

Analyse moderner Ansätze zur bildbasierten Klassifikation von Dokumenten

Hochschule Bochum
Studiengang: Informatik

vorgelegt am:
von: Florian Jung
aus: Hamm
Betreuer:

Inhaltsverzeichnis

Abbildungsverzeichnis	IV
1 Einführung	1
1.1 Situationsbeschreibung	1
1.2 Motivation und Forschungsfragen	1
1.3 Vorgehensweise	2
1.4 Relevanz der Thematik	2
2 Grundlagen	2
2.1 Optimierung	2
2.2 Klassifikation	3
2.2.1 Mustererkennung	3
2.3 Optical character recognition	4
2.4 Maschinelles lernen	5
2.4.1 Grundkonzept	5
3 Convolutional Neural Networks	6
3.1 Convolutional layer	7
3.2 Stride	7
3.3 Padding	8
3.4 Pooling Layer	8
4 Transformer	9
4.1 Attention	9
5 Überblick der Ansätze	10
6 Metriken für Evaluationen von Klassifikationen	12
6.1 Wahl der Informationen für Metriken	13
6.2 Wahl der Metriken	13
7 Daten für das Training	14
7.1 Aufteilung der Daten	15
8 Analyse des Datensatzes	15
8.1 Email	16
8.2 Quittungen	17
8.3 Notizen	18
8.4 Fragebögen	19
8.5 Spezifikationen	20
8.6 Datensatz im Bezug zur Wahl der Ansätze	21
9 Abwägung der Ansätze	21
9.1 Deep Learning als genereller Ansatz für Klassifikationen	21
9.2 Deep Learning als Ansatz für Bildklassifikationen	22

10 Wahl eines Ansatzes	24
10.1 LayoutLM	26
10.2 LayoutLMv3	28
11 Prognostizierung des Ergebnisses	32
12 Praktische Umsetzung	33
12.1 Vorbereitung der Entwicklungsumgebung	33
12.2 Beschaffung der Dokumentenbilder	33
13 Implementation von LayoutLMv3	33
13.1 Prüfung des CUDA Toolkits	34
13.2 Installation der Abhängigkeiten und Bibliotheken	34
13.3 Implementation des OCR Moduls	36
13.4 Pre Processing	40
13.5 Datensatzobjekt	40
13.6 Training	42
13.7 Testen	48
14 Anhang	50
14.0.1 Optimierungsprobleme	50
14.0.2 Gradienten	51
14.0.3 Binäre Klassifikationsprobleme	51
14.0.4 Multi-Klassen Klassifikationsprobleme	52
14.0.5 Multi-Label Klassifikationsprobleme	52
14.0.6 Künstliche Neuronen	52
14.0.7 Aktivierungsfunktion	53
14.0.8 Pre-Processing	55
14.0.9 Data Augmentation	56
14.1 Encoder	61
14.2 Decoder	61
14.2.1 Scaled Dot-Product Attention	62
14.2.2 Multi-Head Attention	63
14.3 Template basierende Methoden	63
14.4 Graph basierende Methoden	64
14.5 Handgemachte feature basierenden Methoden	65
14.6 Deep feature basierende Methode	66
14.7 Text und visuell basierende Methoden	67
14.8 Text und strukturbasierende Methoden	70
14.9 Erwartete Kosten	70
14.10 Verwechslungsmatrix	70
14.11 Conda	71
14.12 CUDA	72
15 Literaturverzeichnis	73
16 Eigenständigkeitserklärung	77

Abbildungsverzeichnis

1	Convolution	7
2	Max Pooling	8
3	Überblick der Ansätze	10
4	Überblick der Ansätze	13
5	Überblick der Ansätze	16
6	Überblick der Ansätze	17
7	Überblick der Ansätze	18
8	Überblick der Ansätze	19
9	Überblick der Ansätze	20
10	Darstellung mehrere Reinforcement Learning Iterationen in einem Rennspiel	22
11	DARTS Architektur	23
12	DARTS Architektur	25
13	DARTS Architektur	27
14	DARTS Architektur	28
15	LayoutLMv3 Architektur	29
16	LayoutLMv3 Architektur	31
17	LayoutLMv3 Architektur	34
18	LayoutLMv3 Architektur	37
19	LayoutLMv3 Architektur	39
20	LayoutLMv3 Architektur	42
21	LayoutLMv3 Architektur	45
22	LayoutLMv3 Architektur	45
23	LayoutLMv3 Architektur	46
24	LayoutLMv3 Architektur	47
25	LayoutLMv3 Architektur	48
26	Scaled Dot-Product Attention	62
27	Multi-Head Attention	63
28	Überblick der Ansätze	64
29	Überblick der Ansätze	64
30	Überblick der Ansätze	66
31	Text und visuell basierendes Modell	67
32	Überblick der Ansätze	68
33	Überblick der Ansätze	69
34	Überblick der Ansätze	69
35	Überblick der Ansätze	71

