ANNIKA THELEN

TABELLENSTRUKTURERKENNUNG MIT NEURONALEN NETZWERKEN

TABELLENSTRUKTURERKENNUNG MIT NEURONALEN NETZWERKEN

ANNIKA THELEN

Bachelorarbeit Informatik

Betreuer: Dr. Moritz Wolter
Gutachter: Dr. Moritz Wolter, Prof. Dr. Hildegard Kühne
HPC/A-Lab
Mathematisch-Naturwissenschaftliche Fakultät
Rheinische Friedrich-Wilhelms-Universität Bonn

10. November 2024

Annika Thelen: *Tabellenstrukturerkennung mit Neuronalen Netzwerken* ,
Bachelorarbeit Informatik, © 10. November 2024

DANKSAGUNG

Ich danke meinem Betreuer Dr. Moritz Wolter herzlich für die umfassende Betreuung und Unterstützung.

Bei Prof. Dr. Hildegard Kühne möchte ich mich für ihre Unterstützung und ihre hilfreichen Anregungen herzlich bedanken.

Außerdem möchte ich mich für die von Dr. Moritz Wolter bereitgestellte Möglichkeit, den CLAIX-2023-ML-Cluster der RWTH Aachen nutzen zu können, bedanken.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ALGORITHMENVERZEICHNIS

FORMELVERZEICHNIS

Teil I

EINLEITUNG UND LITERATUR

1 EINLEITUNG

Durch Deep-Learning-Ansätze ist Digitalisierung und Transkribierung von Dokumenten wesentlich einfacher geworden. Besonders uneinheitliche Tabellenstrukturen stellen dabei aber weiterhin ein Problem dar [47].

Ein Beispiel dazu sind historische Dokumente, die oft handgeschriebene und uneinheitliche oder uneindeutige Tabellen enthalten, wodurch die Erfassung der Tabellenstruktur maßgeblich erschwert wird. Auch sind die Tabellen oft nicht perfekt achsenparallel auf der Dokumentseite orientiert, da sie von Hand gezeichnet oder gesetzt wurden. Da deswegen die Annahmen über Tabellenstrukturen, die für viele Transkribierungsansätze verwendet werden [47], nicht zwingend zutreffen, wird die Übertragung von diesen auf historische Tabellen erschwert.

Herkömmliche Layouterkennung bei bestehenden Diensten wie Transkribus [56] scheitert somit oft an der Strukturerkennung der Tabellen [43]. Lösungen sind oft wenig flexibel, sodass seperate Modelle für Tabellenerkennung, Strukturerkennung innerhalb der Tabelle und Optical Character Recognition (dt. optische Zeichenerkennung) (OCR) des Texts in der Tabelle verwendet werden [40]. Bei Transkribus werden so beispielsweise Tabellenstrukturerkennung, Baselineerkennung und Texterkennung als separate Schritte ausgeführt [44]. Aber die dort angebotene Tabellenstrukturerkennung ist für die Anwendung auf komplexen Tabellenstrukturen wie denen im BonnTable-Datensatz (2.4.1) [64] nicht ausreichend.

Auch die Verwendung von OCR stellt für historische Dokumente eine besondere Herausforderung dar, da diese oft strukturell uneinheitlicher als moderne Dokumente sind (z.B. überlappender Text). Außerdem werden je nach Epoche und Herkunft verschiedene von Englisch und vom zeitgenössischen equivalent abweichende Sprachen (und im Falle von Deutsch beispielsweise unübliche Sonderzeichen) verwendet, wodurch herkömliche kommerzielle OCR-Lösungen schlechtere Ergebnisse als auf vergleichbaren zeitgenössischen englischen Textausschnitten liefern [2]. Hinzu kommt, dass aufgrund des Alters und des Annotationsaufwands historischer Dokumente nur eine vergleichsweise geringe Menge von Trainingsdokumenten verfügbar ist.

Die Digitalisierung von historischen Dokumenten ist zu Preservierungszwecken wichtig, so berichtet der Fototechnische Ausschuss der KLA, dass „viele Archive mittelfristig 5-10% ihrer Bestände digitalisieren wollen“[28]. Für die historische Forschung, beispielsweise in

der Wirtschaftsgeschichte, ist es dabei wichtig, dass diese Digitalisierung auch Transkription und Strukturerkennung von historischen Dokumenten beinhaltet, um das Arbeiten mit Big Data (z.B. Preisentwicklung von Gütern über Jahrzehnte) zu ermöglichen [64].

Zu ähnlichen Problemen kommt es auch bei der Transkription von modernen Tabellen im Real-Life-Context aus (Hand)Kameraaufnahmen. Dieser Anwendungsfall ist im Alltag, beispielsweise beim Abfotographieren einer handgeschriebenen Tabelle von der Tafel während einer Präsentation, relevant. Hier sind die Strukturerkennung und OCR aufgrund von Verdeckungen und Verzerrungen im Bild sowie unvorhersehbaren weiteren Bildelementen oft schwierig [47].

Durch den Trainingsaufwand und die Verwendung von kaskadiertem OCR wird die Dokumentverarbeitungspipeline oft rechnerisch sehr kostenintensiv und inflexibel sowohl in Bezug auf unterstützte Sprachen als auch auf Dokumenttypen [31].
OCR-freie, multimodale Ende-zu-Ende-Transformer stellen im generellen Dokumentenverarbeitungskontext eine mögliche Lösung für viele der OCR-bedingten Probleme dar, da sie ohne Anwendung von externer OCR oder mehrschrittigen Pipelines Strukturerkennung und Trankribierung durchführen können [31].

In letzter Zeit wurden transformerbasierte Ansätze erfolgreich zur Tabellenstrukturerkennung auf modernen digitalen Dokumenten angewendet [49]. So erreichte Tabletransformer [51] sehr gute Ergebnisse auf PubTables-1M [51], einem Tabellendatensatz bestehend aus digital erzeugten Papers.
In der folgenden Arbeit soll betrachtet werden, ob transformerbasierte Tabellenstrukturerkennung, insbesondere mit einem multimodalen Ende-zu-Ende-Transformer, auch auf schwierigen, nicht-digitalen Tabellen erfolgreich ist. Dazu wird ein solcher Ende-zu-Ende-Ansatz bezüglich der Tabellenstrukturerkennung auf zwei historischen Datensätzen und einem Real-Life-Context-Tabellendatensatz mit einem Faltungsnetz und einem transformerbasierten Ansatz zur Tabellenstrukturerkennung vergleichend bewertet.

1.1 AUFBAU

In Teil I ist die zugrundeliegende Literatur für diese Arbeit zu finden. Zuerst werden in Kapitel 2 die technischen Grundlagen von Deep Learning – besonders Transformers und CNNs – erläutert. Außerdem werden die verwendeten Datensätze und Metriken erklärt. In Kapitel 3 werden dann die verwendeten Modelle genauer vorgestellt.

In Teil II werden die durchgeführten Experimente einschließlich Ergebnisse erläutert. Zunächst werden in Kapitel 4 Implementationsdetails und das Training auf dem HPC Cluster der RWTH Aachen beschrieben. Die Experimente bezüglich der Tabellenstrukturerkennung und deren Ergebnisse sind in Kapitel 5 zu finden. Abschließend findet sich das Fazit sowie ein Ausblick in Kapitel 6.

2 GRUNDLAGEN

2.1 DEEP LEARNING GRUNDLAGEN

Im Folgenden werden die wichtigsten Deep-Learning-Konzepte, die als fundamentale Grundlagen für die Inhalte dieser Arbeit relevant sind, erläutert.

2.1.1 *Stochastischer Gradientenabstieg [25]*

Der Stochastic Gradient Descent (dt. Stochastischer Gradientenabstieg) (SGD) (siehe Algorithmus 1) ist der zugrundeliegende Optimierungs-Algorithmus der meisten Deep-Learning-Anwendungen [25] und ist eine Erweiterung des Gradientenabstiegs [25] (siehe Abbildung 1).
Die Kostenfunktion

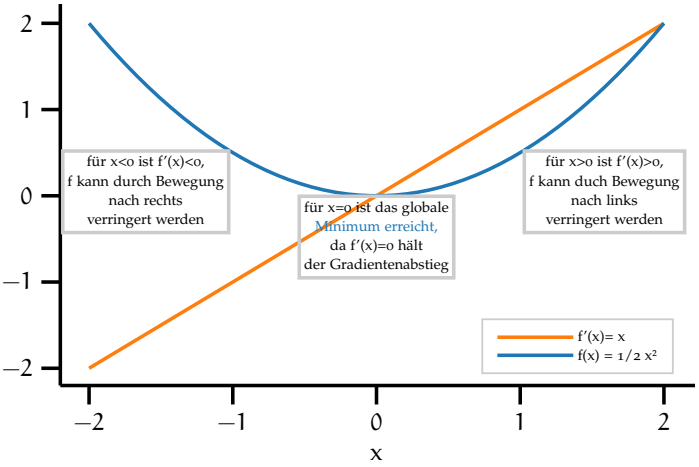


Abbildung 1: Anschauungsbeispiel Gradientenabstieg nach [25]

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} L(f(x^{(i)};\theta), y^{(i)}) \tag{1}$$

zwischen den Modellvorhersagen $f(x^{(i)};\theta)$ und den Targets $y^{(i)}$ wird mithilfe des Gradientenabstiegs minimiert (hier mit Kosten $L(f(x^{(i)};\theta))$ pro Trainingsbeispiel). Dafür wird der Gradient

$$\nabla_\theta J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta L(f(x^{(i)};\theta), y^{(i)}) \tag{2}$$

(partielle Ableitungen der Kostenfunktion L nach Komponenten θ_i von θ) berechnet. Die Kostenfunktion L kann dann durch Bewegung in Richtung des negativen Gradienten verringert werden (siehe Abbildung 1).

Algorithm 1 Stochastic Gradient Descent (dt. Stochastischer Gradientenabstieg) [25]

Require: Lernraten-Schedule $\epsilon_1, \epsilon_2, ..$
Require: Initialer Parameter θ
  $k \leftarrow 1$
  while Stop-Bedingung nicht erfüllt **do**
    Einen Minibatch mit m' Beispielen aus dem Trainingsset $\{x^{(1)}, ..., x^{(m')}\}$ mit zugehörigen Targets $y^{(i)}$ ziehen
    Gradientenschätzung berechnen:
    $\hat{g} \leftarrow \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} L(f(x^{(i)}; \theta), y^{(i)})$
    Gradientenupdate anwenden: $\theta \leftarrow \theta - \epsilon_k \hat{g}$
    $k \leftarrow k + 1$
  end while

Für große Trainingssets ist dies aber ineffektiv, da die Berechnungskosten in O(m) liegen und so mit größerer Anzahl Beispiele enorm ansteigen.

Deshalb wird beim Stochastic Gradient Descent (dt. Stochastischer Gradientenabstieg) (siehe Algorithmus 1) der Umstand genutzt, dass es sich bei dem Gradienten um einen Erwartungswert handelt, der erwartungstreu durch Berechnen des durchschnittlichen Gradienten auf einem Minibatch aus m' unabhängig und identisch verteilten (eng. independent and identically distributed) (i.i.d.) gezogenen Beispielen geschätzt werden kann.

Die Lernrate ϵ_k muss hierbei in der Praxis graduell reduziert werden, da durch das Schätzen des Gradienten konstantes Noise erzeugt wird, das auch am globalen Minimum bestehen bleibt.

2.1.2 *Adam [25] und AdamW [38]*

Adam ist ein Algorithmus mit adaptiver Lernrate, der im Gegensatz zu RMSProp Momentum durch die Schätzung des 1sten Moments direkt miteinbezieht und das Bias der Momente korrigiert [25].

Momentum ist hierbei eine Methode, das Lernen (bspw. bei kleinen aber konsistenten Gradienten) zu Beschleunigen, indem ein exponentiell verfallender gleitender Durchschnitt der vergangenen Gradienten (= Schätzung des 1sten Moments) berechnet und als Update verwendet wird. Das k-te Moment bezeichnet in der Stochastik den Erwartungswert der k-ten Potenz einer Zufallsvariable X [59], das

Algorithm 2 Adam [25] und AdamW [38] Algorithmen im Vergleich, $\odot$ bezeichnet hier das Hadamard-Produkt (also die elementweise Multiplikation).

Require: Schrittgröße ϵ
Require: Exponentielle Verfallraten ρ_1, ρ_2 für die Schätzung der Momente in $[0, 1)$
Require: Kleine Konstante δ für numerische Stabilität
Require: Initiale Parameter θ
Require: Weight Decay $w \in \mathbb{R}$
  initialisiere Variablen für das 1ste und 2te Moment mit $s = 0, r = 0$
  initialisiere Zeitschritt $t = 0$
  while Stop-Bedingung nicht erfüllt **do**
    Einen Minibatch mit m' Beispielen aus dem Trainingsset $\{x^{(1)}, ..., x^{(m')}\}$ mit zugehörigen Targets $y^{(i)}$ ziehen
    Gradientenschätzung berechnen:
    $g \leftarrow \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} L(f(x^{(i)}; \theta), y^{(i)})$
    $t \leftarrow t + 1$
    verzerrte Schätzung des 1sten Moments updaten:
    $s \leftarrow \rho_1 s + (1 - \rho_1) g$
    verzerrte Schätzung des 2ten Moments updaten:
    $r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g$
    das Bias im 1sten Moment korrigieren $\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$
    das Bias im 2ten Moment korrigieren $\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$
    Update berechnen: $\Delta\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$ (elementweise angewendet)
    Update anwenden: $\theta \leftarrow \theta + \Delta\theta - w\theta$
  end while

1ste Moment kann also hier als Erwartungswert des Gradienten gesehen werden. Beim 2ten Moment handelt es sich analog um den Erwartungswert der quadrierten Gradienten, geschätzt als exponentiell verfallender gleitender Durchschnitt der vergangenen quadrierten Gradienten.

Im Updateschritt wird dann das 1ste Moment durch die Wurzel des 2ten Moments elementweise skaliert, sodass die Lernrate für Parameter mit größeren partiellen Ableitungen der Kostenfunktion stärker abnimmt als für Parameter mit kleineren Ableitungen. Dies führt zu besserer Approximierung in Richtungen mit kleineren Gradienten. Um eine zu große Reduzierung der Lernrate durch Skalierung zu verhindern, werden exponentielle Verfallraten verwendet, sodass weiter zurückliegende Gradienten weniger stark ins Gewicht fallen und irgendwann "vergessen"werden.

AdamW ist eine Erweiterung von Adam, bei der zusätzlich Weight Decay (Regularisierung) im Updateschritt angewendet wird, was dazu dient, die Gewichte/Parameter klein zu halten, da dies zu weniger Overfitting und damit besserer Generalisierung beiträgt.

AdamW wird als Optimizer für alle in dieser Arbeit verwendeten Modelle genutzt.

2.1.3 *Faltungsschicht [25]*

Das Convolutional Neural Network (dt. Faltungsnetzwerk) (CNN) zeichnet sich durch Verwendung von Faltungsschichten aus, die eine Konvolution (auch: Faltung) zwischen der Eingabe B und einem wesentlich kleineren Kernel K ausführen. Sowohl bei der Eingabe als auch bei dem Kernel handelt es sich dabei i.A. um multidimensionale Arrays. Die Konvolution ist im 2-dimensionalen als

$$(K * B)(i,j) = \sum_m \sum_n B(i-m, j-n)K(m,n) \tag{3}$$

definiert [25]. Sie ist eng verwandt mit der Kreuzkorrelation, die im 2-dimensionalen als

$$(K \star B)(i,j) = \sum_m \sum_n B(i+m, j+n)K(m,n) \tag{4}$$

definiert ist [25].

Konvolution und Kreuzkorrelation unterscheiden sich im diskreten Fall lediglich dadurch, dass bei der Konvolution der Filterkernel im Verhältnis zum Bild geflippt ist, d.h. wenn m, n zunehmen, nimmt der Index des Bildes ab und der Index des Kernels zu (oder umgekehrt). Aufgrund dieser Eigenschaft ist die Konvolution im Gegensatz zur Kreuzkorrelation kommutativ.

Die Kreuzkorrelation wird im Kontext der Bildverarbeitung zum Template Matching genutzt, d.h. bei dem Kernel K handelt es sich um einen Bildausschnitt, der auf dem Eingabebild B gefunden werden soll [55] [53].

Die Position auf dem Eingabebild B, wo die Template am Besten mit dem Bild überlapt, ist dann $(\hat{i},\hat{j}) = \text{argmax}_{(i,j)}(K \star B)(i,j)$ [53] (wenn K und B normalisiert sind). Die Kreuzkorrelation $(K \star B)(i,j)$ kann hier auch als Skalarprodukt (6) $k^T w(i,j)$ der durch Vektorisierung von W, K entstandenen Vektoren w, k interpretiert werden [55], wobei W ein Ausschnitt aus B mit derselben Größe wie K um Koordinatenposition (i, j) ist.

Es wird also die Position gefunden, wo die Template und der von ihr abgedeckte Bildausschnitt aus B am ähnlichsten sind.

Analog dazu kann man sich auch die Konvolution in Faltungsschichten als eine Art Template Matching vorstellen. Das Modell lernt Kernels und berechnet mit Konvolution die Ähnlichkeit zwischen dem Kernel und der von ihm abgedeckten Eingabe für alle Eingabepositionen (bei Stride=1), der Kernel bewegt sich also als Sliding Window über das Bild (siehe Abbildung 2). Die so entstandene Ausgabe nennt sich Feature Map (dt. Merkmalskarte).

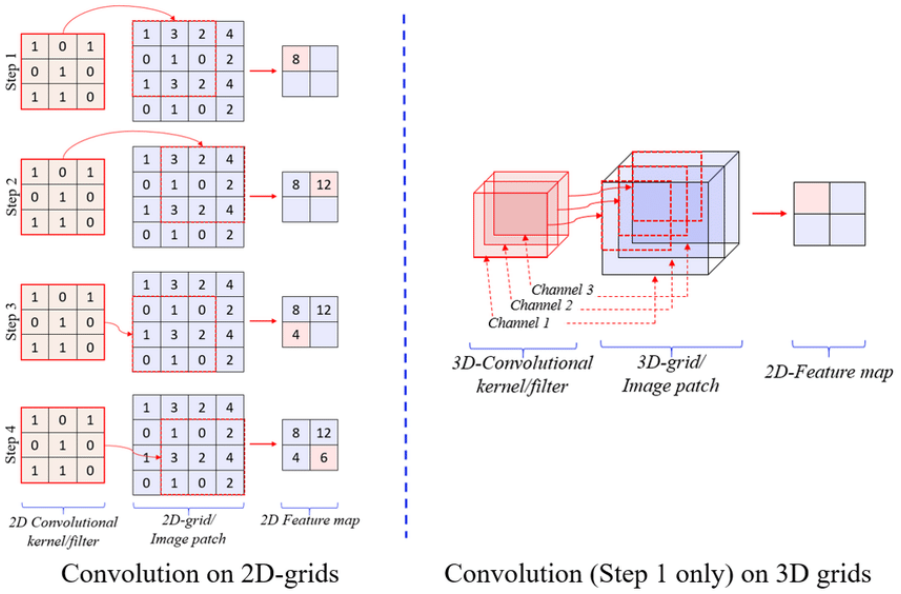


Abbildung 2: Faltung in 2D und 3D, Darstellung aus [5]

Da die Elemente des Kernels in diesem Kontext von dem Modell gelernt werden, ist es für die praktische Implementation unerheblich, ob die Konvolution oder die Kreuzkorrelation implementiert wird (die gelernten Kernels sind dann lediglich geflippt) [25]. Deshalb wird zumeist die konzeptuell einfachere Kreuzkorrelation implementiert.

Im realen Anwendungsfall hat die Eingabe einer Faltungsschicht meist die Dimensionen (N, H, W, C_{in}) (=Batchgröße, Höhe, Breite, Anzahl Channel/Featuremaps (Tiefe)) [14] und in der Faltungsschicht werden C_{out} dreidimensionale Kernels (H_K, W_K, C_{in}) auf die Eingabe angewendet [29] (siehe Abbildung 2). Die Ausgabe sind dann C_{out} Feature Maps, deren Größe von der Eingabehöhe und -breite und den Parametern der Faltung (Stride, Padding, Dilation) abhängt [14] (Breite analog)

$$H_{out} = \frac{H_{in} + 2\text{padding}[0] - \text{dilation}[0](\text{Kernel Size}[0] - 1) - 1}{\text{stride}[0]} + 1. \tag{5}$$

2.1.4 *ResNet [26]*

Beim Residual Network (ResNet) handelt es sich um ein tiefes Faltungsnetz, welches oft als Backbone-Netzwerk für andere Neuronale Netze verwendet wird. Um das Problem der Degradation – also das Abnehmen der Test- und Trainnigsgenauigkeit mit zunehmender Netzwerktiefe – zu lösen, optimiert das ResNet die sogenannte Residualfunktion anstatt der eigentlich zu optimierenden Funktionen.

Die zugrundeliegende Annahme ist, dass die Genauigkeiten eines tiefen Faltungsnetzes (bei sonst identischem Aufbau) mindestens ge-

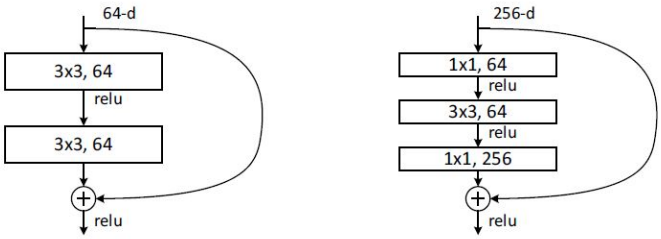


Abbildung 3: Residualblöcke, Darstellung aus [26]

nauso gut wie die eines flacheren Netzwerks sein sollte, da die zusätzlichen Schichten dann einfach eine Identitätsfunktion abbilden würden. Das Degradationsproblem ist deshalb ein Zeichen dafür, dass die Approximation der Identitätsfunktion in Faltungsnetzwerken nicht trivial ist. Die Residualfunktion vereinfacht diese wesentlich, da dadurch, anstatt direkt die Funktion $H(x)$ zu approximieren, stattdessen die Residualfunktion $F(x) = H(x) - x$ approximiert und dann die Identität x addiert wird (siehe Abbildung 3). Damit gilt $H(x) = F(x) + x$, sodass die Gewichte sich nun lediglich null annähren müssen, um eine Identitätsfunktion zu approximieren.

Da der Aufbau des ResNet bereits in meinem Projektgruppenbericht beschrieben wurde, sind genauere Details im Anhang A.2 zu finden.

2.2 TRANSFORMERGRUNDLAGEN

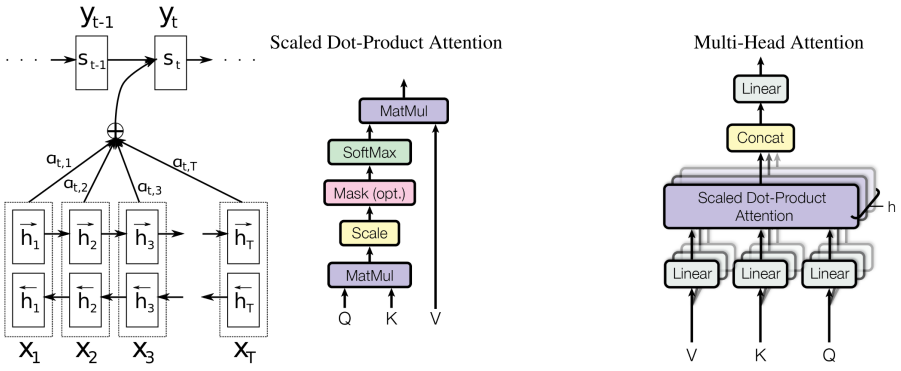
2.2.1 *Attention und der standard Encoder-Decoder-Transformer*

Transformer sind aus dem Modellieren von Sprachsequenzen hervorgegangen. Vor Transformern wurde das Recurrent Neural Network (dt. rückgekoppeltes neuronales Netz) (RNN) für Sequenzmodellierungs- und Transductionsaufgaben wie Übersetzungen verwendet [7].

In diesem Zusammenhang führte 2016 [7] Additive Attention ein (siehe Abbildung 4 (a)).

Das Prinzip von Attention kann im Allgemeinen so beschrieben werden, dass das Modell lernt, welcher Teil einer Eingabesequenz für den jeweiligen Punkt der Ausgabesequenz relevant ist. Die Ausgabe ist dann eine Abbildung von einer Query der Ausgabesequenz und Key-Value-Paaren der Eingabesequenz inform einer mit der Ähnlichkeit der Query und Keys gewichteten Summe der korrespondierenden Values.

Bahdanau et al. setzen dies als ein bidirektionales RNN um, wo die Sequenz von einem Forward- und einem Backward-RNN verarbeitet wird, sodass zwei Sequenzen von Hidden States entstehen, die jeweils vom vorhergehenden Element (in der jeweiligen Richtung) und



(a) Additive Attention in [7]

(b) Dot Product Attention in [57]

Abbildung 4: Attention in [57] und [7] im Vergleich

dem vorhergehenden Hidden State abhängen. Für jedes Element der Eingabesequenz werden die zugehörigen Forward- und Backward-Hidden States verknüpft, dies wird Annotation genannt. Da RNNs auf neuere Inputs fokusiert sind, ist durch die Annotation der gesamte Kontext eines Wortes mit Fokus auf das umliegende Gebiet gegeben. Für jedes Target in der Ausgabesequenz wird aus den Annotationen ein Kontextvektor c als gewichtete Summe des Zusammenhangs zwischen dem Target und den Elementen der Eingabesequenz mit einem trainierbaren Feed Forward Netzwerk berechnet. Der Kontextvektor kann als erwartete Annotation über alle Annotationen gesehen werden. Er wird dann als Eingabe zur Berechnung der Wortwahrscheinlichkeit (mit dem Hidden State des RNN und der vorhergehenden Ausgabe) verwendet. Dies konstituiert also einen additiven Attention-Mechanismus mit c als Ausgabe und den Annotationen als Values, die mit dem Zusammenhang zwischen Ein- und Ausgabe gewertet werden.

[57] präsentiert 2017 das Transformer-Modell, was allein mit Scaled-Dot-Product-Attention funktioniert und ohne Faltungsschichten oder RNN-Elemente auskommt.

Scaled Dot-Product-Attention nutzt anstatt eines Feed-Forward-Networks das Skalarprodukt zur Bestimmung der Ähnlichkeit zwischen Keys und Queries und ist daher schneller und platzeffizienter sowie parallelisierbar, da dieses effizient durch Matrixmultiplikationen umgesetzt werden kann. Das Skalarprodukt zweier Vektoren x, y ist als

$$x\dot{y} = |x||y|\cos(x, y) = x^T y \tag{6}$$

definiert [1] [25], ein Element $C_{i,j}$ einer Matrix $C = AB$ kann also als Skalarprodukt zwischen der i-ten Zeile von A und der jten Spalte von B aufgefasst werden [25].

