Die Embedding-Matrix E wird aus einem Trainingskorpus gelernt. Eine gängige Methode dafür ist *Word2Vec* [MCCD13], die lässt sich als ein einfaches zweischichtiges neuronales Netz auffassen, das aus einer Eingabeschicht, einer Projektionsschicht und einer Ausgabeschicht besteht.

Dabei entsteht jedes Trainingsbeispiel durch die Betrachtung eines „Fenster" von c benachbarten Wörtern (typischerweise $c = 5$), die als Kontextwörter dienen. Die Wörter werden dabei als unabhängige Einheiten behandelt, da sie im Vokabular durch Indizes repräsentiert sind. Zu diesem Ansatz wurden 2 Varianten vorgestellt: *Continuous Bag of Words* (CBOW) und *Skip-gram*.

Im CBOW ist das Trainingskriterium, das aktuelle (Mittel-)Wort korrekt aus dem Kontext zu klassifizieren. Diese Architektur wird als Bag-of-Words-Modell bezeichnet, da die Reihenfolge der Wörter innerhalb des Kontextes vollständig ignoriert wird. Gegeben eine Sequenz von Trainingswörtern $w_1, w_2, \ldots, w_n$, berechnet CBOW nach [WXCH17] die Vektor-Repräsentation des aktuellen Wortes in dem Hidden Layer als Mittelwert der Vektoren der Kontextwörter $x = \sum_{w_i \in c} e(w_i)$, wobei $e(w_i)$ das Embedding des Kontextwortes w_i bezeichnet. Das Optimierungsziel besteht darin, die Log-Likelihood über das gesamte Trainingskorpus D zu maximieren:

$$\mathcal{L} = \sum_{(w,c)\in D} \log p(w \mid c), \qquad p(w \mid c) = \frac{\exp\left(e'(w)^\top x\right)}{\sum_{w'\in V} \exp\left(e'(w')^\top x\right)} \tag{2.3.1}$$

Hierbei bezeichnet $e'(w)$ das Output-Embedding von w und c ist die Menge der entsprechenden Kontextwörter. Die Wahrscheinlichkeit $p(w \mid c)$, dass das Zielwort w im Kontext c auftritt, wird durch eine Softmax-Funktion beschrieben.

Das Skip-gram hingegen dreht die Vorhersagerichtung um: Statt das aktuelle Wort aus dem Kontext vorherzusagen, wird das (Mittel-)Wort als Eingabe gegeben, und die Kontextwörter dienen als Zielwerte. Das Skip-gram-Modell maximiert nach [MSC+13] die durchschnittliche logarithmische Wahrscheinlichkeit

$$\mathcal{L} = \frac{1}{n} \sum_{w} \sum_{c_i \in c} \log p(c_i \mid w), \qquad p(c_i \mid w) = \frac{\exp\left(e'(c_i)^\top e(w)\right)}{\sum_{w'\in V} \exp\left(e'(w')^\top e(c_i)\right)} \tag{2.3.2}$$

Ein wesentliches Merkmal ist, dass semantisch ähnliche Wörter in räumlich nahe Positionen im Embedding-Raum abgebildet werden. Ein klassisches Beispiel für diesen Effekt ist die Vektoroperation $\mathrm{vec}(\text{“Madrid”}) - \mathrm{vec}(\text{“Spain”}) + \mathrm{vec}(\text{“France”}) \simeq \mathrm{vec}(\text{“Paris”})$, die zeigt, dass semantische Beziehungen im Vektorraum explizit abgebildet werden können [MCCD13].

Positional Encoding

Da die vom Transformer gelernten Repräsentationen unabhängig von der Reihenfolge der Eingabetokens sind, ergibt sich bei der Verarbeitung sequenzieller Daten das Problem [CMB23, S. 371–372]. Um diese Einschränkung zu überwinden, wird den Token-Embeddings eine Positionskodierung hinzugefügt, die die Reihenfolge der Tokens innerhalb der Sequenz abbildet. Die Token-Embeddings und die Positionscodierungen werden komponentenweise addiert, um das finale Eingabe-Embedding zu erhalten, was als Eingang für die erste Schicht des Encoder-Stacks und des Decoder-Stacks dient.

Die Positionskodierungen besitzen dieselbe Dimension d wie die Eingabe-Embeddings, damit beide Vektoren komponentenweise addiert werden können. Die von Vaswani et al. [VSP+23] gewählten Positional Encoding an der Position pos in der Sequenz und für die Dimension i basieren auf Sinus- und Kosinusfunktionen unterschiedlicher Frequenzen lautet:

$$PE_{(pos,\,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{2.3.3}$$

$$PE_{(pos,\,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{2.3.4}$$

Die Kodierungswerte an den geraden und ungeraden Positionen werden entsprechend durch Gleichung 2.3.3 bzw. 2.3.4 berechnet.

2.3.2 *Attention*

Attention ist ein zentrales Konzept in Transformer-Architekturen. Die Idee dieser Mechanismen wurde ursprünglich als Erweiterung von RNNs eingeführt [BCB16]. Später zeigten Vaswani et al. [VSP+23], dass sich die Leistung deutlich verbessern lässt, wenn auf die rekurrente Struktur vollständig verzichtet wird und das Modell ausschließlich auf dem Attention-Mechanismus basiert.

Attention ermöglicht es dem Modell, bei der Verarbeitung eines Elements einer Sequenz dynamisch relevante Teile der gesamten Eingabesequenz zu gewichten. Im Fall von Textsequenzen handelt es sich dabei um Token-Embeddings (vgl. Abschnitt 2.3.1), während bei Bilddaten entsprechende visuelle Token-Embeddings verwendet werden. Dabei wird für jedes Token eine gewichtete Kombination aller anderen Tokens berechnet, wodurch kontextabhängige Repräsentationen entstehen. Abbildung 2.3.1 illustriert

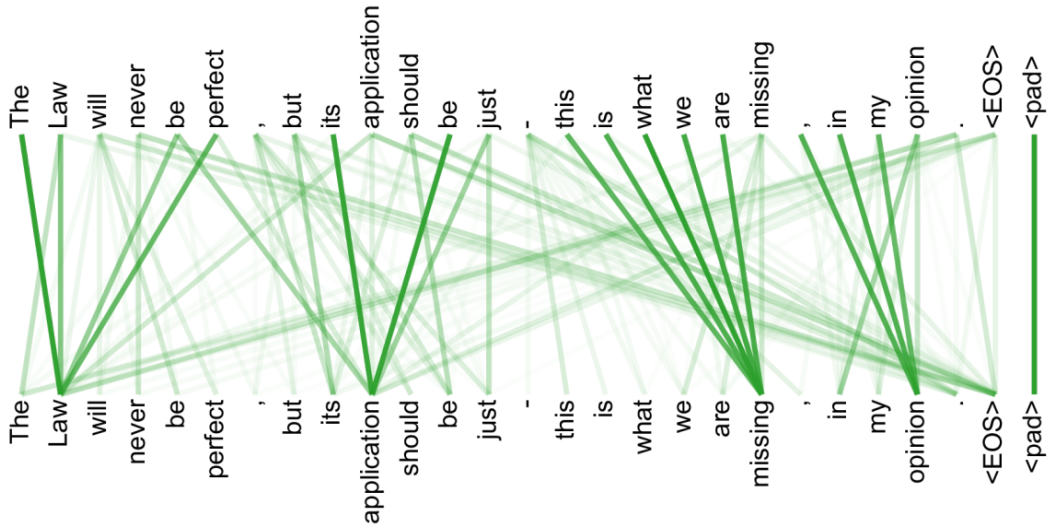


Abbildung 2.3.1: Beispiel nach [CMB23, S. 360] für gelernte Attention-Gewichte in einem Transformer-Modell von [VSP+23]. Die Verbindungen zwischen den Tokens veranschaulichen, welche Wörter beim Verarbeiten eines bestimmten Tokens besonders stark berücksichtigt werden. Die Stärke der Aufmerksamkeit wird durch die Farbintensität der Linien angezeigt.

die in einem Transformer gelernten Attention-Gewichte anhand eines Beispielsatzes. In dem Beispiel richtet das Wort *missing* besonders große Attention-Gewichte auf *what*, *we* und *are* (erkennbar an der stärkeren Farbintensität). Das Modell hebt damit die Abhängigkeit zwischen *missing* und dem Ausdruck *what we are* hervor.

Die Menge von Eingabevektoren $x_1, \ldots, x_n$ ist in einem Embedding-Raum gegeben, die in eine neue Menge der Ausgabevektoren $y_1, \ldots, y_n$ abgebildet werden sollen. Im Fall von Self-Attention besitzen die Ausgaben die gleiche Länge wie die Eingabesequenz, liegen jedoch in einem neuen Raum, der reichhaltigere semantische Strukturen abbilden kann. Dabei hängt der Wert von y_i nicht nur vom entsprechenden Eingabevektor x_i ab, sondern von allen Eingabevektoren $x_1, \ldots, x_n$.

Self-Attention

Um zu bestimmen, wie stark ein Token auf ein anderes Token „achten“ soll, muss zunächst berechnet werden, wie ähnlich diese beiden Vektoren sind. Dieser Mecha-

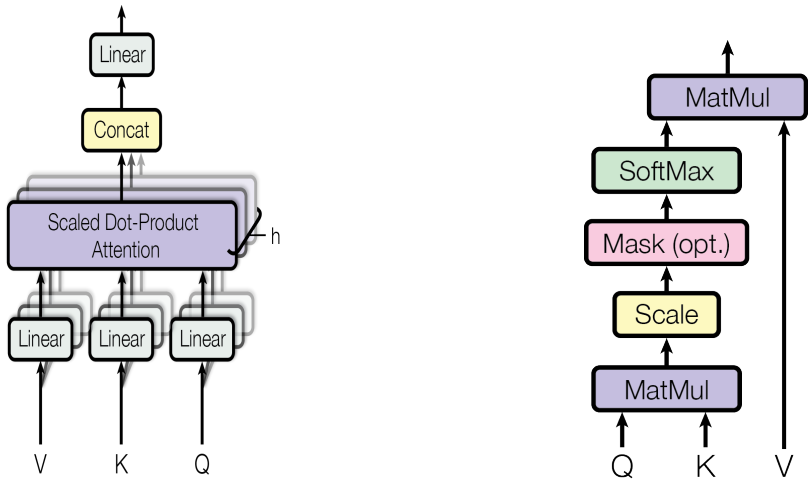


Abbildung 2.3.2: Aufbau der Architektur von Multi-Head Attention (links) und Scaled Dot-Product Attention (rechts) aus [VSP+23].

nismus heißt *Self-Attention* [VSP+23], da dieselbe Eingabesequenz genutzt wird, um Queries, Keys und Values zu erzeugen. Dazu wird die Eingabematrix X jeweils mit eigenen Gewichtsmatrizen multipliziert:

$$Q = XW^{(q)}, \quad K = XW^{(k)}, \quad V = XW^{(v)}$$

Die Gewichtsmatrizen $W^{(q)}, W^{(k)} \in \mathbb{R}^{d \times d_k}$ und $W^{(v)} \in \mathbb{R}^{d \times d_v}$ sind Parameter, die während des Trainings gelernt werden. Dabei legt d_k die Dimension der Query- und Key-Vektoren fest (üblicherweise $d_k = d$), während d_v die Dimension der Ausgaben bestimmt. Die Gewichtsmatrizen sind notwendig, weil nicht alle Merkmale eines Token-Vektors gleich wichtig für die Berechnung der Ähnlichkeit sind. Durch die lernbaren Parameter kann das Modell bestimmte Merkmale hervorheben und andere abschwächen.

Die Stärke der Beziehung zwischen dem Token x_i und Token x_j ergibt sich aus dem Skalarprodukt $q_i^\top k_j$, das ein Maß für die Ähnlichkeit zwischen den beiden Tokens darstellt. Um daraus gültige Attention-Gewichte zu berechnen, wird die Softmax-Funktion angewandt:

$$w_{ij} = \text{Softmax}\left(q_i^\top k_j\right) \tag{2.3.5}$$

Die Softmax-Funktion sorgt dafür, dass die berechneten Koeffizienten normalisiert werden, sodass sie nicht negativ sind und sich über alle Tokens hinweg zu eins sum-

mieren. In diesem Zusammenhang hat die Softmax-Funktion keine probabilistische Bedeutung, sondern dient lediglich der Normalisierung der Gewichte.

Die Ausgabevektoren y_i ergeben sich anschließend als gewichtete Summe der Value-Vektoren:

$$y_i = \sum_{j=1}^{n} w_{ij} v_j. \tag{2.3.6}$$

Zur kompakteren Darstellung der Formel (2.3.5) und (2.3.6) wird die Berechnung häufig in Matrixschreibweise formuliert. Dabei bezeichnet $Y \in \mathbb{R}^{n \times d}$ die Ausgabematrix, deren Zeilen den berechneten Vektoren y_i entsprechen.

$$Y = \text{Softmax}\left(QK^T\right) V. \tag{2.3.7}$$

Ein Problem der Softmax-Funktion besteht darin, dass die Gradienten für Eingaben mit großem Betrag exponentiell (großen Logits) klein werden können. Um diesen Problem zu vermeiden, wird das Skalarprodukt der Query- und Key-Vektoren vor der Anwendung der Softmax-Funktion normalisiert. Dies geschieht, indem das Produkt durch die Standardabweichung, also die Quadratwurzel von d_k, geteilt wird. Die Ausgabe der Attention-Schicht ergibt sich damit zu

$$Y = \text{Attention}(Q, K, V) \equiv \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V.$$

Dieses Verfahren wird als *Scaled Dot-Product Self-Attention* bezeichnet und wird in der Abbildung 2.3.2 abgebildet.

Multi-Head Self-Attention

Die Multi-Head Self-Attention (MHA) [VSP+23] stellt eine Erweiterung des Self-Attention-Mechanismus dar. Anstatt nur eine einzelne Self-Attention-Berechnung durchzuführen, werden mehrere dieser Mechanismen parallel ausgeführt. Jede einzelne dieser Berechnungen wird als Head bezeichnet. Durch die parallele Nutzung mehrerer Heads kann das Modell verschiedene Repräsentationen und Abhängigkeiten innerhalb der Sequenz erfassen, da jeder Head unterschiedliche Aspekte der Eingabe fokussieren kann [TWW22, S. 67]. MHA stellt eine zentrale Komponente moderner Transformermodelle dar und hat sich insbesondere im Natural Language Processing (NLP) [DCLT19] sowie zunehmend in der Computer Vision [DBK+21] als Standardarchitektur durchgesetzt.

Jeder Head $i \in \{1, \ldots, h\}$ verarbeitet eine andere Projektion der Eingabedaten, führt eine eigene Scaled Dot-Product Attention aus und extrahiert unterschiedliche Beziehungsmuster innerhalb der Sequenz. Die Ergebnisse aller h Köpfe werden anschließend konkateniert und mit einer weiteren Gewichtsmatrix $W^O \in \mathbb{R}^{h d_v \times d}$ multipliziert. Das beschriebene Vorgehen ist in Abbildung 2.3.2 dargestellt. Formal lässt sich dies wie folgt beschreiben:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{Head}_1, \ldots, \text{Head}_h) \cdot W^O, \\ \text{Head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.3.8}$$

Im Originalpapier von Vaswani et al. [VSP+23] wird mit $h = 8$ parallelen Heads und $d_k = d_v = \frac{D}{h} = 64$ gearbeitet.

2.3.3 *Transformer-Modelle*

Der originale Transformer-Modell von Vaswani et al. [VSP+23] basiert auf der Encoder-Decoder-Architektur und ist in Abbildung 2.3.3 dargestellt.

Als Eingabe der Modelle dient eine Sequenz von Wörtern, die zunächst tokenisiert (vgl. Abschnitt 2.3.1) und in Token-Embeddings (vgl. Abschnitt 2.3.1) umgewandelt wird. Da der Attention-Mechanismus keine Informationen über die Reihenfolge der Tokens enthält, wird zusätzlich eine Positionscodierung eingebracht, um die sequenzielle Struktur des Textes abzubilden. Die Token-Embeddings werden daher mit entsprechenden Positions-Embeddings (vgl. Abschnitt 2.3.1) addiert, die für jedes Token seine Position in der Sequenz repräsentieren. Die resultierenden Vektoren durchlaufen anschließend mehrere hintereinander verbundenen Encoder-Layer, wobei jeder Layer auf den Ausgaben des vorherigen Layers aufbaut. Formal transformiert der Encoder eine Eingabesequenz $x = (x_1, \ldots, x_n)$ in eine Sequenz kontinuierlicher Repräsentationen $z = (z_1, \ldots, z_n)$. Diese finale Encoder-Ausgabe z dient dem Decoder als Kontextinformation.