1 Einführung

1.1 Situationsbeschreibung

Industrien und Arbeitsumgebungen werden regelmäßig mit dem Begriff der “Digitalisierung” konfrontiert. Das Lagern und Verarbeiten von Unterlagen und Daten hat viele unterschiedlich Vorteile. Eines der Vorteile der Digitalisierung von Daten ist die effiziente Weiterverarbeitung, wie Manipulation, Sortierung und Verteilung. Große Unternehmen müssen sich mit einer zunehmenden Menge von Daten beschäftigen, was außerdem eine wachsende Herausforderung für das verantwortliche Personal ist. Dokumentenmanagementsysteme bieten durch Softwarelösungen eine Plattform in welcher Weiterverarbeitungen über ein Netzwerk möglich gemacht werden.

Die Art und Weise der Weiterverarbeitung steht häufig in Relation der Natur der zu verarbeiteten Daten. Das heißt der Inhalt der Daten spielt eine Rolle. Es muss also eine Beurteilung passieren, welche die Daten in gewisse Kategorien unterteilt. Das seit einigen Jahren wachsende Thema der “Künstlichen Intelligenz”, beziehungsweise des “Machine Learning” unterstützt bei solchen Thematiken. Diese Arbeit beschäftigt sich mit der Fragestellung wie der derzeitige Stand des “Machine Learning” beziehungsweise des “Deep Learning” genutzt werden kann um notwendige Arbeit wie Klassifizierungen effizient zu automatisieren.

1.2 Motivation und Forschungsfragen

Machine Learning baut auf mathematischen Modellen auf die in der Anwendung signifikante Vorteile in Klassifizierungsproblemen erbringen kann. Zusätzlich beschäftigt sich die Thematik “Deep Learning” mit dem Abstrahieren des Machine Learning, wobei die Modelle in eine Vielzahl sogenannter “Schichten”, oder “Layer” unterteilt werden. Das kann in der richtigen Umsetzung zu einer Lösung von komplexeren Problemen führen. In dieser Arbeit wird der derzeitige Stand der Forschung bearbeitet und “State-of-the-art” Lösungen untersucht und mit dieser Problemstellung verglichen.

1.3 Vorgehensweise

Maschinelles lernen bezieht sich auf mathematische Modelle in welchen durch “Training” das Modell angepasst wird. Bei einem Klassifizierungsproblem werden konkrete Objekte normiert durch alle Schichten des Modells getragen und anschließend wird der sogenannte “Fehler” ermittelt. Der Fehler wird aus dem erwarteten Ergebnis des Modells bekannt. Abhängig vom Fehler werden die “Gewichte” des Modells angepasst. Abstrahiert formuliert “lernt das Modell die Klassen kennen”.

Maschine Learning, und vor allem Deep Learning, benötigen eine große Menge an Daten. Diese Daten müssen außerdem der zugehörigen Klasse zugewiesen werden. Diese Daten werden teilweise aus dem eigenen Haus erlagt, also werden konkrete Daten genommen die auch für die Lösung genutzt werden. Außerdem werden öffentlich zugängliche “Datasets” genutzt. Diese Daten werden dann in drei unterschiedliche Mengen unterteilt. Das Training Set, Validation Set und das Test Set [1]. Sobald ein passendes Modell für die Problemstellung gefunden wurde wird dieses Modell antrainiert. Die Problemstellung bezieht sich auf die Dokumentenklassifizierung, daher werden Dokumente aller unterschiedlichen Typen, die für unsere Klassifizierung genutzt werden, mit dem zugehörigen “Label” mehrmals durch das Modell getragen um die Klassifizierung anzulernen. Nach der Lernphase beginnt die Testung und Validierung. Sobald diese Prozesse abgeschlossen sind werden Metriken genutzt und mit der Erwartung des Ziels verglichen. Reflektiert die Qualität des Modells zufriedenstellende Werte ist die Arbeit erfolgreich beendet.