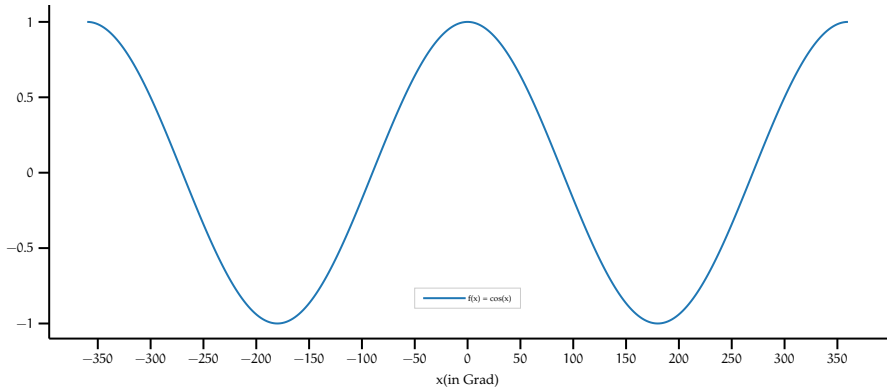


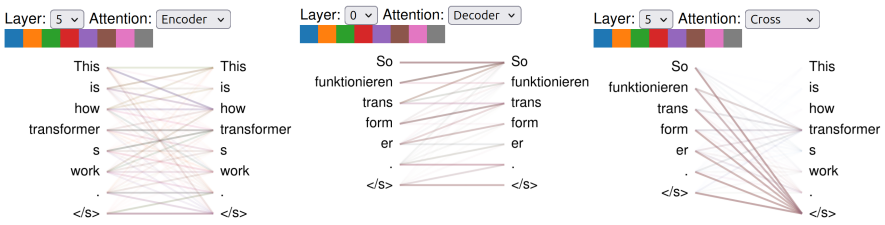
Abbildung 5: Cosinusfunktion

Die Eigenschaft des Skalarprodukts als Ähnlichkeitsmaß ergibt sich aus dem Cosinusterm. Wie in Abbildung 5 ersichtlich, wird die Cosinusfunktion genau dann -1, wenn der Winkel zwischen den Vektoren 180° beträgt, genau dann 1, wenn der Winkel entweder 0° oder 360° ist und 0, wenn die Vektoren orthogonal (90°) zueinander stehen. Für zwei Vektoren gegebener Länge ist das Skalarprodukt also dann maximal, wenn sie in dieselbe Richtung zeigen, und minimal, wenn sie in die entgegengesetzte Richtung zeigen [1].

Die d_{model}-dimensionalen Eingaben (Queries, Keys, Values) der Dot-Product-Attention (siehe Abbildung 4 (b)) werden mit lernbaren linearen Transformationen auf d_v (Queries) bzw. d_k (Keys, Values) Dimensionen projiziert. Da die Matrixmultiplikation als lineare Transformation eines Vektor-Sets gesehen werden kann [9], wird dies durch die Matrixmultiplikation der jeweiligen Eingabe-Embedding-Matrix mit lernbaren Gewichtsmatrizen (je eine Matrix für Key, Query, Value) implementiert [3]. Das Skalarprodukt zwischen Queries und Keys wird ebenfalls mithilfe einer Matrixmultiplikation zwischen den dadurch enstehenden Query- und Key-Matrizen implementiert. Das Ergebnis des Skalarprodukts wird skaliert, um zu große Werte und daraus resultierende Instabilität zu verhindern, und dann an Softmax gegeben, sodass sich die Relationswerte zwischen einer Query und den verschiedenen Keys zu 1 aufsummieren. Mit diesen Werten werden dann die Values gewichtet, was ebenfalls durch eine Matrixmultiplikation zwischen der Softmax-Ausgabematrix und der Value-Matrix implementiert wird, wodurch eine d_v-dimensionale Ausgabe entsteht.

Transformer nutzen Multi-Head-Attention, d.h. die Keys, Values und Queries werden pro Head mit anderen gelernten linearen Projektionen transformiert. Die Ergebnisse der verschiedenen Heads werden dann verknüpft und durch eine weitere lineare Transformation auf die Eingabe-Dimension d_{model} projiziert [3].

Die Multi-Head-Attention wird im Encoder als Self-Attention-Sublayer genutzt (Abbildung 6 (a)), d.h. sowohl die Queries als auch die Key-



(a) Encoder Self-Attention (b) Decoder Self-Attention (c) Decoder Cross-Attention

Abbildung 6: Visualisierung von Self-Attention and Cross-Attention mit bertviz [58] (das genutzte Modell ist Helsinki NLP/opus-mt-en-de [54])

Value-Paare sind Projektionen der Eingabe-Sequenz, es wird die Relation einer Position zu den anderen Positionen in der Sequenz bestimmt.

Im Decoder wird Masked-Self-Attention genutzt (Abbildung 6 (b)), wo im Gegensatz zu der Self-Attention im Encoder jede Position der Ausgabesequenz nur zu den vorhergehenden Positionen attended und nicht zur gesamten Sequenz. Dadurch wird in Kombination mit dem Offsets des Outputs um eine Position sichergestellt, dass die Modellvorhersage ausschließlich von zur Inferenzzeit bekannten Vorhersagen abhängt [57]. Ausserdem werden im Decoder Cross-Attention-Sublayer verwendet, bei denen die Key-Value-Paare aus dem Encoder kommen und von den Queries von Ausgabesequenzpositionen attended werden (Abbildung 6 (c)). Hier wird also bestimmt, welche Teile der Eingabesequenz für eine Position der Ausgabesequenz am relevantesten sind.

Insgesamt besteht ein Encoder-Layer (dt. Schicht) jeweils aus einem Self-Attention-Sublayer gefolgt von einem Positionsweisen Feed-Forward-Netz, das unabhängig und identisch auf jede Position angewandt wird (siehe Abbildung 7). Beide Sublayer nutzen Residual Connections (siehe 2.1.4).

Ein Decoder-Layer besteht aus einem Masked-Self-Attention-Sublayer, gefolgt von einem Cross-Attention-Sublayer und einem Positionsweisen Feed-Forward-Netz (siehe Abbildung 2.1.4). Auch hier werden Residual Connections angewendet. Darauf folgt noch eine finale lineare Schicht, die den Ausgabevektor des Decoders auf die Größe des Ausgabevokabulars projiziert. Durch das finale Softmax wird dies in eine Wahrscheinlichkeit für jedes Wort im Vokabular umgewandelt [3].

Der Decoder ist autoregressiv bei der Inferenz, d.h. bei der Generierung von Sequenzen wird das Element, das am wahrscheinlichsten auf die bisherige Ausgabesequenz (anfangs leer) folgt, vorhergesagt (als Wahrscheinlichkeitsverteilung über dem Vokabular), und wird

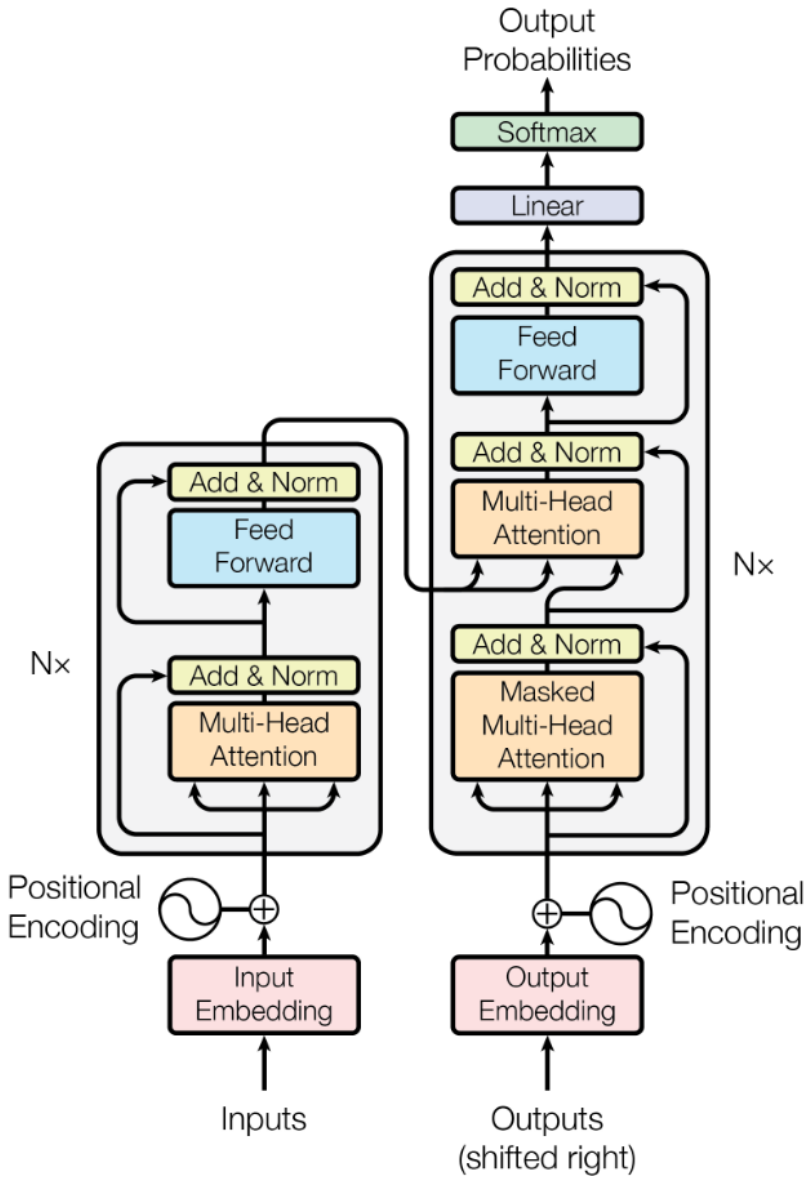


Abbildung 7: Standard-Transformerarchitektur aus [57]

dann im nächsten Schritt an die Ausgabesequenz angehängt [57]. Das Training ist i.A. aber nicht autoregressiv, anstatt die Vorhersagen der vorherigen Schritte als Eingabe zu Nutzen, wird stattdessen die (masked) Ground Truth verwendet (=Teacher Forcing) [30].

Wegen der vielseitigen Einsetzungsmöglichkeiten und Flexibilität der Attention-Layers sind Transfomer in Natural Language Processing Tasks und darüber hinaus weit verbreitet. Beispiele für den Einsatz von Transformern stellen generative vortrainierte Transformer (eng. Generative Pre-trained Transformers) (GPT) dar.

2.2.2 *Visual Transformer Encoders*

Um das Prinzip der Attention auf Bilder anzuwenden, gibt es zwei wesentliche Architekturen: **ViT** und **Swin**.

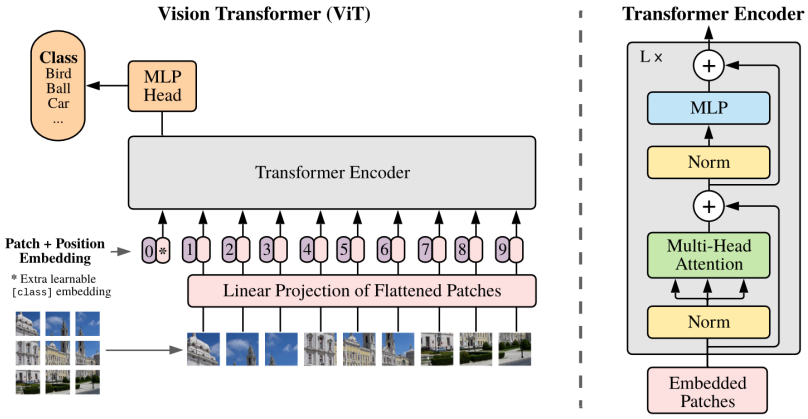


Abbildung 8: ViT [17]

ViT (Abbildung 8) zeichnet sich durch die Methode zum Umwandeln eines Bildes in die vom Transformer erwartete Sequenz von 1-dim. Token-Embeddings aus.

Zuerst werden die Eingabebilder in eine Sequenz von abgeflachten 2d-Patches mit Dimension $Nx(P^2C)$ umgeformt, wobei die Input-Patchgröße P im Paper 16 ist. C bezeichnet die Channels und N die Anzahl der Patches in der Sequenz (abhängig von Bild- und Patchgröße). Im Gegensatz zu **SWIN!** (2.2.2) ist ViT also nicht hierarchisch.

Da die latenten (verborgenen) Vektoren eine konstante Größe D haben, werden die Patches dann mit einem trainierbaren linearen Embedding auf D Dimensionen projiziert.

Dann werden lernbare 1-dimensionale Positions-Embeddings angehängt (und lernbare Embeddings zur Klasifizierung vorangehängt).

Als Encoder wird der Standard-Transformer-Encoder aus [57] verwendet (mit MLP-Head zur Klassifizierung).

Der Standard-ViT skaliert die Eingabebilder vor dem Umfromen in Patches auf eine vordefinierte Eingabegröße, wobei das Seitenverhältnis nicht beibehalten wird. Pix2Struct (2.3.2) schlägt eine Abänderung vor, die variable Eingabegrößen möglich macht.

Der **Shifted Window (Swin) Transformer** [37] (Abbildung 9) ist ein hierarchischer Vision Transformer, der sich durch die Verwendung von Shifted Window-basierter Self-Attention in den Transformer-Blöcken auszeichnet. Dafür wird das Eingabebild in nicht-überlappende Patches und Fenster, die jeweils aus mehreren Patches bestehen, aufgeteilt, wobei Self-Attention nur für die Patches innerhalb des zugehörigen

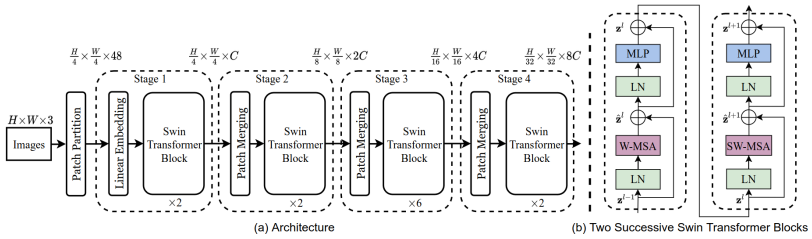


Abbildung 9: Swin [37]

Fensters berechnet wird. Um den Zusammenhang zwischen den nicht-überlappenden Fenstern herzustellen, werden für zwei aufeinanderfolgende Swin-Blöcke die Fenster im zweiten Block im Vergleich zum ersten Block um die halbe Fenstergröße verschoben (sodass nun andere Patches in den jeweiligen Fenstern attended werden).

Der Swin Transformer nutzt Patch-merging Layers, die jeweils eine 2x2-Nachbarschaft von Patches zusammenfassen(2x Downsampling der Auflösung), sodass eine hierarchische Repräsentation mit denselben Feature-Map-Auflösungen wie bei Backbone-CNNs wie ResNet (2.1.4) entsteht.

Da die Self-Attention nur lokal im jeweiligen Fenster und nicht wie bei ViT global berechnet wird, liegt die Komplexität bei Swin linear in der Bildgröße, während ViT quadratische Komplexität hat.

2.3 RELATED WORK

2.3.1 *Historische Layouterkennung und Dokumentverarbeitung*

In der bestehenden Literatur zur Historischen Dokumentverarbeitung kommt es zumeist zu einer Trennung von Layouterkennung und Texterkennung. Der Fokus der wissenschaftlichen Arbeiten liegt i.A. auf der Layouterkennung, während die Texterkennung mithilfe gängiger Engines für OCR wie Tesseract erfolgt. [1] **Global Surface Air Temperature (GloSAT)** verwendet das CascadeTabNet (2.3.3) zum Training auf dem GloSAT Datensatz (2.4.2), der aus historischen Tabellen besteht. Zusätzlich wird im GloSAT-Paper eine Postprocessessing-Methode für Tabellen vorgestellt (2.5.5).

ICDAR2019_cTDaR ist ein Datensatz zur Tabellenerkennung und Tabellenstrukturerkennung, der eine Mischung aus historischen und digialen Tabellen enthält [49].

Weitere Arbeiten zu historischer Dokumentverarbeitung sind im Anhang A.3 zu finden.

1 https://github.com/tesseract-ocr/tesseract

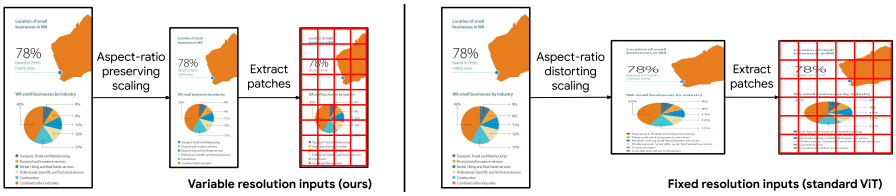


Abbildung 10: Vergleich der Eingabebild-Patches bei ViT (2.2.2) und Pix2Struct 2.3.2 (Darstellung aus [32])

2.3.2 *OCR-freie Ende-zu-Ende Dokumentverarbeitung*

Kürzlich haben mehrere Paper die Realisierbarkeit von multimodaler, Transformer-basierter, OCR-freier Ende-zu-Ende Dokumentverarbeitung mit kompetitiven Ergebnissen unter Beweis gestellt. Hier wird keinerlei Input von OCR-Engines wie Tesseract benötigt, da der Text von dem Modell implizit gelernt wird. Exemplarisch wird an dieser Stelle lediglich Pix2Struct vorgestellt, da es als Kontext für das in dieser Arbeit verwendete Kosmos-2.5 (3.2) relevant ist. Weitere Modelle und Ergänzungen zu Pix2Struct sind im Anhang (A.4) zu finden.

Pix2Struct [32] hat das Ziel, ein einziges vortrainiertes Ende-zu-Ende Bild-zu-Text Transformer-Modell zur Vefügung zu stellen, dass für eine Vielzahl von Aufgaben finetuned werden kann.

Als Eingabe enthält das Modell ein Bild (raw pixels), und gibt dann ähnlich wie Donut (A.4) Text in Form von Tokensequenzen aus.

Das Modell selbst ist ein weitgehend standardmäßiges Bild-Encoder-Text-Decoder-Modell auf Basis von ViT, welches aber so abgeändert wird, dass variable Eingabeauflösungen möglich sind (siehe Abbildung 10). Das Eingabebild wird dafür immer so skaliert, dass die maximale Anzahl von Bild-Ausschnitten einer vorgegebenen Größe extrahiert werden können, die in eine gegebene Sequenzlänge passen, wobei das ursprüngliche Seitenverhältnis beibehalten wird. Es werden 2-dimensionale absolute Positional-Embeddings für die Patches verwendet, um die Kompatibilität mit verschiedenen Auflösungen und Seiteverhältnissen sicherzustellen.

Das Modell wird dadurch im Vergleich zum ursprünglichen ViT, welches das Eingabebild immer auf eine vorgegebene Größe skaliert, robuster gegenüber variierenden Seitenverhältnissen und ermöglicht On-the-Fly-Anpassungen der Auflösung und der Sequenzlänge.

2.3.3 *Tabellenstrukturerkennung*

Im Feld der Tabellenstrukturerkennung existieren eine Vielzahl von Datensätzen und Modellarchitekturen. Hier werden lediglich die Modelle vorgestellt, die als Kontext für die im Experimentalteil verwen-

deten Datensätze relevant sind. Ein weiteres Modell und ein weiterer Datensatz sind im Anhang zu finden (A.5).

CascadeTabNet [42] ist das Faltungsnetz, welches im in der GloSAT-Veröffentlichung für Tabellen- und Tabellenstrukturerkennung genutzt wurde. Es handelt sich hier um einen Ende-zu-Ende-Ansatz, bei dem Tabellen- und Tabellenstrukturerkennung gleichzeitig durchgeführt werden. Die Modellarchitektur kombiniert die Architektur des Cascade Region-based Convolutional Neural Network (R-CNN) mit einem High-Resolution Net (HRNet)-Backbone, da beide Modelle für genaue Objekterkennung entwickelt wurden. ResNet50 (2.1.4) wird ebenfalls als Backbone genutzt, um die Feature Map aus dem Bild zu extrahieren.

Bei **Cycle-CenterNet** handelt es sich um eine Erweiterung von CenterNet [18], die im Paper zum WTW-Datensatz [47] vorgestellt wurde. CenterNet repräsentiert BoundingBoxen als ein Keypoint in der Mitte der BoundingBox und ein Eckpunkte-Paar (links oben und rechts unten) und stellt eine Erweiterung von CornerNet dar (welches BoundingBoxen als zwei Keypoints linke obere und rechte untere Ecke repräsentiert). Cycle-CenterNet erweitert CenterNet durch Hinzufügen eines Cycle-Pairing-Moduls mit einem korrespondierenden Pairing-Loss, um die Beziehung zwischen dem Eckpunkten und den Mittelpunkten zu lernen. Mithilfe dieser Information können die nach dem Prinzip von CenterNet gefundenen Zellen verbunden und Zeilen und Spalten extrahiert werden. Da dieses Modell von dem Unternehmen Alibaba verwendet wird, wurde der Modellcode nicht veröffentlicht.

2.4 GENUTZTE DATENSÄTZE

2.4.1 *Bonner Tabellendatensaz*

Anmerkung: Als Teil meiner Projektgruppe war ich an der Annotation dieses Datensatzes beteiligt. Dieser Abschnitt zum Bonner Tabellendatensaz beruht auf dem gleichnahmigen Abschnitt in meinem Projektgruppenbericht [64].

Der Bonner Tabellendatensatz (hier: BonnData) besteht aus 616 Scans von preußischen Immediatzeitungsseiten mit Tabellen aus dem Rheinland um 1820. Die Tabellenstruktur ist fein (Segmentation einzelner Zellen) annotiert (Annotationsbeispiel siehe u.a. Abbildung 17), die umliegenden Textregionen sind teilweise markiert. Der Text selbst ist aber (auch innerhalb der Tabellen) nicht transkribiert.

Es sind überwiegend handschriftliche (514) aber auch einige gedruckte (101) Scans mit Tabellen in verschiedenen Formaten vorhanden, aufgrund derer die Scans in 5 Kategorien eingeteilt wurden (siehe Tabelle 1). Auf einigen Scans sind mehrere (bis zu 4) Tabellen vorhanden, oft handelt es sich bei einem Scan um eine Doppelseite des Ursprungsdokuments.

Tabelle 1: Kategorisierung der Bilder des Bonner Tabellendatensatzes

KATEGORIE	BEDEUTUNG	ANZAHL
0	Tabelle	157
1	keine Tabelle	105
2	minimale Tabelle	104
3	Wort oder Satz, der für mehrere Zeilen-/Spalten gilt, ist quer über die Zeilen/-Spalten geschrieben	25
4	Tabelle zum Zustand der öffentlichen Kassen	71
undefiniert	nicht endgültig zugeordnet	155

2.4.2 *GloSAT [40]*

Anmerkung: Dieser Abschnitt zum GloSAT-Datensatz beruht auf dem gleichnahmigen Abschnitt in meinem Projektgruppenbericht [64].

Global Surface Air Temperature (GloSAT) ist ein Datensatz bestehen aus 500 Bildern von sowohl handgeschriebenen als auch gedruckten historischen Tabellen mit meteorologischen Daten.

Es sind sowohl grobe als auch feine Segmentationen der Tabellenzellen vorhanden (siehe Abbildung 23), und Kopfzeile (Header) und Wertebereich der Tabelle (Table Body) sind markiert.

Es handelt sich zumeist um Einzelseiten, auf denen oft mehrere Tabellen vorhanden sind.

Der Text ist weder innerhalb noch außerhalb der Tabellen transkribiert und Textfelder sind nicht annotiert.

2.4.3 *Wired Table in the Wild [47]*

Bei Wired Table in the Wild (WTW) handelt es sich um einen Datensatz von annotierten modernen Tabellen aus dem Real-Life-Kontext, wie bspw. Photos von Nahrungsmitteletiketten oder Whiteboards mit Tabellen sowie Bilder von Websites. Es liegen 10970 Trainingsbeispiele und 3611 Testbeispiele vor, die in 7 Kategorien aufgeteilt sind (Beispielbilder siehe Abbildung 24, occblu=occluded or blurred).

Da die Bilder der Tabellen im Real-Life-Kontext aufgenommen wurden, ist die Tabellenstruktur – beispielsweise aufgrund von Verdeckungen und Verformungen – unregelmäßig und schwer zu erkennen. Aufgrund der Vielfalt der aufgenommenen Tabellen ist deren Layout sehr uneinheitlich. Text außerhalb der Tabelle sowie mehrere Tabellen pro Bild kommen im Datensatz vor. Der Text ist grundsätzlich nicht annotiert.

Der Datensatz beschränkt sich bewusst auf „wired “Tables, also Tabellen mit Trennlinien, da Tabellen ohne Trennlinien keine klaren

Richtlinien zur Gruppierung geben und somit schwer zu annotieren sind [47].

Die GroundTruth liegt sowohl als achsenparallele (anhand xmin, ymin, xmax, ymax definierte) als auch als nicht-achsenparallele (über die Koordinaten aller vier Ecken definierte) BoundingBox vor. Für die GroundTruth-Annotationen des Test-Datensatzes existieren zwei Versionen, in dieser Arbeit wird immer die neuere aktualisierte Version zur Evaluation verwendet.

2.5 METRIKEN UND METHODEN

Anmerkung: Dieser Abschnitt zu Metriken und Methoden beruht in Teilen auf meinem Projektgruppenbericht [64].

2.5.1 *Jaccard-Koeffizient [27]*

Der Jaccard-Koeffizient ist als

$$J(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{7}$$

definiert. Er ist ein Ähnlichkeitsmaß für zwei Sets und wird als das Verhältnis von der Schnittmenge (eng. Intersection) zu der Vereinigung (eng. Union) von diesen definiert. Deshalb wird er auch als Intersection over Union (IoU) bezeichnet.

2.5.2 *Intersection over Detection [61]*

Intersection over Detection (IoD) ist ein Maß für die Ähnlichkeit zwischen zwei Sets, das dem IoU ähnelt. Er ist als

$$IoD(A, B) = \frac{|A \bigcap B|}{|B|} \tag{8}$$

definiert, wobei B die Vorhersage des Modells ist.

Der Unterschied zum IoU besteht darin, dass beim IoD das Verhältnis der Schnittmenge der Sets zu der Fläche von dem Vorhersage-Set berechnet wird.

2.5.3 *F1-Score [19]*

Bei dem F1-Score handelt es sich um das harmonische Mittel von Precision und Recall. Dieser ist definiert als

$$F_1 = 2\frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}, \tag{9}$$

Precision beschreibt dabei das Verhältnis von den richtigen Vorhersagen (True Positives) zu der Gesamtmenge der vorhergesagten Elemente.

Recall beschreibt dagegen das Verhältnis der richtig vorhergesagten Elemente (True Positives) zu der Gesamtmenge der Ground-Truth-Elemente (True Positives und False Negatives).

2.5.4 *Gewichteter F1-Score [40]*

Der Weighted Average F1 Score (dt. Gewichteter F1-Score) (Wf1) ist definiert als

$$WF_1 = \frac{\sum_{i=1}^{n} F1@IoU_i \cdot IoU_i}{\sum_{i=1}^{n} IoU_i}, \tag{10}$$

wobei $F1@IoU_i$ der F1-Wert am i-ten IoU-Threshold und n die Länge der Liste von IoU-Thresholds ist, über die der Weighted F1-Score berechnet werden soll.