Die Ausgabe des Encoders wird an jede Decoder-Schicht weitergegeben. Der Decoder erzeugt daraufhin eine Vorhersage für das wahrscheinlichste nächste Token in der Zielsequenz. Am Anfang der Eingabesequenz des Encoders wird ein spezielles Start-Token `<start>` eingefügt, das den Beginn der Sequenz markiert. Damit die Eingabesequenz für den Decoder um eine Position nach rechts verschoben wird und das Token x_n als Eingabe für die Vorhersage von y_{n+1} dient . Die bis zu diesem Zeitpunkt erzeugten Tokens werden Schritt für Schritt wieder in den Decoder eingespeist, um das jeweils nächste Token zu erzeugen. Dieser Prozess wird solange wiederholt, bis

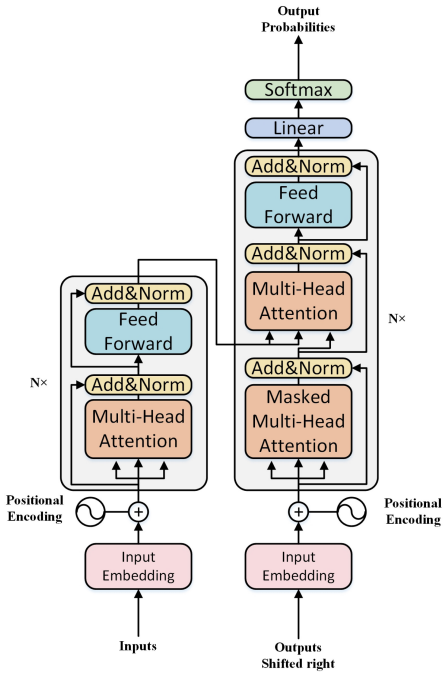


Abbildung 2.3.3: Aufbau der Transformer-Architektur nach [VSP+23] mit einem Encoder-Stack (links) zur Verarbeitung der Eingabesequenz und einem Decoder-Stack (rechts) zur Generierung der Ausgabe.

entweder ein spezielles End-of-Sequence-Token `<eos>` vorhergesagt wird oder eine zuvor definierte maximale Sequenzlänge erreicht ist, die durch die Eingabegröße des Transformers bestimmt wird. Auch hier baut jeder Decoder-Layer auf dem Output des vorherigen Layers auf.

Encoder

Der Encoder ist in der linken Hälfte von Abbildung 2.3.3 dargestellt. Der Encoder-Stack besteht aus N (N = 6 im ursprünglichen Transformer-Modell [VSP+23]) identischen Schichten, die sequenziell angeordnet sind. Der Input der ersten Encoder-Schicht ist die eingebettete Eingabesequenz, angereichert mit Positionscodierungen. Die Ausgabe einer Schicht dient als Eingabe für die nächste. Die letzte Encoder-Schicht liefert kontextreiche Vektoren, die als Eingabe für den Decoder dienen.

Jede Schicht enthält zwei Unterschichten: Die erste ist ein *Multi-Head-Self-Attention* (vgl. Abschnitt 2.3.2), die zweite ist ein *Position-Wise-Feedforward-Netzwerk* (vgl. Ab-

schnitt 2.2.1). Die FFN-Unterschicht besteht aus einem einfachen zweischichtigen, vollständig verbundenen neuronalen Netz. Im Gegensatz zu einer gemeinsamen Verarbeitung der gesamten Sequenz von Embeddings wird jedoch jedes Token-Embedding unabhängig voneinander transformiert. Aus diesem Grund wird diese Schicht als Position-Wise FFN bezeichnet. Die Ausgabe jeder Unterschicht wird über eine *Residualverbindung* [HZRS16] mit ihrer jeweiligen ursprünglichen Eingabe addiert, und auf die Summe wird anschließend eine *Layer-Normalisierung* (vgl. Abschnitt 2.2.1) angewendet. Normalisierung und Residualverbindungen sind bewährte Techniken, die das Training tiefer neuronaler Netze effizienter und stabiler machen.

Die Ausgabe der Multi-Head Attention Unterschicht ergibt sich formal durch die Anwendung einer Residual-Verbindung, gefolgt von einer Layer-Normalisierung. Dabei werden die Eingaben x gleichzeitig als Query, Key und Value genutzt. Die Berechnung erfolgt wie folgt:

$$\text{Output} = \text{LayerNorm}(\mathbf{x} + \text{MultiHead}(x, x, x)).$$

Zusätzlich enthält der Transformer-Stack ein kleines Position-Wise Feedforward-Netzwerk (FFN), das auf jede Position unabhängig und identisch angewendet wird. Das Feedforward-Netzwerk besteht aus zwei linearen Transformationen mit einer nichtlinearen Aktivierungsfunktion ReLU dazwischen. Auch hier wird eine Residual-Verbindung mit anschließender Layer-Normalisierung verwendet. Die vollständige Transformation des Eingabesignals $\mathbf{x}$ durch den Feedforward-Teil lässt sich wie folgt darstellen:

$$\begin{aligned} \text{Output} &= \text{LayerNorm}(x + \text{FFN}(x)) \\ \text{FFN}(x) &= \max(0, xW_1 + b_1)W_2 + b_2 \end{aligned} \tag{2.3.9}$$

Decoder

Der Decoder ist in der rechten Hälfte von Abbildung 2.3.3 dargestellt. Der Decoder-Stack besteht analog zum Encoder aus N identischen Schichte. Jede dieser Schichten enthält drei Unterschichten. Neben der Self-Attention und dem Feed-Forward-Netzwerk wie im Encoder kommt im Decoder eine zusätzliche Self-Attention-Schicht hinzu. Dabei erfüllt jede Multi-Head-Attention-Schicht im Decoder eine eigene spezifische Funktion. Der Input der ersten Schicht ist die eingebettete Zielsequenz y, ergänzt durch Positionscodierungen. Jede Schicht verarbeitet zwei Informationsquellen: die eigene vorherige Decoder-Ausgabe über Masked Self-Attention und die Encoder-Ausgabe über Encoder-Decoder Attention.

Die erste Unterschicht, die *Masked-Self-Attention* [VSP+23], ermöglicht es jedem Token, Informationen ausschließlich aus früheren Tokens der Decoder-Eingabesequenz zu beziehen, jedoch nicht aus zukünftigen Positionen. Dadurch wird verhindert, dass der Decoder während des Trainings unrechtmäßig auf zukünftige Informationen zugreift und so lediglich das nächste Token aus der Eingabe „abschreibt“ [CMB23, S. 384–385]. Wie im [NP20] beschrieben, wird in jedem Attention-Kopf die Maskierung durch Addition einer Matrix $M \in \mathbb{R}^{d \times d}$ realisiert, wobei die Einträge für unzulässige (zukünftige) Positionen mit $-\infty$ belegt sind. Bei der anschließenden Anwendung der Softmax-Funktion führen dazu, dass die entsprechenden Wahrscheinlichkeiten exakt null werden, da gilt $e^{-\infty} = 0$. Die Berechnung nach [NP20, XZ23] erfolgt analog zur regulären Attention, jedoch mit Maskierung:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V; \quad M(i,k) = \begin{cases} 0 & \text{falls } i \leqslant k, \\ -\infty & \text{falls } i > k \end{cases} \tag{2.3.10}$$

Die zweite Unterschicht ist die *Encoder-Decoder Multi-Head Attention* (auch als Cross-Attention bezeichnet). Hierbei dienen die Ausgaben des Encoders gleichzeitig als Queries Q und Keys K, während die Ausgabe des vorherigen Decoder-Layers als Values V verwendet wird. Diese Schicht erlaubt es dem Decoder, gezielt Informationen aus dem Encoder-Kontext zu beziehen und relevante Teile der Eingabesequenz zu fokussieren. Dadurch wird eine direkte Verbindung zwischen dem aktuell generierten Decoder-Token und dem gesamten Input ermöglicht.

Analog zum Encoder werden Residualverbindungen um jede der Unterschichten herum eingesetzt, gefolgt von einer Layer-Normalisierung.

Die finale Ausgabe des Decoder-Stacks, bezeichnet als $z \in \mathbb{R}^{T \times d}$, wobei T die Zielsequenzlänge ist, wird durch eine lineare Projektion und eine Softmax-Funktion in Wahrscheinlichkeiten über das Zielvokabular überführt:

$$\hat{y}_t = \text{Softmax}(z_t W_o + b_o)$$

Dabei ist $W_o \in \mathbb{R}^{d \times |V|}$ die Gewichtsmatrix der linearen Projektion, $b_o \in \mathbb{R}^{|V|}$ ein Bias-Term und $|V|$ die Größe des Zielvokabulars. Der Vektor $\hat{y}_t \in \mathbb{R}^{|V|}$ enthält die vorhergesagten Wahrscheinlichkeiten für das nächste Token an Position t.

Das Token mit der höchsten Wahrscheinlichkeit wird typischerweise durch

$$y_t = \arg\max_j \hat{y}_{t,j}$$

ausgewählt.

2.3.4 *Vision Transformers*

Vision Transformer (ViT) [DBK+21] verwendet im Unterschied zu klassischen Transformern (vgl. Abschnitt 2.3) keine Wort- oder Subword-Tokens, sondern kleine Bildausschnitte (Patches) als Eingabetokens. Ein weiterer Unterschied ist, dass ViT ausschließlich einen Encoder nutzt.

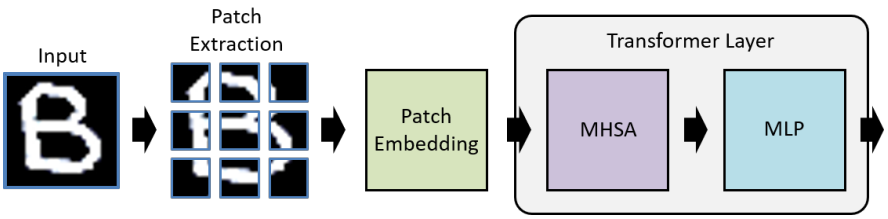


Abbildung 2.3.4: Vision-Transformer-Pipeline nach [IK23]: Das Bild wird in feste Patches zerlegt, jedes Patch linear in einen Embedding-Raum projiziert, mit Positions-Embeddings addiert und die entstehende Sequenz schließlich von einem Transformer-Encoder verarbeitet.

Das Eingangsbild $x \in \mathbb{R}^{H \times W \times C}$, wobei C die Anzahl der Farbkanäle angibt ($C = 3$ für RGB-Bilder), wird durch ein Patch-Extraction-Modul in nicht überlappende Patches der Größe $P \times P$ unterteilt (siehe Abbildung 2.3.4). Jeder Patch wird anschließend zu einem Vektor der Dimension $\mathbb{R}^{P^2 \cdot C}$ abgeflacht. Auf diese Weise entsteht eine Sequenz von Patch-Vektoren $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, wobei $N = \frac{HW}{P^2}$ die Gesamtzahl der Patches pro Bild ist. Um diese Vektoren in den Modellraum zu überführen, wird jeder Patch-Vektor mit einer lernbaren linearen Projektion in eine Dimension d abgebildet.

Zusätzlich wird ein spezielles `[class]`-Token vorangestellt, dessen Repräsentation später für die Bildklassifikation genutzt wird. Um die räumliche Struktur der Patches zu berücksichtigen, werden zu den Embeddings Positions-Embeddings addiert. Die vollständige Eingabesequenz lautet somit nach [DBK+21]

$$x = [x_{\text{class}}, x_1 W_E, \ldots, x_n W_E] + E_{\text{pos}}$$

wobei $W_E \in \mathbb{R}^{(P^2 \cdot C) \times d}$ die Projektionsmatrix und E_{pos} die Positions-Embeddings bezeichnen. Die resultierende Vektorsequenz x dient anschließend als Eingabe für einen Standard-Transformer-Encoder (vgl. Abschnitt 2.3.3).

3 VERWANDTE ARBEITEN

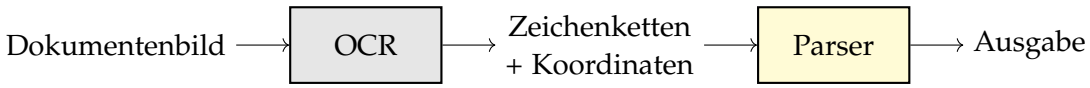
In diesem Kapitel werden zentrale Forschungsarbeiten im Bereich der Handschrifterkennung und Informationsextraktion aus gescannten Dokumentenbildern vorgestellt. Anschließend wird auf Methoden zur Konfidenzbewertung eingegangen, die eine zuverlässige Einschätzung der Vertrauenswürdigkeit von Modellvorhersagen ermöglichen.

3.1 HANDSCHRIFTERKENNUNG UND INFORMATIONSEXTRAKTION

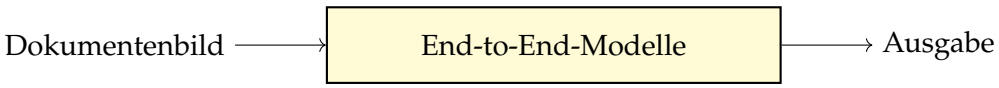
Frühere Ansätze für die Handschriftenerkennung (HTR) nutzten häufig Hidden-Markov-Modelle (HMM) [MLEY+00, BB08]. In einigen Arbeiten wurde HMM mit Recurrent Neural Network (RNN) in Kombination mit einer Connectionist Temporal Classification (CTC)-Verlustfunktion [GFGS06] eingesetzt. Ein Nachteil der HMM-basierten Ansätze bestand jedoch darin, dass sie nur die aktuelle Beobachtung und einer kleinen Anzahl vorheriger Zustände, wodurch längere Abhängigkeiten in der Zeichenfolge vernachlässigt wurden.

Die Methoden [BM17, SNBTL20, Pui17] setzen auf Convolutional Neural Networks (CNN) kombiniert mit RNN. Dabei werden zunächst mehrere Convolutional Layers eingesetzt, um lokale Merkmale aus Textzeilenbildern zu extrahieren. Anschließend verarbeiten Recurrent Layers, meist Bi-directional Long Short-Term Memory (BLSTM) [VDN16], diese Merkmale sequenziell und geben auf Grundlage kontextueller Abhängigkeiten Zeichenwahrscheinlichkeiten aus. Diese Modelle werden häufig als CRNN-CTC-Architekturen bezeichnet, da sie die CTC-Verlustfunktion verwenden, um mit unterschiedlich langen Label- und Vorhersagesequenzen umzugehen, ohne eine explizite Zeichensegmentierung zu benötigen. Ein Nachteil solcher Architekturen ist jedoch die lange Trainingszeit, die vor allem durch die sequentielle Verarbeitung in den rekurrenten Schichten verursacht wird.

Traditionelle Ansätze zur Dokumentenanalyse basieren oft auf der Verwendung von Optical Character Recognition (OCR), um Textinhalte zu extrahieren. Diese OCR-Ergebnisse bilden anschließend die Grundlage für weiterführende Parsing- und Verarbeitungsmethoden [XLC+20, XXL+22] (vgl. Abbildung 3.1.1(a)).



(a) OCR-basierter Ansatz



(b) Ocr-freier End-to-End Ansatz

Abbildung 3.1.1: Vergleich von OCR-basierten und End-to-End-Ansätzen zur Informationsextraktion aus Dokumentenbildern nach [OBK+24]. (a) Traditionelle OCR-basierte Verfahren nutzen von einem externen OCR-Modul erzeugte Textzeichenketten und deren Koordinaten, die anschließend durch einen Parser weiterverarbeitet werden. (b) End-to-End-Modelle hingegen generieren die Ausgaben direkt aus dem Dokumentenbild durch sequenzbasiertes Decoding, ohne auf externe OCR-Ergebnisse zurückzugreifen.

Die LayoutLM-Modellfamilie [XLC+20], zu der auch die Varianten LayoutLMv2 [XXL+22], LayoutLMv3 [HLC+22] und LayoutXLM [XLC+21] kombiniert erkannte Tokens mit Layout-Informationen sowie visuellen Merkmalen, die durch OCR extrahiert werden. Diese Repräsentationen dienen als Eingabe für ein Transformer-Modell, das auf BERT [DCLT19] basiert und für verschiedene Aufgaben trainiert wird, darunter Form Understanding (Erfassen von Formularstrukturen), Receipt Understanding (Rechnungs- bzw. Belegverstehen) sowie die Klassifikation von Dokumentenbildern.

Die aktuellen Fortschritte in der HTR zeigen, dass Transformer-basierte Modellen zunehmend den Stand der Technik bestimmen. Im Gegensatz zu CRNN-CTC-Modellen lernen Transformer-basierte Modelle, Bildbereiche direkt mit den entsprechenden Zielsequenzen zu verknüpfen, sodass das Netzwerk gezielt auf die relevanten Bildregionen fokussiert [PV21], ohne den Umweg über rekurrente Verarbeitungsschritte. Transformer benötigen bei visuellen Aufgaben tendenziell mehr Rechenressourcen als CRNNs, können aber Vorteile bei der parallelen Verarbeitung sequentieller Daten bieten und sind daher eine leistungsfähige Alternative für HTR.

Diese auf OCR basierenden Ansätze zeigen vielversprechende Ergebnisse, leiden jedoch unter hohen Rechenkosten, eingeschränkter Sprachflexibilität und der Fehlerpropagation in nachfolgende Verarbeitungsschritte. In den letzten Jahren hat sich das Paradigma der Handschrift- und Dokumentenerkennung zunehmend in Richtung einer End-to-End-Perspektive entwickelt (vgl. Abbildung 3.1.1(b)). Ziel ist es, Informa-

tionen direkt aus dem vollständigen Dokumentenbild in einem einzigen Schritt ohne vorherige explizite Texterkennung oder Layoutanalyse zu extrahieren. Zu den bekanntesten dieser neueren Ansätze zählen *DAN* [CCP23a], *Dessurt* [DMP+22] und *Nougat* [BCSS23]. Diese Modelle können durch das Fine-Tuning in unterschiedlichen Anwendungsfeldern wie Drucktext- und Szene-Text-Erkennung, Dokumentenverständnis oder Visual Question Answering eingesetzt werden.