1.4 Relevanz der Thematik

Der viel umfassende Begriff “Künstliche Intelligenz”

2 Grundlagen

2.1 Optimierung

Maschinelles Lernen verfolgt die Grundidee der Optimierung, bei der ein Problem definiert wird und einem Optimierungsprozess iterativ nach der “besten” Lösung gesucht

wird.

2.2 Klassifikation

Klassifikation ist Teil des Ansatzes “überwachtes Lernen”, auch bekannt als supervised learning [4]. Bei einer Klassifizierung werden Objekte genommen und anhand der Eigenschaften der Objekte, vordefinierte Klassen zugeordnet. Das Programm, welches diese Entscheidungen vornimmt, wird als Klassifikator bezeichnet. Ein Objekt erhält dann ein Label, was die Informationen darüber enthält, in welche Klasse der Klassifikator das Objekt zugewiesen hat.

Der Klassifikator in einem Klassifikationsproblem in diesem Kontext hat also die Funktion eine Entscheidung zu treffen, bei welcher die einzige veränderliche Variable die Eigenschaften des Objekts sind, da sich die Klassen in die sie eingeteilt werden sollen nicht ändern. Ein Beispiel für einen solchen Prozess ist die Entscheidung ob sich auf Bildern ein Hund oder eine Katze befindet. Dieser eine Klassifikator kann mehrere unterschiedliche Bilder untersuchen, die Klassen sind aber die gleichen.

Klassifizierungsprobleme werden in unterschiedlichen Arten unterschieden: Binäre Klassifikationsprobleme, Multi-Klassen Klassifikationsprobleme und Multi-Label Klassifikationsprobleme.

2.2.1 Mustererkennung

Der Klassifikator trifft die Entscheidung welche Labels Daten gegeben werden sollen. Um diese Aufgabe ausführen zu können muss der Klassifikator lernen unter welchen Umständen Daten die Bedingungen für die unterschiedlichen Labels erfüllen.

Der derzeitige Stand der Forschung hat viele unterschiedliche Ansätze untersucht, welche sich unter Anderem darunter Unterscheidet mit welchen Arten von Daten sich das Problem befasst. Die Gemeinsamkeiten der Ansätze ist, dass das Ziel das gleiche ist: Der Klassifikator soll durch Trainingsdaten ein Muster erkennen.

Beispielsweise wird durch ein sogenanntes “Training” eine Sequenz in einem Satz, oder Kanten in einem Bild gesucht. Mit einer Menge von Trainingsdaten die genügt, kann der Klassifikator Gemeinsamkeiten zwischen Daten erkennen, die in die selbe Klasse gehören. Diese Gemeinsamkeiten können auch “Muster” genannt werden. Es werden

also nicht alle Qualitäten einer Klasse gelernt, sondern nur das was ausreicht um diese Klassen zu unterscheiden.

Hat der Klassifikator diese Muster erlernt, kann er die Daten, die klassifiziert werden sollen auf diese Muster untersuchen.

2.3 Optical character recognition

Optical character recognition (OCR) ist ein System, das Text in einer Bilddatei erkennt und in ein Maschinenformat konvertiert [7]. Dadurch werden aus Pixelformaten, tatsächliche Zeichen in Formaten wie Buchstaben generiert.

OCR folgt der gleichen Grundidee wie Klassifizierung. Ein OCR System extrahiert sogenannte “Features” aus der Pixelrepräsentation. Features sind in diesem Kontext die Eigenschaften eines Objekts, welche dabei helfen das Objekt einer bestimmten Klasse zuzuordnen. Die Forschung beschäftigt sich unter anderem mit dem Themenbereich der handwritten OCR . Diese Form unterscheidet sich zwar nicht in der Grundidee des Systems, muss sich allerdings mit komplexeren Rahmenbedingungen auseinandersetzen. Beispielsweise kann die Schrift von Person zu Person variieren. Zwar kann auch die Schriftart von digitalen Dokumenten unterschiedlich sein, wie zum Beispiel Arial zu Comic Sans, aber innerhalb einer digitalen Schriftart bleiben alle identischen Zeichen optisch identisch. Handschrift bringt eine Varianz mit sich, bei der beispielsweise jedes “F” mit einer geringen Abweichung anders aussieht. Dieses Problem erhöht die notwendige Komplexität eines OCR Systems.