Der mittlere F1-Wert wird immer mit dem zugehörigen IoU-Threshold gewichtet, sodass gute F1-Werte bei höheren IoU-Thresholds stärker zu dem Gesamtwert beitragen.

Das GloSAT-Paper berechnet den Weighted F1-Score über die IoU-Thresholds {0.6, 0.7, 0.8, 0.9}.

2.5.5 *Postprocessing mit DBScan [40]*

Das DBScan-Postprocessing von GloSAT erwartet als Eingabe eine Liste von Zellen sowie die sie umgebende Tabellenregion.

Die Zellenkoordianten werden mithilfe der Tabellenkoordinaten normalisiert und die Zellen werden dann seperat in X- und Y-Richtung mit 1D-DBScan geclustered. Für jeden Cluster wird dann der Durchschnittswert ermittelt, Outliers (dt. Ausreißer) werden wahlweise entweder aussortiert oder angehängt.

Die so entstandenen Koordinaten werden dann mit den Tabellenkoordinaten wieder zurücktransformiert und als Zeilen- bzw. Spaltenkoordinaten verwendet.

3 NETZWERKARCHITEKTUR

3.1 FASTER R-CNN [45] [22]

Das Faster R-CNN ist ein regionsbasiertes Faltungsnetz zur Objekterkennung. Es ist eine Kombination von einem Region Proposal Network (dt. Regionsvorschlagsnetzwerk) (RPN) und von Fast-R-CNN als

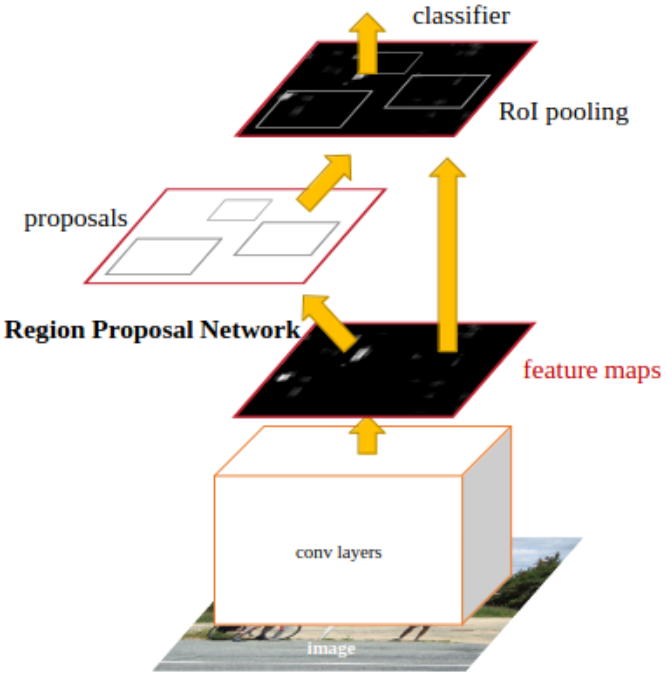


Abbildung 11: Aufbau des Faster R-CNN, Darstellung aus [45]

Detektionsnetzwerk, sodass keine externe Methode zum Finden von Kandidatenregionen benötigt wird. Das RPN is dabei ein vollständiges Faltungsnetz und beide Netze teilen sich Faltungsschichten, wodurch die benötigte Zeit, um Regionsvorschläge zu generieren, reduziert wird.

In meiner Projektgruppe wurde mit diesem Modell Tabellenstrukturerkennung auf ausgeschnittenen historischen Tabellen durchgeführt. Da das Faster R-CNN deshalb schon in meinem Projektgruppenbericht beschrieben wurde, sind weitere Details im Anhang (A.7) zu finden.

3.2 KOSMOS-2.5 [39]

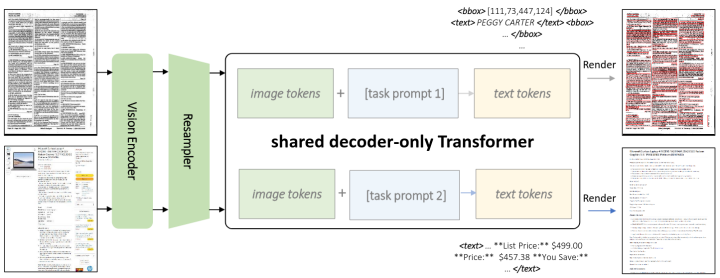
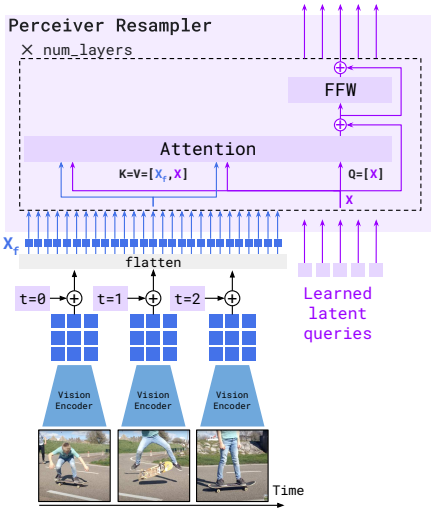


Abbildung 12: Aufbau von Kosmos-2.5, Darstellung aus [39]




```python
def perceiver_resampler(
    x_f,  # The [T, S, d] visual features (T=time, S=space)
    time_embeddings,  # The [T, 1, d] time pos embeddings.
    x,  # R learned latents of shape [R, d]
    num_layers,  # Number of layers
):
  """The Perceiver Resampler model."""

  # Add the time position embeddings and flatten.
  x_f = x_f + time_embeddings
  x_f = flatten(x_f)  # [T, S, d] -> [T * S, d]
  # Apply the Perceiver Resampler layers.
  for i in range(num_layers):
    # Attention.
    x = x + attention_i(q=x, kv=concat([x_f, x]))
    # Feed forward.
    x = x + ffw_i(x)
  return x
```

Abbildung 13: Aufbau des Perceiver-Resampler-Moduls (Darstellung aus [4])

Kosmos-2.5 [39] ist ein multimodales Decoder-Only-Modell, dass zwei verwandte Transkribierungsaufgaben für Dokumente umsetzt.

Neben der Erstellung eines strukturierten Markdown-Outputs aus einem Bild ist hier auch Text-Lokalisierung mit BoundingBoxen und simultaner Texterkennung als Output möglich.

Trainingsinformationen sind im Anhang A.8 zu finden.

3.2.1 *Aufbau*

Als Encoder für das Eingabebild wird ein auf dem ViT (siehe 2.2.2) basierender Vision Encoder verwendet, der mit den Gewichten des Encoders des Pix2Struct-large-Modells (siehe 2.3.2) initialisiert wird. Ebenfalls wird die Strategie, um variable Größen der Eingabebilder zu ermöglichen, von Pix2Struct übernommen (siehe 2.3.2).

Der Decoder basiert auf Magneto [60], einen transformerbasierten Sprach-Decoder der Sublayernorm verwendet, und wird mit dessen Gewichten initialisiert. Dabei wird die Layernormalisierung bei der Multi-Head-Attention (siehe Abbildung 7) vor der Projektion der Queries, Keys und Values und vor der Ausgabeprojektion durchgeführt. Bei Feed-Forward-Layers wird die Layernormalisierung vor der Eingabe- und vor der Ausgabeprojektion durchgeführt [60].

Encoder und Decoder werden durch ein Perceiver-Resampler-Modul [4] verbunden (siehe Abbildung 13), das die Größe der Bild-Einbettungen durch Attentive Pooling auf eine vorgegebene Größe reduziert.

3.2.1.1 *Perceiver-Resampler-Modul [4]*

Im Original-Paper wird das Perceiver-Resampler-Modul mit Videos verwendet, weshalb hier pro Frame eine gelernte zeitliche Enkodie-

rung zum Output des ViT addiert wird. Bilder werden als 1-Frame-Videos betrachet.

Es werden keine zusätzlichen räumlichen Enkodierungen addiert, da diese aus dem jeweiligen Vision Encoder vorhanden sind. Es wird die Cross-Attention zwischen gelernten latenten Query-Vektoren und der Konkatenation aus dem ViT-Output und den latenten Query-Vektoren berechnet. Die Anzahl Outputs entspricht der Anzahl Query-Vektoren, die Größe der Output-Bildeinbettung des ViT wird also auf diese Größe reduziert.

3.2.2 *Eingabe*

Die Eingabe von Kosmos-2.5 ist eine Kombination aus Text und Bild. Das Bild wird immer mithilfe des Vision Encoders eingebettet, während es sich bei dem Text entweder um reinen Markdown-Text oder um Text mit BoundingBoxen handeln kann.

Ist Ersteres der Fall, wird eine normale Text-Tokenization mit SentencePiece durchgeführt. Bei der layout-basierten Task werden BoundingBoxes und zugehörige Textgebiete wie bei Kosmos 2 modelliert (wessen Ansatz auf Pic2Seq zurückgeht). Dazu werden die kontinuierlichen BoundingBox-Koordinaten in diskrete Location Tokens umgewandelt. Höhe und Breite des Eingabebildes werden dafür je in P (Hyperparameter) Segmente unterteilt, woder PxP Bins entstehen, denen jeweils ein Location-Token als Repräsentation der Koordinaten in der Bin zugeteilt werden, welche zum Wort-Vokabular hinzugefügt werden. Die maximale Sequenzlänge L der variablen Bildauflösungsrepräsentation aus Pix2Struct (siehe 2.3.2) beschreibt auch die Maximallänge pro Bilddimension, sodass hier 2L+2 Lokalisierungs-Tokens entstehen. Die verschiedenen Datentypen werden als Groundtruth (und Eingabe) in einer Struktur verknüpft, die "hyperlink" in Markdown ähnelt und sich je nach Aufgabe unerscheiden kann.

3.3 TABLETRANSFORMER [51]

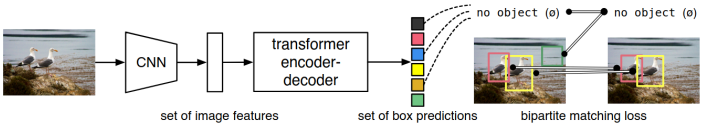


Abbildung 14: Aufbau von DETR, Darstellung aus [12]

Bei dem TableTransformer handelt es sich um einen Detection Transformer (DETR) [12] mit ResNet18-Backbone, der spezifisch auf Tabellenerkennung bzw. Tabellenstrukturerkennung trainiert wurde (Pipeline mit 2 Modellen). Zellen werden nicht direkt erkannt sondern nur implizit als Schnittmenge von Zeilen und Spalten definiert. DETR ent-

fernt viele der Aspekte eines Regionsvorschlagsnetzwerks wie Faster R-CNN (3.1), die Vorwissen über das zu lösende Problem vorraussetzen (z.B. generieren von Anchors). Trainingsdetails sind im Anhang A.9 zu finden.

3.3.1 *Aufbau*

DETR besteht aus einem Faltungsnetz (hier ResNet18 (siehe 2.1.4)) als Backbone-Netzwerk, gefolgt von einem Encoder-Decoder-Transformer. Im Gegensatz zum Standard-Transformer ist DETR nicht autoregressiv, pro Durchlauf durch den Decoder werden N Objektvorhersagen gemacht, wobei $N \in \mathbb{N}$ fest ist und wesentlich größer als die Normalanzahl Objekte pro Bild gewählt wird (100 im Paper). Um N unterschiedliche Vorhersagen zu machen, sind N unterschiedliche Eingabe-Embeddings nötig. Diese sogenannten Object Queries, sind gelernte Positional Encodings die mit 0 initialisiert werden. Die N Ausgabe-Embeddings aus dem Decoder werden getrennt voneinander an zwei geteilte Detection-Heads gegeben, ein geteiltes Feed-Forward-Netz (hier 3-Schichten-Perzeptron) zur Bounding-Box-Vorhersage, und eine lineare Schicht zur Projektion auf die Klassenlabels mit Softmax als Aktivierungsfunktion zur Klassifizierung.

Teil II

EXPERIMENTE UND DISKUSSION

4

IMPLEMENTATION UND CODEBASE

Der für diese Arbeit produzierte Code ist auf Github einsehbar [63].

Für die verschiedenen Modelle habe ich vorhandene Implementationen (siehe 5.1) verwendet. Außerdem habe ich teilweise Code aus meiner Projektgruppe [41] wiederverwendet und erweitert (Preprocessing von GloSAT und BonnData Datensätzen wurde wiederverwendet und Trainings- und Custum-Datensatz-Code wurde abgeändert). Alle Verwendungen von fremdem Code sind im Github als solche gekennzeichnet.

Der Code wurde größtenteils in Python programmiert wobei u.a. die Libraries pytorch, pandas, beautifulsoup, numpy, transformers (von huggingface) und torch-lightning Verwendung fanden.

Zum Linting und Testing wurde Nox [1] verwendet.

Das Training und die Auswertung wurde größtenteils auf dem Unieigenen GPU-Cluster „Bender “ [2] durchgeführt, in diesem Zusammenhang wurden auch eigene Bashscripts und Slurm Jobscripts geschrieben.

4.1 MULTI-NODE-TRAINING AUF HPC CLUSTER DER RWTH AACHEN

Der Tabletransformer (3.3) wurde zusätzlich auf dem WTW-Datensatz (2.4.3) auf 12 GPUs (3 Nodes) auf dem Cluster „Aix-la-Chapelle (CLAIX)“ der RWTH Aachen [3] trainiert. Dafür wurde basierend auf [4] mit Torch-Lightning ein LightningModule erstellt. Da der Cluster die Z-Shell verwendet, wurden in diesem Zusammenhang auch Z-Shellscripts geschrieben.

Wie in Abbildung 15 zu sehen ist, nimmt die Trainingsgeschwindigkeit wie erwartet mit der Anzahl von GPUs zu. Allerdings ist die Kommunikation zwischen verschiedenen Rechenknoten (eng. Node) meist zeitaufwendiger als GPU-zu-GPU-Kommunikation auf einem Rechenknoten [8], sodass die Trainingsgeschwindigkeit mit dem Training auf mehreren Rechenknoten stark abnimmt (siehe Tabelle 2). In Zukunft besteht hier also noch Optimierungspotential. Deswegen ist es hier am Effizientesten, auf einem Rechenknoten (mit 4 GPUs) zu

1 https://nox.thea.codes/en/stable/index.html
2 https://www.hpc.uni-bonn.de/en/systems/bender
3 https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/
4 https://github.com/NielsRogge/Transformers-Tutorials/blob/master/DETR/Fine_tuning_DetrForObjectDetection_on_custom_dataset_(balloon).ipynb

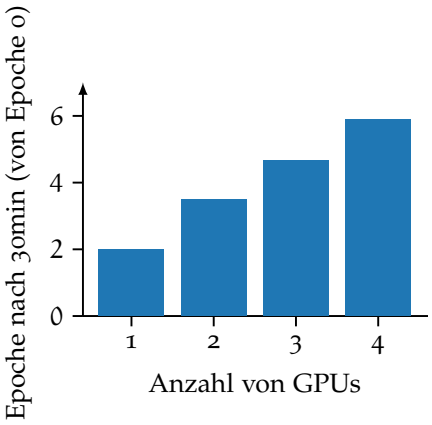


Abbildung 15: Trainingfortschritt von TableTransformer (3.3) auf Wired Table in the Wild 2.4.3 Datensatz in Abhängigkeit der Anzahl von GPUs auf einer Node (in 30min)

#Nodes	GPUs/Node	Epoch@30min
1	1	2
1	2	3,5
1	3	4,67
1	4	**5,89**
2	2	2,76
2	4	4,97
3	4	5,62

Tabelle 2: Trainingfortschritt in Epochen (von Epoche 0) von TableTransformer (3.3) auf Wired Table in the Wild 2.4.3 Datensatz in Abhängigkeit der Anzahl von Rechenknoten und GPUs (in 30min)

trainieren, da dies den besten Payoff zwischen Trainingsgeschwindigkeit und verbrauchten Resourcen hat.

5 EXPERIMENTE UND DISKUSSION

5.1 TABELLENSTRUKTUR-ERKENNUNG MIT KOSMOS-2.5, TABLETRANSFORMER UND FASTER R-CNN

Kosmos-2.5 (3.2) ist für die Anwendung auf Tabellen besonders interessant, da es als einziges der Ende-zu-Ende Transformer-Modelle (2.3.2) eine Ausgabe von (zeilenweisen) BoundingBoxen und damit eine Ortung der erkannten Strukturen auf dem Ausgangsbild ermöglicht. Auch die Anwendung von PostProcessing-Methoden auf die BoundingBoxen wird so möglich. Deshalb habe ich Kosmos-2.5 zur Evaluation der Tabellenstrukturerkennung bei Ende-zu-Ende-Transformermodellen mit OCR ausgewählt.

Da bisher nur Inferenzcode veröffentlich wurde, habe ich den gegebenen Model-Checkpoint as-is auf die historischen Bonner (2.4.1) und GloSAT Tabellendatensätze(2.4.2) sowie auf den WTW-Datensatz (2.4.3), der aus Real-Life-Bildern von deformierten und teilweise verdeckten modernen Tabellen besteht, angewandt. Hierzu habe ich ein

Tabelle 3: Ergebnisse der Tabellenstrukurerkennung (mit Thresholds {0.5,0.6,0.7,0.8,0.9} für WF1-Scores) (Evaluiert auf Vorhersagen im Tabellengebiet und nur auf Bildern mit (validen) Vorhersagen, Kosmos-2.5-Werte ohne Finetuning nach 5.1; WTW pro Subset evaluiert, Beispielbilder für Subsets siehe Abbildung 24)

dataset	subset over WTW	network	WF1 (2.5.4)		testset size	files w/o valid pred
			IoU	IoD		
BonnData (2.4.1)	-	Kosmos-2.5	0.015406	0.371667	53	17
		Faster R-CNN	**0.372770**	**0.608328**		0
		TableTransformer	0.067434	0.255319		0
GloSAT (2.4.2)	-	Kosmos-2.5	0.000827	0.634133	41	5
		Faster R-CNN	**0.556608**	**0.720681**		0
		TableTransformer	0.097294	0.297704		0
Wired Table in the Wild (WTW) (2.4.3)	Curved	Kosmos-2.5	0.024906	**0.913326**	427	19
		Faster R-CNN	**0.827490**	0.912131		0
		TableTransformer	0.265596	0.563543		0
	extremeratio	Kosmos-2.5	0.002490	0.789565	1525	76
		Faster R-CNN	**0.948594**	**0.973403**		0
		TableTransformer	0.133527	0.393499		0
	inclined	Kosmos-2.5	0.003702	0.659157	670	41
		Faster R-CNN	**0.971505**	**0.981353**		0
		TableTransformer	0.091069	0.285322		0
	muticolorandgrid	Kosmos-2.5	0.008715	0.513382	741	159
		Faster R-CNN	**0.677734**	**0.705799**		0
		TableTransformer	0.135698	0.257752		0
	occblu	Kosmos-2.5	0.005925	0.355674	72	20
		Faster R-CNN	**0.778279**	**0.838957**		0
		TableTransformer	0.061695	0.140593		0
	overlaid	Kosmos-2.5	0.001670	0.423599	75	17
		Faster R-CNN	**0.906893**	**0.931330**		1
		TableTransformer	0.117184	0.300393		0
	simple	Kosmos-2.5	0.000121	0.781224	93	9
		Faster R-CNN	**0.999540**	**1**		0
		TableTransformer	0.028958	0.071948		0

Tabelle 4: Ergebnisse von TableTransformer (3.3) mit Evaluation auf Zeilen/Spalten (Evaluiert auf Vorhersagen im Tabellengebiet und nur auf Bildern mit (validen) Vorhersagen, Kosmos-2.5-Werte ohne Finetuning wie beschrieben in 5.1)

dataset	subset over WTW	wf1		testset size	files w/o valid pred
		IoU	IoD		
BonnData (2.4.1)	-	0.291508	0.779167	53	1
GloSAT (2.4.2)	-	0.381578	0.728168	41	0
WTW (2.4.3)	Curved	0.439445	0.947788	427	0
	extremeratio	0.508365	0.981476	1525	0
	inclined	0.513688	0.902900	670	0
	muticolorandgrid	0.460649	0.798580	741	1
	occblu	0.386308	0.824607	72	0
	overlaid	0.563441	0.910857	75	1
	simple	0.614729	0.997754	93	0

Tabelle 5: Vergleich meiner Ergebnisse mit Faster R-CNN auf dem WTW Datensatz mit denen des ursprünglichen Papers [47]

Herkunft der Werte	Prec@0.9 over IoU	Rec@0.9 over IoU	F1@0.9 over IoU
Parsing Table Structures in the Wild [47]	0.721000	0.615000	0.664000
meine Reproduktion	0.854	0.607	0.710

Bashscript geschrieben geschrieben, dass den veröffentlichten Inferenzcode [1] verwendet.

Der gegebene Model-Checkpoint wurde auf das Transkribieren aller Textzeilen von Dokumenten, die teilweise Tabellen beinhalten, trainiert, die Traingsdaten für die Layout-Task bestanden dabei überwiegend (80%) aus digitalen Quellen und zu 20% aus gescannten Dokumenten des IIT-CDIP-Datensatzes, der mit Störungen versehene Scans von Dokumenten aus den 1990er Jahren mit handgeschriebenem Text, Flecken, etc. enthält [52].

Als Metrik habe ich den Weighted F1-Score (2.5.4) verwendet. Zusätzlich zu dem Weighted F1-Score über IoU (2.5.1) habe ich auch den WF1-Score über IoD (2.5.2) als gewichteten F1-Score über IoD an den IoD Thresholds berechnet. Die Thresholds wurden für beide Metriken identisch als {0.5,0.6,0.7,0.8,0.9} gewählt. Die Richtigkeit der Berechnung der IoU-Metrik wird durch mehrere Tests, die ich geschrieben und über Pytest durchgeführt habe, überprüft. Die im folgenden aufgeführten Metriken werden über einen (Unter-)Datensatz berechnet, indem die True Positives, False Positives und False Negatives für jedes Bild berechnet und über den Datensatz aufsummiert werden. Dann werden aus diesen Gesamtwerten wie in 2.5 beschrieben die Metriken berechnet. Bei GloSAT [2] und in meiner Projektgruppe ([41]) wurde die Berechnung nach demselben Prinzip durchgeführt. Weiterhin werden die Metriken auch einzeln pro Bild berechnet, dies ist in dieser Arbeit aber aufgrund der Datensatzgröße nicht aufgeführt.

Als Baseline habe ich mit Faster R-CNN [20] (3.1) ein Faltungsnetz gewählt, dass auch im WTW-Paper [47] als Baseline verwendet wurde und auf dem wir in meiner Projektgruppe [41] vielversprechende Erebnisse bezüglich Srukturerkennung auf historischen Tabellenausschnitten erreicht hatten.

Hierfür habe ich die TorchVision-Implementation [21] mit den Default-Gewichten aus einem Pretraining auf dem COCO-Datensatz [3] initialisiert und dann auf dem jeweiligen Datensatz Finetuning durchge-

1 https://github.com/microsoft/unilm/tree/master/kosmos-2.5
2 https://github.com/stuartemiddleton/glosat_table_dataset/blob/main/dla/src/eval_ICDAR_wF1.py
3 https://pytorch.org/vision/stable/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html#torchvision.models.detection.FasterRCNN_ResNet50_FPN_Weights

Tabelle 6: Ergebnisse der Tabellenstrukurerkennung auf R-CNN mit zufälliger Initialisierung der Gewichte (mit Thresholds {0.5,0.6,0.7,0.8,0.9} für WF1-Scores)(Evaluiert auf Vorhersagen im Tabellengebiet und nur auf Bildern mit (validen) Vorhersagen, Kosmos-2.5-Werte ohne Finetuning wie beschrieben in 5.1)

dataset	subset over WTW	wf1		testset size	files w/o valid pred
		IoU	IoD		
BonnData (2.4.1)	-	0.112288	0.425305	53	2
GloSAT (2.4.2)	-	0.084267	0.378818	41	0
WTW (2.4.3)	Curved	0.764097	0.882783	427	1
	extremeratio	0.927106	0.968242	1525	0
	inclined	0.962743	0.977668	670	0
	muticolorandgrid	0.669707	0.703417	741	0
	occblu	0.746128	0.822285	72	0
	overlaid	0.851398	0.898822	75	0
	simple	0.998494	0.999702	93	0

führt. Mit dieser Initialisierung (Tabelle 3) werden eindeutig bessere Ergebnisse als mit einer zufälligen Initialisierung (Tabelle 6) erreicht.

Beide Modelle erhalten beim Training achsenparallele rechteckige Zellen-BoundingBoxen (im Format (xmin,ymin,xmax,ymax)) als GroundTruth und sagen achsenparallele Zellen-BoundingBoxen in demselben Format vorher.

Als drittes und letztes Modell habe ich noch den TableTransformer (3.3) ausgewählt, da er einen transformerbasierten Ansatz für die Tabellenstrukturerkennung (ohne OCR) darstellt und auf PubTables-1M sehr gute Ergebnisse erreicht hat [51]. Er erhält beim Training ebenfalls achsenparallele rechteckige BoundingBoxen als GroundTruth, allerdings handelt es sich hier um Zeilen- und Spalten- (sowie Tabellen-) BoundingBoxen (siehe 3.3). Um die Vergleichbarkeit mit den anderen Modellen zu wahren, wird die Evaluation anhand Zellen-BoundingBoxen, die aus den Zeilen- und Spalten-BoundingBoxen als deren Schnittmengen extrahiert wurden, durchgeführt. Außerdem wird das Modell lediglich mit Zeilen-, Spalten- und Tabellenboundingboxen finegetuned, da die anderen Kategorien auf den hier betrachteten Datensätzen nicht definiert sind. Zusätzlich wird auch eine Evaluation der detektierten Zeilen- und Spalten-BoundingBoxen anhand der Groundtruth-Zeilen und -Spalten durchgeführt. Es wurde die Modellimplementation von HuggingFace [4] verwendet, und mit dem Table-Structure-Recognition-Checkpoint [5] initialisiert, der auf ausgeschnittenen Tabellen aus PubTables1M trainiert wurde.

Sowohl Faster R-CNN als auch TableTransformer wurden von mir auf dem Gesamtbild finegetuned. Die jeweiligen Train-Test-Splits wurden dabei für WTW aus dem Paper [51] bzw. im Falle von dem Bonner Tabellendatensatz und GloSAT aus meiner Projektgruppe [41] reprodu-

4 https://huggingface.co/docs/transformers/model_doc/table-transformer
5 https://huggingface.co/microsoft/table-transformer-structure-recognition

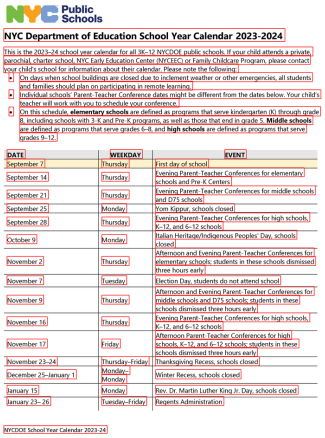
ziert, um Vergleichbarkeit mit vorherigen Ergebnissen sicherzustellen. Das Finetuning fand jeweils für 250 Epochen statt und ein Train-Valid-Split wurde zusätzlich erstellt, um Early-Stopping zu ermöglichen.