Der erster Ansatz in diese Richtung war das *DAN* (Document Attention Network) [CCP23a]. Dieses Modell nutzt einen CNN-basierten Encoder zur Extraktion von Bildmerkmalen sowie einen Transformer-Decoder zur zeichenweisen Sequenzgenerierung. Damit gelang es erstmals, Dokumentenseiten ohne vorherige Segmentierung vollständig und autoregressiv zu transkribieren. Als Weiterentwicklung wurde *Faster DAN* [CCP23b] vorgeschlagen. Hierbei wird die Verarbeitung beschleunigt, indem die Texterkennung auf Zeilenebene parallelisiert durchgeführt wird. Dieser Geschwindigkeitsvorteil geht jedoch oft mit einer höheren Fehlerrate im Vergleich zum ursprünglichen DAN-Modell einher. Dieser Ansatz sind weiterhin auf CNN-Backbones angewiesen und stellen damit keine vollständig Transformer-basierte End-to-End-Architektur dar.

Donut, Pix2Struct, Dessurt, und Nougat sind vollständig Transformer-basierten Vision-Encoder-Decoder-Modellen, die End-to-End trainiert werden. Trotz ähnlicher Grundidee bestehen zwischen diesen Modellen deutliche Unterschiede. *Pix2Struct* setzt einen Vision Transformer (ViT)[DBK+21] als Encoder ein. *Dessurt* hingegen kombiniert einen kleinen CNN-Encoder, der die Eingaben in Feature-Arrays umwandelt, mit einem Transformer-Decoder. *Donut* und *Nougat* verwenden die Swin-Transformer-Architektur[LLC+21] als Encoder. Dabei ist *Nougat* eine speziell für akademische Dokumente angepasste, feinabgestimmte Variante von Donut. Donut teilt viele Vorteile mit Dessurt. Beide nutzen einen Swin-Encoder [LLC+21], allerdings ist die Swin-Architektur von Dessurt flacher und schmaler aufgebaut als die von Donut.

3.2 KONFIDENZBEWERTUNG

Hendrycks und Gimpel [HG18] etablierten einen grundlegenden Baseline-Ansatz zur Konfidenzschätzung. Dabei wird die Softmax-Funktion auf die Logits des Klassifikators angewendet, um diese in eine Wahrscheinlichkeitsverteilung über alle möglichen Klassen zu transformieren. Aus dieser Verteilung wird anschließend die höchste Wahrscheinlichkeit extrahiert, die als Konfidenzwert des Modells interpretiert wird. Dieser Ansatz weist jedoch einige Einschränkungen auf. Da die Softmax-Funktion exponentielles Wachstum nutzt, entstehen häufig sehr hohe Konfidenzwerte, auch wenn die

zugrunde liegende Vorhersage falsch ist [GPSW17]. Dadurch neigen Softmax-basierte Schätzungen zu einer Überkonfidenz.

Ein alternativer Ansatz ist der sogenannte Trust Score [JKGG18], der von Jiang et al. vorgeschlagen wurde. Der zugrunde liegende Gedanke ist, das Verhältnis zwischen dem Abstand einer Testeingabe zur nächsten Klasse, die von der Vorhersage abweicht, und dem Abstand zur vorhergesagten Klasse zu betrachten. Ist der Abstand zur vorhergesagten Klasse deutlich größer als zu einer alternativen Klasse, deutet dies darauf hin, dass die Vorhersage des Modells möglicherweise fehlerhaft ist. Der Nachteil dieses Ansatzes liegt jedoch in seiner mangelnden Skalierbarkeit, da die Berechnung nächster Nachbarn in großen Datensätzen sehr aufwendig ist [JKGG18].

Bayesianische Ansätze [KG17, GG16, BCKW15] zur Schätzung von Unsicherheit haben in den letzten Jahren großes Interesse geweckt, da sie eine probabilistische Interpretation von Modellen ermöglichen, indem sie ganze Verteilungen über die Gewichte schätzen. Diese Methoden sind jedoch oft sehr rechenaufwendig. Gal und Ghahramani [GG16] schlugen mit Monte Carlo Dropout (MCDropout) vor, bei der Unsicherheiten durch wiederholtes stochastisches Durchlaufen des Netzwerks geschätzt werden. Dabei wird Dropout vor jeder Gewichtsschicht angewendet, und mehrere Vorhersagen des Netzwerks werden zur Approximation der posterioren Vorhersageverteilung genutzt. Durch wiederholte stochastische Vorhersagen desselben Modells kann so eine Näherung der posterioren Verteilung über die Vorhersagen gewonnen werden. Mathematisch entspricht ein neuronales Netzwerk mit beliebiger Tiefe und nichtlinearen Aktivierungen unter Dropout einer Näherung eines probabilistischen tiefen Gaussian-Prozesses [DL13].

Eng mit der Konfidenzbewertung verknüpft ist das Forschungsfeld der Confidence Calibration, die darauf abzielt, die vorhergesagten Wahrscheinlichkeiten eines Modells an die tatsächliche Eintrittswahrscheinlichkeit der Ereignisse anzupassen. Bei schlecht kalibrierten Netzen kann es jedoch vorkommen, dass falsche Vorhersagen mit sehr hoher Konfidenz getroffen werden. Zu den gängigen Verfahren zählen die Post-Processing-Kalibrierung, bei der das Modell unverändert bleibt und lediglich die Ausgaben im Nachhinein angepasst werden. Dazu zählen Methode wie Binning-Methoden [ZE01, ZE02, PNCH15], Platt Scaling [Pla00] und Temperature Scaling [GPSW17].

4 METHODIK

4.1 MODELLARCHITEKTUR

Aufbauend auf der Entwicklungen des Transformers wurde das Donut-Modell von Kim et al. [KHY+22] vorgestellt, um visuelle Informationen direkt aus Dokumentenbildern zu extrahieren und diese anschließend in strukturierter Form zu dekodieren. In dieser Arbeit dient das Donut-Modell als zentrale Grundlage für die vorgeschlagene Lösung zur datengetriebenen Informationsextraktion.

Der Ablauf von Donut (siehe Abbildung 4.1.1) gliedert sich in vier Schritte. Zunächst erhält das Modell ein Dokumentbild $x \in \mathbb{R}^{H \times W \times 3}$ als Eingabe. Ein visueller Encoder auf Basis des Swin Transformers [LLC+21] wandelt das Bild in eingebettete Repräsentationen $\{z_i \in \mathbb{R}^d \mid 1 \leqslant i \leqslant n\}$ um. Die resultierenden Repräsentationen $\{z\}$ werden anschließend von einem vortrainierten BART-Modell [LLG+19] dekodiert, das eine Token-Sequenz $(y_i)_{i=1}^{M}$ erzeugt. Diese wird in strukturierter Form im JSON-Format ausgegeben und enthält die extrahierten Informationen, etwa ein Datum.

Das Pretraining von Donut erfolgte auf 11 Millionen gescannten englischen Dokumenten aus dem IIT-CDIP-Korpus, ergänzt durch 2 Millionen synthetische Dokumente mit Wikipedia-Texten in Chinesisch, Japanisch, Koreanisch und Englisch. Das vortrainierte Donut-Modell dient als Grundlage für die in dieser Arbeit vorgestellte Lösung zur datengetriebenen Informationsextraktion.

Im Folgenden wird die übergeordnete Architektur von Swin und BART im Detail beschrieben.

4.1.1 *SWIN Encoder*

Eine Übersicht der Swin Transformer-Architektur ist in Abbildung 4.1.2 dargestellt. Ähnlich wie bei Vision Transformers (vgl. Abschnitt 2.3.4) werden 2D-Bilder zunächst in eine Sequenz von Patch-Embeddings überführt. In der ursprünglichen Implementierung [LLC+21] wird eine Patch-Größe von 4×4 verwendet, woraus sich für jedes Patch eine Feature-Dimension von $4 \times 4 \times 3 = 48$ ergibt.

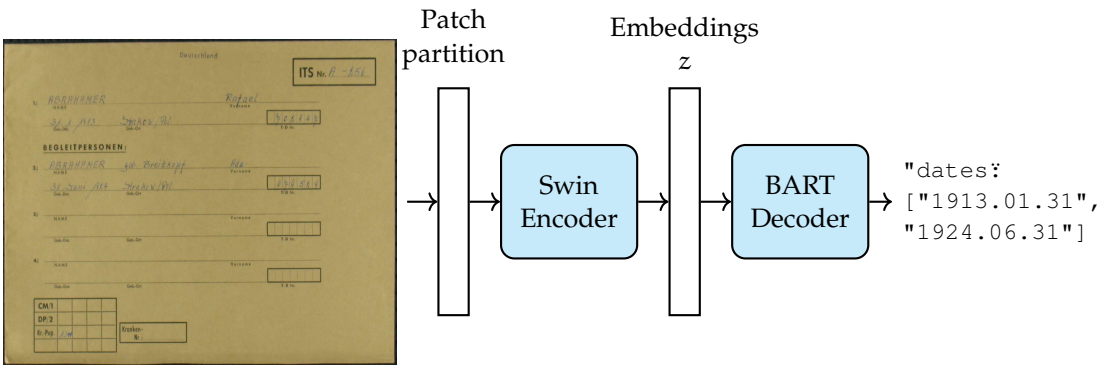


Abbildung 4.1.1: Pipeline des Donut-Modells: Das Eingabedokument wird in Patches zerlegt und durch den Encoder in Embeddings überführt. Der Decoder generiert daraus eine Token-Sequenz, die in ein strukturiertes Ausgabeformat transformiert wird.

Daraufhin folgen mehrere Swin-Transformer-Blöcke, die auf die Patch-Tokens angewendet werden. Die erste Stufe aus linearer Einbettung und Swin-Transformer-Blöcken wird als Stage 1 bezeichnet. Die Anzahl der Tokens bleibt dabei konstant bei $\frac{H}{4} \times \frac{W}{4}$.

Der Swin Transformer ersetzt die Standard Multi-Head Self-Attention (MSA) (vgl. Abschnitt 2.3.2) durch *Window-based Multi-head Self-Attention* (W-MSA) und *Shifted Window-based Multi-head Self-Attention* (SW-MSA), während andere Schichten unverändert bleiben. Wie in Abbildung 4.1.2 (b) dargestellt, besteht jeder Swin-Transformer-Block aus zwei aufeinanderfolgenden Unterschichten. Die erste verwendet W-MSA, die Self-Attention innerhalb lokaler Fenster berechnet. Die zweite setzt SW-MSA ein, bei der die Fenster verschoben werden, um den Informationsaustausch zwischen benachbarten Fenstern zu ermöglichen. In SW-MSA erfolgt die Verschiebung der Fenster im Vergleich zur vorherigen Schicht um jeweils $\lfloor \frac{M}{2} \rfloor$ Pixel in horizontaler und vertikaler Richtung, wobei M die Fenstergröße bezeichnet (siehe Abbildung 4.1.3).

Die Berechnung innerhalb zweier aufeinanderfolgender Swin-Transformer-Blöcke lässt sich nach [LLC+21] wie folgt darstellen

$$\begin{aligned} \hat{z}^{\ell} &= \text{W-MSA}(\text{LayerNorm}(z^{\ell-1})) + z^{\ell-1}, \\ z^{\ell} &= \text{MLP}(\text{LayerNorm}(\hat{z}^{\ell})) + \hat{z}^{\ell}, \\ \hat{z}^{\ell+1} &= \text{SW-MSA}(\text{LayerNorm}(z^{\ell})) + z^{\ell}, \\ z^{\ell+1} &= \text{MLP}(\text{LayerNorm}(\hat{z}^{\ell+1})) + \hat{z}^{\ell+1} \end{aligned} \tag{4.1.1}$$

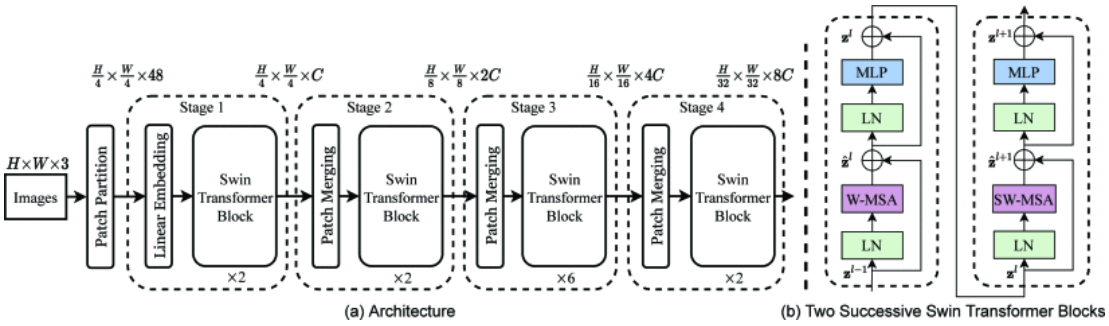


Abbildung 4.1.2: Übersicht der Swin Transformer Architektur [LLC+21]. (a) Das Eingabebild (H×W×3) wird in Patches unterteilt und eingebettet. Darauf folgen vier hierarchische Stufen mit abwechselnden Patch-Merging-Layern und Swin Transformer Blocks. (b) Jeder Swin Transformer Block besteht aus Window-basiertem Multi-Head Self-Attention (W-MSA bzw. SW-MSA), Layer-Normalisierung, Residualverbindungen und einem MLP.

wobei Multilayer-Perzeptron (MLP) definiert ist als:

$$\mathrm{MLP}(x) = \mathrm{GELU}\left(xW_1 + b_1\right)W_2 + b_2$$

Bei der Berechnung der Self-Attention wird zusätzlich Positionsinformation berücksichtigt. Swin verzichtet auf globale Position-Embeddings und verwendet stattdessen relative Position Biases in jeder Attention-Berechnung. Die Attention wird formal durch die Gleichung nach [LLC+21]

$$\mathrm{Attention}(Q, K, V) = \mathrm{Softmax}\left(\frac{QK^\top}{\sqrt{d}} + B\right)V \tag{4.1.2}$$

beschrieben. Hierbei ist $B \in \mathbb{R}^{M^2 \times M^2}$ der relative Position Bias, der die Abhängigkeit zwischen den Patches berücksichtigt. Durch die Hinzufügung von B kann das Modell nicht nur den Inhalt der Patches, sondern auch deren relative Position zueinander in die Attention-Berechnung einbeziehen.

Der Swin Transformer erzeugt eine hierarchische Merkmalsdarstellung indem er zunächst mit kleineren Patches arbeitet und diese in tieferen Schichten des Modells sukzessive zu größeren Regionen zusammenführt. In einem weiteren Schritt wird eine *Patch-Merging*-Schicht eingesetzt. Die Patch-Merging-Schicht reduziert die räumliche Auflösung, indem sie jeweils 2×2 benachbarte Patches zusammenfasst. Diese vier Vektoren werden zunächst konkateniert und anschließend linear in einen $2C$-dimensionalen Raum projiziert. Auf diese Weise werden gleichzeitig die Anzahl der

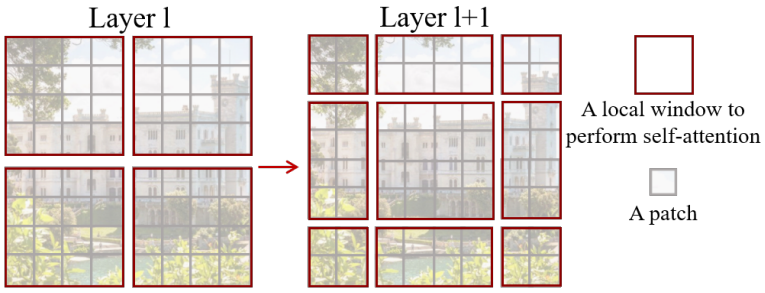


Abbildung 4.1.3: Darstellung des Shifted Window Mechanismus über zwei aufeinanderfolgende Schichten nach [LLC+21]. In Schicht l wird Self-Attention lokal innerhalb nicht überlappender Fenster berechnet. In Schicht $l+1$ werden die Fenster verschoben, sodass Patches aus verschiedenen ursprünglichen Fenstern gemeinsam berücksichtigt werden können.

Tokens verringert und die Kanal-Dimension erhöht. Anschließend folgen weitere Swin-Blöcke zur Merkmalsextraktion bei reduzierter Auflösung $\frac{H}{8} \times \frac{W}{8}$. Diese Kombination bildet die Stage 2. Zwei weitere Blöcke (Stage 3 und Stage 4) folgen dem gleichen Prinzip und reduzieren die Auflösung weiter auf $\frac{H}{16} \times \frac{W}{16}$ bzw. $\frac{H}{32} \times \frac{W}{32}$.

Diese hierarchische Architektur erlaubt es dem Modell, Token unterschiedlicher Größe über die Layer hinweg zu verarbeiten. Dadurch werden visuelle Merkmale auf mehreren Skalenebenen zuverlässig extrahiert.