1965 wurde die “Advance reading machine IBM 1287” präsentiert, welche dazu in der Lage war handgeschriebene Zahlen zu identifizieren [8]. Moderne Forschung im Bereich des maschinellen Lernen hat sich mit vielen unterschiedlichen Ansätzen beschäftigt Modelle zu optimieren. Bekannte Ansätze sind Support Vector Machines, Random Forests, k Nearest Neighbor und Decision Trees. Außerdem wurden Ansätze untersucht, wie man Daten verarbeiten kann. Diese Ansätze kombiniert haben dabei geholfen und derzeitige Genauigkeit von OCR Systemen zu erhöhen.

Zum Beispiel können Farbformate in Graustufenformate konvertiert werden ource . Graustufenformate besitzen eine reduzierte Dimension als Farbformate. Rauschen ist

außerdem ein gängiges Problem, was die Erkennung von Zeichen erschweren könnte. Dafür gibt es Prozesse die Rauschen erkennen. Das Behandeln von einzelnen Segmenten, als beispielsweise einzelne Zeilen oder Sätzen kann außerdem die Genauigkeit erhöhen.

2.4 Maschinelles lernen

2.4.1 Grundkonzept

Diese Arbeit beschäftigt sich stark mit der Idee des “Lernens”. Lernen wird definiert als Erwerb von geistigen, körperlichen und sozialen Kenntnissen und Fertigkeiten. Dieser Erwerb wird unterteilt in absichtlichem, beiläufigem, individuellem oder kollektivem Erwerb [9]. Relevant ist, dass “Lernen” per Definition mit “Veränderung” in Relation steht: “Aus lernpsychologischer Sicht wird Lernen als ein Prozess der relativ stabilen Veränderung des Verhaltens, Denkens oder Fühlens, als verarbeitete Wahrnehmung der Umwelt oder Bewusstwerdung eigener Regungen, aufgefasst” -Prof. Dr. Thomas Bartscher Außerdem wird definiert, dass Lernen ein Prozess ist, der erfahrungsbezogen konstruiert.

Viele dieser Aspekte lassen sich beispielsweise in der Optimierung von Klassifikationsproblemen wiederfinden. Die Verarbeitung von Trainingsdaten repräsentiert den Aspekt der “Erfahrung” im Kontext des Lernens und die Anpassung der Parameter des Modells die “Veränderung”.

Die Forschung der Biologie hat sich mit der Fragestellung befasst, wie Menschen lernen. [10] Der derzeitige Stand der Wissenschaft hat Ansätze gefunden wo sich Gedanken oder Erinnerungen befinden und es ist möglich geworden Neuronen und deren Komponenten zu modellieren [11]. Dieser Fortschritt der mittlerweile detaillierten Darstellung der Funktionsweise des menschlichen Gehirns bringt außerdem die Möglichkeit diese Funktionsweise in theoretischen Modellen nachzustellen.

Die Idee ist, dass wenn das menschliche Lernen verstanden wurde, dieses Lernen künstlich nachzustellen. Das ist das Grundkonzept von künstlichen neuronalen Netzen im Bereich des maschinellen Lernen.

Das menschliche Gehirn besteht aus 100 Milliarden Nervenzellen [12], die auch Neuronen genannt werden. Neuronen haben Relevanz im Kontext des Lernprozesses da diese

vereinfacht definiert “Gewichte” speichern.

Das biologische Abbild eines Neurons ist deutlich komplexer als die programmatische Repräsentation.

Biologische Neuronen “kommunizieren” über elektrische Signale miteinander. Diese Signale haben die Form von kurzen Impulsen. Die Kreuzungen zwischen den Kanälen, welche die Neuronen verbinden werden durch “Synapsen” realisiert. Die Kanäle, die aus den Zellen herausragen werden “Dendriten” genannt.

Jedes Neuron hat eine direkte Verbindung zu mehreren tausenden weiteren Neuronen. Die elektrischen Signale, die zwischen Neuronen transportiert werden, werden auf den Neuronen bei denen die Signale ankommen aufsummiert. Im Biologischem Vorbild sind es also mehrere Tausend Signale, die die Summe des eingehenden Signals eines Neurons darstellt. Wenn die Summe der Signale eine gewisse Grenze überschreitet “feuert” das Neuron. Es sendet dann also ein Signal an die verbundenen folgende Neuronen über Kanäle die “Axon” genannt werden. Neuronen können mit eingehender Stimulierung ihre Eigenschaften ändern. Zum Beispiel können sich die Konditionen ändern, unter welchen ein Neuron feuert.