Als Optimizer wurde bei allen Modellen AdamW (2.1.2) verwendet.

Da auf einigen der Bilder in den Datensätzen auch (teilweise nicht annotierte) Textfelder ausserhalb des Tabellengebietes vorhanden sind und nicht alle Modelle auf reine Tabellenerkennung finegetuned werden konnnten, wurden nur vorhergesagte BoundingBoxen, die innerhalb eines Tabellenbereiches liegen, für die folgende Evaluation gewertet. Die erzielten Genauigkeitswerte wurden lediglich über die Bilder gemittelt, auf denen valide Vorhersagen im Tabellenbereich existierten. Hiervon abweichende Evaluationsarten sind im folgenden explizit in der Tabelle oder in dem zugehörigen Textabschnitt angegeben.

TableTransformer und Faster R-CNN sagen beide Scores für ihre BoundingBoxen voraus, nach denen gefiltert werden kann oder Non-Maxima-Suppression angewandt werden kann.. Da dies bei Kosmos-2.5 nicht der Fall ist, sind die folgenden Werte nicht gefiltert, damit die Ergebnisse vergleichbar bleiben. Die gefilterten Ergebnisse sind in 5.2.1 zu finden.

Auf dem WTW-Datensatz mussten die Ergebnisse des Papers [47] auf Faster R-CNN reproduziert werden (siehe Tabelle 5). Dies war nötig, da kein Checkpoint oder Trainingscode für dieses Modell zur Verfügung gestellt wurde. Hierfür wurde der originale Test/Train-Split verwendet und kein Early-Stopping durchgeführt, da es keinen originalen Valid-Split gibt. Ebenfalls wurden alle vorhergesagten BoundingBoxen bewertet, um Vergleichbarkeit mit den Ergebnissen aus dem Paper zu wahren. Als Vergleichsmetrik wurde wie im Paper der F1-Score über IoU am Threshold 0.9 verwendet. Da kein Evaluationscode zur Verfügung gestellt wurde, habe ich meinen eigenen Code zur Berechnung verwendet, dessen Richtigkeit ich mit einem Test (über Pytest) überprüft habe. Überraschenderweise wurden dabei teilweise bessere Werte als im originalen Paper verzeichnet. Eine mögliche Ursache dafür könnte sein, dass das im Paper verwendete Faster R-CNN auch nicht-rechteckige BoundingBoxen vorhersagen kann und vermutlich zur Berechnung des IoU verwendet, was die Vorhersagen und den IoU beeinflusst. Als GroundTruth liegt im Datensatz (2.4.3) für jede BoundingBox sowohl eine achsenparallele (anhand xmin, ymin, xmax, ymax definiert) als auch eine über die Koordinaten aller vier Ecken identifizierte nicht-achsenparallele Representation vor. Ich habe immer die rechteckige verwendet, da das von mir verwendete R-CNN achsenparallele BoundingBoxen erwartet. Außerdem wurde in [47] Post-Processing auf das Ergebnis von Faster R-CNN angewendet, um die Zellen in Tabellen zu gruppieren und Zeilen/Spalten-Informationen zu erhalten, was ebenfalls die


NYC Public Schools

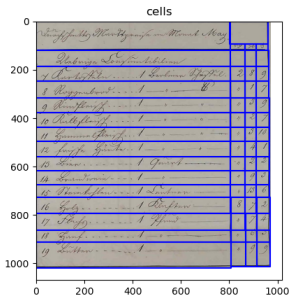
NYC Department of Education School Year Calendar 2023-2024

This is the 2023–24 school year calendar for all 3K–12 NYCDOE public schools. If your child attends a private, parochial, charter school, NYC Early Education Center (NYCEEC), or Family Childcare Program, please contact your child's school for information about their calendar. Please note the following:

- On days when school buildings are closed due to inclement weather or other emergencies, all students and families should plan on participating in remote learning.
- Individual schools' Parent-Teacher Conference dates might be different from the dates below. Your child's teacher will work with you to schedule your conference.
- On this schedule, **elementary schools** are defined as programs that serve kindergarten (K) through grade 8, including schools with 3-K and Pre-K programs, as well as those that end in grade 5. **Middle schools** are defined as programs that serve grades 6–8, and **high schools** are defined as programs that serve grades 9–12.

DATE	WEEKDAY	EVENT
September 7	Thursday	First day of school
September 14	Thursday	Evening Parent-Teacher Conferences for elementary schools and Pre-K Centers
September 21	Thursday	Evening Parent-Teacher Conferences for middle schools and D75 schools
September 25	Monday	Yom Kippur, schools closed
September 28	Thursday	Evening Parent-Teacher Conferences for high schools, K–12, and 6–12 schools
October 9	Monday	Italian Heritage/Indigenous Peoples' Day, schools closed
November 2	Thursday	Afternoon and Evening Parent-Teacher Conferences for elementary schools; students in these schools dismissed three hours early
November 7	Tuesday	Election Day; students do not attend school
November 9	Thursday	Afternoon and Evening Parent-Teacher Conferences for middle schools and D75 schools; students in these schools dismissed three hours early
November 16	Thursday	Evening Parent-Teacher Conferences for high schools, K–12, and 6–12 schools
November 17	Friday	Afternoon Parent-Teacher Conferences for high schools, K–12, and 6–12 schools; students in these schools dismissed three hours early
November 23–24	Thursday–Friday	Thanksgiving Recess, schools closed
December 25–January 1	Monday–Monday	Winter Recess, schools closed
January 15	Monday	Rev. Dr. Martin Luther King Jr. Day, schools closed
January 29–26	Tuesday–Friday	Regents Administration

NYCDOE School Year Calendar 2023-24



(a) Annotationen, auf denen Kosmos-2.5 trainiert wurde, Darstellung aus [39]

(b) Annotationen der Tabellendatensätze

Abbildung 16: Annotationsunterschiede

genauen Koordinaten der BoundingBoxen verändert. Weiterhin sind im Paper keine genaueren Angaben über das Training vorhanden, sodass auch Unterschiede in Gewichtsinitialisierung, Trainingsdauer oder Optimierung ein (Mit-)Grund sein können. Laut Paper sollen Angaben zum Training im „supplemental material“ zu finden sein, das ich aber nicht auffinden konnte. Im „supplementary material“ wurden keine Angaben zum Training gemacht. Ich habe die Evaluation um Vergleichbarkeit sicherzustellen sowohl mit der aktualisierten GroundTruth als auch mit der alten Version durchgeführt und dort für eine Genauigkeit von 3 Nachkommastellen dieselben Ergebnisse erreicht.

5.1.1 *Ergebnisdiskussion*

Es fällt sofort auf, dass die WF1-Scores über den IoU bei Kosmos-2.5 unabhängig vom Datensatz sehr schlecht (fast null) sind.

Dies ist zum Teil darauf zurückzuführen, dass Kosmos-2.5 primär auf die Erkennung von Text trainiert wurde und nicht speziell auf Tabellenzellen (siehe Abbildung 16), weshalb die von Kosmos-2.5 erkannten BoundingBoxen wesentlich enger um den Text gelegt sind als die annotierten BoundingBoxen, welche die gesamte Breite der Tabellenzellen umfassen.

Da der WF1-Score eine Gewichtung der F1-Scores an den IoU-Thresholds darstellt und überhaupt erst IoU-Werte größer gleich 0.5 als Treffer gelten, führt dies schnell zur Verschlechterung der Metrik, weil selbst eine gute Übereinstimmung mit der GroundTruth durch die Größenunterschiede nicht zwingen zu einem guten IoU führt.

Aufgrund des Trainingsfokus auf Texterkennung werden auch leere Zellen nicht erkannt.

Weiterhin werden von Kosmos-2.5 im Vergleich zu den anderen Modellen auf unverhältnismäßig vielen Bildern keine validen Vorhersagen gemacht. Dies wird in 5.1.1.1 für BonnData genauer betrachtet (siehe Abbildung 17) und hängt ebenfalls zumindest teilweise mit den Annotationsunterschieden und der As-Is-Anwendung zusammen.

Diese Probleme sind also vermutlich auf Annotationsunterschiede zurückzuführen, weshalb man wahrscheinlich durch Finetuning auf einem der Tabellendatensätze eine Verbesserung herbeiführen kann. Damit sind sie kein Anzeichen, dass das Modell für Tabellenstrukturerkennung grundsätzlich ungeeignet ist.

Die IoD-Wf1-Scores (2.5.2, 2.5.3) sind auf Kosmos-2.5 wesentlich besser, was sich dadurch erklären lässt, dass beim IoD nur durch die Fläche der Vorhersage geteilt wird, sodass die wesentlich größeren GroundTruth BoundingBoxen das Ergebnis nicht verzerren.

Trotzdem sind sie schlechter als die Vergleichswerte mit dem Faster R-CNN. Es ist zu vermuten, dass die Faster R-CNN Werte deshalb besser sind, weil das Modell auf den spezifischen Datenzsätzen trainiert wurde, sodass die genauen Strukturen besser erkannt werden können. Dies wird dadurch unterstrichen, dass bei GloSAT die Differenz zwischen den Scores von Faster R-CNN und Kosmos-2.5 wesentlich kleiner als auf dem BonnData Datensatz ist. Da GloSAT der einheitlichere Datensatz ist, ist die Übertragung von einem generell auf Dokumenten trainierten Modell besser möglich als bei dem uneinheitlicheren BonnData-Datensatz, weshalb die Differenz kleiner ist.

Während der TableTransformer leicht bessere IoU-Werte als Kosmos-2.5 erziehlt, schneidet Kosmos-2.5 bei den IoD-Werten auffallend besser ab. Dies liegt wahrscheinlich daran, dass der TableTransformer Zellen –abgesehen von den hier nicht näher betrachteten Sonderkategorien– lediglich indirekt als Schnittmenge von Zeilen und Spalten vorhersagt, was die vorhersagbaren Tabellenstrukturen einschränkt. Während die Vermutung, dass Tabellenzeilen und -spalten durchgängig sind und Zellen mit wenigen Ausnahmen als Schnittmengen von diesen existieren, für moderne digitale Tabellen oft zutreffen ist, gilt sie nicht für die historischen und/oder unregelmäßigem Tabellen, die hier betrachtet werden. Die Ergebnisse der Zeilen- und Spalten-Evaluation auf dem TableTransformer (Tabelle 4) bestätigen dies, da sie sowohl für IoU als auch für IoD wesentlich besser als die Zellen-Evaluation ausfallen, was zeigt, dass das Modell bei dem Finetuning auf dem Datensatz grundsätzlich konvergiert.

Die IoD- und IoU-WF1-Scores von Kosmos-2.5 auf dem Wired Table in the Wild Datensatz (Tabelle 3) fallen in einen ähnlichen Bereich wie

die Werte von BonnData und GloSAT, was darauf hindeutet, dass die Erkennung von historischen Tabellen kein grundsätzlich schwereres Problem für das Modell als die Erkennung moderner teilweise uneinheitlicher Tabellen ist. Dies deutet auch erneut auf den Umstand hin, dass das Kosmos-2.5-Modell auf Dokumenterkennung größtenteils digitaler Dokumente trainiert wurde [47], und deshalb auf verschiedenen Tabellenarten, die den in solchen Dokumenten vorkommenden Tabellen nicht entsprechen, nicht gut abschneidet.

Bei den vom Faster R-CNN erzielten Ergebnissen (Tabelle 3) fällt hingegen auf, dass der erzielte IoU auf den historischen Datensätzen deutlich unter dem auf WTW erzielten liegt. Auch die erzielten IoD-WF1-Werte sind ähnlich, und im Falle von 2.4.1 sogar auffallend geringer, wie der niedrigste auf WTW erzielte Score (muticolorandgrid). Bei der Zeilen- und Spaltenerkennung mit dem TableTransformer sieht es ähnlich aus, auch hier weisen die beiden historischen Datensätze die schlechtesten IoU- und IoD-Scores auf. Allerdings ist hier überraschenderweise der IoD-Score auf BonnData etwas höher als auf GloSAT. Dies könnte damit zusammenhängen, dass die Komplexität von BonnData durch die Zeilen- und Spaltenaufteilung der GroundTruth teilweise vereinfacht wird. Die Gesamtdatensatzgröße und Größe der Testdatensätze könnte hierbei eine Rolle spielen, jedoch schneiden sowohl FasterR-CNN (auf Zellen) als auch TableTransformer (auf Zeilen/Spalten) auch bei einigen der kleineren WTW-Kategorien, die ähnliche Testdatensatzgrößen wie die historischen Daten besitzen, ähnlich oder besser ab.

Dass die Modelle auf WTW tendenziell besser abschneiden ist wahrscheinlich auf die klarer erkennbaren Trennlinien und Textblöcke in Wired Table in the Wild (WTW) zurückzuführen, da dieser bewusst ausschließlich aus „wired “ Tabellen mit klaren Trennlinien besteht [47].

5.1.1.1 *Bonner Tabellendatensatz*

Auf dem **Bonner Tabellendatensatz** (2.4.1) sind die Ergebnisse im Vergleich zu den anderen Datensätzen am schlechtesten.

Bei Anwendung von Kosmos-2.5 auf das Gesamtbild (ohne vorheriges Ausschneiden der Tabellen) werden auf einigen Bildern keine Vorhersagen im Tabellenbereich gemacht (bei 17 von 53 Bildern). Dies ist zumeist darauf zurückzuführen, dass der vorhandene Checkpoint von Kosmos-2.5 nicht auf historischen Dokumentseiten oder historischen Tabellen trainiert wurde, sodass andere Bildelemente vorrangig erkannt werden.

Teilweise handelt es sich hierbei um die umliegenden Textblöcke, aber auch Störelemente wie die Maßstabsleiste, die auf einigen Bildern des Testdatensatzes vorhanden ist, stellen teilweise ein Problem dar (Bsp. siehe Abbildung 17).

Tabelle 7: WF1-Score (2.5.4) von IoD (2.5.2) im Vergleich über einzelne Kategorien des Bonner Tabellendatensatzes (2.4.1) (mit Thresholds {0.5,0.6,0.7,0.8,0.9})(Evaluiert auf Vorhersagen im Tabellengebiet und nur auf Bildern mit (validen) Vorhersagen, Kosmos-2.5-Werte ohne Finetuning wie beschrieben in 5.1)

category	model	wf1:IoD	testset size	files w/o pred
normale Tabelle	Kosmos-2.5	0.269317	22	7
	Faster R-CNN	**0.620608**	22	0
	TableTransformer	0.166120	22	1
minimale Tabelle	Kosmos-2.5	0.361992	21	8
	Faster R-CNN	**0.563356**	21	0
	TableTransformer	0.391793	21	0
Konflikttabelle	Kosmos-2.5	0.435217	3	1
	Faster R-CNN	**0.568821**	3	0
	TableTransformer	0.367707	3	0
Kassentabelle	Kosmos-2.5	0.226268	7	1
	Faster R-CNN	**0.635299**	7	0
	TableTransformer	0.336125	7	0

Es ist wahrscheinlich, dass diese Probleme sich beide durch ein Finetuning auf einem ausreichend großen Tabellendatensatz verbessern würden. Weiterhin könnten die Maßstableisten in einem manuellen Processing-Schritt aus dem Datensatz entfernt werden, da sie keine relevanten Informationen enthalten.

Im Vergleich mit den Ergebnissen mit R-CNN sind die erreichten WF1-Metriken mit Kosmos-2.5 und TableTransformer zwar weitaus schlechter (Tabelle 3), allerdings ist das Ergebnis von Kosmos-2.5 auf einigen der Bilder wesentlich besser, als dies vermuten lässt (siehe Abbildung 18). Mit dem TabellenTransformer werden dagegen, ähnlich wie in geringerem Ausmaß mit Faster R-CNN, zu viele BoundingBoxen vorhergesagt (siehe Abbildung 19). Dies ist aber für beide Netzwerke zu erwarten, da eine feste Anzahl von Objekten pro Bild vorhergesagt wird, und kann so für beide Modelle durch Score-Filtering behoben werden (5.2.1)

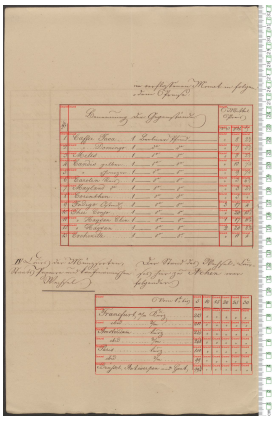
Bei Betrachtung der Ergebnisse auf den einzelnen Tabellenkategorien von BonnData (Tabelle 7) bezüglich des WF1 über IoD fällt auf, dass die verhältnismäßig am Besten und am Schlechtesten erkannten Tabellenkategorien bei den Modellen nicht übereinstimmen.

So schneidet Kosmos-2.5 auf den Kassentabellen am schlechtesten ab, während das Faster R-CNN dort die besten Werte erhält. Der TableTransformer schneidet auf den normalen Tabellen am Schlechtesten ab. Die minimalen Tabellen werden von Kosmos-2.5 verhältnismäßig besser erkannt als von Faster R-CNN, was wahrscheinlich darauf zurückzuführen ist, dass diese Tabellen regulärem Text ähneln und Kosmos-2.5 auf Textlokalisierung (und OCR) in Dokumenten

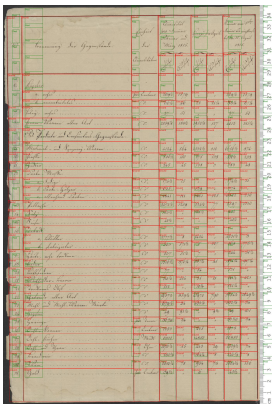
Da die vorhandene Kategorisierung der Testbilder des Bonner Tabellendatensatzes teilweise unvollständig oder uneindeutig war, habe ich diese manuell vervollständigt, um einen Vergleich der Kategorien zu ermöglichen. Dies war daher möglich, dass ich in meiner Projektgruppe an der Annotation des Datensatzes beteiligt und somit mit den Annotationsprinzipien vertraut war [64].

Tabelle 8: Vergleich der Resultate auf dem Gesamtbild (alle BoundingBoxen ausgewertet) und auf ausgeschnittenen Tabellen (Modellcheckpoints aus Projektgruppe [41]) auf dem Bonner Tabellendatensatz 2.4.1 mit Faster R-CNN (mit Thresholds {0.5,0.6,0.7,0.8,0.9}). Die Modelle wurden immer mit der Default-Initialisierung von Pytorch (Coco V1) [21] initialisiert und ggf. auf GloSat weiter vortrainiert, es wurde für max. 250 Epochen mit Early Stopping trainiert.

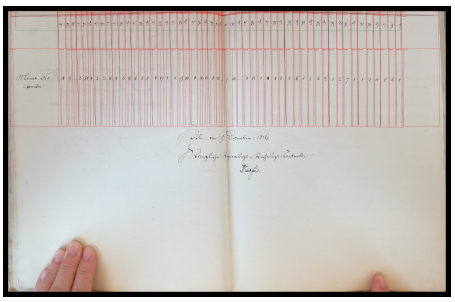
category	tablecutout	pretrained on GloSat	wf1:iou	testset size
normale Tabelle	no	yes	0.446685	22
		no	0.407988	22
	yes	yes	0.472387	28
		no	0.464772	28
minimale Tabelle	no	yes	0.272925	21
		no	0.206345	21
	yes	yes	0.266400	29
		no	0.297805	29
Konflikttabelle	no	yes	0.347740	3
		no	0.304936	3
	yes	yes	0.427133	3
		no	0.428965	3
Kassentabelle	no	yes	0.359619	7
		no	0.329149	7
	yes	yes	0.357364	8
		no	0.380403	8



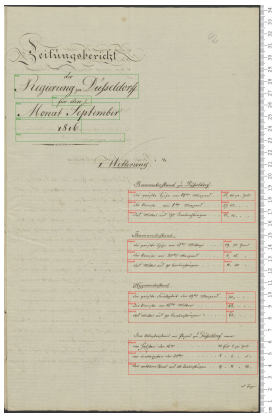
(a) Maßstabsleiste am Bildrand anstatt Bildinhalt erkannt



(b) Spaceing Problems on dense tables



(c) Vertikale Tabellen



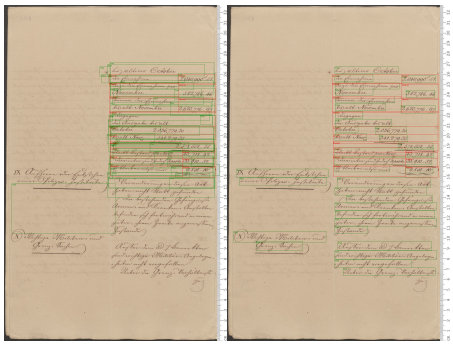
(d) Erkennung von Text anstatt Tabellen

Abbildung 17: Beispiele von Fehlerquellen bei as-is-Anwendung von Kosmos-2.5

trainiert wurde. Dies deutet wiederrum darauf hin, dass Kosmos-2.5 mit einem Finetuning auf dem historischen Tabellendatensatz in den für Faster R-CNN schwierigen Kategorien eventuell besser abschneiden könnte. Hier ist aber zu beachten, dass in dieser Kategorie auf 8 von 21 Seiten keine validen Vorhergesagen von Kosmos-2.5 gemacht wurden.

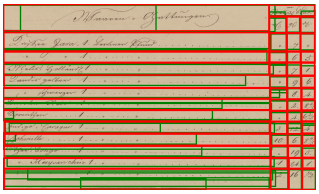
Im Vergleich mit den Resultaten von Faster R-CNN auf vorausgeschnittenen Tabellen (Modellcheckpoints aus [41]) in Tabelle 8 bezüglich des WF1 über IoU sind die Ergebnisse auf dem Gesamtbild (alle vorhergesagten BoundingBoxen evaluiert) nur auf den Konflikttabellen, die einen sehr kleinen Teil des Datensatzes ausmachen, deutlich schlechter. Mit Pretraining auf Glosat ist er auf den normalen Tabellen leicht schlechter und auf den minimalen Tabellen und Kassentabellen sogar etwas besser. Das Pretraining auf GloSAT ist für die Anwendung auf dem Gesamtbild grundsätzlich erfolgreich, während

(a) Ergebnis Faster R-CNN Projektgruppe [41]

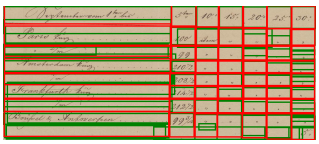


(b) Ergebnis Faster R-CNN Gesamtbild

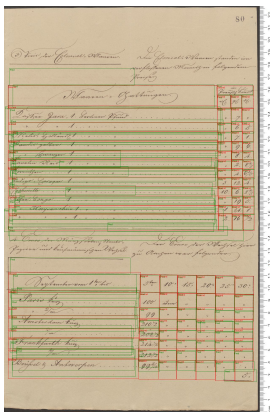
(c) Ergebnis Kosmos-2.5



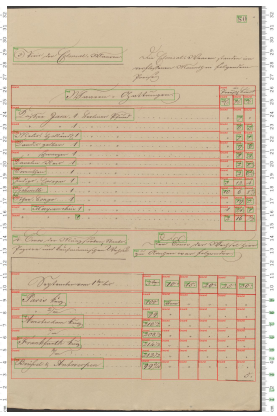
(d) Ergebnis Faster R-CNN Projektgruppe [41] Tabelle 1



(e) Ergebnis Faster R-CNN Projektgruppe [41] Tabelle 2



(f) Ergebnis Faster R-CNN Gesamtbild



(g) Ergebnis Kosmos-2.5

Abbildung 18: Performance mit Kosmos-2.5 entgegen der Aussage des Wf1-Score teilw. besser als mit Faster R-CNN(Vorhersage in Grün)

es für Anwendung auf den ausgeschnittenen Tabellen eher negativ ist (leicht schlechtere Ergebnisse in 3 Kategorien). Dies könnte damit zusammenhängen, dass durch das Pretraining die Erkennung von Tabellenelementen im Gegensatz zu Textelementen verbessert wird, die genauen Tabellenstrukturen und -annotationen aber zu unterschiedlich sind, um bei reiner Tabellenstrukturerkennung auf ausgeschnittenen Tabellen zu einer Verbesserung zu führen. Die Pipeline lässt sich also mit geringen Einbüßungen bezüglich Genauigkeit dadurch vereinfachen, dass das Faster R-CNN direkt auf dem Gesamtbild trainiert und angewendet wird, anstatt wie bei der Projektgruppe [41] zwei R-CNNs separat auf Tabellenerkennung und Tabellenstrukturerkennung auf ausgeschnittenen Tabellen zu trainieren.