Neben der Standardarchitektur Swin-B (Base) wurden mehrere Varianten des Swin Transformers von Ze Liu Et Al. [LLC+21] vorgeschlagen, die sich hinsichtlich Modellgröße und Rechenaufwand unterscheiden. Diese werden als Swin-T (Tiny), Swin-S (Small), und Swin-L (Large) bezeichnet. Alle Varianten folgen demselben Grundprinzip, variieren jedoch in den Architektur-Hyperparametern, insbesondere in der Kanalanzahl C der ersten Stufe sowie in der Anzahl der geschalteten Swin-Transformer-Blöcken pro Stage. Swin-T stellt die kleinste Variante dar und nutzt $C = 96$ mit einer Schichtenkonfiguration von $(2, 2, 6, 2)$, während Swin-S mit gleicher Kanalanzahl eine größere Tiefe von $(2, 2, 18, 2)$ aufweist. Swin-B erhöht die Dimension auf $C = 128$ bei ebenfalls $(2, 2, 18, 2)$, und Swin-L bildet die größte Ausführung mit $C = 192$ und derselben Tiefe. In allen Varianten wird eine Fenstergröße von $M = 7$ verwendet.

Im Donut-Modell kommt die Swin-B-Variante als visueller Encoder mit leichten Modifikationen zum Einsatz. Die Anzahl der Blöcke pro Stage sowie die Fenstergröße wurden dabei auf $(2, 2, 14, 2)$ bzw. 10 gesetzt [KHY+22].

4.1.2 *BART Decoder*

Das BART-Modell wurde ursprünglich von Lewis et al. [LLG+19] als Denoising Autoencoder [VLBM08] entwickelt. Ein Denoising Autoencoder ist ein neuronales Modell, das darauf trainiert wird, aus absichtlich verrauschten Eingaben die ursprüngliche, unverrauschte Sequenz zu rekonstruieren. BART nutzt verschiedene Verfahren (siehe Abbildung 4.1.4 (b)) zur gezielten Verrauschung von Texten. Beim Token Masking werden zufällige Tokens durch [_] ersetzt, bei der Token Deletion vollständig entfernt. Das Text Infilling ersetzt ganze Textspannen variabler Länge (gezogen aus einer Poisson-Verteilung mit $\lambda = 3$) durch ein einzelnes [_]. Zudem werden in der Sentence Permutation die Satzreihenfolgen vertauscht, während die Document Rotation den Text zyklisch verschiebt, indem ein zufälliges Token als neuer Beginn gewählt wird. Dieses Verfahren stärkt die Robustheit des Modells gegenüber fehlerhaften oder unvollständigen Eingaben und verbessert zugleich seine Fähigkeit zur semantischen Generalisierung.

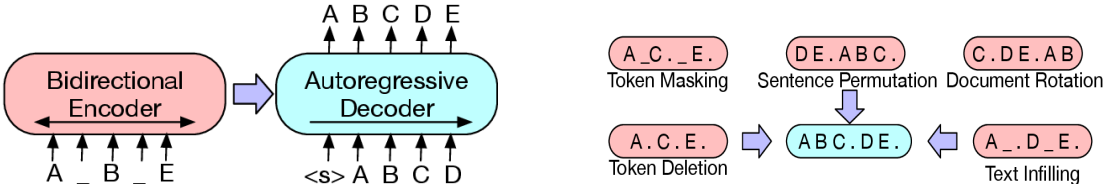


Abbildung 4.1.4: (a) Übersicht der BART-Architektur [LLG+19]. Der bidirektionale Encoder erfasst vollständige Kontextinformationen der Eingabe, während der autoregressive Decoder die Zielsequenz Schritt für Schritt generiert. (b) Verrauschungsverfahren, die beim Pretraining von BART eingesetzt werden.

Die ursprüngliche Architektur von BART besteht aus zwei Hauptkomponenten: einen bidirektionalen Encoder, der dem von BERT (Bidirectional Encoder Representations from Transformers) [DCLT19] ähnelt, und einem autoregressiven Decoder, der dem GPT (Generative pre-trained Transformers) [RNS+18] entspricht (siehe Abbildung 4.1.4 (a)). Im Donut-Modell wird der Encoder-Teil von BART nicht verwendet. Stattdessen übernimmt der Swin Transformer die Rolle der Eingabekodierung [VSP+23]. Folglich konzentriert sich die nachfolgende Erklärung ausschließlich auf den BART-Decoder.

Der Decoder von BART orientiert sich konzeptionell am GPT-Modell [RNS+18]. In GPT besteht jede Decoderschicht aus einer Masked-Multi-Head Self-Attention, gefolgt von positionsweisen Feed-Forward-Netzwerken. GPT ermöglicht, jedes Token der

Ausgabesequenz Schritt für Schritt generieren werden zu können, wobei ausschließlich die zuvor generierten Tokens als Kontext verwendet werden. Im Unterschied zum GPT-Decoder enthält der BART-Decoder in jeder Schicht zusätzlich eine *Cross-Attention* (Encoder-Decoder-Attention), die die Verbindung zu den Encoder-Ausgaben herstellt. Damit entspricht die Architektur des BART-Decoders im Wesentlichen dem Standard-Transformer-Decoder, wie in Abschnitt 2.3.3 beschrieben.

Im ursprünglichen BART-Modell bestehen die Decoder-Architekturen je nach Modellgröße aus 6 oder 12 Schichten. Donut verwendet jedoch lediglich die ersten 4 Schichten des vortrainierten, multilingualen BART-Decoders [LGG+20], um ein günstiges Verhältnis von Genauigkeit und Rechenaufwand sicherzustellen. Die Gewichte werden dabei aus einem öffentlich verfügbaren multilingualen BART-Modell[1] [LGG+20] initialisiert.

4.2 KONFIDENZMASSE

Die zuverlässige Einschätzung der *Sicherheit* einer Modellvorhersage sind entscheidend für den praktischen Einsatz von Modellen. Hierbei bezeichnet *Sicherheit* den Grad, mit dem das Modell davon ausgeht, dass seine Vorhersage korrekt ist, während *Konfidenz* einen quantitativen Wert darstellt, der diese Sicherheit ausdrückt [CTBH+19]. Je höher dieser Wert ist, desto sicherer ist sich das Modell in Bezug auf das ausgegebene Ergebnis. Ein geeignetes Konfidenzkriterium sollte dabei so gestaltet sein, dass falsche Vorhersagen mit niedrigen Werten einhergehen und korrekte Vorhersagen mit hohen Werten [CTBH+19]. In der vorliegenden Arbeit wird die Konfidenz genutzt, um die Zuverlässigkeit der Datumserkennung zu bewerten und damit die Qualität der Ergebnisse zu messen.

Zur Quantifizierung der Modellsicherheit im Rahmen der Sequenzvorhersage ist die Berechnung geeigneter Konfidenzmaße von zentraler Bedeutung. Grundlage bildet dabei die durch eine Softmax-Normalisierung über das Vokabular erhaltene Token-Wahrscheinlichkeit bzgl. Pseudowahrscheinlichkeit am Dekodierschritt t

$$\sigma(z_t)^{(v)} = \frac{\exp\left(z_t^{(v)}\right)}{\sum_{u \in \mathcal{V}} \exp\left(z_t^{(u)}\right)}, \qquad v \in \mathcal{V}. \tag{4.2.1}$$

wobei $z_t^{(v)}$ die Logits für das Token v darstellen. Das jeweils vorhergesagte Token y_t ist das mit der höchsten Wahrscheinlichkeit $y_t = \arg\max_{v \in \mathcal{V}} \sigma(z_t)^{(v)}$.

Die Log-Pseudowahrscheinlichkeit des gewählten Tokens y_t wird als $c_t = \ln\left(\sigma(z_t)^{(y_t)}\right)$ definiert. Diese Werten dient als Grundlage für die Berechnung der

1 https://huggingface.co/hyunwoongko/asian-bart-ecjk

Sequenzkonfidenz. In der Arbeit wird Log-Softmax anstelle von Softmax verwendet, da das Arbeiten im Log-Raum numerisch stabiler ist, insbesondere bei sehr kleinen Wahrscheinlichkeiten, die sonst zu Rundungsfehlern führen könnten [GBC16, S. 79]. Im Folgenden werden drei verschiedene Konfidenzmaße eingeführt, die unterschiedliche Aspekte der Modellunsicherheit erfassen.

Das erste Maß, das innerhalb dieser Arbeit eingeführt wird, ist die geometrische Mittelkonfidenz (GAVG), definiert als

$$\text{GAVG} \;=\; \exp\left(\frac{1}{T}\sum_{t=1}^{T} c_t\right), \tag{4.2.2}$$

wobei T die Sequenzlänge ist. Dieses Maß entspricht dem geometrischen Mittel der der maximalen Pseudowahrscheinlichkeiten je Token, äquivalent zur exponentiellen mittleren Log-Wahrscheinlichkeit der gewählten Tokens. Intuitiv erfasst GAVG die mittlere Sicherheit des Modells über die gesamte Sequenz hinweg. Ein hoher Wert zeigt an, dass das Modell über die gesamte Sequenz hinweg konsistent sichere Vorhersagen getroffen hat, während ein niedriger Wert auf Unsicherheit hinweist.

Das zweite Maß MIN fokussiert dagegen ausschließlich auf das unsicherste Token innerhalb der Sequenz:

$$\text{MIN} = \min_t \exp\left(c_t\right). \tag{4.2.3}$$

Hierbei bestimmt das schwächste Token die gesamte Konfidenz. Ein hoher MIN bedeutet, dass selbst das unsicherste Token mit hoher Wahrscheinlichkeit korrekt vorhergesagt wurde, sodass keine signifikante Unsicherheit in der Sequenz vorhanden ist. Umgekehrt weist ein kleiner MIN darauf hin, dass mindestens ein Token sehr unsicher war, wodurch die gesamte Sequenz potenziell fehlerhaft ist.

Eine weitere Möglichkeit, die Vorhersagesicherheit zu messen, ist die *Top-2-Margin*, die den Abstand der beiden wahrscheinlichsten Kandidaten pro Token einer Sequenz betrachtet. Für jedes Token wird die Differenz der Wahrscheinlichkeiten der Top-2-Kandidaten als *Margin* definiert. Die Sequenzkonfidenz wird dann als Mittelwert über alle Tokens aggregiert:

$$\text{T2M} = \frac{1}{T}\sum_{t=1}^{T} \text{Margin}_t, \quad \text{Margin}_t = \exp\left(c_t^{(1)}\right) - \exp\left(c_t^{(2)}\right) \tag{4.2.4}$$

wobei $c_t^{(1)}$ und $c_t^{(2)}$ die höchsten Log-Pseudowahrscheinlichkeit des Tokens y_t sind. Die Konfidenz ergibt sich somit aus dem Abstand zwischen der Wahrscheinlichkeit

des vorhergesagten Tokens und der des zweitbesten Tokens. Ein hoher Top-2-Margin signalisiert, dass das Modell eine klare Entscheidung für das gewählte Token getroffen hat, während ein niedriger Wert auf eine Unsicherheit zwischen mehreren Kandidaten hindeutet.

Bei der Berechnung der Konfidenzmaße werden bestimmte Spezialtokens wie das Öffnen- und Schließen-Symbol `[`, `]` sowie das `<eos>` (End-of-Sequence) Token explizit ausgeschlossen. Der Grund dafür ist, dass diese Tokens keine semantische Bedeutung im extrahierten Inhalt tragen, sondern lediglich der Strukturierung dienen. Ihre Vorhersage ist in der Regel trivial und führt daher zu künstlich hohen Wahrscheinlichkeiten, die die Konfidenzschätzung verzerren würden. In Abschnitt 5.4.2 werden alle drei Konfidenzmaße genutzt und verglichen, um ein umfassendes Bild der Vorhersagesicherheit im DONUT-Modell zu erhalten.

Dabei ist zu beachten, dass sich die hier betrachteten Konfidenzwerte immer auf die gesamte vorhergesagte Sequenz eines Bildes beziehen. Da ein Bild mehrere Datumseinträge enthalten kann, geben die Maße nicht die Sicherheit für ein einzelnes Datum wieder, sondern die aggregierte Sicherheit aller im Bild enthaltenen Datumssequenzen.

5 EVALUATION

5.1 DATENSÄTZE

Für die Experimente werden insgesamt zwei Datensätze verwendet. Der erste Datensatz besteht aus ausgeschnittenen Bildausschnitten, die ausschließlich Datumsangaben enthalten. Dadurch kann das Modell die visuelle Struktur und typische Textmuster von Datumsformaten erlernen. Der zweite Datensatz umfasst vollständige Dokumentseiten, auf denen das zuvor gelernte Wissen über Datumsangaben im Kontext ganzer Seiten angewendet wird, um diese korrekt zu erkennen und zu lokalisieren. Im Folgenden werden beide Datensätze näher vorgestellt.

5.1.1 *Death Certificates 2*

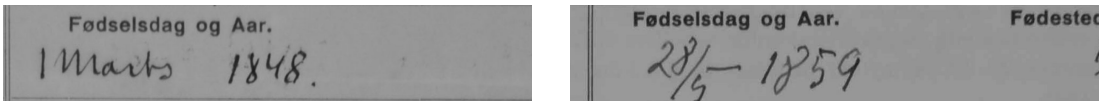
Der Datensatz Death Certificates 2 (DC-2) aus der DARE[1] Datenbank [DJS+22] umfasst handschriftliche Datumsangaben, die aus dänischen Todesurkunden stammen. Die DARE-Datenbank umfasst insgesamt sechs verschiedene Datensätze, die sich in Herkunft und Format der Datumsangaben unterscheiden (z. B. TT-MM-JJJJ, TT-MM-JJ oder TT-MM). Neben dänischen Todesurkunden finden sich darin auch Datumsangaben aus dänischen Polizeiberichten, Beerdigungsunterlagen und Dokumenten von Krankenpflegeheimen sowie zwei Datensätze aus dem schwedischen Register zu Todesursachen. Für diese Arbeit wurde der Datensatz DC-2 ausgewählt, da die Datumsangaben dort in der Regel einzeilig geschrieben sind und überwiegend dem Format TT-MM-JJJJ entsprechen (siehe Abbildung 5.1.1), was mit dem im zweiten Trainingsdatensatz verwendeten Format übereinstimmt. In einigen Fällen fehlt jedoch die Jahresangabe oder das Format weicht zu JJJJ-MM-TT ab oder wird der Monatsangabe nicht als Zahl, sondern als ausgeschriebenes Wort dargestellt.

Die ursprünglichen Quelldokumente weisen teils starke Unterschiede in ihrer Größe auf, mit Breiten von bis zu 1400 Pixeln und Höhen bis 356 Pixel. Für die weitere Verarbeitung wurden alle Bilder einheitlich auf eine Größe von 1400×250 Pixeln skaliert.

1 https://www.kaggle.com/datasets/sdusimonwittrock/dare-database

Größe der Sets des DC-2 Datensatzes				
	Training	Validierung	Test	Gesamt
Bilder	150.439	5.000	8.338	163.777

Tabelle 5.1.1: Größe der Sets des DC-1 Datensatzes



(a) `"date": "01-03-1848"` (b) `"date": "28-05-1859"`

Abbildung 5.1.1: Beispiele verschiedener Datumsformate aus dem DC-2 Datensatz mit zugehöriger JSON-Annotation

Jedes dieser Bilder zeigt genau eine handschriftliche Datumsangabe. Die zugehörige Ground Truth folgt dabei dem Format `"date": "TT-MM-JJJJ"` und stellt das jeweils enthaltene Datum in standardisierter Form dar. Der Datensatz ist in Trainings-, Validierungs- und Testmengen unterteilt, die 150.439, 5.000 bzw. 8.338 Bilder enthalten. Eine Übersicht zur Aufteilung ist in Tabelle 5.1.1 dargestellt.

5.1.2 *CM1-COVER*

Die CM1 (Care and Maintenance) umfasst Formulare, die zwischen 1947 und 1952 von der International Refugee Organization (IRO) und ihren Vorgängerorganisationen zusammengetragen wurden. Sie dokumentieren die Betreuung von Displaced Persons (DPs) in den westlichen Besatzungszonen Deutschlands sowie in Italien, Österreich, der Schweiz und England. Zentrales Dokumententyp ist der Fragebögen aus dem Care and Maintenance Programm der IRO, den DPs zur Beantragung humanitärer Hilfe ausfüllen mussten [Arc].

Die Scans der CM1-Dokumente wurden von den Arolsen Archives [2] bereitgestellt. Jeder Eintrag im Datensatz umfasst mehrere Dokumente eines individuellen Verwaltungsverfahrens, typischerweise bestehend aus einem Deckblatt (Cover), einem ausgefüllten Antragsformular sowie weiteren begleitenden persönlichen Unterlagen [WTM+25]. Die in dieser Arbeit untersuchten Aufgaben beziehen sich ausschließlich auf die Deckblätter, die strukturierte, handschriftlich eingetragene Informationen zur antragstellenden Person sowie zu bis zu drei begleitenden Angehörigen enthalten.