Da menschliche Gehirne die Fähigkeit aufweisen zu lernen, ist es eine Theorie, dass ein Modell, dass einem menschlichen Gehirn ähnelt, ähnliche Eigenschaften aufweist. Um eine ähnliche Architektur zu modellieren muss eine Struktur gefunden werden, die Neuronen und ihre Eigenschaften nachstellen.

3 Convolutional Neural Networks

Die Architektur von Convolutional Neural Networks (CNN) ist ähnlich wie die der künstlichen Neuronalen Netze. Die Netze bestehen aus künstlichen Neuronen die durch einen Lernprozess optimiert werden [29] CNNs werden hauptsächlich für Bildverarbeitung genutzt. CNNs befassen sich mit dem Problem, dass Bilddateien rechnerisch aufwendig sind. Ein Lösungsansatz für die Reduzierung der rechnerischen Komplexität ist “Faltung” und “Pooling”. Ein Convolutional Neural Networks besteht aus convolutional layers, pooling layers und fully connected layers.

3.1 Convolutional layer

Ein Convolutional Layer besteht aus einem zweidimensionalen “Kernel”. Die Werte die sich in dem Kernel befinden sind trainierbar, lassen sich also im Trainingsprozess anpassen. Diese Map wird dann für die sogenannte “Faltung” genutzt. Die Eingabe repräsentiert das Bild in Pixelwerten und stellt dadurch ebenfalls eine zweidimensionale Vektor dar. Der Kernel, der im durchschnitt eine sehr kleine Dimension hat, wird in einem Convolutional layer über den Eingangsvektor “geschoben”. Es wird dann also für jede mögliche Stelle des Eingabevektors eine Matrix erzeugt, welche die selbe Dimension wie der Kernel hat und den Inhalt der betrachteten Stellen des Eingabevektors. Mit dieser Matrix wird ein Kreuzprodukt mit dem Kernel erzeugt. Das Kreuzprodukt wird dann in eine neue Matrix übertragen. Dieser Prozess wird pro Convolutional Layer so häufig durchgeführt, bis jede Stelle des Eingabevektors für ein Kreuzprodukt genutzt wurde.