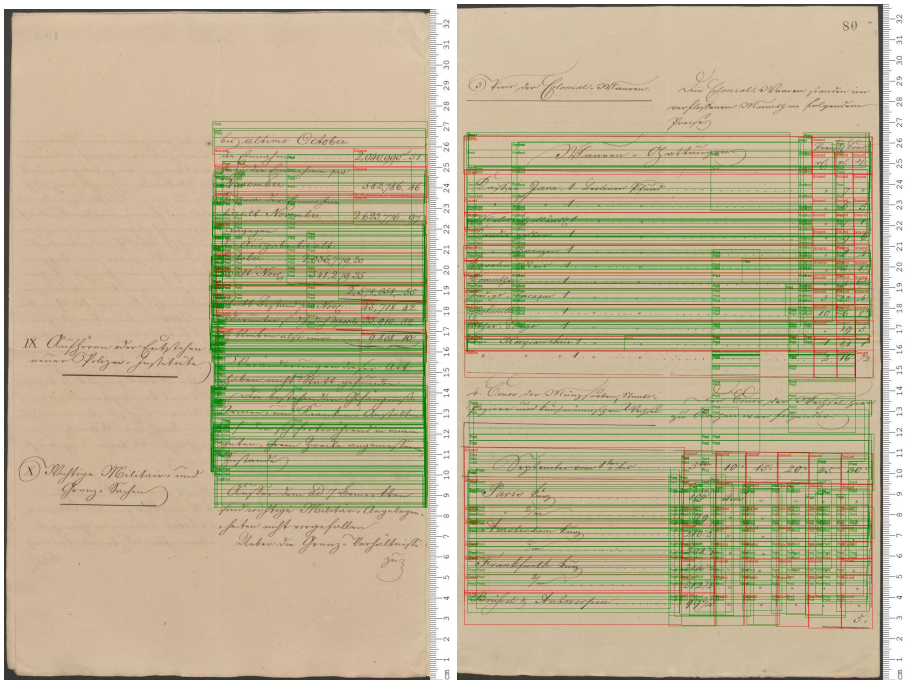
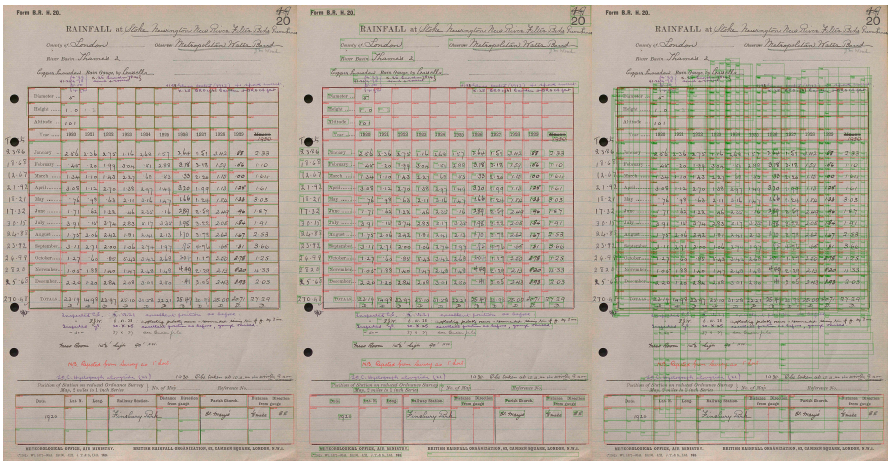


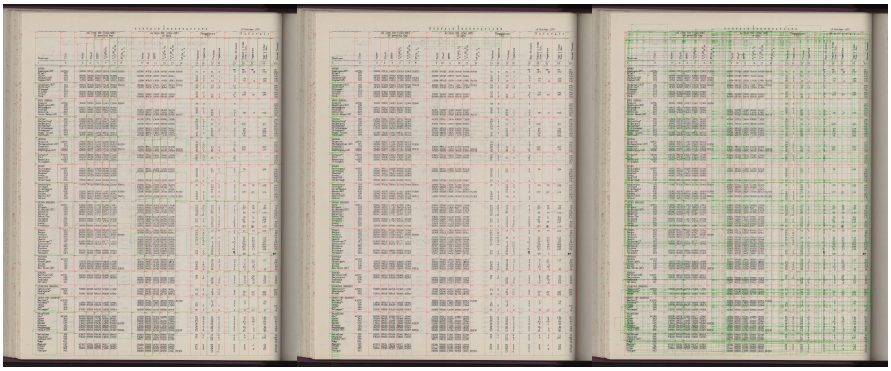
Abbildung 19: Ergebnisse von TableTransformer auf dem BonnData-Datensatz (Zellen)

Tabelle 9: Ergebnisse auf GloSAT-Datensatz 2.4.2 (mit Thresholds {0.6,0.7,0.8,0.9}) (bei Gesamtbild alle BoundingBoxen evaluiert) und Ergebnisse des GloSAT-Paper [40], Checkpoint für Faster R-CNN mit ausgeschnittenen Tabellen aus [41]

model	variant	wf1	
		IoU	IoD
Faster R-CNN	full image	0.516229	0.703391
Kosmos-2.5	full image	0.000183	0.537526
TableTransformer	full image	0.061609	0.232119
Faster R-CNN [41]	tablecutout	0.596642	0.767239
Cascade TabNet [40]	individual cell	0.050000	-
	coarse cell	0.370000	-



(a) Ergebnis Faster R-CNN (b) Ergebnis Kosmos-2.5 (c) Ergebnis TableTransformer (Zellen)



(d) Ergebnis Faster R-CNN (e) Ergebnis Kosmos-2.5 (f) Ergebnis TableTransformer (Zellen)

Abbildung 20: Beispielbilder der Ergebnisse aus GloSAT (2.4.2) (GroundTruth in rot)

5.1.1.2 *GloSAT Tabellendatensatz 2.4.2*

In Tabelle 9 sind meine Ergebnisse mit Detektion auf ganzem Bild (Evaluation auf allen BoundingBoxen), und die Ergebnisse des Glosat-Paper und von Faster R-CNN je auf ausgeschnittenen Tabellen (Checkpoint Faster R-CNN ist aus meiner Projektgruppe [41], wurde neu evaluiert) aufgeführt. Für beide Evaluationskriterien schneidet Faster R-CNN deutlich am Besten ab.

Was die Detektion auf dem ganzen Bild angeht ist auffällig, dass der Tabletransformer einen deutlich schlechteren IoD-Score als Kosmos-2.5 und Faster R-CNN hat, während Kosmos-2.5bezüglich IoU, jedoch mit wesentlich geringerem Abstand zum TableTransformer, am schlechtesten ist. Die von Kosmos-2.5 vorhergesagten Zellen treffen die GroundTruth also ähnlich gut wie das Faster R-CNN, sind aber, wie schon bei 5.1.1 erwähnt, wesentlich kleiner, weshalb der IoU so schlecht ist. Hinzu kommt, dass die GloSAT-Annotation teilwei-

Tabelle 10: Vergleich meiner Ergebnisse mit denen aus [47]

Herkunft der Werte	Modell	Prec@0.9 over IoU	Rec@0.9 over IoU	F1@0.9 over IoU
Wired Table in the Wild (WTW) Paper [47]	Cycle-CenterNet+PairingLoss	0.780000	0.785000	0.783000
	CenterNet	0.742000	0.721000	0.731000
	TridenNet	0.645000	0.655000	0.650000
	Cascade R-CNN	0.774000	0.653000	0.709000
	Faster R-CNN Parsing Table Structures in the Wild	0.721000	0.615000	0.664000
Meine Ergebnisse (aktualisierte GroundTruth)	Faster R-CNN meine Reproduktion	0.854000	0.607000	0.710000
	Kosmos-2.5	0.000050	0.000064	0.000056
	TableTransformer (Cells)	0.008460	0.109972	0.015711

se grob sind und mehrere Zeilen als eine Zelle zusammenfassen, was diesen Effekt verstärkt. Wie ebenfalls schon in 5.1.1 erwähnt, werden leere Tabellenzellen aufgrund des Trainings ebenfalls nicht vorhergesagt (Abbildung 20 oben). Zusätzlich hat Kosmos-2.5 scheinbar Probleme bei sehr großen Tabellen mit klar definierten Spalten, dort wird nur die linke Spalte erkannt (siehe Abbildung 20 unten). Ähnliches ist auch bei meinen Anwendungstests auf historischen Zeitungsseiten aufgetreten (siehe C.2).

Mit Faster R-CNN tritt auf großen Tabellen das Problem auf, dass einige Zellen nicht vorhergesagt werden, auch wenn die vorhandenen Vorhersagen zutreffend sind. Dies könnte durch Anpassen der Maximalanzahl von vorhergesagten BoundingBoxen behoben werden, was allerdings zu Overprediction auf kleineren Tabellen führen könnte. Letzteres könnte aber warscheinlich durch Filtering (5.2.1) gemildert werden.

Bei dem TableTransformer kommt es erwarteterweise ohne Filtering wieder zur Overprediction. Die Zellengenauigkeit ist hier schlechter als erwartet, da die Tabellen im GloSAT-Datensatz regelmäßiger sind als die von BonnData und somit die Zellermittlung aus Zeilen und Spalten eher zutreffen sollte. Die Schlechtigkeit der Zellenvorhersage scheint teilweise darauf zurückzugehen, dass die Zellen bei einigen Tabellen lediglich durch Whitespace abgegrenzt sind und mehrere Textzeilen zu einem Block zusammengefügt werden (siehe Abbildung 27), sodass die Tabellenstruktur schwerer zu erlernen ist.

5.1.1.3 *Wired Table in the Wild Datensatz 2.4.3*

Im Vergleich mit den Ergebnissen aus [47] (Tabelle 10) schneidet die Faster R-CNN-Reproduktion recht gut ab. Die erreichte Precision ist deutlich besser als die im WTW-Paper, während der Recall leicht schlechter als das Faster R-CNN von [47] und deutlich schlechter als die CenterNet-basierten Modelle [47] ist. Als gesamter F1-Score ergibt sich damit ein ähnlicher Wert wie für das Cascade R-CNN, die CenterNet-basierten Ansätze schneiden also besser ab. Die Resultate auf Kosmos-2.5 und TableTransformer (aktualisierte GroundTruth) sind weitaus schlechter als alle anderen Ergebnisse. Ein Vergleich mit den Unterkategorien ist nicht möglich, da im Paper nur Werte für Adjacency Relation gegeben sind.

Im Vergleich meiner Ergebnisse über die einzelnen Unterkategorien (Tabelle 3, Beispielbilder siehe Abbildung 24) schneidet Faster R-CNN ebenfalls am Besten ab. Die einzige Ausnahme stellt hier der IoD-Wert für den Curved-Unterdatensatz dar, wo Kosmos-2.5 einen leicht besseren Wert erreicht. Dies zeigt erneut, dass die Präzision der Kosmos-Vorhersagen hier gut ist.

5.2 POSTPROCESSING

Anmerkung: Da die Beispielbilder viel Platz einnehmen, sind diese im Anhang B.2 zu finden.

5.2.1 *Filtering der Ergebnisse von Faster R-CNN und TableTransformer anhand ihres Scores*

Modelle mit einer festen Anzahl vorhergesagter BoundingBoxen wie Faster R-CNN und TableTransformer sagen Scores für jede von ihnen vorhergesagte BoundingBox voraus, nach denen die Ergebnis-BoundingBoxen gefiltert werden können. Dadurch kann die Anzahl der vorhergesagten BoundingBoxen auf Faster R-CNN und TableTransformer reguliert, und falsche BoundingBoxen mit niedrigen Confidence-Scores eliminiert werden.

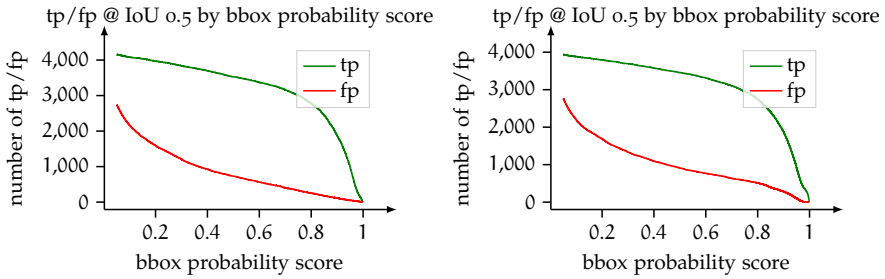
Um einen guten Filtering-Threshold zu finden, habe ich wie in meiner Projektgruppe [41] die Anzahl der True Positives und False Positives am IoU-Threshold 0.5 in Abhängigkeit des Probability Score Thresholds auf dem Testdatensatz gemessen (21, 22). Der IoU wird also immer über diejenigen BoundingBoxen berechnet, deren Probability Score größer oder gleich dem momentanen Threshold ist. Als Filtering-Threshold habe ich dann den kleinsten Wert größer 0 gewählt, wo die Differenz zwischen True Positives und False Positives maximal ist.

Um die Anwendung des Filterings auf neue Daten zu simulieren, habe ich außerdem die Filtering-Thresholds basierend auf dem Valid-Datensatz berechnet und auf die jeweiligen Testdatzensätze angewendet. Um Vergleichbarkeit mit den nicht-gefilterten Werten (Abbildung 3) zu wahren, werden hier auch die Werte aufgeführt, die bei der ausschließlichen Evaluierung auf den gefilterten BoundingBoxen, die innerhalb einer der annotierten Tabellen liegen, entstehen.

5.2.1.1 *R-CNN (3.1)*

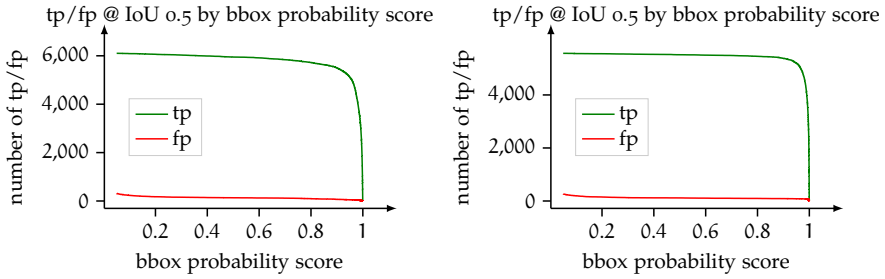
Mit diesen Filtering-Thresholds (21) wurden mit Faster R-CNN ähnliche, teilweise leicht bessere IoU- und IoD-Werte als ohne Filtering erreicht (Vgl. Tabelle 11). Weiterhin fällt auf, dass die Werte, die sich auf alle vorhergesagten BoundingBoxen beziehen, nur geringfügig schlechter sind als diejenigen, die sich nur auf die in einer Tabelle liegenden BoundingBoxen beziehen. Bei Anwendung auf neue Daten kann hier also ähnliche Genauigkeit erwartet werden.

Gleichzeitig hat die Anzahl der BoundingBoxen auf den Bildern wesentlich abgenommen, sodass diese um einiges lesbarer werden



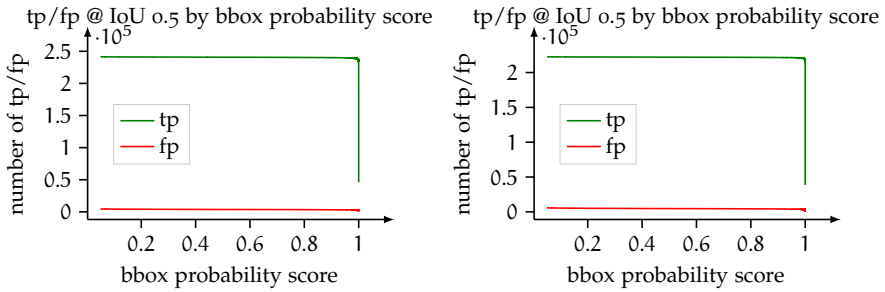
(a) Test-Threshold (b) Valid-Threshold

BonnData



(c) Test-Threshold (d) Valid-Threshold

GloSAT



(e) Test-Threshold (f) Valid-Threshold

WTW

Abbildung 21: Threshold Graphen der in Tabelle 3 aufgeführten R-CNN-Modell-Checkpoints

Tabelle 11: Ergebnisse auf Faster R-CNN mit auf Validdatensatz berechneten Filtering-Thresholds, an IoU und IoD Thresholds {0.5,0.6,0.7,0.8,0.9} (Vgl. ungefiltert siehe Tabelle 3; Vgl. Test-Filterthreshold siehe Tabelle 13 (für nur Tabellengebiet), Beispielbilder siehe Abbildung 26)

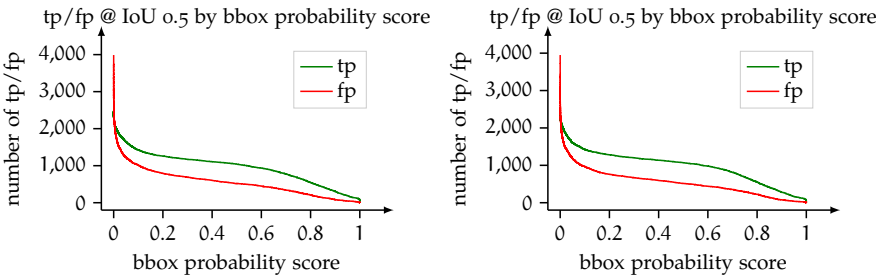
dataset	subset over WTW	evaluierte BoundingBoxen	wf1		testset size	files w/o valid pred
			IoU	IoD		
BonnData	-	alle	0.405968	0.588976	53	0
		nur in tabellengebiet	0.410478	0.595536	53	0
GloSAT	-	alle	0.562962	0.717869	41	0
		in tabellengebiet	0.561087	0.716008	41	0
Wired Table in the Wild (WTW) (2.4.3)	Curved	alle	0.826715	0.905353	427	0
		nur in tabellengebiet	0.838248	0.915848	427	0
	extremeratio	alle	0.954207	0.976462	1525	0
		nur in tabellengebiet	0.949301	0.972105	1525	0
	inclined	alle	0.969379	0.978445	670	0
		nur in tabellengebiet	0.971658	0.980729	670	0
	muticolorandgrid	alle	0.633685	0.660486	741	0
		nur in tabellengebiet	0.677588	0.704979	741	0
	occblu	alle	0.771025	0.829830	72	0
		nur in tabellengebiet	0.778637	0.837774	72	0
	overlaid	alle	0.903351	0.923942	75	0
		nur in tabellengebiet	0.909618	0.931324	75	2
	simple	alle	0.999540	1	93	0
		nur in tabellengebiet	0.999540	1	93	0

und nun besser zur eventueller Weiterverarbeitung in einer Pipeline geeignet sind.

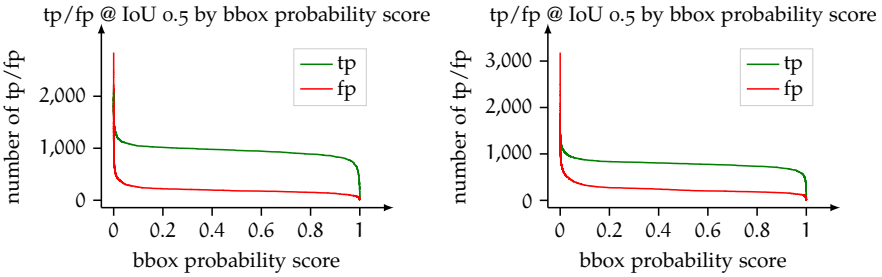
Das Filtering an den Valid-Thresholds stellt auf Faster R-CNN also einen klaren Erfolg dar und ist gut auf neue Daten übertragbar.

5.2.1.2 *TableTransformer (3.3)*

Gegenüber Tabelle 4 sind die IoU-Werte mit Filtering and Valid-Thresholds (12) entweder besser oder gleichbleibend. Auf dem Gesamtbild kommt es wie erwartet zu einem Präzisionsverlust gegenüber der Ergebnisse nur im Tabellengebiet, aber es werden ähnliche oder bessere Werte wie nur im Tabellengebiet ohne Filtering erreicht. Was IoD-Werte angeht, ist größtenteils eine leichte Abnahme gegenüber Tabelle 4 zu erkennen, aber es handelt sich trotzdem insgesamt um gute Werte. Auch bezüglich der Zellenevaluation (Abbildung 14) sind die Ergebnisse gegenüber Abbildung 3 ungefähr gleich oder besser, besonders die Werte bei GloSAT erhöhen sich. Trotzdem erziehlt der TableTransformer weiterhin deutlich schlechtere IoD-Werte als die anderen Modelle. Filtering ist also auch auf dem TableTransformer definitiv von Vorteil, da die Werte sich in ähnlichem Bereich wie ohne Filtering befinden, aber überschüssige und unwahrscheinliche Vorhersagen erfolgreich rausgefiltered werden (Beispielbilder siehe Abbildung 27). Test- und Valid-Filtering-Thresholds unterscheiden sich nur für den GloSAT-Datensatz (Tabelle 15).

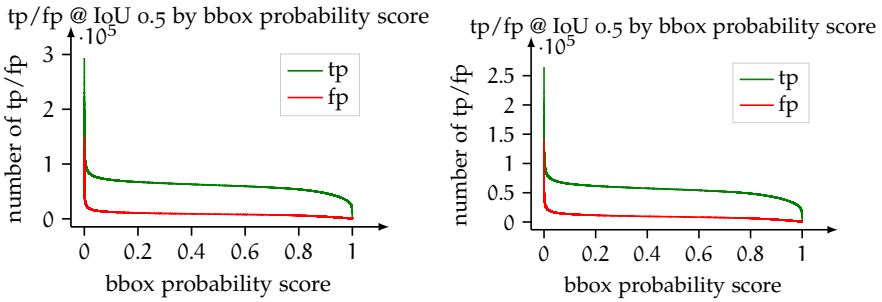


BonnData



(c) Test-Threshold (d) Valid-Threshold

GloSAT



BonnData

Abbildung 22: Threshold Graphen der in Tabelle 3 aufgeführten TableTransformer-Modell-Checkpoints, Test- und Valid-Thresholds für BonnData und WTW identisch

Tabelle 12: Ergebnisse von TableTransformer (3.3) mit Valid-Filtering und Evaluation auf Zeilen/Spalten (Vgl. ungefiltert siehe Tabelle 4, Beispielbilder siehe Abbildung 27)

	dataset	evaluierte BoundingBoxen	wf1 IoU	wf1 IoD	testset size	files w/o valid pred
Wired Table in the Wild (WTW) (2.4.3)	BonnData	alle	0.303879	0.679196	53	2
		nur in tabellengebiet	0.293265	0.673068		4
	GloSAT	alle	0.506143	0.690509	41	0
		in tabellengebiet	0.492475	0.653894		4
	Curved	alle	0.487804	0.919395	427	0
		nur in tabellengebiet	0.496119	0.932231		1
	extremeratio	alle	0.482958	0.969483	1525	0
		nur in tabellengebiet	0.655761	0.937123		0
	inclined	alle	0.502179	0.895966	670	0
		nur in tabellengebiet	0.853020	0.941643		
	muticolorandgrid	alle	0.434388	0.767842	741	0
		nur in tabellengebiet	0.509738	0.764802		1
	occblu	alle	0.366400	0.797228	72	0
		nur in tabellengebiet	0.435310	0.801167		
	overlaid	alle	0.542142	0.867527	75	0
		nur in tabellengebiet	0.563441	0.910857	75	1
	simple	alle	0.587711	0.989301	93	0
		nur in tabellengebiet	0.614729	0.997754	93	0

5.2.2 *DBScan-PostProcessing aus [40]*

Anmerken: Die Tabellen sind in Anhang B.1 zu finden.

Auch die Wirksamkeit der im GloSAT-Paper vorgestellten Postprocessessing-Methode mithilfe von DBScan 2.5.5 wurde auf den Ergebnissen der verschiedenen Modelle getestet. Da wir lediglich die Strukturerkennung auf dem Gesamtbild durchführen und keine seperate Tabellenerkennung durchgeführt wird, der GloSAT-Algorithmus aber Tabellenkoordinaten zusätzlich zu den BoundingBox-Koordinaten erwartet, werden die vorhergesagten BoundingBoxen pro Seite erst mithilfe von DBScan in Tabellen geclustered. Der GloSAT-Algorithmus wird dann auf die BoundingBoxen der so entstandenen Tabellen angewendet. Außerdem werden während des Postprocessessing alle ungültigen BoundingBoxen aus den Modellvorhersagen entfernt.

Dieses Postprocessessing führt bei Anwendung ohne Filtering, wie in Tabelle 16 zu sehen ist, zu einer Verschlechterung der Ergebnisse gegenüber Tabelle 3. Dies geht teilweise darauf zurück, dass die Tabellenkoordinaten durch Clustering ermittelt werden, aber Tabelle 17 zeigt, dass selbst bei Verwendung der GroundTruth-Tabellen die erreichten Werte schlechter als ohne PostProcessing sind. Das DBScan-Postprocessessing scheint also nicht auf andere Modelle oder andere Datensätze übertragbar zu sein.

6 FAZIT UND AUSBLICK

In dieser Arbeit wurden drei Modelle mit unterschiedlichen zugrundeliegenden Architekturen (Faster R-CNN, Kosmos-2.5, TableTransformer) bezüglich Tabellenstrukturerkennung auf drei schwierigen Datensätzen (BonnData, GloSAT, Wired Table in the Wild) evaluiert und verglichen.

Insgesamt erreicht der TableTransformer auf Zeilen/Spalten gute IoD-Werte (Tabelle 3). Die Genauigkeit wird aber besonders auf den indirekt bestimmten Zellen durch die Annahmen über die Tabellenstruktur eingeschränkt, die bei der Modellerstellung zugrunde gelegt wurden und für unregelmäßige und historische Tabellen nicht zutreffen (z.b. durchgehende Zeilen und Spalten, Zellen immer als Schnittmenge von Zeile und Spalte). Er ist also aufgrund dieser nicht zutreffenden Annahmen für die Anwendung auf komplexe Tabellen eher ungeeignet. Für diesen Anwendungsfall wäre der Modellaufbau so anzupassen, dass Zellenerkennung direkt trainiert werden kann. Alternativ könnte man das bestehende Pre-Processing der Annotationen um die im Ursprungs-Paper verwendeten Kategorien (siehe A.9) erweitern, die diese Unregelmäßigkeiten teilweise abdecken. Außerdem wäre dann auch, falls möglich, eine Optimierung des derzeit ineffektiven Multi-Node-Trainings (siehe Tabelle 2) sinnvoll.

Die Ergebnisse der Zeilen- und Spaltenerkennung zeigen aber, dass auch transformerbasierte Modelle bei Training auf komplexen Tabellen trotz der verhältnismäßig kleinen Datensatzgröße konvergieren. Im Gegensatz zum TableTransformer wird Kosmos-2.5 nur auf Bounding-Box-Texterkennung ohne deren Kategorisierung trainiert und kann Zellen somit direkt erkennen, ist also potentiell besser für unregelmäßige Tabellen geeignet. Da in dieser Arbeit kein Finetuning auf dem Modell durchgeführt werden konnte und die reinen Inferenzwerte sehr schlecht waren, kann hier kein finaler Schluss über die Anwendbarkeit eines OCR-freien Ende-zu-Ende-Transformers auf schwierigen Tabellen gezogen werden. Es ist aber wahrscheinlich, dass mit einem geeigneten Finetuning weitaus bessere Werte als mit reiner Inferenz erreicht werden, da viele der auftretenden Probleme auf die Trainingsbedingungen des Modells zurückzuführen sind.

Im 08/2024 wurde zudem angekündigt, dass die baldige Integration von Kosmos-2.5 in Transformers von HuggingFace erfolgt [1] [2], bisher scheint diese aber noch nicht abgeschlossen zu sein. Wenn das Modell fertig in HuggingFace implementiert ist, sollte auch Finetu-

1 https://huggingface.co/microsoft/kosmos-2.5
2 https://github.com/huggingface/transformers/pull/31711

ning problemlos möglich sein. Weiterhin ist mit HuggingFace und Torch Lightning eine Einbettung als LightningModule analog zu der Einbettung von TableTransformer (4.1) leicht möglich, um Multi-GPU-Training zu ermöglichen. Für ein Finetuning werden allerdings geeignete GroundTruth-Daten in Form von BoundingBoxen mit transkribiertem Text benötigt, um auch die OCR-Fähigkeiten des Modells zu trainieren.

Momentan ist auf keinem der hier betrachteten Tabellendatensätze vollständig transkribierter Text vorhanden. Falls die Transkribierung derart erweitert werden würde, wäre es aussichtsreich, Kosmos-2.5 (im Falle der deutschen historischen Datensätze mit einem anderen Tokenizer) finezutunen. Im Idealfall wären dann Tabellenstrukurerkennung und OCR in einem Schritt möglich.

Faster R-CNN hat bei weitem die besten Ergebnisse für die Zellenerkennung erreicht. Für die höchste BoundingBox-Genauigkeit auf historichen Tabellen ist also weiterhin Faster R-CNN mit einem separaten OCR zu empfehlen. Der Genauigkeitsunterschied zwischen Tabellenstrukurerkennung auf dem Gesamtbild und auf der vorausgeschnittenen Tabelle ist aber mit Filtering gering genug (Tabelle 11), dass man die Pipeline gegenüber dem in meiner Projektgruppe umgesetzten Ansatz [64] vereinfachen kann, indem man direkt Tabellenstrukurerkennung auf dem Gesamtbild anwendet.