Der vollständige Datensatz umfasst insgesamt 123.550 digitalisierte Dokumentbilder mit 184.647 annotierten Datumsangaben. Zur Modellentwicklung wurde der Datensatz in drei

2 https://collections.arolsen-archives.org/en/archive/3-2-1

Größe der Sets des CM1-COVER Datensatzes				
	Training	Validierung	Test	Gesamt
Bilder	113.237	3.427	6.886	123.550

Tabelle 5.1.2: Größe der Sets des CM1-COVER Datensatzes

Teile aufgeteilt: 113.237 Bilder für das Training, 3.427 für die Validierung und 6.886 für den abschließenden Test. Eine tabellarische Übersicht dieser Aufteilung ist in Tabelle 5.1.2 dargestellt.

Ein Dokumentbild kann bis zu vier Datumsangaben enthalten, die üblicherweise dem Format `TT-MM-JJJJ` folgen. Die zugehörige Ground-Truth-Annotation wird entsprechend als JSON angegeben, beispielsweise `"dates": ["1913-01-31", "1924-06-31"]`. Ein anschauliches Beispiel ist in Abbildung 5.1.2 dargestellt.

5.2 EVALUATIONSMETRIKEN

In diesem Kapitel werden die Metriken beschrieben, die zur Bewertung der Modellleistung verwendet werden. In dieser Arbeit werden die Mean Character Error Rate (mCER) und die Sequence Accuracy (SeqAcc) verwendet, da sie sowohl die Genauigkeit auf Zeichen- als auch auf Sequenzebene abbilden.

5.2.1 *Mean Character Error Rate*

Die gebräuchlichste Metrik zur Bewertung von Handschriftenerkennungssystemen (HTR) ist die Character Error Rate (CER). CER misst den Anteil der fehlerhaft erkannten Zeichen im Vergleich zur Annotation.

Zur Berechnung wird die Anzahl der fehlerhaften Bearbeitungsoperationen bzw. Einfügungen, Löschungen und Ersetzungen gezählt. Diese Zahl wird dann durch die Gesamtanzahl der Zeichen in der Annotation geteilt:

$$\text{CER} = \frac{\#\text{Einfügungen} + \#\text{Löschungen} + \#\text{Ersetzungen}}{\text{Länge der Annotation}} \tag{5.2.1}$$

Für ein Testset wird die mittlere CER (mCER) über alle N Stichproben gebildet:

$$\text{mCER} = \frac{1}{N} \sum_{i=1}^{N} \text{CER}_i \tag{5.2.2}$$

Ein niedriger mCER-Wert zeigt eine gute Erkennungsleistung, während ein hoher Wert viele Fehler deutet. Es ist zu beachten, dass in Fällen, in denen die vorhergesagte Datumssequenz leer ist, der Zeichenfehlerratewert (CER) in dieser Arbeit standardmäßig auf 1.0 gesetzt wird.

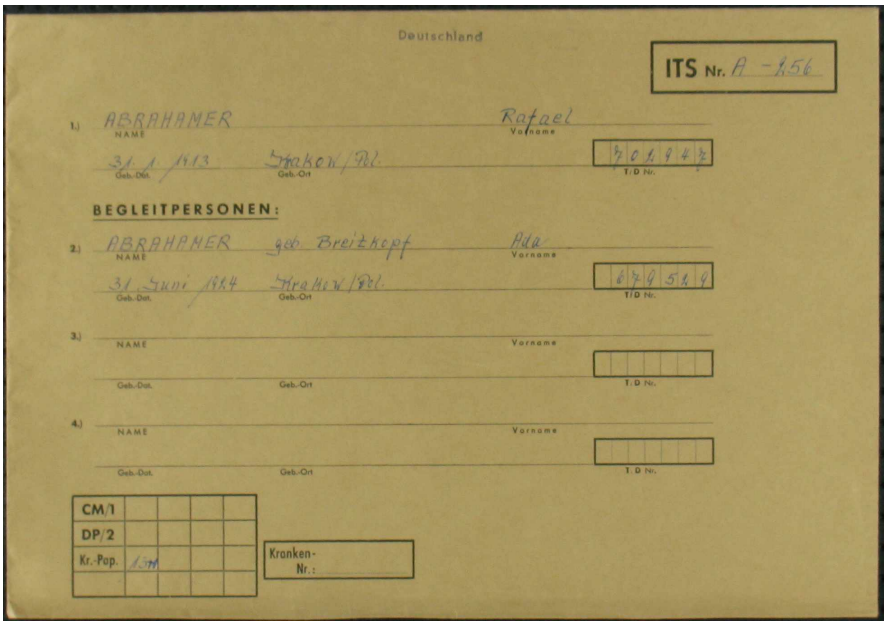


Abbildung 5.1.2: CM1-COVER Beispielbild mit zugehöriger JSON-Annotation: `"dates: ["1913-01-31", "1924-06-31"]`

Da ein Dokumentbild mehrere Datumsangaben enthalten kann, wird zunächst für jedes Bild die mCER berechnet. Anschließend werden die mCER über den gesamten Datensatz aggregiert.

5.2.2 *Sequence Accuracy*

Eine weitere Metrik zur Bewertung von Transkriptionsqualität ist die Sequence Accuracy (SeqAcc). Dabei gilt eine Vorhersage nur dann als korrekt, wenn alle Zeichen bzw. Token in einer Sequenz vollständig richtig erkannt wurden. Ist auch nur eines der Token falsch, wird die gesamte Vorhersage als fehlerhaft gewertet. Dies macht die SeqAcc besonders streng und unterscheidet sie deutlich von Metriken wie der CER, da sie die Transkription als Ganzes bewertet. Typischerweise ist die SeqAcc daher deutlich niedriger als zeichenbasierte Metriken. Die Berechnung erfolgt gemäß:

$$\text{SeqAcc} = \frac{\text{Anzahl korrekt erkannter Sequenzen}}{\text{Gesamtanzahl der Sequenzen}} \tag{5.2.3}$$

Da ein Bild mehrere Datumsangaben enthalten kann, wird die SeqAcc zunächst pro Datumssequenz ermittelt. Die SeqAcc pro Bild ergibt sich als Durchschnitt dieser Sequenzen, und über den gesamten Datensatz wird wiederum der Mittelwert aller Bild-SeqAcc berechnet.

5.2.3 *Klassifikationsleistung*

Um die Wirkung der Konfidenzschranken auf die Freigabe historischer Dokumente zu bewerten (siehe Abschnitt 5.4.4), wird die Klassifikationsleistung des Modells in Bezug auf die 100-Jahres-Grenze untersucht. Dabei ist entscheidend, ob ein Dokument korrekt als „veröffentlichbar“ oder „nicht veröffentlichbar“ eingestuft wird.

Zur quantitativen Analyse werden zwei zentrale Maße herangezogen: *Precision* und *Relative Error Reduction* (RER). Die Grundlage bildet die in Tabelle 5.2.1 dargestellte Konfusionsmatrix. Sie unterscheidet vier Fälle: Wird eine Instanz, die tatsächlich positiv ist, korrekt als positiv erkannt, spricht man von einem True Positive (TP). Erfolgt dagegen eine positive Vorhersage für eine eigentlich negative Instanz, handelt es sich um ein False Positive (FP). Analog bezeichnet man korrekt negative Vorhersagen als True Negative (TN), während falsch negative Vorhersagen als False Negative (FN) gelten.

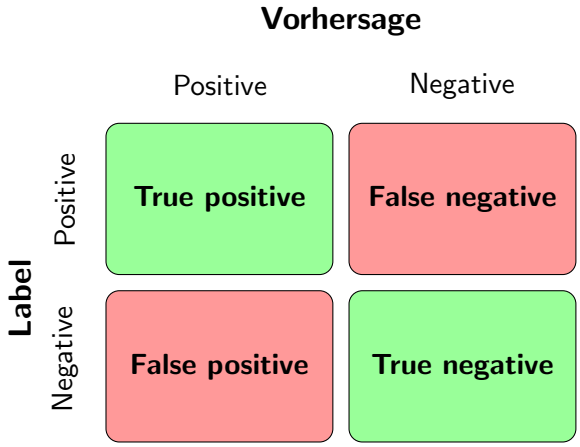


Abbildung 5.2.1: Konfusionsmatrix eines binären Klassifikators.

Auf Basis dieser Größen lässt sich die Precision nach [Pow20] definieren. Sie beschreibt den Anteil korrekt positiver Klassifikationen an allen positiven Vorhersagen und ergibt sich formal zu:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.2.4}$$

Eine hohe Precision bedeutet, dass nur ein geringer Anteil der als positiv klassifizierten Instanzen tatsächlich fehlerhaft ist.

Ein weiteres Maß zur Bewertung der Modellleistung ist die Relative Error Reduction (RER), wie sie in [VWN23] zur vergleichenden Analyse verschiedener Modelle vorgeschlagen wurde. RER misst die relative Verringerung der Fehlklassifikationen gegenüber einer Referenzmethode

(Baseline). Angenommen, die Baseline erreicht eine Genauigkeit von 80% und ein verbessertes Modell erzielt 85%, so ergibt sich eine RER von $\frac{5}{20} = 25\%$. Formal lässt sich RER wie folgt ausdrücken:

$$\text{RER} = \frac{\text{Err}_{\text{baseline}} - \text{Err}_{\text{Methode}}}{\text{Err}_{\text{baseline}}} \times 100\%, \tag{5.2.5}$$

wobei $\text{Err}_{\text{baseline}}$ die Anzahl der Fehler in der Baseline beschreibt und $\text{Err}_{\text{Methode}}$ die Fehlerzahl des betrachteten Verfahrens.

In dieser Arbeit wird die RER auf die False Positives (FP) angewendet. Dazu wird $\text{FP}_{\text{baseline}}$ als die Anzahl an False Positives in der Baseline (ohne Konfidenzfilter) definiert. FP_τ bezeichnet hingegen die Zahl der False Positives bei Anwendung eines Konfidenzschwellwertes τ. Die RER_τ ergibt sich damit formal zu:

$$\text{RER}_\tau(\text{FP}) = \frac{\text{FP}_{\text{baseline}} - \text{FP}_\tau}{\text{FP}_{\text{baseline}}} \times 100\%. \tag{5.2.6}$$

Dieses Maß verdeutlicht, in welchem Ausmaß die Einführung einer Konfidenzschranke zur Reduktion fehlerhafter positiver Vorhersagen beiträgt. Damit lässt sich quantitativ belegen, dass die Konfidenzbewertung ein wirksames Mittel zur Verbesserung der Filterung über Modellvorhersagen darstellt.

5.3 TRAININGSAUFBAU

Für die Experimente dieser Arbeit wurde das vortrainierte Donut-Modell [3] auf dem Consolidated Receipt Dataset (CORD) [PSL+19] als Basis-Modell für das Fine-Tuning verwendet. Dieses Modell wurde speziell für die Aufgaben der Dokumenten-Informationsextraktion auf dem CORD trainiert. Der Datensatz umfasst 800 Trainings-, 100 Validierungs- und 100 Testbeispielen. Die enthaltenen Texte sind in lateinischer Schrift verfasst. Insgesamt werden 30 unterschiedliche Informationsfelder extrahiert, darunter beispielsweise Menüname, Menge, Gesamtpreis und weitere relevante Felder.

Als technische Grundlage diente Python 3.9 zusammen mit dem PyTorch-Framework für das Deep Learning. Alle Trainingsvorgänge wurden auf einer NVIDIA GeForce RTX 3070 mit 8GB Grafikspeicher ausgeführt, was hinsichtlich Speicherkapazität und Rechenleistung gewisse Einschränkungen mit sich brachte.

Das Training erfolgte unter Verwendung des Adam-Optimizers [KB17] mit einer Lernrate von 3×10^{-5}. Aufgrund der begrenzten Hardware-Ressourcen wurde das Training auf die Batch-Größe von 1 beschränkt und mit einer maximal möglichen Eingabegröße von 640×320 Pixeln durchgeführt. Um den Lernprozess zu stabilisieren, wurde eine Warmup-Phase von 10.000 Iterationen eingeführt. Die maximale Token-Länge wurde auf 50 Zeichen begrenzt, um die Effizienz zu gewährleisten. Diese Hyperparameter blieben über alle im Rahmen dieser Arbeit untersuchten Experimente hinweg unverändert.

3 https://huggingface.co/naver-clova-ix/donut-base-finetuned-cord-v2/tree/official

Zur Vermeidung von Overfitting wurde ein Early Stopping [Pre12] eingesetzt. Dabei wird die Modellleistung auf dem Validierungsdatensatz während des Trainings überwacht. Sobald sich die Leistung nicht weiter verbessert oder sich verschlechtert, wird der Trainingsprozess automatisch gestoppt. Dieses Verfahren verhindert, dass das Modell übermäßig stark auf Rauschen im Trainingsdatensatz reagiert und verbessert somit die Generalisierungsfähigkeit.

5.4 ERGEBNISSE

In diesem Kapitel werden die Ergebnisse der Experimente zur Leistung der Donut-Modelle nach dem Finetuning mit dem DC2- und CM1-COVER-Datensatz vorgestellt. Der Fokus liegt dabei auf zwei Aspekten: zum einen auf der Abhängigkeit der Modellleistung von der Größe der verfügbaren Trainingsdaten, zum anderen auf der Bewertung der Modellsicherheit anhand verschiedener Konfidenzmaße. Um darüber hinaus den praktischen Nutzen der Konfidenzprüfung zu verdeutlichen, wurde ein Simulationsszenario entworfen, das reale Veröffentlichungssituationen nachbildet.

5.4.1 *Reduktion der Trainingsdaten*

Die Bewertung der Modellleistung in Abhängigkeit von der Verfügbarkeit von Trainingsdaten ist ein wichtiger Aspekt. Eine gängige Methode besteht darin, die Menge an Trainingsdaten zu reduzieren, indem nur ein zufällig ausgewählter Teil des ursprünglichen Datensatzes verwendet wird. Dieser Experiment beantwortet 2 Fragestellungen: Erstens, wie stark die Leistung des Modells vom Umfang der verfügbaren Trainingsdaten abhängt, und zweitens, welchen Einfluss ein vorheriges Finetuning auf dem DC2-Datensatz im Vergleich zum Basismodell hat.

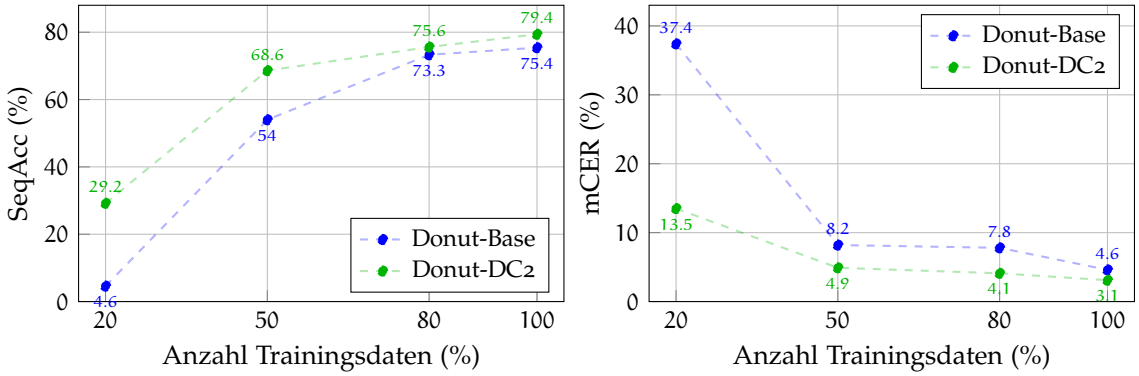
Konkret wurde der Trainingsdatensatz des CM1-COVER-Datensatzes in hierarchisch aufgebaute Teilmengen unterteilt. Dies bedeutet, dass jede kleinere Teilmenge vollständig in der nächstgrößeren enthalten ist. Insgesamt wurden vier Trainingsstufen definiert, die 20%, 50%, 80%, 100% des ursprünglichen Trainingsdatensatzes umfassen. Die Validierungs- und Testmengen blieben über alle Experimente hinweg unverändert (Siehe Tabelle 5.4.1).

Mit diesen Datensplits wurden anschließend zwei Modellvarianten untersucht. Zum einen das Donut-Basismodell, das ohne zusätzliches Finetuning verwendet wurde (Donut-Base). Zum anderen das Donut-Modell, das zuvor bereits auf dem DC2-Datensatz vortrainiert wurde (Donut-DC2). Beide Modelle wurden separat mit den vier unterschiedlichen Trainingsmengen trainiert und anschließend auf denselben Testdaten evaluiert.

Die Auswertung der in Abbildung 5.4.1 dargestellten Ergebnisse zeigt, dass Donut-DC2 in allen Konfigurationen deutliche Leistungsgewinne gegenüber dem Basismodell Donut-Base erzielt. Bereits bei geringen Trainingsanteilen zeigt sich ein deutlicher Vorteil von Donut-DC2. Während Donut-Base bei nur 20% Trainingsdaten noch eine sehr hohe mCER von 37.4 aufweist, reduziert Donut-DC2 diesen Wert auf 13.5. Damit werden fast zwei Drittel der Fehler eliminiert. Mit zunehmendem Anteil an Trainingsdaten verbessert sich die Leistung beider Modelle deutlich. Donut-Base zeigt dabei einen sprunghaften Leistungsanstieg zwischen 20% und 50% Trainingsdaten und erreicht bei 100% Daten eine Sequenzgenauigkeit von 75.4% bei

Datensatz	Train				Val	Test
	20%	50%	80%	100%		
CM1-COVER	22,647	56,618	90,589	113,237	3,427	6,886

Tabelle 5.4.1: Die Aufteilung des Datensatzes erfolgt in Trainings-, Validierungs- und Testmengen. Die Trainingsmengen sind hierarchisch aufgebaut, wobei jede kleinere Teilmenge vollständig in der jeweils nächstgrößeren enthalten ist.