Was Post-Processing-Methoden angeht, so ist das im GloSAT-Paper vorgestellte DB-Scan-Postprocessessing (2.5.5, 5.2.2) nicht gut auf andere Datensätze übertragbar, und selbst auf dem GloSAT-Datensatz (2.4.2) ist es bei Tabellenstrukurerkennung mit anderen Modellen nicht zielführend. Alternativ könnte man die ähnliche in [47] vorgestellte Post-Processing-Methode, bei welcher die detektierten Tabellenzellen heuristisch zu Tabellen gruppiert werden und dann Zeilen- und Spaltenanfang und -ende zugeteilt bekommen, vergleichend implementieren. Da dieser Ansatz davon ausgeht, dass lediglich Zell-BoundingBoxen zur Vefügung stehen, werden wahrscheinlich bessere Ergebnisse als in 5.2.2 erreicht werden.

Teil III

ANHANG

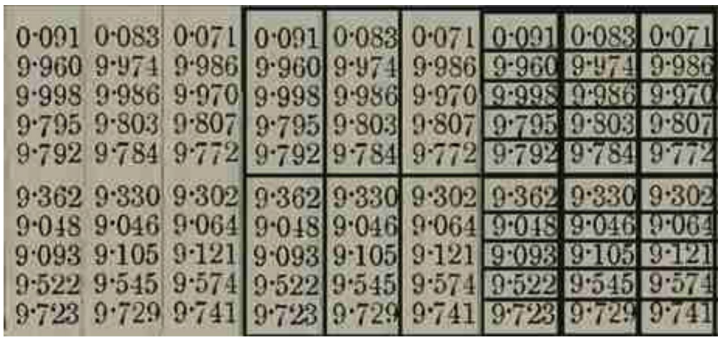
A

ERGÄNZUNGEN ZUM LITERATURTEIL

A.1 DETAILS ZU DATENSÄTZEN

Source ID; Region, Timeframe	**Images / Tables / Headers**	**Page Style; Table Style**
20cr_DWR_MO; India, 1970's	24 / 31 / 31	Printed; Borderless
20cr_DWR_NOAA; India, 1930's	24 / 24 / 24	Printed; Semi-bordered
20cr_Kubota; Philippines, 1900's	24 / 28 / 28	Printed; Semi-bordered
20cr_Natal_Witness Africa, 1870's	26 / 26 / 26	Printed; Semi-bordered
Ben Nevis UK, 1890's	97 / 137 / 82	Printed; Semi-bordered
DWR UK and world 1900's	93 / 139 / 139	Mixed; Semi-bordered
WesTech Rodgers Arctic 1880's	82 / 164 / 82	Mixed; Semi-bordered
WR_10_years UK, 1830's to 1930's	97 / 129 / 129	Mixed; Bordered
WR_Devon_Extern UK, 1890's to 1940's	33 / 33/ 33	Mixed; Bordered
Total	500 / 710 / 573	

(a) Datensatzstatistik [40]



(b) Annotationsbeispiel, in der Mitte grobe Segmentierung und rechts Segmentierung der individuellen Zellen, Darstellung aus [40]

Abbildung 23: Der GloSAT-Datensatz [40]

A.2 DETAILS ZU RESNET (SIEHE 2.1.4)

Anmerkung: Dieser Abschnitt zum ResNet beruht auf dem gleichnahmigen Abschnitt in meinem Projektgruppenbericht [64].

A.2.0.1 *Aufbau*

Die Residualfunktion wird als sogenannter Residualblock realisiert. Dieser besteht aus mehreren Faltungsschichten, deren Ausgabe mit der Ausgabe sogenannter Shortcut Connections (auch Identity Skip Connections), die eine Identitätsabbildung durchführen, addiert wird (3 links).

Die ResNet-Architektur kann unterschiedlich tief sein (z.B. resnet50 mit 50 Schichten). Das ResNet besteht immer aus mehreren aufeinanderfolgenden Stapeln von Residualblöcken, deren Größe je nach Tiefe variiert und zwischen denen Downsampling (Verkleinerung der Feature Map) durch eine Faltung mit Stride 2 durchgeführt und gleichzeitig die Dimension (Anzahl der Feature Maps) verdoppelt wird.

Bei Dimensionsverdopplung gibt es zwei mögliche Implementationen der Shortcut Connection, um Ein- und Ausgabedimension anzugleichen: Projektion der Eingabe mithilfe einer 1x1-Faltung (=Projection Shortcut) oder parameterlose Identity Skip Connection mit Zero Padding. Letzteres ist bei tiefen Netzwerken (ab 50 Schichten) empfehlenswert, da sie parameterlos ist und so die Modellgröße und die Trainingszeit kleiner bleiben.

Um die Trainingszeit bei tiefen Netzen weiter zu reduzieren, wird dort eine Bottleneck-Architektur für die Residualblöcke verwendet (3 rechts). Durch 1x1 Faltungsschichten wird die Dimension erst verkleinert und dann nach Anwendung einer 3x3-Faltung -dem Bottleneck- wiederhergestellt.

A.3 WEITERE ARBEITEN BEZÜGLICH HISTORISCHER DOKUMENTVERARBEITUNG

dhSegment [6] ist ein generischer Deep Learning Ansatz zur Segmentierung von historischen Dokumenten. Im ersten Schritt werden aus den Eingabebildern mithilfe eines auf dem U-Net [46] basierenden Modells mit ResNet-Backbone Merkmalskarten herausgebildet. Aus diesen wird dann im zweiten Schritt durch Postprocessessing mit gängigen Methoden wie Connected Component Analysis der gewünschte Output extrahiert.

Mit **Reading Yesterdays News** [50] wurden an der Universität Bonn bereits sehr gute Segmentierungsergebnisse auf dem zugehörigen Bonn Newspapers Datensatz erreicht. Hierfür wurde dhSegment als Modell verwendet, wobei zusätzlich zu dem ursprünglichen Modell auch eine abgeänderte Variante mit Convolutional Block Attention Modules verwendet wurde. Zum Pretraining fand hier der Europeana-Datensatz Verwendung.

PHD [11] adaptiert Pixel [48] als bildbasierten OCR-freien Sprachencoder von historischen Dokumenten, um die Probleme mit OCR bei historischen Dokumenten zu vermeiden, und zeigt erfolgreiche Anwendung auf Fragenbeantwortung bei historischen Zeitungen.

Auch semantische Suche von ähnlichen Zeitungsausschnitten ist möglich. Pixel beruht auf der Vit-MAE-Architektur, wobei der Decoder ähnlich wie bei BERT zum Finetuning durch einen passenden Classification-Head ersetzt wird. Soweit es aus dem Paper ersichtlich war, wurde hier keine Anwendung auf Segmentierungsaufgaben versucht.

A.4 WEITERE MODELLE FÜR OCR-FREIE ENDE-ZU-ENDE DOKUMENTVERARBEITUNG

Donut [31] ist ein transformerbasierter End-to-End Ansatz zur Dokumentenverstehung, bei dem aus einem Input von Bildern von Dokumentseiten (und Prompts) Json-Files generiert werden. Außerdem enthält das Donut-Release mit SynthDoG einen skalierbaren synthetischen Dokumentgenerator zum Erstellen von Trainingsdatensätzen in anderen Sprachen. Dieser sampled die jeweilige Sprache von Wikipedia, ist also für historische Dokumente, die andere Sprache oder Zeichen benutzen, nur begrenzt hilfreich.

Es handelt sich bei Donut um eine Encoder-Decoder-Architektur. Je nach Prompt sind verschiedenartige Tasks wie reine OCR (als Vortraining), Fragenbeantwortung, Klassifizierung und komplette Strukturerkennung mitsamt Transkribierung möglich.

Donut benutzt den Swin Transformer 2.2.2 zum Einbetten der Eingabebilder. Die letzendlich hierbei enstandenen Encoder-Output-Embeddings werden dann dem Decoder für die Cross-Attention übergeben.

Bei dem Decoder handelt es sich um einen mBart-Decoder [36], der neben den Encoder-Embeddings noch einen Task-abhängigen Prompt als Eingabe erhält. Donut nutzt zum Embedden des Texts den sprachunabhängigen XLM-RoBERTa-Tokenizer, der auf Sentence Piece basiert.

Donut benutzt Teacher-Forcing als Trainingsstrategie, während des Trainings erhält das Modell also in jedem Schritt die Ground Truth als Input.

Dessurt [15] ist ebenfalls ein transformerbasierter End-to-End-Ansatz zur Dokumentenverarbeitung, der Donut im Aufbau sehr ähnlich ist. Unterschiede sind in der Berechnung der Cross-Attention zu finden. Während bei Donut eine klassische Encoder-Decoder Architektur vorliegt, bei der lediglich im Decoder die Cross-Attention mit den Encoder-Embeddings berechnet wird, nutzt Dessurt eine 3-Stream-Architektur(Bild, Prompt, Antwort/Ausgabe), die es ermöglicht, in einem modifizier-

ten SWIN-Encoder nicht nur die Shifted-Window-Self-Attention sondern auch Cross-Attention mit dem Prompt zu berechnen. Das Output-Decoding wir mit Greedy Search ermittelt, es wird also in jedem Schritt das wahrscheinlichste Token für die Ausgabe gewählt.

Nougat [10] ist ein transformerbasierter End-to-End-Ansatz zur Verarbeitung von PDFs zu wissenschaftlichen Forschungsarbeiten.
Aus den Eingabebildern wird hier Formattierter Markup-Text generiert, der direkt in Latex-Code umgewandelt werden kann. Ein besonderer Fokus liegt auf der korrekten Latex-Formattierung von wissenschaftlichen Formeln und Graphen.
Das genutzte Modell folgt der Donut-Architektur, allerdings wird ein anderer Tokenizer benutzt, der spezifisch auf wissenschaftliche Texte ausgelegt ist.

Nougat generalisiert auf ähnliche Dokumente wie beispielsweise alte Mathebücher.

Bei **Pix2Struct** 2.3.2 wird HTML-Parsing als Self-supervised Pre-Training Methode verwendet. Dabei werden Self-supervised Paare von Screenshots und zugehörigem vereinfachten HTML-Sourcecode verwendet, wobei verschiedene Pretraining-Strategien wie BART-like masked text und Donut-like OCR-Pretraining verwendet. Vortraining ist wesentlich stabiler und das Finetuning is genauer, wenn vor dem HTML-Parse-Vortraining ein kurzes Vortraining mit einer simplen Text-Dekodierungsaufgabe aus Bildern durchgeführt wird.

Das Finetuning unterscheidet sich je nach Task, wobei Prompts (wie Fragen bei Visual Question Answering) direkt als Bildheader gerendert werden. Bei Aufgaben, die sich nur auf bestimmte Bildausschnitte beziehen, werden diese durch eine BoundingBox eingegrenzt.

Durch diese Neuerungen erreich Pix2Struct i.A. bessere Ergebnisse als Donut (A.4).

Matcha [35] nutzt Pix2Struct (2.3.2) als Basismodell und führt zusätzliches Pretraining durch Chart Derendering (Code oder zugrundeliegende Tabelle aus Chart extrahieren) und mathematical reasoning (Antwort einer als Bild gerenderten mathematischen Fragestellung ermitteln). Damit wurden konsistente Verbesserungen gegenüber Pix2Strukt auch in Chart- und Plotunabhängigen Anwendungsbereichen wie Dokumente und Natural Images erzielt.

UReader [62] nutzt ein Multimodal Large Language Model (MLLM) und eine innovative Cropping-Technik der Eingabebilder, um hochauflösende Bilder zu verarbeiten.

Als MLLM wird im Paper mPLUG-Owl gewählt, andere Modelle sind aber ebenfalls möglich.

Zuerst wird das Bild im Shape Adaptive Cropping Modul gecropt und die resultierenden Crops laufen gleichzeitig durch einen Visual Encoder und einen Visual Abstractor. Damit im Large Language Model (LLM) räumliche Relationen zwischen den Crops zur Verfügung stehen, werden diese erst an ein Crop Position Encoding Modul weitergegeben, bevor sie als Eingabe für das LLM verwendet werden. freeze the origin language model and adopt the low-rank adaptation approach(LoRA) eezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks.

DocPedia [23] ist ein Large Multimodal Model (LMM), das als Decoder anstelle eines light language decoders (wie bei Donut, Pix2Struct, KOSMOS 2.5) ein LLM nutzt. Es kann hochauflösende Eingabebilder verarbeiten. Die JPEG-Eingabebilder werden zuerst mithilfe von Discrete Cosine Transform (dt. Diskrete Kosinustransformation) (DCT) in den Frequenzbereich transformiert, bevor sie an einen Swin Transformer [37] als Visual Encoder gegeben werden. Die Ausgabe-Embeddings werden dann mit einer Linearen Schicht der Eingabedimension von Vicuna [13], dem verwendeten LLM, angeglichen.

Der Output der Linearen Schicht wird dann mit dem tokenized Prompt verknüpft und als Eingabe an das LLM gegeben.

Das Training erfolgt in zwei Schritten. Im ersten Vortrainingsschicht wird das LLM eingefroren, sodass der Vision Encoder und die Lineare Schicht mit Low Level Image tasks wie Texterkennung und -lesen lernen, mit Eingaben im Frequenzbereich umzugehen. Im zweien Schritt wird dann das gesamte Netz im Fine Tuning mit einer Mischung aus den Low Level Aufgaben aus dem Pretraing und neuen High level Aufgaben aus den Bereichen Dokument- und Szenenverstehen trainiert.

A.5 WEITERE INFORMATIONEN ZUR TABELLENSTRUKTURERKENNUNG

Table Reconstruction Aligned to Corner and Edges (TRACE) stellt einen neuen Ansatz zur Tabellenstrukturerkennung vor. Anstatt separat Tabellen- und Tabellenstrukturerkennung durchzuführen, wird hier ein Ende-zu-Ende-Modell vorgeschlagen, dass die bekannte Struktur von Tabellen nutzt und Ecken und Kanten der Tabelle erkennt, woraus dann die Tabellenstrukur rekonstruiert wird. Die Kanten werden dabei als implizit oder explizit klassifiziert, sodass auch Tabellen mit unregelmäßigem Aufbau erkannt werden können.

TableBank [33] ist ein Benchmark-Datensatz für Tabellenerkennung

und -strukturerkennung, der ca. 417000 Tabellen aus Latex- und Word-Dokumenten enthält. Für die Tabellenerkennung sollen BoundingBoxen generiert werden, während für die Tabellenstrukturerkennung HTML-Tag-Sequenzen generiert werden sollen, welche die Anordnung der Zeilen und Spalten sowie den Typ der Zelle repräsentieren.

A.6 KOSTENFUNKTIONEN

Anmerkung: Dieser Abschnitt zur Kostenfunktionen beruht auf meinem Projektgruppenbericht [64].

A.6.1 *Kreuzentropie-Kostenfunktion (Cross-Entropy-Loss)[34]*

Die binäre Kreuzentropie-Kostenfunktion, auch log-loss genannt, ist definiert als

$$CE(p_t) = -\log(p_t) \tag{11}$$

mit

$$p_t = \begin{cases} p & \text{wenn y=1} \\ (1-p) & \text{sonst} \end{cases}, \tag{12}$$

Ground Truth $y \in \{-1, 1\}$ und
$p \in [0, 1]$, welches die vom Modell geschätze Wahrscheinlichkeit für die Klasse $y = 1$ ist. Die Kreuzentropie-Kostenfunktion dient dazu, Abweichungen der geschätzen Wahrscheinlichkeit für die Ground Truth p_t von 1 zu bestrafen. Je weiter p_t von 1 abweicht, desto größer der Loss. Allerdings ist die Kreuzentropie-Kostenfunktion auch für gut klassifizierbare Beispiele (Wahrscheinlichkeit » 0.5) noch hoch, weshalb oft ein klassenspezifischer Balancing-Faktor

$$\alpha_t = \begin{cases} \alpha & \text{wenn y=1} \\ (1-\alpha) & \text{sonst} \end{cases}, \tag{13}$$

wobei α ein Hyperparameter ist, verwendet wird.
Die Kreuzentropie-Kostenfunktion ist so modifizierbar, dass sie auch für Mehrklassen-Klassifizierung verwendet werden kann.

A.6.2 *Smooth L1-Loss [24]*

Der Smooth L1-Loss ist als

$$L_{loc}(t^u, v) = \sum_i \text{smooth}_{L1}(t_i^u - v_i) \tag{14}$$

definiert, wobei u die betrachtete Klasse, t ein 4-Tupel der vorhergesagten Offsets und v ein 4-Tupel der Targets ist, mit

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & \text{sonst} \end{cases}. \tag{15}$$

Die Differenz zwischen Vorhersage und Target wird bei smooth_{L1} nur für absolute Differenzen kleiner 1 quadriert, sodass man bei kleinen Differenzen die Vorteile von einer quadratischen Kostenfunktion hat, ohne dass es bei großen Outliers zu wesentlich größeren quadrierten Fehlern kommt (was eine Schwachstelle der L_2-Kostenfunktion ist).

A.7 Details zum Faster R-CNN (siehe 3.1)

Anmerkung: Dieser Abschnitt zum Faster R-CNN wurde aus dem gleichnahmigen Abschnitt in meinem Projektgruppenbericht [64] übernommen.

A.7.1 *Aufbau*

A.7.1.1 *Geteilte Faltungsschichten*

Die geteilten Faltungsschichten werden auch als Backbone-Netzwerk bezeichnet, zumeist werden Blöcke eines tiefen Faltungsnetzes wie ResNet (siehe 2.1.4) verwendet [20].

Die Ausgabe der geteilten Faltungsschichten ist eine Convolutional Feature Map. Die Höhe und Breite der Ausgabe-Feature Map ist normalerweise wesentlich schmaler als das Eingabebild, das genaue Größenverhältnis hängt dabei von der verwendeten Backbone-Architektur ab.

A.7.1.2 *Regionsvorschlagsnetzwerk*

Das Regionsvorschlagsnetzwerk besteht aus einer nxn-Faltungsschicht, die man als nxn-Sliding-Window auf der Ausgabe-Feature Map der geteilten Faltungsschichten verstehen kann. Im Paper wird n=3 gewählt, wobei aufgrund des vorausgehenden geteilten tiefen Faltungsnetzes das Receptive Field auf dem Eingabebild groß ist.
Darauf folgen dann zwei (2) 1x1-Sibling-Faltungsschichten, eine für die Box-Regression (reg), die die Koordinaten der BoundingBoxes vorhersagt, und eine für die Box-Classification (cls), was die Wahrscheinlichkeit ist, dass der Regionsvorschlag ein Objekt enthält (=Objectness Score).

Pro Sliding-Window, also pro Punkt nach Anwendung der nxn-Faltungsschicht, werden k Regionsvorschläge generiert. Die Ausgabe der Box-Regression-Schicht hat also eine Tiefe von 4k, da 4 Koordinaten pro BoundingBox vorhergesagt werden, während die Box-Classification-Schicht eine Ausgabe mit Tiefe von 2k hat (Wahrscheinlichkeiten für Objekt und Hintergrund). Die k Regionsvorschläge werden relativ zu k Anchor-Boxen angegeben, die auf dem jeweiligen

nxn-Fenster zentriert sind und sich in Größe und Seitenverhältnis unterschieden. Es wird also eine 'Pyramide von Anchor-Boxen' verwendet, sodass verschiedene Größenordnungen von BoundingBoxes basierend auf einer einzelnen Ausgabe-Feature Map vorhergesagt werden können. Dies spart Rechenzeit, da nicht z.B. Feature Maps für verschiedene Bildgrößen berechnet werden müssen, und stellt zudem sicher, dass ein geteiltes Backbone-Netzwerk verwendet werden kann, da Fast-R-CNN- Detektionsnetzwerke eine einzelne Ausgabe-Feature Map verwenden. Gesamt hat eine Feature Map der Größe WxH mit $H, W \in \mathbb{N}$ also W*H*k Anchor-Boxes. Im Paper werden drei (3) verschiedene Größen und drei (3) verschiedene Seitenverhältnisse verwendet, sodass k=9.

Da es sich um ein vollständiges Faltungsnetz handelt, sind sowohl die Anchor-Boxen als auch die Regionsvorschläge relativ zu den Anchor-Boxen translationsinvariant.

A.7.1.3 *Fast-R-CNN [20]*

Für jeden im Regionsvorschlagsnetzwerk gefundenen Regionsvorschlag (Region of Interest (RoI)) wird aus der Ausgabe-Feature-Map der geteilten Faltungsschichten eine Feature Map mit gleichbleibendender Anzahl Features und mit räumlicher Größe HxW mit $H, W \in \mathbb{N}$ extrahiert. Höhe H und Breite W sind dabei Schicht-spezifische Hyperparameter, die unabhängig von dem betrachteten Regionsvorschlag gewählt werden. Diese räumliche Verkleinerung geschieht durch Max-Pooling des von der vorhergesagten RoI (BoundingBox) in der Ausgabe-Feature Map aufgespannten Fensters (= RoI-Pooling). Für eine Gesamtmenge von N Regionsvorschlägen hat der Output die Größe NxHxWxF, wobei $F \in \mathbb{N}$ die Anzahl der Filter der Ausgabe-Featuremap des gemeinsamen Faltungsnetzes ist.

Die extrahierten Feature Maps werden dann an zwei (2) vollständig verknüpfte Schichten (Fully Connected Layer) gegeben, worauf die Sibling-Netzwerke für Klassifizierung und Box-Regression folgen.

Das Klassifizierungs-Netzwerk besteht aus einem Fully-Connected-Layer, der den Input auf K+1 Objektklassen mit $K \in \mathbb{N}$ (K Objektklassen + Hintergrund) abbildet, gefolgt von Softmax als Aktivierungsfunktion, sodass Wahrscheinlichkeiten für jede der K+1 Klassen ausgegeben werden. Für eine Gesamtmenge von N Regionsvorschlägen hat der Output also eine Größe von Nx(K+1).

Das Box-Regression-Netzwerk verfeinert die Box-Regression aus dem RPN, indem es den Input mit einem Fully-Connected-Layer auf 4*K Outputs abbildet, die für jede der K-Objektklassen einen Bounding-Box-Offset relativ zu dem Regionsvorschlag enkodieren. Die Hintergrund-Klasse wird hier ignoriert, da diese keine Ground-Truth-Boxen hat. Für eine Gesamtmenge von N Regionsvorschlägen hat der Output also eine Größe von Nx4K.

A.7.1.4 *Training und Loss*

Ingesamt werden im Netzwerk vier unterschiedliche Losses berechnet, zwei im RPN und zwei im Fast-R-CNN.

Die Box-Regressions im RPN und im Fast-R-CNN nutzen beide den Smooth L1-Loss (siehe A.6.2).

Für die Klassifizierungen wird beim RPN die binäre Kreuzentropie (siehe A.6.1) und bei Fast-R-CNN die nicht-binäre Kreuzentropie verwendet.

Da das RPN und das Fast-R-CNN bei unabhängigem Training die Faltungsschichten unterschiedlich verändern würden, muss bei geteilten Faltungsschichten eine Technik entwickelt werden, um ein Faltungsnetz mit geteilten Features zu ermöglichen. Die Veröffentlichung stellt hierfür drei mögliche Methoden vor:

- Alternating training: Erst RPN trainiern, Regionsvorschläge zum Trainieren des Fast-R-CNN verwenden. Das auf Fast-R-CNN trainierte Netz wird dann zur Initialisierung vom RPN genutzt, diese Schritte werden iterativ wiederholt. Dieser Ansatz wurde bei dem für Tests im Paper verwendeten Netzwerk angewendet.
- Approximate joint training: Das Faltungsnetz wird wie im Paper beschrieben von RPN und Fast-R-CNN während des Trainings geteilt. Im Forward-Pass werden die Regionsvorschläge bei dem Trainieren des Fast-R-CNN als unveränderlich und vorberechnet behandelt, während bei der Backward-Propagation die Losses von RPN und Fast-R-CNN kombiniert werden. Hier werden also die vom RPN vorhergesagten BoundingBoxes als Eingabe für das RoI-Pooling in Fast-R-CNN bei der Berechnung der Ableitungen ignoriert, es handelt sich um eine Approximation.
- Non-approximate joint training: Durch Verwendung eines nach den Koordinaten der Regionsvorschläge differenzierbaren RoI-Pooling-Layers wird dieser Aspekt beim Backwards-Pass miteinbezogen.

A.8 DETAILS ZU KOSMOS-2.5 (SIEHE 3.2)

A.8.1 *Training*

Als Kostenfunktion wird die Kreuzentropie-Kostenfunktion (A.6.1) auf die nächste Token-Vorhersage angewandt.

Das "Full Sentence"Training-Format von RoBERTa wird adaptiert, d.h. nebeneinanderliegende Sätze werden von einem oder mehr Dokumenten mit einem Seperator Token zwischen Dokumenten gesampelt, bis eine vorgegebene Größe an Zeichen gegeben ist, und es gibt

im Gegensatz zu BERT kein Next-Sentence-Prediciton Loss. AdamW wird als Optimizer verwendet.

Für die layout-basierte Task (BoundingBoxen+Text) wurden zum Pretraining IIT-CDIP, arXiv-Papers, PowerPoint-Folien, open-domain digitale PDFs und Website-Screenshots verwendet. IIT-CDIP, was im Gegensatz zu den anderen digitalen Daten ein Datensatz bestehend aus gescannten Dokumenten ist, machte 10% der Trainingsdaten aus.