(a) SeqAcc für verschiedene σ-Werte.

(b) mCER für verschiedene σ-Werte.

Abbildung 5.4.1: Vergleich von Donut-Base und Donut-DC2 bei unterschiedlichen Trainingsanteilen (20%, 50%, 80%, 100%).

einer mCER von 4.6%. Donut-DC2 skaliert hingegen kontinuierlich mit zunehmender Datenmenge und übertrifft das Basismodell in allen Fällen. Unter Verwendung des vollständigen Trainingssatzes erreicht das Modell den höchsten Leistungswert mit 79,4% Sequenzgenauigkeit bei einer mCER von 3,1%.

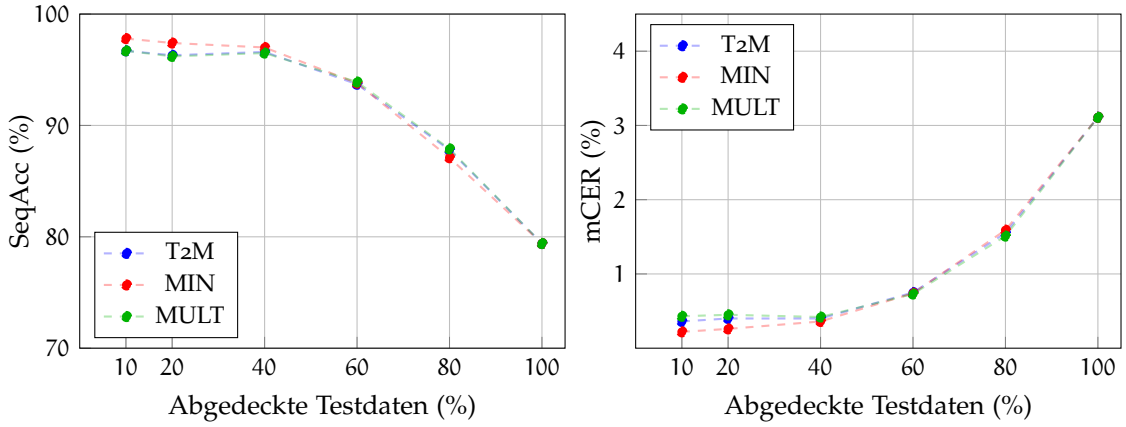
Die Ergebnisse verdeutlichen zudem, dass der Nutzen zusätzlicher Trainingsdaten mit wachsender Datenmenge abnimmt. Besonders für Donut-Base ist der Sprung von 20% auf 50% Trainingsdaten erheblich (37.4 auf 8.2), während die Verbesserung von 80% auf 100% vergleichsweise gering ausfällt (7.8 auf 4.6). Ein ähnliches Bild zeigt sich bei Donut-DC2, wenn auch auf niedrigerem Niveau. Dies weist darauf hin, dass die Modelle ab einem bestimmten Punkt weitgehend gesättigt sind.

Zusammenfassend lässt sich festhalten, Donut-DC2 in allen Szenarien deutlich überlegen ist, sowohl bei kleiner als auch bei großem Trainingssatz. Das Vortraining auf dem DC2-Datensatz wirkt daher wie ein leistungsfähiges Transfer-Learning.

5.4.2 *Modellsicherheit durch Konfidenzmaße*

Zur Quantifizierung der Zuverlässigkeit der Vorhersagen des DONUT-DC2-Modells zu quantifizieren, wurden die im Abschnitt 4.2 vorgestellten Konfidenzmaße MULT, MIN und T2M untersucht.

Zur Evaluation wurde für jede Konfidenzmetrik folgender Ansatz gewählt: Zunächst werden alle Testdaten nach ihrem Konfidenzwert absteigend sortiert. Anschließend werden sukzessive die obersten 10%, 20%, 40%, 60%, 80% sowie schließlich den gesamten Datensatz (100%) berücksichtigt. Für jede dieser Teilmengen wurden die mittlere SeqAcc und mCER berechnet. So konnte überprüft werden, ob eine hohe Konfidenz tatsächlich mit einer besseren Vorhersagequalität korreliert.



(a) (b)

Abbildung 5.4.2: Vergleich der Konfidenzmetriken T2M, MIN und MULT hinsichtlich (a) Sequenzgenauigkeit (SeqAcc) und (b) mittlerem Character Error Rate (mCER) des Donut-DC2 Modells. Die Auswertung erfolgt auf sukzessiven Teilmengen des Testdatensatzes, die nach absteigendem Konfidenzwert sortiert wurden (10% bis 100%).

Ein wesentlicher Aspekt, der sich aus den Ergebnissen (siehe Abbildungen 5.4.2) ableiten lässt, ist der deutliche Trade-off zwischen Abdeckungsrate und Zuverlässigkeit. Während die obersten 10–40% der nach Konfidenz sortierten Vorhersagen eine nahezu fehlerfreie Erkennung (mit einer Sequenzgenauigkeit von über 97,0% bei allen drei Konfidenzmaße) liefern, nimmt die Genauigkeit mit steigender Abdeckung erwartungsgemäß ab. Für die praktische Anwendung stellt sich damit die Frage nach einem geeigneten Gleichgewicht: Soll ein System nur die sichersten Vorhersagen automatisch übernehmen und damit eine geringere Abdeckung, aber hohe Verlässlichkeit gewährleisten, oder ist eine vollständige automatische Verarbeitung wünschenswert, obwohl dies ein höheres Fehlerrisiko mit sich bringt.

Ein ähnliches Bild zeigt sich auf Zeichenebene bei den mCER. In den obersten 10% erreicht MIN mit 0,22(%) den niedrigsten Fehlerwert, während MULT (0,43(%)) und T2M (0,36(%)) leicht darüber liegen. Auch bei 20% und 40% bleibt MIN vorn und weist damit durchgängig die geringsten Fehler auf. Dies verdeutlicht, dass die nach Konfidenzmaßen ausgewählten Vorhersagen nicht nur weniger komplette Sequenzfehler enthalten, sondern auch auf Zeichenebene präzise sind.

Mit zunehmender Abdeckung sinkt die Genauigkeit erwartungsgemäß ab, da auch weniger sichere Vorhersagen in die Bewertung einbezogen werden. Bei größeren Anteilen (60% und 80%) fällt die SeqAcc in allen Fällen deutlich ab und liegt zwischen 93–87%. Parallel dazu steigen die mCER spürbar an. Wird schließlich die gesamte Testmenge betrachtet, so erreichen alle drei Konfidenzmaße identische Ergebnisse von 79,4% SeqAcc bei einer mCER von 3,11%. Dieses Resultat entspricht der Leistung des Modells ohne Anwendung einer Konfidenzschwelle (vgl. Experiment 5.4.2 mit Donut-DC1 bei 100% Trainingsdaten).

Insgesamt lässt sich somit festhalten, dass alle drei Konfidenzmaße in der Lage sind, zuverlässig zwischen sicheren und unsicheren Vorhersagen zu unterscheiden. Unter ihnen erweist sich MIN als das stabilste und leistugsfähigste Maß, da es in den oberen Konfidenz-Segmenten (Top-10% bis Top-40%) konstant die höchste Sequenzgenauigkeit und gleichzeitig die niedrigste mCER liefert. Dies deutet darauf hin, dass die schlechteste Token-Wahrscheinlichkeit tatsächlich ein aussagekräftiger Indikator für die Gesamtzuverlässigkeit einer Sequenz ist. GAVG und T2M zeigen dagegen nahezu identische Verläufe und liegen insgesamt etwas unterhalb von MIN, da sie die Unsicherheit über die gesamte Sequenz mitteln und daher insgesamt etwas weniger strikt sind.

5.4.3 *Datensatzreduktion & Konfidenzbewertung*

In diesem kombinierten Experiment werden die beiden zuvor beschriebenen Experimente zusammengeführt: die Reduktion der Trainingsdaten (vgl. Abschnitt 5.4.1) sowie die Modellsicherheit durch Konfidenzmaße (vgl. Abschnitt 5.4.2). Dazu wurde der CM1-COVER-Trainingsdatensatz in hierarchische Teilmengen aufgeteilt. Für jede Trainingsstufe wurde das Modell Donut-DC2 trainiert und die Vorhersagen mithilfe der Konfidenzmetriken ausgewertet. Die Testdaten wurden dabei nach Konfidenzwert sortiert, und für Abdeckungsraten von 10% bis 100% wurden die SeqAcc sowie die mCER berechnet. Die Ergebnisse werden in der Abbildung 5.4.3 dargestellt.

Bei einer Reduktion der Trainingsdaten auf 20% wird der Einfluss der Verfügbarkeit von Trainingsdaten auf die Wirksamkeit der Konfidenzmetriken deutlich sichtbar. Die Sequenzgenauigkeit liegt zunächst bei rund 76% für die obersten 10% der nach Konfidenz sortierten Testdaten und sinkt kontinuierlich auf etwa 29% bei vollständiger Abdeckung. Gleichzeitig steigt die mCER von rund 3% auf 13,5%. In diesem Szenario gelingt es den Konfidenzwerten noch nicht, zuverlässig zwischen guten und schlechten Vorhersagen zu unterscheiden.

Mit 50% Trainingsdaten verbessert sich die Situation erheblich. Die Accuracy steigt bei niedrigen Abdeckungsraten (10–40%) auf über 90%, und der mCER bleibt in diesem Bereich

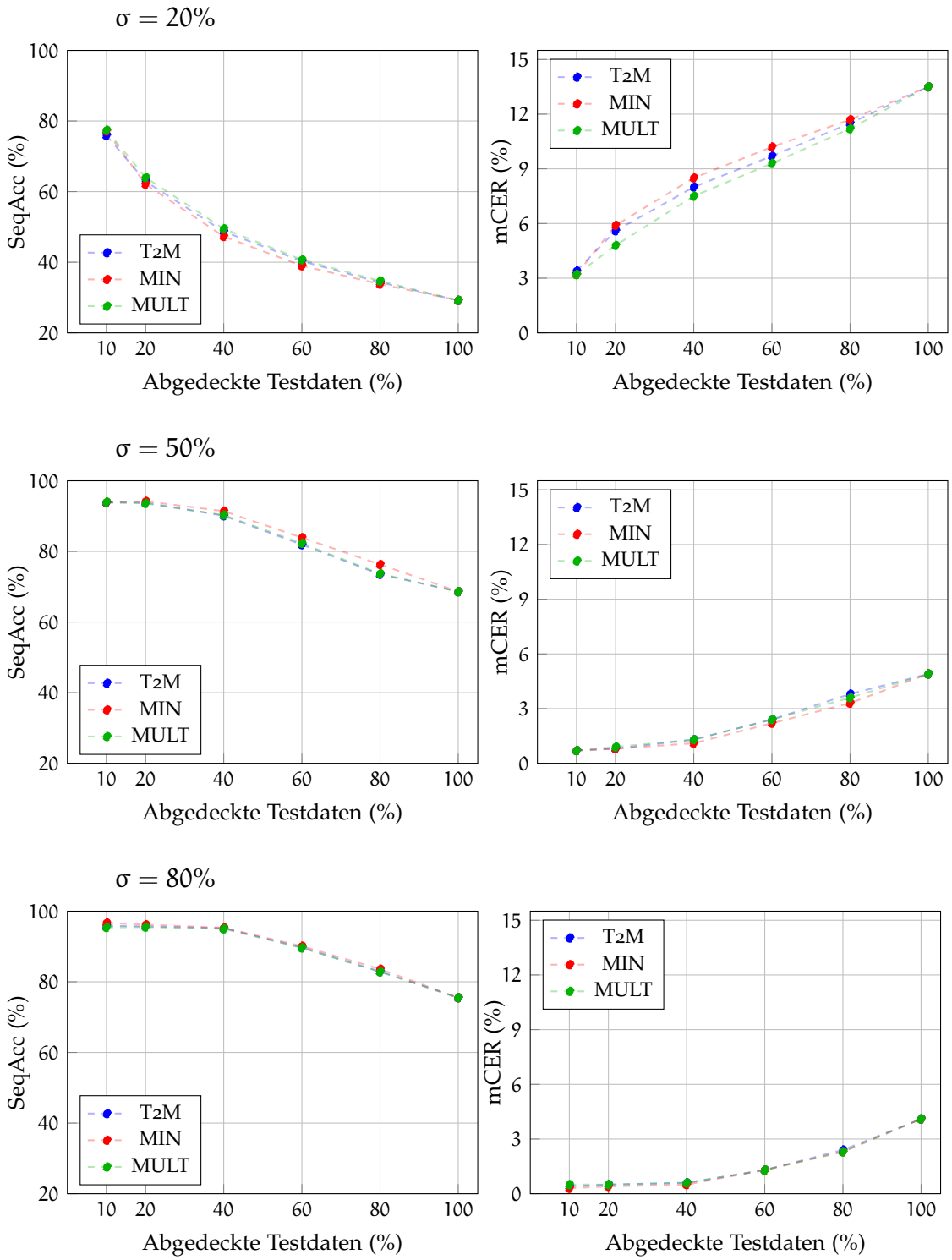


Abbildung 5.4.3: Vergleich der Konfidenzmetriken T2M, MIN und MULT hinsichtlich SeqAcc (links) und mCER (rechts) des Donut-DC2 Modells bei unterschiedlichen Trainingsanteilen (20%, 50%, 80%). Die Auswertung erfolgt auf sukzessiven Teilmengen des Testdatensatzes, die nach absteigendem Konfidenzwert sortiert wurden.

gering. Die geringen Unterschiede zwischen den Konfidenzmetriken deuten darauf hin, dass das Modell in diesem Szenario bereits ein stabiles Unsicherheitsmaß erlernt hat.

Bei 80% Trainingsdaten werden in den oberen Konfidenzbereichen (10–40%) weiterhin SeqAcc über 95% und sehr niedrigen mCER erzielt. Erst bei höherer Coverage nimmt die Leistung stärker ab, bleibt jedoch nur geringfügig schlechter als bei voller Trainingsmenge (vgl. Abschnitt 5.4.2). Dies deutet darauf hin, dass der Nutzen zusätzlicher Trainingsdaten ab einem Umfang von etwa 80% deutlich abnimmt und die Konfidenzmaße auch in diesem Kontext zuverlässig arbeiten.

5.4.4 *Fehlervermeidung durch Konfidenzschranken*

Um den praktischen Nutzen der Konfidenzprüfung zu demonstrieren, wurde ein Simulationsszenario entworfen. Dokumente sollen nur dann veröffentlicht werden, wenn für alle extrahierten Geburtsdaten gilt, dass seit der Geburt mindestens 100 Jahre vergangen sind. Da die CM1-Dokumente ohnehin fast alle älter als 100 Jahre sind, wurde ein fiktiver Stichtag eingeführt, um realistischere Prüfbedingungen zu schaffen.

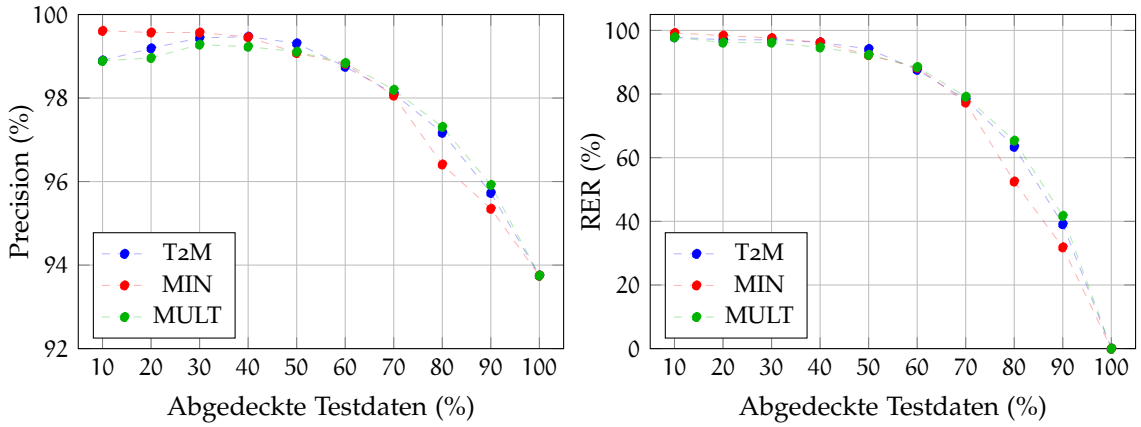
Um die Robustheit der Ergebnisse zu gewährleisten, wurde das Experiment nicht nur für einen einzelnen Stichtag, sondern für mehrere Referenzdaten im Zeitraum vom 01.01.2010 bis zum 01.01.2020 (in Jahresschritten) durchgeführt. Diese Auswahl ist repräsentativ, da die meisten Geburtsdaten in den CM1-Dokumenten in den Jahren 1900–1930 liegen. Somit lassen sich unterschiedliche Szenarien abbilden, in denen die 100-Jahres-Grenze gerade erreicht oder knapp verfehlt wird.