A.9 DETAILS ZUM TABLETRANSFORMER (SIEHE 3.3)

A.9.1 *Training [51]*

DETR zeichnet sich durch das bipartite Matching beim Loss aus, durch den das Set der Ground Truth optimal dem Set der Vorhersagen zugeordnet wird. Jede der N Vorhersagen wird dabei mit genau einer Ground Truth (falls weniger als N Ground Truths vorhanden, werden diese mit kein-Objekt ($\varnothing$) gepadded) bipartit so zugeordnet. Dies wird als Finden der Permutation $\hat{\sigma} \in \mathfrak{S}$ des Vorhersagensets modelliert, sodass die Matching Cost zwischen dem Vorhersagenset und dem Ground-Truth-Set minimiert wird:

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}}{\operatorname{argmin}} \sum_{i=1}^{N} L_{match}(y_i, \hat{y}_{\sigma(i)}). \tag{16}$$

Dieser ist dabei als

$$L_{match}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{1}_{\{c_1 \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_1 \neq \varnothing\}} L_{Box}(b_i, \hat{b}_{\sigma(i)}) \tag{17}$$

definiert [12], wobei y das Ground-Truth-Set, $\hat{y}$ das Vorhersagen-Set, $\sigma(i)$ der Index an Stelle i der Permuation σ, b die Ground-Truth-Bounding-Box und $\hat{b}_{\sigma(i)}$ die BoundingBox der Vorhersage mit Index $\sigma(i)$ ist. $\hat{p}_{\sigma(i)}(c_i)$ ist die Wahrscheinlichkeit der Ground Truth Klasse c_i für die Vorhersage mit Index $\sigma(i)$. $\mathbb{1}_{\{c_1 \neq \varnothing\}}$ ist die Indikatorfunktion, der Term ist also null, wenn die Ground-Truth-Klasse kein Objekt ist, und ansonsten 1. Die Matching Cost einer Vorhersage mit einer Kein-Object-Ground-Truth ist also immer null. Hier wird bewusst die Wahrscheinlichkeit direkt – und nicht die Log-Wahrscheinlichkeit – verwendet, da der Term so eine ähnliche Größenordnung wie 18 hat. Dies führt empirisch zu besseren Ergebnissen. Der Bounding-Box-Loss L_{Box} ist dabei eine Kombination aus Generalized IoU-Loss (2.5.1) und L1-Loss :

$$L_{Box}(b_i, \hat{b}_{\sigma(i)}) = \lambda_{IoU} L_{IoU}(b_i, \hat{b}_{\sigma(i)} + \lambda_{L1} |b_i - \hat{b}_{\sigma(i)}|_1), \tag{18}$$

wobei $\lambda_{IoU}, \lambda_{L1} \in \mathbb{R}$ Hyperparameter sind. Dies wurde deshalb gewählt, weil der L1-Loss nicht maßstabsunabhängig ist und die BoundingBoxen hier direkt und nicht als Offsets vorhergesagt werden, sodass dadurch der L1-Loss auch bei ähnlichen relativen Fehlern für große Bounding-Boxes unverhältnismäßig höher ist. Der IoU-Loss ist dagegen maßstabsunabhängig und soll diesen Aspekt ausgleichen. Als Loss-Funktion wird dann der Hungarian Loss über alle Matched Pairs von der Optimierung berechnet, eine lineare Kombination von negative Log-Likelihood (A.6.1) als Klassifier-Loss und dem oben definierten Box-Loss:

$$L_{\text{Hungarian}(y,\hat{y})=\sum_{i=1}^{N}} -\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_1 \neq \varnothing\}} L_{Box}(b_i, \hat{b}_{\hat{\sigma}(i)}). \tag{19}$$

Auch hier wird AdamW (2.1.2) als Optimierer verwendet.

Im Paper werden zwei verschiedene TableTransformers trainiert, einer für die Tabellenerkennung und einer für die Tabellenstrukturerkennung (auf ausgeschnittenen Tabellen). Für letzteren werden als Strukturkategorien „Tabelle“, „Zeile“, „Spalte“, „Zeilen-Kopfzeile“ und „Spalten-Kopfzeile“ definiert. Zellen werden nicht direkt erkannt sondern nur implizit als Schnittmenge von Zeilen und Spalten definiert.

B

ERGÄNZUNGEN ZUM EXPERIMENTALTEIL

B.1 ZUSÄTZLICHE TABELLEN

Tabelle 13: Ergebnisse auf Faster R-CNN mit auf Testdatensatz berechneten Filtering-Thresholds, an IoU und IoD Thresholds {0.5,0.6,0.7,0.8,0.9}, nur Tabellengebiet evaluiert (Vgl. ungefiltert siehe Tabelle 3; Vgl. Valid-Filterthreshold siehe Tabelle 11)

dataset	subset over WTW	wf1		testset size	files w/o valid pred
		IoU	IoD		
BonnTables 2.4.1	-	0.410414	0.593694	53	0
GloSAT	-	0.560898	0.718319	41	0
WTW (2.4.3)	Curved	0.833895	0.914630	427	0
	extremeratio	0.949324	0.973048	1525	0
	inclined	0.971978	0.981265	670	0
	muticolorandgrid	0.677738	0.705534	741	0
	occblu	0.778536	0.838583	72	0
	overlaid	0.908779	0.931335	75	2
	simple	0.999540	1	93	0

Tabelle 14: Ergebnisse von TableTransformer (3.3) mit auf Valid-Datensatz berechnetem Threshold und Evaluation auf Zellen (Vgl. ungefiltert siehe Tabelle 3, Beispielbilder siehe Abbildung 27)

dataset	subset WTW	wf1		testset size	files w/o valid pred
		IoU	IoD		
BonnTables	-	alle	0.069711	0.176360	53	1
		nur in tabellengebiet	0.070850	0.178070		
GloSAT	-	alle	0.217855	0.382060	41	0
		in tabellengebiet	0.220788	0.385907		2
Wired Table in the Wild (WTW) (2.4.3)	Curved	alle	0.384236	0.624418	427	0
		nur in tabellengebiet	0.390000	0.632214		
	extremeratio	alle	0.153967	0.233835	1525	0
		nur in tabellengebiet	0.154809	0.235568		
	inclined	alle	0.326184	0.370911	670	0
		nur in tabellengebiet	0.326913	0.371668		
	muticolorandgrid	alle	0.190953	0.295859	741	0
		nur in tabellengebiet	0.192323	0.298156		1
	occblu	alle	0.097282	0.146207	72	0
		nur in tabellengebiet	0.098312	0.147900		
	overlaid	alle	0.103035	0.281496	75	0
		nur in tabellengebiet	0.099878	0.274292	75	1
	simple	alle	0.051954	0.057768	93	0
		nur in tabellengebiet	0.051954	0.057768	93	0

Tabelle 15: Ergebnisse von TableTransformer (3.3) auf GloSAT-Datensatz mit auf Test-Datensatz berechnetem Filtering-Threshold nur in Tabellengebiet

	wf1		testset size	w/o valid pred
	IoU	IoD		
Cell	0.211835	0.390883	41	0
Row/Col	0.484324	0.663944	41	2

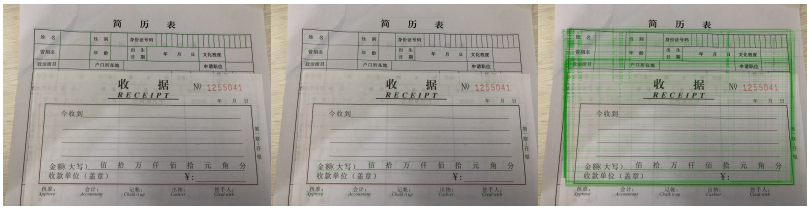
Tabelle 16: DBScan-Postprocessessing (5.2.2) (ohne Filtering)

dataset	subset WTW	model	wf1		testset size	images with valid pred after postprocessing
			IoU	IoD		
BonnTables	-	Kosmos-2.5	0.012618	0.263844	53	43
		Faster R-CNN	0.061998	0.154821		53
		TableTransformer	0.000407	0.010553		53
GloSAT	-	Kosmos-2.5	0.001768	0.489187	41	40
		Faster R-CNN	0.467734	0.676925		41
		TableTransformer	0.001005	0.007065		41
Wired Table in the Wild (WTW) (2.4.3)	Curved	Kosmos-2.5	0.023648	0.561111	427	420
		Faster R-CNN	0.259072	0.444238		426
		TableTransformer	0.001764	0.014038		427
	extremeratio	Kosmos-2.5	0.005960	0.732489	1525	1426
		Faster R-CNN	0.092625	0.360748		1525
		TableTransformer	0.000137	0.002867		1525
	inclined	Kosmos-2.5	0.006991	0.604133	670	630
		Faster R-CNN	0.253149	0.536629		670
		TableTransformer	0.002648	0.005473		670
	muticolorandgrid	Kosmos-2.5	0.017925	0.359059	741	624
		Faster R-CNN	0.566298	0.742080		741
		TableTransformer	0.000204	0.001895		586
	occblu	Kosmos-2.5	0.009074	0.204815	72	58
		Faster R-CNN	0.243798	0.419772		72
		TableTransformer	0.000103	0.000425		72
	overlaid	Kosmos-2.5	0.002966	0.468743	75	58
		Faster R-CNN	0.189791	0.346628		71
		TableTransformer	0.000568	0.006048		75
	simple	Kosmos-2.5	0.000377	0.804184	93	87
		Faster R-CNN	0.080254	0.500104		93
		TableTransformer	0.000000	0.000144		90

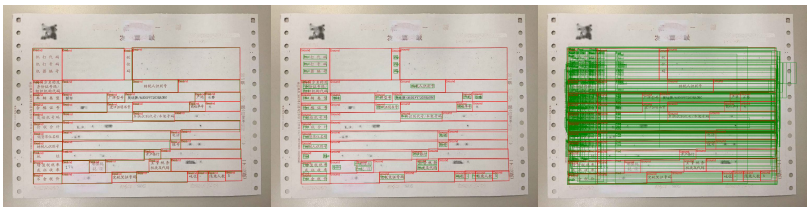
B.2 ZUSÄTZLICHE BEISPIELBILDER

Tabelle 17: DBScan-Postprocessessing (5.2.2) Ergebnisse mit Ground-Truth-Tabellen

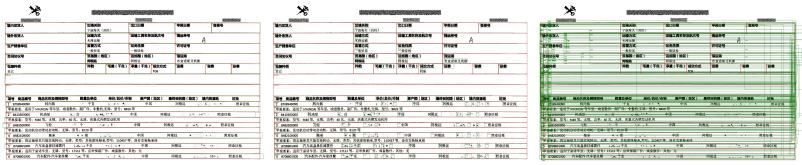
dataset	subset WTW	network	wf1 IoU with table ground truth	testset size
BonnTables	-	Kosmos-2.5	0.016553	53
		Faster R-CNN	0.072176	
GloSAT	-	Kosmos-2.5	0.002030	41
		Faster R-CNN	0.523963	
Wired Table in the Wild (WTW) (2.4.3)	Curved	Kosmos-2.5	0.031006	427
		Faster R-CNN	0.217465	
	extremeratio	Kosmos-2.5	0.007249	1525
		Faster R-CNN	0.124228	
	inclined	Kosmos-2.5	0.014315	670
		Faster R-CNN	0.274064	
	muticolorandgrid	Kosmos-2.5	0.042109	741
		Faster R-CNN	0.572369	
	occblu	Kosmos-2.5	0.009970	72
		Faster R-CNN	0.266087	
	overlaid	Kosmos-2.5	0.020538	75
		Faster R-CNN	0.158344	
	simple	Kosmos-2.5	0.004426	93
		Faster R-CNN	0.115144	



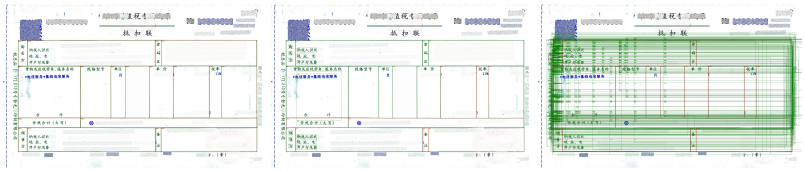
(a) overlaid Faster R-CNN (b) overlaid Kosmos-2.5 (c) overlaid TableTransformer (cells)



(d) Inclined Faster R-CNN (e) Inclined Kosmos-2.5 (f) Inclined TableTransformer (cells)



(g) extremeratio Faster R-CNN (h) extremeratio Kosmos-2.5 (i) extremeratio TableTransformer (cells)



(j) simple Faster R-CNN (k) simple Kosmos-2.5 (l) simple TableTransformer (cells)

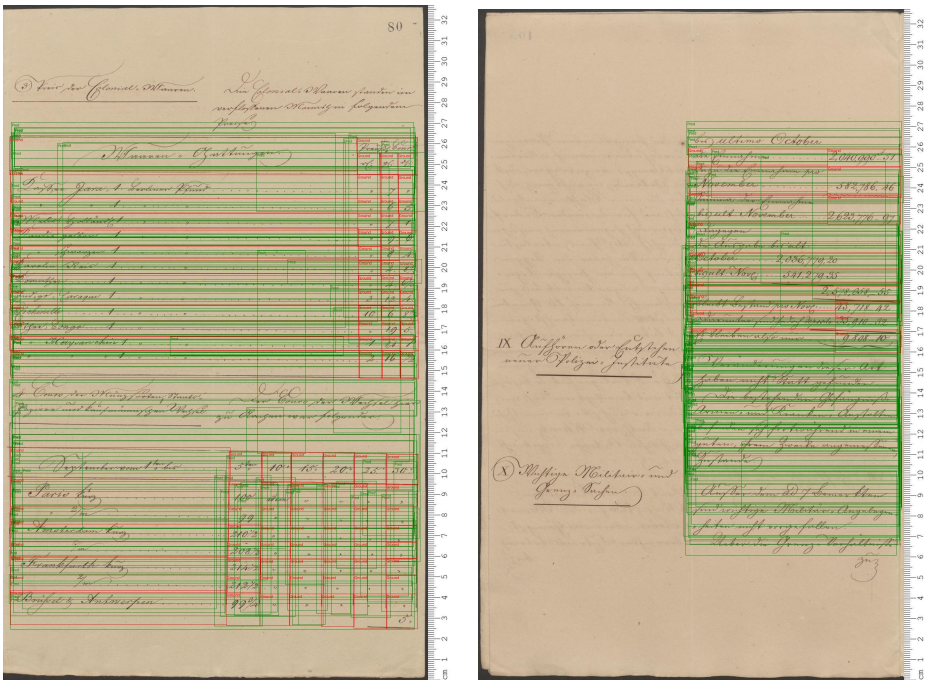
Abbildung 24: Beispielbilder der Ergebnisse auf dem Wired Table in the Wild (WTW)-Datensatz (2.4.3) (GroundTruth in rot)

(m) curved Faster R-CNN (n) curved Kosmos-2.5 (o) curved TableTransformer (cells)

(p) muticolorandgrid Faster R-CNN (q) muticolorandgrid Kosmos-2.5 (r) muticolorandgrid TableTransformer (cells)

(s) occblu Faster R-CNN (t) occblu Kosmos-2.5 (u) occblu TableTransformer (cells)

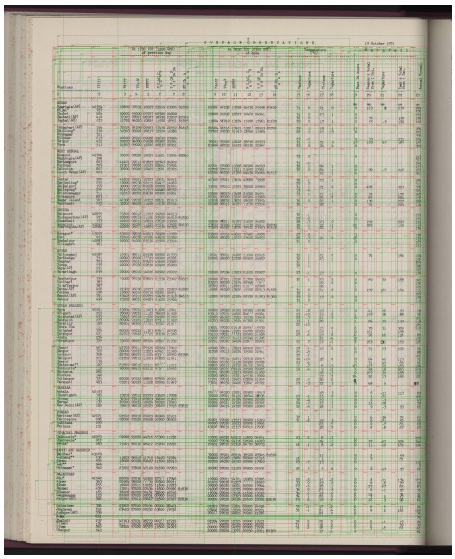
Abbildung 24: Beispielbilder der Ergebnisse auf dem Wired Table in the Wild (WTW)-Datensatz (2.4.3) (GroundTruth in rot) (fortgesetzt)



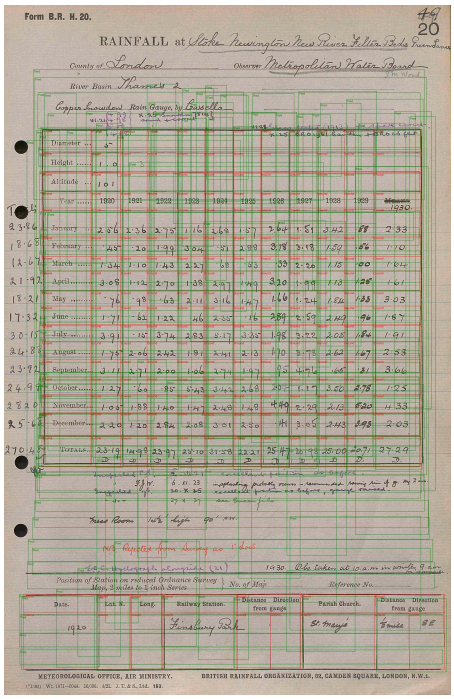
(a) Zeilen/Spalten (b) Zeilen/Spalten

BonnTables 2.4.1

Abbildung 25: Beispielbilder der Zeilen/Spaltenerkennung mit TableTransformer (GroundTruth in rot)



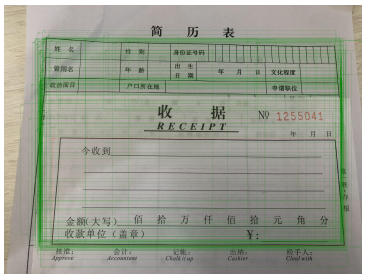
(c) Zeilen/Spalten



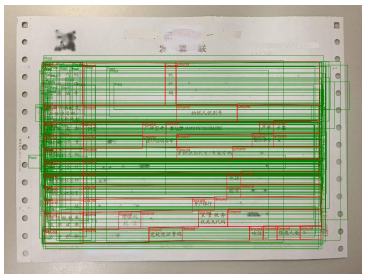
(d) Zeilen/Spalten

GloSAT 2.4.2

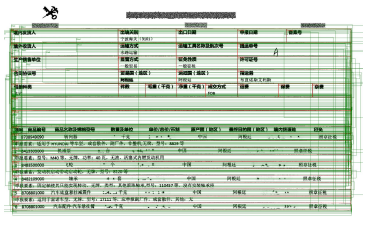
Abbildung 25: Beispielbilder der Zeilen/Spaltenerkennung mit TableTransformer (GroundTruth in rot) (fortgesetzt)



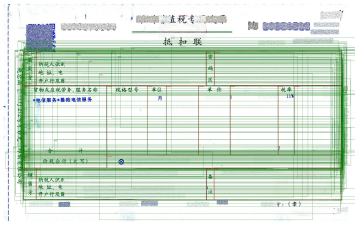
(e) overlaid Zeilen/Spalten



(f) Inclined Zeilen/Spalten



(g) extremeratio Zeilen/Spalten

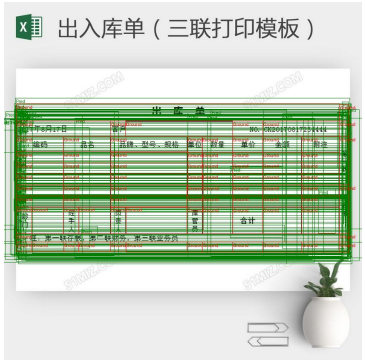


(h) simple Zeilen/Spalten

Wired Table in the Wild (WTW) 2.4.3

Abbildung 25: Beispielbilder der Zeilen/Spaltenerkennung mit TableTransformer (GroundTruth in rot) (fortgesetzt)

(i) curved Zeilen/Spalten

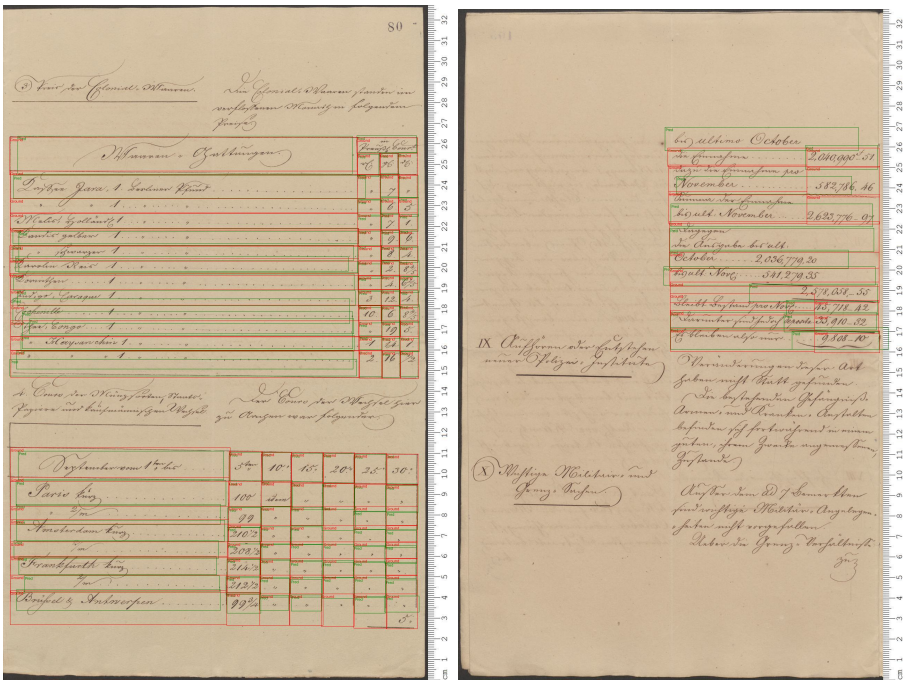


(j) muticolorandgrid Zeilen/Spalten

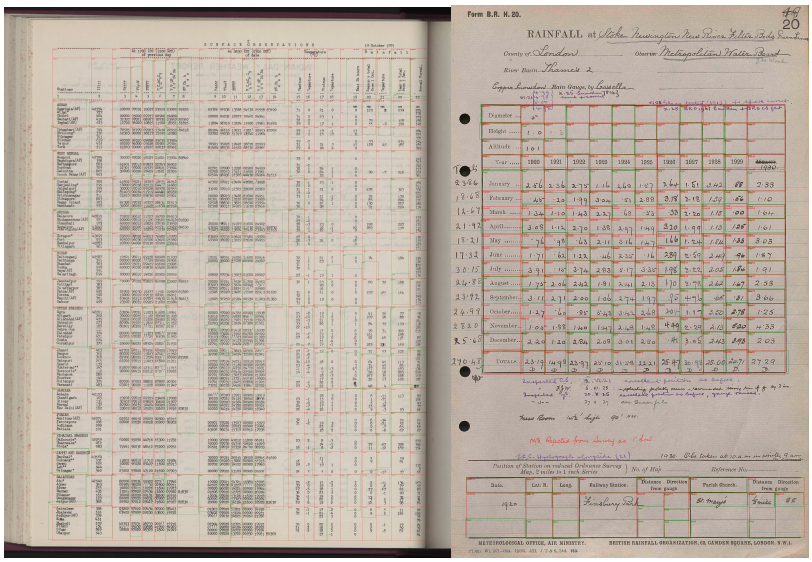
(k) occblu Zeilen/Spalten

Wired Table in the Wild (WTW) 2.4.3

Abbildung 25: Beispielbilder der Zeilen/Spaltenerkennung mit TableTransformer (GroundTruth in rot) (fortgesetzt)

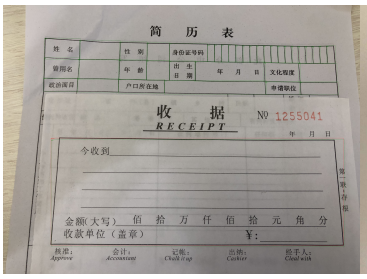


BonnTables 2.4.1

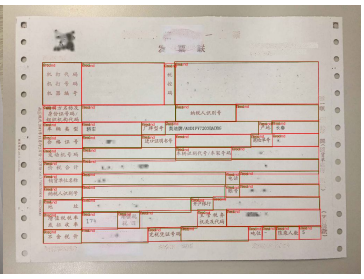


GloSAT 2.4.2

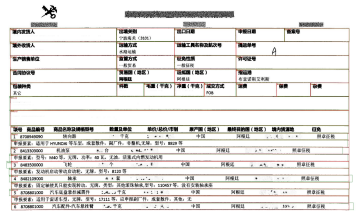
Abbildung 26: Gefilterte Beispielbilder aus den Testdatensätzen mit Faster R-CNN (Filterthreshold auf Valid-Datensatz berechnet, GroundTruth in rot) (fortgesetzt)



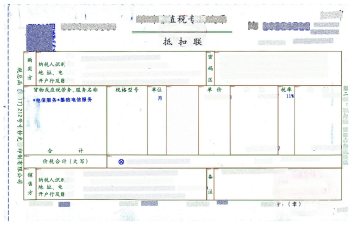
(e) overlaid



(f) Inclined



(g) extremeratio



(h) simple

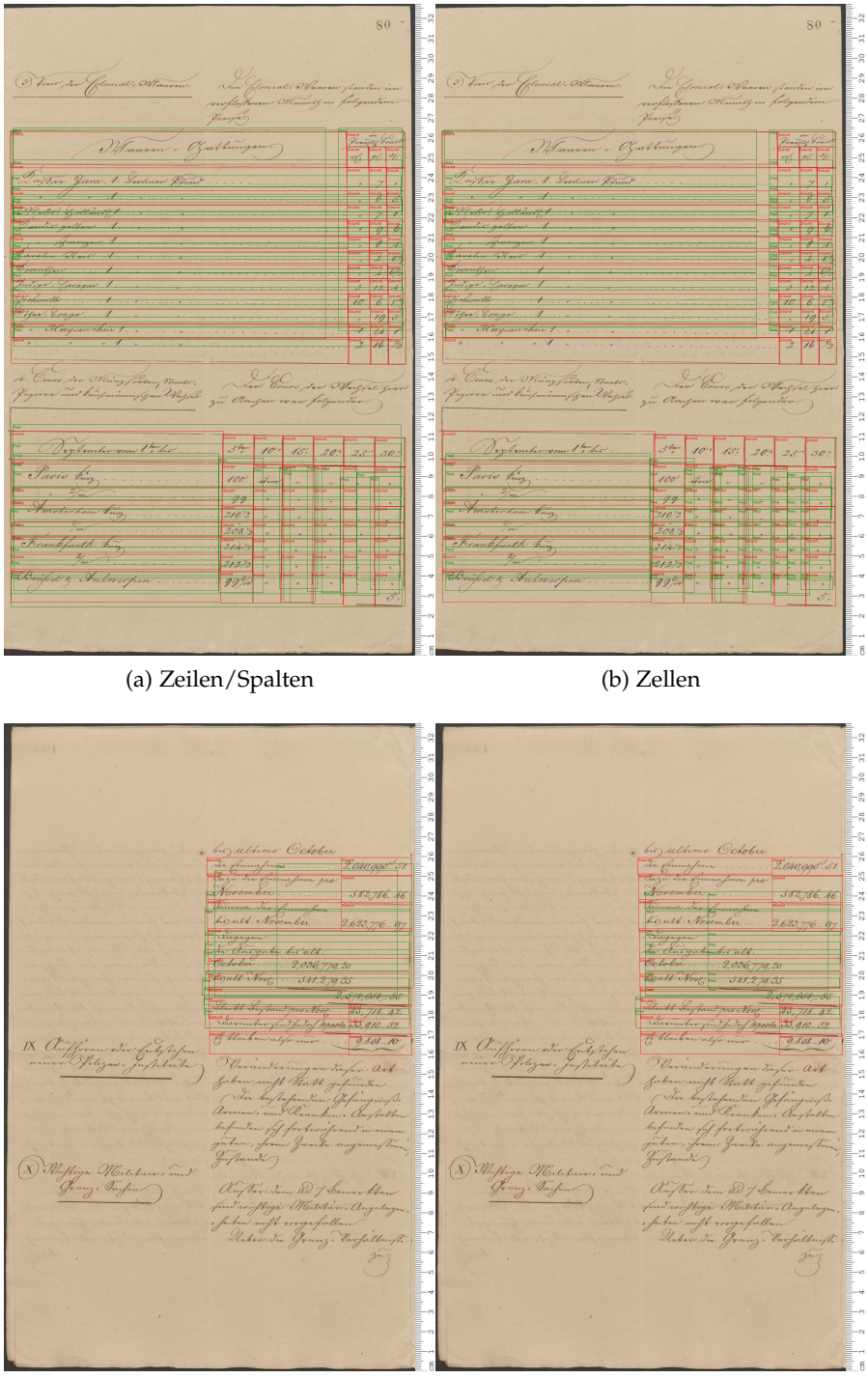
(i) curved

(j) muticolorandgrid

(k) occblu

Wired Table in the Wild (WTW) 2.4.3

Abbildung 26: Gefilterte Beispielbilder aus den Testdatensätzen mit Faster R-CNN (Filterthreshold auf Valid-Datensatz berechnet, GroundTruth in rot) (fortgesetzt)

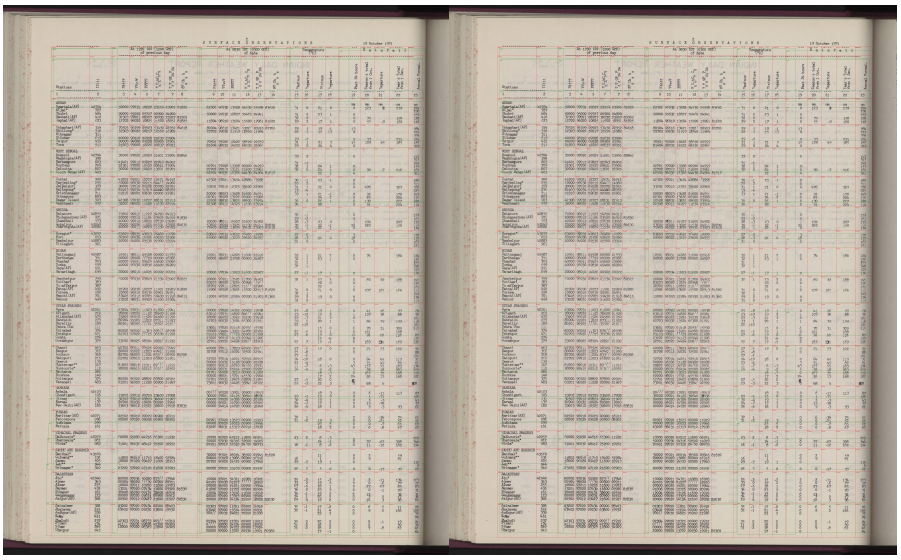


(a) Zeilen/Spalten (b) Zellen

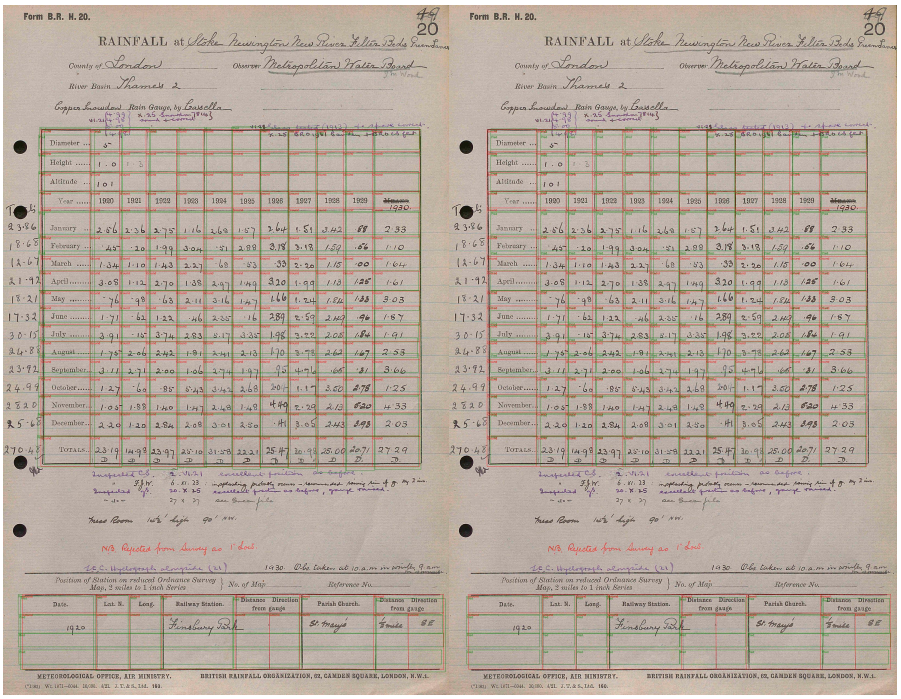
(c) Zeilen/Spalten (d) Zellen

BonnTables 2.4.1

Abbildung 27: Gefilterte Beispielbilder aus den Testdatensätzen mit Table-Transformer (Filterthreshold auf Valid-Datensatz berechnet, GroundTruth in rot)



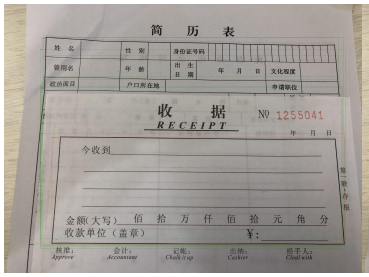
(e) Zeilen/Spalten (f) Zellen



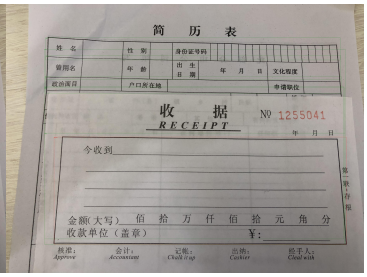
(g) Zeilen/Spalten (h) Zellen

GloSAT 2.4.2

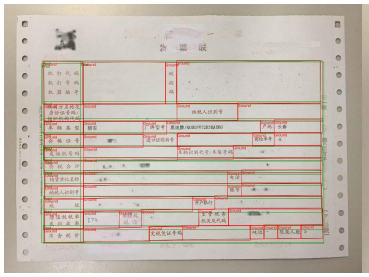
Abbildung 27: Gefilterte Beispielbilder aus den Testdatensätzen mit Table-Transformer (Filterthreshold auf Valid-Datensatz berechnet, GroundTruth in rot) (fortgesetzt)



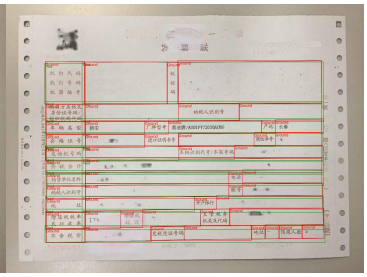
(i) overlaid Zeilen/Spalten



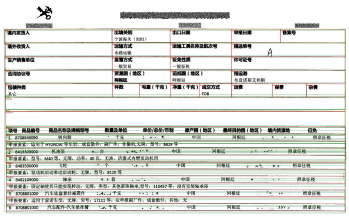
(j) overlaid Zellen



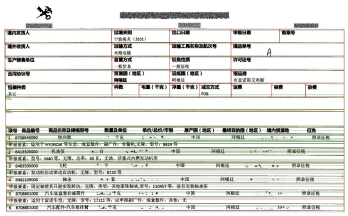
(k) Inclined Zeilen/Spalten



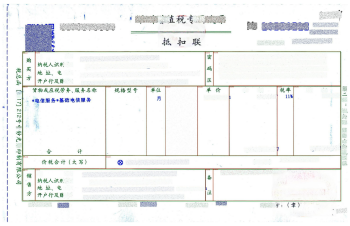
(l) Inclined Zellen



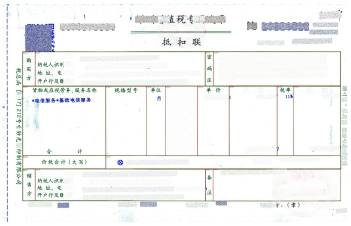
(m) extremeratio Zeilen/Spalten



(n) extremeratio Zellen



(o) simple Zeilen/Spalten



(p) simple Zellen

Wired Table in the Wild (WTW) 2.4.3

Abbildung 27: Gefilterte Beispielbilder aus den Testdatensätzen mit Table-Transformer (Filterthreshold auf Valid-Datensatz berechnet, GroundTruth in rot) (fortgesetzt)

(q) curved Zeilen/Spalten (r) curved Zellen

(s) muticolorandgrid Zeilen/Spalten (t) muticolorandgrid Zellen

(u) occblu Zeilen/Spalten (v) occblu Zellen

Wired Table in the Wild (WTW) 2.4.3 (fortgesetzt)

Abbildung 27: Gefilterte Beispielbilder aus den Testdatensätzen mit Table-Transformer (Filterthreshold auf Valid-Datensatz berechnet, GroundTruth in rot) (fortgesetzt)

C

HISTORISCHE ZEITUNGSDATENSÄTZE

C.1 GENUTZTE DATENSÄTZE

C.1.1 *American Stories [16]*

Der American Stories Datensatz von Melissa Dell besteht aus rund 20M historischen Zeitungsseiten aus den Jahren 1780-1940, mit einem Fokus auf Zeitungen von vor 1920.

Es sind Zeitungen aus allen 50 US-Staaten vorhanden.

Das Erstellen eines Datensates in dieser Größenordnung war mithilfe von OCR und Deep Learning Methoden möglich. Der Text auf den Seiten wird automatisch bezüglich seiner Lesbarkeit eingeordnet.

	(1) Total Boxes	(2) Articles	(3) Text Bounding Boxes Headlines	(4) Captions	(5) Bylines	(6) Images	(7) Other Bounding Boxes Ads	(8) Tables	(9) Mastheads
Legible	-	335M	368M	9.7M	14.7M	-	-	-	-
Illegible	-	26M	27M	0.9M	2.5M	-	-	-	-
Borderline	-	77M	22M	1.3M	1.2M	-	-	-	-
Total	1.14B	438M	417M	11.9M	18.4M	9.1M	221M	16.3M	4.9M

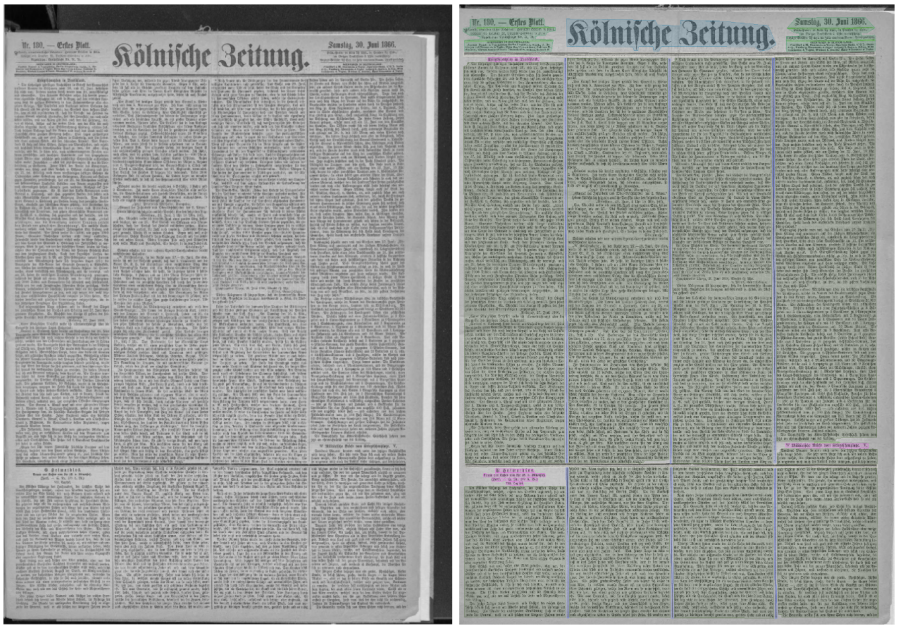
Datensatzstatistik [16]

Annotationsbeispiel [16]

C.1.2 *Chronicling Germany Zeitungsdatensatz [50]*

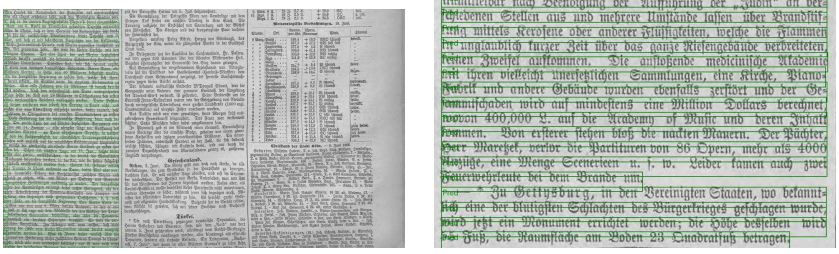
Bei Chronicling Germany handelt es sich um einen wesentlich kleineren Datensatz von rund 520 annotierten Seiten aus den Jahren aus 1866 und 1924.

Die verschiedenen annotierten Kategorien beinhalten article, header, heading, caption, separators, table, image und advertisements.



Annotationsbeispiel [50]

C.2 AS-IS-ANWENDUNG VON KOSMOS-2.5 AUF AMERICAN STORIES UND CHRONICLING GERMANY ZEITUNGSDATENSÄTZEN



(a) (b)

Abbildung 28: Kosmos-2.5 auf Zeitungscrops aus C.1.2

Auf dem **American-Stories-Datensatz** (C.1.1) gelangt Kosmos-2.5 schnell in Wiederholungsschleifen, wahrscheinlich da die Seiten sehr groß sind und die Spaltenstruktur von historischen Zeitungen eine Herausforderung darstellt. Allerdings gelingt die Textzeilenerkennung auf einer ausgeschnittenen Zeile des **Chronicling Germany Datensatzes** (Abbildung 28), eine grundsätzliche Verarbeitung der Seiten ist also möglich.

Die gewünschte Textblockerkennung und Kategorisierung ist hier nicht möglich, da das Modell auf reine Zeilenerkennung trainiert wurde und keine Kategorisierung der ausgegebenen BoundingBoxes vorgesehen ist.

Zusammengefasst ist eine Anwendung des Modells auf historischen Zeitungsseiten as-is nicht möglich, hier würde ein reines Finetuning eventuell nicht ausreichen, da die gewünschte Ausgabe (Markierung und Kategorisierund der Zeitungselemente, besonders Anzeigen (bestenfalls mit Text)) sich stark von dem Trainingsziel des vortrainieren Modells (einzelne BoundingBoxes für die Textzeilen) unterscheidet.

ACRONYME

OCR Optical Character Recognition (dt. optische Zeichenerkennung)

CNN Convolutional Neural Network (dt. Faltungsnetzwerk)

VIT Vision Transformer

ADAM Adaptive Moments

I.I.D. unabhängig und identisch verteilten (eng. independent and identically distributed)

SWIN Shifted Window

RNN Recurrent Neural Network (dt. rückgekoppeltes neuronales Netz)

LLM Large Language Model

MLLM Multimodal Large Language Model

LMM Large Multimodal Model

GPT generative vortrainierte Transformer (eng. Generative Pre-trained Transformers)

SGD Stochastic Gradient Descent (dt. Stochastischer Gradientenabstieg)

DCT Discrete Cosine Transform (dt. Diskrete Kosinustransformation)

R-CNN Region-based Convolutional Neural Network

RPN Region Proposal Network (dt. Regionsvorschlagsnetzwerk)

ROI Region of Interest

RESNET Residual Network

IOU Intersection over Union

IOD Intersection over Detection

WF1 Weighted Average F1 Score (dt. Gewichteter F1-Score)

WTW Wired Table in the Wild

GLOSAT Global Surface Air Temperature

DETR Detection Transformer

TRACE Table Reconstruction Aligned to Corner and Edges

FEATURE MAP Feature Map (dt. Merkmalskarte)

HRNET High-Resolution Net

[1] In: aufgerufen am 18.10.24. URL: https://de.wikipedia.org/wiki/Skalarprodukt.

[2] Wissam AlKendi, Franck Gechter, Laurent Heyberger und Christophe Guyeux. "Advancements and Challenges in Handwritten Text Recognition: A Comprehensive Survey". In: *Journal of Imaging* 10.1 (2024). ISSN: 2313-433X. DOI: 10.3390/jimaging10010018. URL: https://www.mdpi.com/2313-433X/10/1/18.

[3] Jay Alammar. "The illustrated transformer (Blog Post)". In: June 27, 2018. URL: https://jalammar.github.io/illustrated-transformer/.

[4] Jean-Baptiste Alayrac u. a. *Flamingo: a Visual Language Model for Few-Shot Learning*. 2022. arXiv: 2204.14198 [cs.CV]. URL: https://arxiv.org/abs/2204.14198.

[5] Fouzia Altaf, Syed Islam, Naveed Akhtar und Naeem Janjua. *Going Deep in Medical Image Analysis: Concepts, Methods, Challenges and Future Directions*. Feb. 2019. DOI: 10.48550/arXiv.1902.05655.

[6] Sofia Ares Oliveira, Benoit Seguin und Frederic Kaplan. "dhSegment: A Generic Deep-Learning Approach for Document Segmentation". In: *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, Aug. 2018. DOI: 10.1109/icfhr-2018.2018.00011. URL: http://dx.doi.org/10.1109/ICFHR-2018.2018.00011.

[7] Dzmitry Bahdanau, Kyunghyun Cho und Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[8] Emily Ann Belli, Jeff Candy, Igor Sfiligoi und Frank Würthwein. "Comparing single-node and multi-node performance of an important fusion HPC code benchmark". In: *Practice and Experience in Advanced Research Computing*. PEARC '22. ACM, Juli 2022, S. 1–4. DOI: 10.1145/3491418.3535130. URL: http://dx.doi.org/10.1145/3491418.3535130.

[9] Matthew N. Bernstein. "Matrix multiplication (Blog Post)". In: December 26, 2020. URL: https://mbernste.github.io/posts/matrix_multiplication/.

[10] Lukas Blecher, Guillem Cucurull, Thomas Scialom und Robert Stojnic. *Nougat: Neural Optical Understanding for Academic Documents*. 2023. arXiv: 2308.13418 [cs.LG].

[11] Nadav Borenstein, Phillip Rust, Desmond Elliott und Isabelle Augenstein. *PHD: Pixel-Based Language Modeling of Historical Documents*. 2023. arXiv: 2310.18343 [cs.CL].

[12] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov und Sergey Zagoruyko. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV]. URL: https://arxiv.org/abs/2005.12872.

[13] Wei-Lin Chiang u. a. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

[14] "Conv2d Dokumentation". In: URL: https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html.

[15] Brian Davis, Bryan Morse, Bryan Price, Chris Tensmeyer, Curtis Wigington und Vlad Morariu. *End-to-end Document Recognition and Understanding with Dessurt*. 2022. arXiv: 2203.16618 [cs.CV].

[16] Melissa Dell. In: URL: https://dell-research-harvard.github.io/resources/americanstories.

[17] Alexey Dosovitskiy u. a. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[18] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang und Qi Tian. *CenterNet: Keypoint Triplets for Object Detection*. 2019. arXiv: 1904.08189 [cs.CV]. URL: https://arxiv.org/abs/1904.08189.

[19] "F1-Score". In: Accessed: 2024-03-08. URL: https://en.wikipedia.org/wiki/F-score.

[20] "Faster R-CNN Erklärung". In: Accessed: 2024-03-08. URL: https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022.

[21] "Faster R-CNN Implementation". In: Accessed: 2024-03-08. URL: https://pytorch.org/vision/stable/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html#torchvision.models.detection.fasterrcnn_resnet50_fpn.

[22] "Faster R-CNN bei learn.microsoft". In: Accessed: 2024-03-08. URL: https://learn.microsoft.com/de-de/cognitive-toolkit/object-detection-using-faster-r-cnn.

[23] Hao Feng, Qi Liu, Hao Liu, Wengang Zhou, Houqiang Li und Can Huang. *DocPedia: Unleashing the Power of Large Multimodal Model in the Frequency Domain for Versatile Document Understanding*. 2023. arXiv: 2311.11810 [cs.CV].

[24] Ross B. Girshick. "Fast R-CNN". In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[25] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren und Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[27] "Jaccard-Koeffizient". In: Accessed: 2023-11-13. URL: https://de.wikipedia.org/wiki/Jaccard-Koeffizient.

[28] Fototechnischen Ausschuss der KLA. "Wirtschaftliche Digitalisierung in Archiven". In: 2016. URL: https://www.bundesarchiv.de/assets/bundesarchiv/de/Downloads/Beitraege/wirtschaftliche-digitalisierung.pdf.

[29] Andrej Karpathy. "CS231n Convolutional Neural Networks for Visual Recognition (Vorlesungsfolien)". In: *CS231n: Deep Learning for Computer Vision*. Stanford University, 2015. URL: https://cs231n.github.io/convolutional-networks/#conv.

[30] Mitesh M. Khapra. "Introduction to Large Language Models (Week 2 Slides)". In: *Introduction to Large Language Models*. Course from AI4Bharat, Department of Computer Science and Engineering, IIT Madras. 2023-2024. URL: https://iitm-pod.slides.com/arunprakash_ai/transformers-a-short-version/fullscreen#/0/50.

[31] Geewook Kim, Teakgyu Hong, Moonbin Yim, Jeongyeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han und Seunghyun Park. *OCR-free Document Understanding Transformer*. 2022. arXiv: 2111.15664 [cs.LG].

[32] Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang und Kristina Toutanova. *Pix2Struct: Screenshot Parsing as Pretraining for Visual Language Understanding*. 2023. arXiv: 2210.03347 [cs.CL].

[33] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou und Zhoujun Li. *TableBank: A Benchmark Dataset for Table Detection and Recognition*. 2019. arXiv: 1903.01949 [cs.CV].

[34] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He und Piotr Dollár. "Focal Loss for Dense Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, S. 2999–3007. DOI: 10.1109/ICCV.2017.324.

[35] Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier und Julian Martin Eisenschlos. *MatCha: Enhancing Visual Language Pretraining with Math Reasoning and Chart Derendering*. 2023. arXiv: 2212.09662 [cs.CL].

[36] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis und Luke Zettlemoyer. *Multilingual Denoising Pre-training for Neural Machine Translation*. 2020. arXiv: 2001.08210 [cs.CL].

[37] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin und Baining Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV].

[38] Ilya Loshchilov und Frank Hutter. "Fixing Weight Decay Regularization in Adam". In: *CoRR* abs/1711.05101 (2017). arXiv: 1711.05101. URL: http://arxiv.org/abs/1711.05101.

[39] Tengchao Lv u. a. *Kosmos-2.5: A Multimodal Literate Model*. 2023. arXiv: 2309.11419 [cs.CL].

[40] Stuart Middleton und Juliusz Ziomek. "GloSAT Historical Measurement Table Dataset: Enhanced table structure recognition annotation for downstream historical data rescue". In: *International Workshop on Historical Document Imaging and Processing (05/09/21 - 06/09/21)*. 2021. URL: https://eprints.soton.ac.uk/450279/.

[41] Annika Thelen (ich) Niklas Kerkfeld. In: URL: https://github.com/Digital-History-Bonn/HistorischeTabellenSemanticExtraction.

[42] Devashish Prasad, Ayan Gadpal, Kshitij Kapadni, Manish Visave und Kavita Sultanpure. *CascadeTabNet: An approach for end to end table detection and structure recognition from image-based documents*. 2020. arXiv: 2004.12629 [cs.CV].

[43] Read-Coop. In: URL: https://readcoop.eu/introducing-table-models-trainable-layout-ai-in-transkribus/.

[44] Read-Coop. In: URL: https://help.transkribus.org/tables#how-to-train-a-table-model.

[45] Shaoqing Ren, Kaiming He, Ross Girshick und Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[46] O. Ronneberger, P.Fischer und T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Bd. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, S. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

[47] Long Rujiao, Wang Wen, Xue Nan, Gao Feiyu, Yang Zhibo, Wang Yongpan und Xia Gui-Song. "Parsing Table Structures in the Wild". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021.

[48] Phillip Rust, Jonas F. Lotz, Emanuele Bugliarello, Elizabeth Salesky, Miryam de Lhoneux und Desmond Elliott. *Language Modelling with Pixels*. 2023. arXiv: 2207.06991 [cs.CL].

[49] Mahmoud Salaheldin Kasem, Abdelrahman Abdallah, Alexander Berendeyev, Ebrahem Elkady, Mohamed Mahmoud, Mahmoud Abdalla, Mohamed Hamada, Sebastiano Vascon, Daniyar Nurseitov und Islam Taj-Eddin. "Deep learning for table detection and structure recognition: A survey". In: *ACM Computing Surveys* 56.12 (2024), S. 1–41.

[50] Christian Schultze, Niklas Kerkfeld, Kara Kuebart, Princilia Weber, Moritz Wolter und Felix Selgert. *Reading yesterday's news. Layout recognition by segmentation of historical newspaper pages*. 2024. arXiv: 2401.16845 [cs.DL].

[51] Brandon Smock, Rohith Pesala und Robin Abraham. *PubTables-1M: Towards comprehensive table extraction from unstructured documents*. 2021. arXiv: 2110.00061 [cs.LG]. URL: https://arxiv.org/abs/2110.00061.

[52] Ian Soboroff. "Complex Document Information Processing (CDIP) dataset, National Institute of Standards and Technology". In: 2022. URL: https://doi.org/10.18434/mds2-2531.

[53] Prof. Dr. Cyrill Stachniss. "Image Template Matching Using Cross Correlation". In: aufgerufen am 20.10.24. URL: https://www.ipb.uni-bonn.de/html/teaching/photo12-2021/2021-pho1-09-matching-cc.pptx.pdf.

[54] Jörg Tiedemann und Santhosh Thottingal. "OPUS-MT — Building open translation services for the World". In: *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*. Lisbon, Portugal, 2020.

[55] Carlo Tomasi. "Image Correlation, Convolution and Filtering". In: aufgerufen am 20.10.24. URL: https://courses.cs.duke.edu/fall15/cps274/notes/convolution-filtering.pdf.

[56] "Transkribus". In: Accessed: 2024-03-08. URL: https://readcoop.eu/de/transkribus/.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser und Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[58] Jesse Vig. "A Multiscale Visualization of Attention in the Transformer Model". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Florence, Italy: Association for Computational Linguistics, Juli 2019, S. 37–42. DOI: 10.18653/v1/P19-3007. URL: https://www.aclweb.org/anthology/P19-3007.

[59] W.Kössler. "Folien Stochastik: Moment und Varianz". In: Humboldt-Universität zu Berlin. URL: https://www2.informatik.hu-berlin.de/~koessler/Stochastik/Stochastik2006/FolienStochastik_11.pdf.

[60] Hongyu Wang u. a. *Foundation Transformers*. 2022. arXiv: 2210.06423 [cs.LG]. URL: https://arxiv.org/abs/2210.06423.

[61] Angchi Xu und Wei-Shi Zheng. "Efficient and Effective Weakly-Supervised Action Segmentation via Action-Transition-Aware Boundary Alignment". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.

[62] Jiabo Ye u. a. *UReader: Universal OCR-free Visually-situated Language Understanding with Multimodal Large Language Model*. 2023. arXiv: 2310.05126 [cs.CV].

[63] Annika Thelen (ich). "Neural-Network-Based-Table-Structure-Recognition". In: URL: https://github.com/Digital-History-Bonn/Neural-Network-Based-Table-Structure-Recognition.

[64] Annika Thelen (ich). "Tabellenerkennung mit Faster-R-CNN anhand von Preußischen Immediatzeitungsberichten". mein Projektgruppenbericht. 2024.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of 10. November 2024 (`classicthesis`).