Für jedes Referenzdatum wird das Modell bewertet, ob es die Dokumente im Hinblick auf die 100-Jahres-Regel korrekt klassifiziert. Das Modell wird folgendermaßen evaluiert:

- True Positives (TP): Dokumente, die zurecht veröffentlicht werden, weil sowohl die Ground-Truth-Daten als auch die vom Modell vorhergesagten Daten älter als 100 Jahre sind.
- False Positives (FP): Dokumente, die veröffentlicht werden, obwohl mindestens ein Datum in den Ground-Truth-Daten jünger als 100 Jahre sind.
- False Negatives (FN): Dokumente, die eigentlich veröffentlicht werden dürften (alle Ground-Truth-Daten >100 Jahre), vom Modell aber zurückgehalten werden.
- True Negatives (TN): Dokumente, die korrekt zurückgehalten werden, da mindestens ein Ground-Truth-Datum noch nicht 100 Jahre alt ist.

Wichtig ist, dass hier nicht geprüft wird, ob das exakte Datum korrekt vorhergesagt wurde. Entscheidend ist allein, ob das Modell mit seiner Vorhersage zu einer richtigen oder falschen Klassifikation bezüglich der 100-Jahres-Grenze führt.

In einem weiteren Schritt wird untersucht, wie viele Dokumente ohne Konfidenzprüfung fälschlicherweise veröffentlicht worden wären und in welchem Maß sich solche Fehler durch eine Konfidenzschranke vermeiden lassen. Dazu werden alle Dokumente zunächst nach ihrem



(a) (b)

Abbildung 5.4.4: Vergleich der Konfidenzmetriken T2M, MIN und MULT hinsichtlich (a) Precision sowie (b) Relative Error Reduction (RER) des Donut-DC2 Modells. Die Auswertung erfolgt auf sukzessiven Teilmengen des Testdatensatzes, die nach absteigendem Konfidenzwert sortiert wurden.

Konfidenzwert sortiert und anschließend nur die obersten 10% bis 100% der Daten berücksichtigt. Das Vorgehen entspricht dabei konzeptionell dem zuvor beschriebenen Experiment 5.4.2, jedoch wird hier nicht die Sequenzgenauigkeit als Bewertungsmaß herangezogen, sondern die Presicion und Relative Error Reduction (RER) (vgl. Abschnitt 5.2.3).

Die Betrachtung der Precision (Siehe Abbildung 5.4.4 (a)) zeigt, dass bei obersten Konfidenzschwellen (10–50%) nahezu alle als veröffentlichbar eingestuften Dokumente korrekt sind. Die Precision liegt hier etwa 99% für alle drei Konfidenzmaße. Mit zunehmender Schwelle sinkt die Precision leicht, da mehr Dokumente mit geringerer Sicherheit freigegeben werden, aber selbst bei 80% bleibt sie noch hoch (ca. 97%).

Parallel dazu verdeutlicht RER die Wirksamkeit der Konfidenzschranken (Siehe Abbildung 5.4.4 (b)). Bereits moderate Schwellenwerte $\sigma = 50\%$ verhindern über 90% der potenziell falschen Veröffentlichungen. Bei $\sigma = 80\%$ liegt die Reduktion noch zwischen ca. 52% und 65%. Wird hingegen die gesamte Datenmenge ohne Konfidenzfilter berücksichtigt ($\sigma = 100\%$), so ergibt sich erwartungsgemäß keine Reduktion, die als Baseline dient.

Die kombinierte Betrachtung von Precision und Reduktion zeigt, dass hohe Konfidenzwerte einerseits die Anzahl korrekt veröffentlichter Dokumente maximieren und andererseits das Risiko von Fehlveröffentlichungen deutlich reduzieren. Insgesamt demonstriert die Analyse, dass die Anwendung von Konfidenzschranken eine effiziente Methode zur sicheren Freigabe historischer Dokumente darstellt.

6 FAZIT

Ziel dieser Arbeit war die Untersuchung eines End-to-End-Ansatzes zur automatisierten Transkription handschriftlicher Datumsangaben in historischen Dokumenten. Hierfür wurde ein Ansatz auf Basis des Donut-Modells gewählt, das Bildinformationen in einem durchgängigen Prozess ohne vorgelagerte OCR-Schritt direkt in Textsequenzen transkribiert. Ergänzend wurden Konfidenzmaße integriert, die eine Einschätzung der Vorhersagesicherheit ermöglichen. Dies ist insbesondere für praktische Anwendungen im Bereich historischer Archive oder der Forschung, wo fehlerhafte Transkriptionen erhebliche Auswirkungen haben können, von großer Bedeutung. Durch die Konfidenzmaße könnte ein halbautomatisches System entwickelt werden. Dazu gehen unsichere Ergebnisse an Menschen zur Überprüfung, sichere Ergebnisse können automatisch übernommen werden.

Das DONUT-Modell basiert auf einer Architektur mit einem Swin-Transformer als Encoder zur visuellen Repräsentation und einem BART-Decoder zur textuellen Sequenzgenerierung. Ein zentraler Baustein war die zweistufige Finetuning-Strategie. Zunächst wurden ausgeschnittene Bildsegmente mit isolierten Datumsangaben aus DC1-Datensatz genutzt, um dem Modell typische Muster von Datumsformaten beizubringen. Anschließend erfolgte ein Training auf vollständigen Dokumentseiten aus CM1-COVER-Datensatz, wodurch das Modell in der Lage war, die zuvor erlernten Muster in einem komplexeren Kontext korrekt zu erkennen und zu lokalisieren.

Die Arbeit hebt mehrere Stärken des Donut-DC2 Modells hervor. Bei der Extraktion von Datumsangaben erreicht das Modell eine Sequenzgenauigkeit von 79,4% und eine mittlere Character Error Rate (mCER) von 3,4, wobei die Integration von Konfidenzwerten zusätzliche Vorteile bietet. Insbesondere lassen sich unsichere Vorhersagen gezielt identifizieren, wodurch die Qualität der extrahierten Daten verbessert werden kann. Dies zeigt sich besonders deutlich bei den obersten 40% der Testdaten mit den höchsten Konfidenzwerten, für die eine Sequenzgenauigkeit von über 96% erzielt wurde.

Zur Veranschaulichung des praktischen Nutzens der Konfidenzprüfung wurde ein Beispiel in Form eines Simulationsszenarios durchgeführt. Dabei sollten Dokumente nur dann veröffentlicht werden, wenn für alle extrahierten Geburtsdaten gilt, dass seit der Geburt mindestens 100 Jahre vergangen sind. Um realistische Prüfbedingungen zu schaffen, wurde ein fiktiver Stichtag eingeführt. Die Betrachtung der Precision in diesem Experiment zeigt, dass bei den obersten Konfidenzschwellen von 10–50% nahezu alle als veröffentlichbar eingestuften Dokumente korrekt sind. Die Precision liegt für alle drei Konfidenzmaße bei etwa 99%, was den praktischen Nutzen der Konfidenzprüfung verdeutlicht.

Gleichzeitig bestehen auch Einschränkungen. Das Modell ist in seiner Leistungsfähigkeit eng an bestimmte Dokumentformate gebunden, da die historischen Dokumente des CM1-COVER-Korpus ein relativ homogenes Layout aufweisen. Auch bleibt die Abhängigkeit von ausreichend

annotierten Trainingsdaten bestehen, wodurch der Einsatz in diversifizierten oder datenarmen Anwendungsszenarien erschwert wird. Schließlich erweisen sich Konfidenzmaße zwar als nützlich, liefern jedoch nicht in allen Fällen eine verlässliche Abbildung der tatsächlichen Unsicherheit.

Für die Zukunft ergeben sich mehrere Ansatzpunkte. Zum einen bietet sich die Erweiterung des Ansatzes auf weitere Dokumentfelder, wie Namen, Beträge oder Adressen, an. Zum anderen bieten aktive Lernstrategien [Set09, Set12] oder semi-supervised Learning [Zhu08, CSZ09] die Möglichkeit, die Dateneffizienz zu steigern und die Abhängigkeit von großen Trainingsmengen zu reduzieren. Darüber hinaus erscheint die Integration in digitale Archivsysteme besonders vielversprechend, um historische Datenbestände automatisiert und qualitätsgesichert zu erschließen.

Insgesamt zeigt diese Arbeit, dass moderne End-to-End-Modelle wie Donut in Verbindung mit geeigneten Finetuning-Strategien und Konfidenzbewertung ein leistungsfähiges Werkzeug zur automatisierung historischer Dokumente darstellen. Damit einen wichtigen Schritt in Richtung einer effizienteren und verlässlicheren Digitalisierung des kulturellen Erbes ermöglichen.

7 ANHANG

Abg. Daten	SeqAcc			mCER		
	T2M	MIN	MULT	t2M	MIN	MULT
10%	96.61	**97.80**	96.63	0.36	**0.22**	0.43
20%	96.33	**97.37**	96.15	0.40	**0.26**	0.45
40%	96.58	**96.95**	96.50	0.40	**0.36**	0.42
60%	93.73	93.76	**93.85**	0.75	0.74	**0.73**
80%	87.82	87.14	**87.85**	1.55	1.59	**1.51**
100%	79.39	79.39	79.39	3.11	3.11	3.11

Tabelle 7.0.1: Die exakten Werte von SeqAcc (%) und mCER (%) in Experiment 5.4.2 in Abhängigkeit vom Anteil der Testdaten, die nach absteigenden Konfidenzwerten ausgewählt wurden.

	Abg. Daten	SeqAcc			mCER		
		T2M	MIN	MULT	t2M	MIN	MULT
$\sigma = 20\%$	10%	75.87	77.22	**77.43**	3.42	**3.20**	**3.20**
	20%	62.86	62.08	**63.99**	5.60	5.85	**4.84**
	40%	48.72	47.29	**49.53**	8.00	8.50	**7.55**
	60%	40.31	38.96	**40.71**	9.73	10.19	**9.35**
	80%	34.33	33.76	**34.72**	11.45	11.72	**11.15**
	100%	29.24	29.24	29.24	13.48	13.48	13.48
$\sigma = 50\%$	10%	93.81	93.91	**93.96**	0.72	**0.70**	**0.70**
	20%	93.85	**94.20**	93.57	0.80	**0.76**	0.88
	40%	90.14	**91.42**	90.29	1.28	**1.13**	1.27
	60%	81.92	**83.90**	82.26	2.40	**2.23**	2.37
	80%	73.46	**76.27**	73.69	3.76	**3.33**	3.65
	100%	68.61	68.61	68.61	4.91	4.91	4.91
$\sigma = 80\%$	10%	95.92	**96.66**	95.44	0.43	**0.34**	0.49
	20%	95.70	**96.25**	95.45	0.47	**0.41**	0.50
	40%	95.20	**95.26**	94.99	0.60	**0.55**	0.58
	60%	89.68	**90.10**	89.63	1.33	**1.26**	1.32
	80%	82.94	**83.61**	82.81	2.41	**2.29**	2.33
	100%	75.54	75.54	75.54	4.06	4.06	4.06

Tabelle 7.0.2: Exakte Werte von SeqAcc (%) und mCER (%) des Donut-DC2 Modells bei unterschiedlichen Trainingsanteilen (20%, 50%, 80%) im Vergleich der Konfidenzmetriken T2M, MIN und MULT aus Experiment 5.4.3. Die Auswertung erfolgt auf sukzessiven Teilmengen des Testdatensatzes (10%, 20%, 40%, 60%, 80%, 100%), die nach absteigendem Konfidenzwert sortiert wurden.

Abg. Daten	Precision			RER		
	T2M	MIN	MULT	t2M	MIN	MULT
10%	98.9	**99.6**	98.9	97.9	**99.2**	97.9
20%	99.2	**99.6**	99.0	97.1	**98.4**	96.2
30%	99.4	**99.6**	99.3	97.1	**97.6**	96.2
40%	**99.5**	**99.5**	99.2	**96.3**	**96.3**	94.7
50%	**99.3**	99.1	99.1	**94.2**	92.3	92.3
60%	**98.8**	**98.8**	**98.8**	87.6	88.3	**88.5**
70%	98.1	98.1	**98.2**	78.3	77.3	**79.2**
80%	97.2	96.4	**97.3**	63.4	52.5	**65.4**
90%	95.7	95.4	**95.9**	39.1	31.8	**41.7**
100%	93.8	93.8	93.8	0.0	0.0	0.0

Tabelle 7.0.3: Die exakten Werte von Precision (%) und Relative Error Reduction (%) über verschiedene Konfidenzschwellenwerte aus Experiment 5.4.4.

LITERATURVERZEICHNIS

[AAS20] Ajit, Arohan ; Acharya, Koustav ; Samanta, Abhishek: A Review of Convolutional Neural Networks. In: *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 2020, S. 1–5

[ACM+15] Afzal, Muhammad Z. ; Capobianco, Samuele ; Malik, Muhammad I. ; Marinai, Simone ; Breuel, Thomas M. ; Dengel, Andreas ; Liwicki, Marcus: Deepdocclassifier: Document classification with deep Convolutional Neural Network. In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 2015, S. 1111–1115

[Agg23] Aggarwal, Charu C.: *Neural Networks and Deep Learning*. Springer Cham, 2023. ISSN 1611–3349

[Arc] Archives, Arolsen: *Mehr Wissen über Displaced Persons*. https://arolsen-archives.org/blick-in-die-sammlung/nachkriegsakten-zu-displaced-persons/, . – Accessed: 2025-07-21

[BB08] Bertolami, Roman ; Bunke, Horst: Hidden Markov model-based ensemble methods for offline handwritten text line recognition. In: *Pattern Recognition* 41 (2008), Nr. 11, S. 3452–3460. – ISSN 0031–3203

[BCB16] Bahdanau, Dzmitry ; Cho, Kyunghyun ; Bengio, Yoshua: *Neural Machine Translation by Jointly Learning to Align and Translate*. https://arxiv.org/abs/1409.0473. Version: 2016

[BCKW15] Blundell, Charles ; Cornebise, Julien ; Kavukcuoglu, Koray ; Wierstra, Daan: *Weight Uncertainty in Neural Networks*. https://arxiv.org/abs/1505.05424. Version: 2015

[BCN18] Bottou, Léon ; Curtis, Frank E. ; Nocedal, Jorge: *Optimization Methods for Large-Scale Machine Learning*. https://arxiv.org/abs/1606.04838. Version: 2018

[BCSS23] Blecher, Lukas ; Cucurull, Guillem ; Scialom, Thomas ; Stojnic, Robert: *Nougat: Neural Optical Understanding for Academic Documents*. 2023

[Bis06] In: Bishop, Christopher: *Pattern Recognition and Machine Learning*. Bd. 16. 2006, S. 140–155

[BKH16] Ba, Jimmy L. ; Kiros, Jamie R. ; Hinton, Geoffrey E.: *Layer Normalization*. https://arxiv.org/abs/1607.06450. Version: 2016

[BM17] Bluche, Théodore ; Messina, Ronaldo: Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* Bd. 01, 2017, S. 646–651

[Bot10] Bottou, Léon: Large-Scale Machine Learning with Stochastic Gradient Descent. In: Lechevallier, Yves (Hrsg.) ; Saporta, Gilbert (Hrsg.): *Proceedings of COMPSTAT'2010*. Heidelberg : Physica-Verlag HD, 2010. – ISBN 978–3–7908–2604–3, S. 177–186

[BRM+20] Boroş, Emanuela ; Romero, Verónica ; Maarand, Martin ; Zenklová, Kateřina ; Křečková, Jitka ; Vidal, Enrique ; Stutzmann, Dominique ; Kermorvant, Christopher: A comparison of sequential and combined approaches for named entity recognition in a corpus of handwritten medieval charters. In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2020, S. 79–84

[BSF94] Bengio, Y. ; Simard, P. ; Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. In: *IEEE Transactions on Neural Networks* 5 (1994), Nr. 2, S. 157–166

[CCP23a] Coquenet, Denis ; Chatelain, Clément ; Paquet, Thierry: DAN: A Segmentation-Free Document Attention Network for Handwritten Document Recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2023), Juli, Nr. 7, S. 8227–8243. – ISSN 1939–3539

[CCP23b] Coquenet, Denis ; Chatelain, Clément ; Paquet, Thierry: Faster DAN: Multi-target Queries with Document Positional Encoding for End-to-End Handwritten Document Recognition. In: *Document Analysis and Recognition - ICDAR 2023*, Springer Nature Switzerland, 2023. – ISBN 9783031416859, S. 182–199

[CMB23] Christopher M. Bishop, Hugh B.: In: *Deep Learning: Foundations and Concepts*, Springer Cham, 2023. – ISBN 978–3–031–45468–4, S. 198–286

[CSZ09] Chapelle, O. ; Scholkopf, B. ; Zien, A. Eds.: Semi-Supervised Learning (Chapelle, O. et al., Eds.; 2006) [Book reviews]. In: *IEEE Transactions on Neural Networks* 20 (2009), Nr. 3, S. 542–542

[CTBH+19] Corbière, Charles ; Thome, Nicolas ; Bar-Hen, Avner ; Cord, Matthieu ; Pérez, Patrick: *Addressing Failure Prediction by Learning Model Confidence*. https://arxiv.org/abs/1910.04851. Version: 2019

[DBK+21] Dosovitskiy, Alexey ; Beyer, Lucas ; Kolesnikov, Alexander ; Weissenborn, Dirk ; Zhai, Xiaohua ; Unterthiner, Thomas ; Dehghani, Mostafa ; Minderer, Matthias ; Heigold, Georg ; Gelly, Sylvain ; Uszkoreit, Jakob ; Houlsby, Neil: *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. https://arxiv.org/abs/2010.11929. Version: 2021

[DCLT19] DEVLIN, Jacob ; CHANG, Ming-Wei ; LEE, Kenton ; TOUTANOVA, Kristina: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: BURSTEIN, Jill (Hrsg.) ; DORAN, Christy (Hrsg.) ; SOLORIO, Thamar (Hrsg.): *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota : Association for Computational Linguistics, Juni 2019, S. 4171–4186

[DFO20] DEISENROTH, Marc P. ; FAISAL, A A. ; ONG, Cheng S.: *Mathematics for machine learning*. Cambridge University Press, 2020

[DJS+22] DAHL, Christian M. ; JOHANSEN, Torben S. D. ; SØRENSEN, Emil N. ; WESTERMANN, Christian E. ; WITTROCK, Simon F.: *DARE: A large-scale handwritten date recognition system*. https://arxiv.org/abs/2210.00503. Version: 2022

[DL13] DAMIANOU, Andreas C. ; LAWRENCE, Neil D.: *Deep Gaussian Processes*. https://arxiv.org/abs/1211.0358. Version: 2013

[DMP+22] DAVIS, Brian ; MORSE, Bryan ; PRICE, Bryan ; TENSMEYER, Chris ; WIGINGTON, Curtis ; MORARIU, Vlad: *End-to-end Document Recognition and Understanding with Dessurt*. 2022

[DSS08] In: DHINGRA, Kapil D. ; SANYAL, Sudip ; SHARMA, Pramod K.: *A Robust OCR for Degraded Documents*. Boston, MA : Springer US, 2008. – ISBN 978–0–387–74938–9, S. 497–509

[FB14] In: FRINKEN, Volkmar ; BUNKE, Horst: *Continuous Handwritten Script Recognition*. London : Springer London, 2014. – ISBN 978–0–85729–859–1, S. 391–425

[FRB+17] FORNÉS, Alicia ; ROMERO, Verónica ; BARÓ, Arnau ; TOLEDO, Juan I. ; SÁNCHEZ, Joan A. ; VIDAL, Enrique ; LLADÓS, Josep: ICDAR2017 Competition on Information Extraction in Historical Handwritten Records. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* Bd. 01, 2017, S. 1389–1394

[GBC16] GOODFELLOW, Ian ; BENGIO, Yoshua ; COURVILLE, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[GFGS06] GRAVES, Alex ; FERNÁNDEZ, Santiago ; GOMEZ, Faustino ; SCHMIDHUBER, Jürgen: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: *Proceedings of the 23rd International Conference on Machine Learning*. New York, NY, USA : Association for Computing Machinery, 2006 (ICML '06). – ISBN 1595933832, S. 369–376

[GG16] GAL, Yarin ; GHAHRAMANI, Zoubin: *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. https://arxiv.org/abs/1506.02142. Version: 2016

[GPSW17] GUO, Chuan ; PLEISS, Geoff ; SUN, Yu ; WEINBERGER, Kilian Q.: *On Calibration of Modern Neural Networks*. https://arxiv.org/abs/1706.04599. Version: 2017

[gQL+19] GUO, He ; QIN, Xiameng ; LIU, Jiaming ; HAN, Junyu ; LIU, Jingtuo ; DING, Errui: *EATEN: Entity-aware Attention for Single Shot Visual Text Extraction*. https://arxiv.org/abs/1909.09380. Version: 2019

[HG18] HENDRYCKS, Dan ; GIMPEL, Kevin: *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*. https://arxiv.org/abs/1610.02136. Version: 2018

[HG23] HENDRYCKS, Dan ; GIMPEL, Kevin: *Gaussian Error Linear Units (GELUs)*. https://arxiv.org/abs/1606.08415. Version: 2023

[HLC+22] HUANG, Yupan ; LV, Tengchao ; CUI, Lei ; LU, Yutong ; WEI, Furu: LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. New York, NY, USA : Association for Computing Machinery, 2022 (MM '22). – ISBN 9781450392037, S. 4083–4091

[HZRS16] HE, Kaiming ; ZHANG, Xiangyu ; REN, Shaoqing ; SUN, Jian: Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016

[IK23] IWANA, Brian K. ; KUSUDA, Akihiro: *Vision Conformer: Incorporating Convolutions into Vision Transformer Layers*. https://arxiv.org/abs/2304.13991. Version: 2023

[JKGG18] JIANG, Heinrich ; KIM, Been ; GUAN, Melody Y. ; GUPTA, Maya: *To Trust Or Not To Trust A Classifier*. https://arxiv.org/abs/1805.11783. Version: 2018

[KB17] KINGMA, Diederik P. ; BA, Jimmy: *Adam: A Method for Stochastic Optimization*. https://arxiv.org/abs/1412.6980. Version: 2017

[Ket17] KETKAR, Nikhil: *Deep learning with Python*. Apress Berkeley, CA, 2017. – ISBN 978–1–4842–2765–7

[KG17] KENDALL, Alex ; GAL, Yarin: *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* https://arxiv.org/abs/1703.04977. Version: 2017

[KHY+22] KIM, Geewook ; HONG, Teakgyu ; YIM, Moonbin ; NAM, JeongYeon ; PARK, Jinyoung ; YIM, Jinyeong ; HWANG, Wonseok ; YUN, Sangdoo ; HAN, Dongyoon ; PARK, Seunghyun: OCR-Free Document Understanding Transformer. In: AVIDAN, Shai (Hrsg.) ; BROSTOW, Gabriel (Hrsg.) ; CISSÉ, Moustapha (Hrsg.) ; FARINELLA, Giovanni M. (Hrsg.) ; HASSNER, Tal (Hrsg.): *Computer Vision – ECCV 2022*. Cham : Springer Nature Switzerland, 2022. – ISBN 978–3–031–19815–1, S. 498–517

[KKY+14] KANG, Le ; KUMAR, Jayant ; YE, Peng ; LI, Yi ; DOERMANN, David: Convolutional Neural Networks for Document Image Classification. In: *2014 22nd International Conference on Pattern Recognition*, 2014, S. 3168–3172

[KNH+22] KHAN, Salman ; NASEER, Muzammal ; HAYAT, Munawar ; ZAMIR, Syed W. ; KHAN, Fahad S. ; SHAH, Mubarak: Transformers in Vision: A Survey. In: *ACM Computing Surveys* 54 (2022), Januar, Nr. 10s, 1–41. http://dx.doi.org/10.1145/3505244. – ISSN 1557–7341

[LB98] LECUN, Yann ; BENGIO, Yoshua: Convolutional networks for images, speech, and time series. In: *The handbook of brain theory and neural networks* (1998)

[Lem12] LEMARÉCHAL, Claude: Cauchy and the gradient method. In: *Doc Math Extra* 251 (2012), Nr. 254, S. 10

[LGG+20] LIU, Yinhan ; GU, Jiatao ; GOYAL, Naman ; LI, Xian ; EDUNOV, Sergey ; GHAZVININEJAD, Marjan ; LEWIS, Mike ; ZETTLEMOYER, Luke: *Multilingual Denoising Pre-training for Neural Machine Translation*. https://arxiv.org/abs/2001.08210. Version: 2020

[LLC+21] LIU, Ze ; LIN, Yutong ; CAO, Yue ; HU, Han ; WEI, Yixuan ; ZHANG, Zheng ; LIN, Stephen ; GUO, Baining: Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, S. 9992–10002

[LLG+19] LEWIS, Mike ; LIU, Yinhan ; GOYAL, Naman ; GHAZVININEJAD, Marjan ; MOHAMED, Abdelrahman ; LEVY, Omer ; STOYANOV, Ves ; ZETTLEMOYER, Luke: *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019

[MCCD13] MIKOLOV, Tomas ; CHEN, Kai ; CORRADO, Greg ; DEAN, Jeffrey: *Efficient Estimation of Word Representations in Vector Space*. https://arxiv.org/abs/1301.3781. Version: 2013

[MLEY+00] MORITA, M.E. ; LETELIER, E. ; EL YACOUBI, A. ; BORTOLOZZI, F. ; SABOURIN, R.: Recognition of handwritten dates on bank checks using an HMM approach. In: *Proceedings 13th Brazilian Symposium on Computer Graphics and Image Processing (Cat. No.PR00878)*, 2000, S. 113–120

[MSC+13] MIKOLOV, Tomas ; SUTSKEVER, Ilya ; CHEN, Kai ; CORRADO, Greg ; DEAN, Jeffrey: *Distributed Representations of Words and Phrases and their Compositionality*. https://arxiv.org/abs/1310.4546. Version: 2013

[NP20] NICOLSON, Aaron ; PALIWAL, Kuldip K.: Masked multi-head self-attention for causal speech enhancement. In: *Speech Communication* 125 (2020), S. 80–96. – ISSN 0167–6393

[OBK+24] OKAMOTO, Yamato ; BAEK, Youngmin ; KIM, Geewook ; NAKAO, Ryota ; KIM, DongHyun ; YIM, Moon B. ; PARK, Seunghyun ; LEE, Bado: *CREPE: Coordinate-Aware End-to-End Document Parser*. https://arxiv.org/abs/2405.00260. Version: 2024

[ON15] O'Shea, Keiron ; Nash, Ryan: *An Introduction to Convolutional Neural Networks*. https://arxiv.org/abs/1511.08458. Version: 2015

[Plaoo] Platt, John: Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In: *Adv. Large Margin Classif.* 10 (2000), 06

[PNCH15] Pakdaman Naeini, Mahdi ; Cooper, Gregory ; Hauskrecht, Milos: Obtaining Well Calibrated Probabilities Using Bayesian Binning. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29 (2015), Feb., Nr. 1

[Pow20] Powers, David M. W.: *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. https://arxiv.org/abs/2010.16061. Version: 2020

[Pre12] Prechelt, Lutz: Early Stopping — But When? In: Montavon, Grégoire (Hrsg.) ; Orr, Geneviève B. (Hrsg.) ; Müller, Klaus-Robert (Hrsg.): *Neural Networks: Tricks of the Trade: Second Edition*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–35289–8, S. 53–67

[PSL+19] Park, Seunghyun ; Shin, Seung ; Lee, Bado ; Lee, Junyeop ; Surh, Jaeheung ; Seo, Minjoon ; Lee, Hwalsuk: CORD: A Consolidated Receipt Dataset for Post-OCR Parsing. (2019)

[Pui17] Puigcerver, Joan: Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* Bd. 1, 2017, S. 67–72

[PV21] Poulos, Jason ; Valle, Rafael: Character-based handwritten text transcription with attention networks. In: *Neural Computing and Applications* 33 (2021), Februar, Nr. 16, S. 10563–10573. – ISSN 1433–3058

[RAS20] Rasamoelina, Andrinandrasana D. ; Adjailia, Fouzia ; Sinčák, Peter: A Review of Activation Function for Artificial Neural Network. In: *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2020, S. 281–286

[RFS+13] Romero, Verónica ; Fornés, Alicia ; Serrano, Nicolás ; Sánchez, Joan A. ; Toselli, Alejandro H. ; Frinken, Volkmar ; Vidal, Enrique ; Lladós, Josep: The ESPOSALLES database: An ancient marriage license corpus for off-line handwriting recognition. In: *Pattern Recognition* 46 (2013), Nr. 6, S. 1658–1669. – ISSN 0031–3203

[RHW86] Rumelhart, David ; Hinton, Geoffrey ; Williams, Ronald: Learning representations by back-propagating errors, Nature Publishing Group UK London, 1986. – ISBN 1476–4687

[RNS+18] Radford, Alec ; Narasimhan, Karthik ; Salimans, Tim ; Sutskever, Ilya u. a.: Improving language understanding by generative pre-training. (2018)

[Rud17] Ruder, Sebastian: *An overview of gradient descent optimization algorithms*. https://arxiv.org/abs/1609.04747. Version: 2017

[RZL17] RAMACHANDRAN, Prajit ; ZOPH, Barret ; LE, Quoc V.: *Searching for Activation Functions*. https://arxiv.org/abs/1710.05941. Version: 2017

[Set09] SETTLES, Burr: Active learning literature survey. (2009)

[Set12] SETTLES, Burr: *Active Learning*. Bd. 6. 2012

[SF17] SUDHOLT, Sebastian ; FINK, Gernot A.: *PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents*. https://arxiv.org/abs/1604.00187. Version: 2017

[SHB16] SENNRICH, Rico ; HADDOW, Barry ; BIRCH, Alexandra: *Neural Machine Translation of Rare Words with Subword Units*. 2016

[SKP97] SVOZIL, Daniel ; KVASNICKA, Vladimír ; POSPICHAL, Jirí: Introduction to multi-layer feed-forward neural networks. In: *Chemometrics and Intelligent Laboratory Systems* 39 (1997), Nr. 1, S. 43–62. – ISSN 0169–7439

[SN12] SCHUSTER, Mike ; NAKAJIMA, Kaisuke: Japanese and Korean voice search. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, S. 5149–5152

[SNBTL20] SOUSA NETO, Arthur F. ; BEZERRA, Byron Leite D. ; TOSELLI, Alejandro H. ; LIMA, Estanislau B.: HTR-Flor: A Deep Learning System for Offline Handwritten Text Recognition. In: *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2020, S. 54–61

[TWW22] TUNSTALL, Lewis ; WERRA, Leandro von ; WOLF, Thomas: *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, Incorporated, 2022 https://books.google.ch/books?id=7hhyzgEACAAJ. – ISBN 1098103246

[VDN16] VOIGTLAENDER, Paul ; DOETSCH, Patrick ; NEY, Hermann: Handwriting Recognition with Large Multidimensional Long Short-Term Memory Recurrent Neural Networks. In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, S. 228–233

[VLBM08] VINCENT, Pascal ; LAROCHELLE, Hugo ; BENGIO, Yoshua ; MANZAGOL, Pierre-Antoine: Extracting and composing robust features with denoising autoencoders. In: *Proceedings of the 25th International Conference on Machine Learning*. New York, NY, USA : Association for Computing Machinery, 2008 (ICML '08). – ISBN 9781605582054, S. 1096–1103

[VSP+23] VASWANI, Ashish ; SHAZEER, Noam ; PARMAR, Niki ; USZKOREIT, Jakob ; JONES, Llion ; GOMEZ, Aidan N. ; KAISER, Lukasz ; POLOSUKHIN, Illia: *Attention Is All You Need*. 2023

[VWN23] VRIES, Wietse de ; WIELING, Martijn ; NISSIM, Malvina: *DUMB: A Benchmark for Smart Evaluation of Dutch Models*. https://arxiv.org/abs/2305.13026. Version: 2023

[WTM+25] Wolf, Fabian ; Tüselmann, Oliver ; Matei, Arthur ; Hennies, Lukas ; Rass, Christoph ; Fink, Gernot A.: *CM1 – A Dataset for Evaluating Few-Shot Information Extraction with Large Vision Language Models*. https://arxiv.org/abs/2505.04214. Version: 2025

[WXCH17] Wang, Qi ; Xu, Jungang ; Chen, Hong ; He, Ben: Two improved continuous bag-of-word models. In: *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, S. 2851–2856

[XLC+20] Xu, Yiheng ; Li, Minghao ; Cui, Lei ; Huang, Shaohan ; Wei, Furu ; Zhou, Ming: LayoutLM: Pre-training of Text and Layout for Document Image Understanding. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, ACM, August 2020 (KDD '20), S. 1192–1200

[XLC+21] Xu, Yiheng ; Lv, Tengchao ; Cui, Lei ; Wang, Guoxin ; Lu, Yijuan ; Florencio, Dinei ; Zhang, Cha ; Wei, Furu: *LayoutXLM: Multimodal Pre-training for Multilingual Visually-rich Document Understanding*. https://arxiv.org/abs/2104.08836. Version: 2021

[XXL+22] Xu, Yang ; Xu, Yiheng ; Lv, Tengchao ; Cui, Lei ; Wei, Furu ; Wang, Guoxin ; Lu, Yijuan ; Florencio, Dinei ; Zhang, Cha ; Che, Wanxiang ; Zhang, Min ; Zhou, Lidong: *LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding*. https://arxiv.org/abs/2012.14740. Version: 2022

[XZ23] Xiao, Tong ; Zhu, Jingbo: *Introduction to Transformers: an NLP Perspective*. https://arxiv.org/abs/2311.17633. Version: 2023

[ZC88] Zhou ; Chellappa: Computation of optical flow using a neural network. In: *IEEE 1988 International Conference on Neural Networks*, 1988, S. 71–78 vol.2

[ZE01] Zadrozny, Bianca ; Elkan, Charles: Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001 (ICML '01). – ISBN 1558607781, S. 609–616

[ZE02] Zadrozny, Bianca ; Elkan, Charles: Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2002), 08

[Zhu08] Zhu, Xiaojin: Semi-Supervised Learning Literature Survey. In: *Comput Sci, University of Wisconsin-Madison* 2 (2008), 07