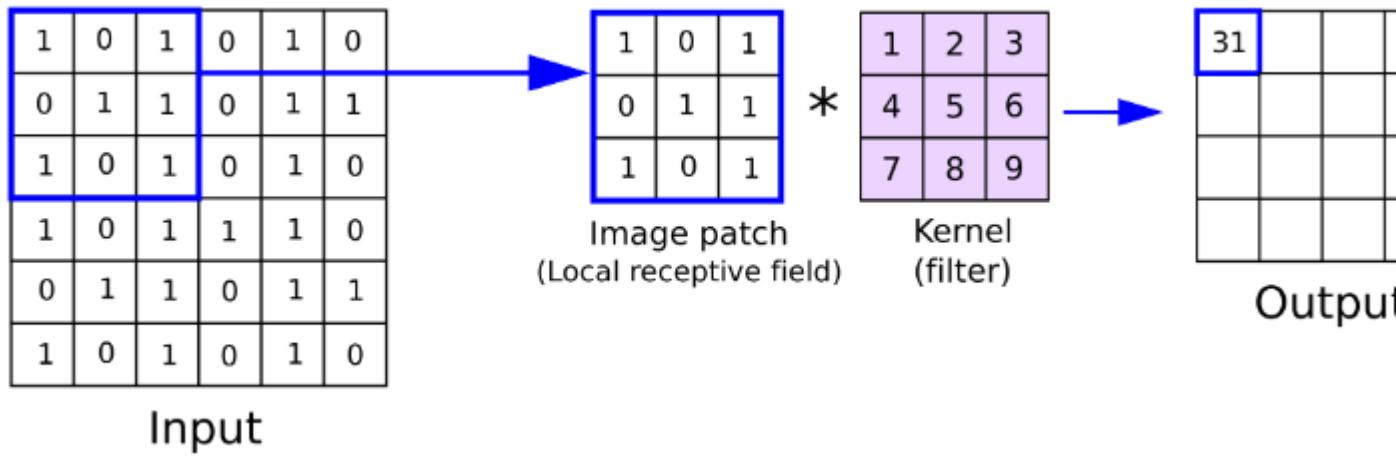


Abbildung 1: Convolution

Quelle: <https://arxiv.org/pdf/1706.03762>

3.2 Stride

Stride ist ein Parameter der das Verhalten der Convolution Layer ändert. Ist der Stride größer als 1 bewegt sich der Kernel mehr als eine Position pro Errechnung des Kreuzproduktes source [30]. Da bei einer höheren Stride auch die Menge der Errechnungen reduziert, reduziert eine höhere Stride auch die Dimension des Ausgangs. Die Größe des Ausgangs ist wie folgt formuliert:

$$O = ((W - K) + 2P)/(S + 1)$$

Wo O die Ausgangsgröße ist, W die Eingangsgröße, K die Kernelgröße und S die Stride. P stellt das Padding dar, was im Folgenden erläutert wird.

3.3 Padding

Padding ist ein weiterer Weg die Dimension des Ausgangs zu manipulieren. “Zero-padding” fügt nullen an den Kanten des Eingangvektors an.

3.4 Pooling Layer

Pooling Layer werden genutzt um die Dimension des Eingangsvektors zu reduzieren. Ein häufig genutzter Pooling Layer ist der sogenannte “Max-pooling layer”. Bei einem Max-pooling layer wird eine activation map, wie bei einer Faltung in einem convolution layer, über den Eingang gelegt und lediglich der höchste Wert, des betrachteten Bereichs, in die Position des Ausgangsvektors übertragen. Parameter wie beispielsweise Kernelgröße und Stride sind auch bei Max Pooling konfigurierbar. source <https://medium.com/@danushidk507/max-pooling-ef545993b6e4>.

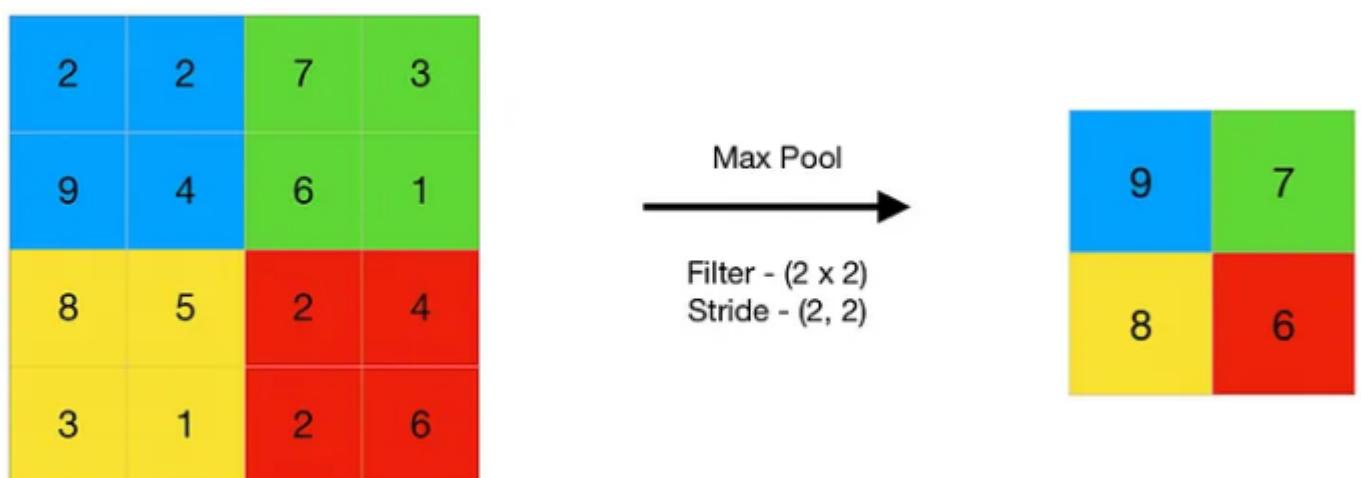


Abbildung 2: Max Pooling

Quelle: <https://medium.com/@danushidk507/max-pooling-ef545993b6e4>

Structure-based Text Transformation betrachtet den Text auf größeren Bereichen als bloß einzelne Wörter. Es werden beispielsweise grammatischen oder sinngemäße Zu-

sammenhänge betrachtet. “Sentence Cropping” verwendet in vielen Fällen Abhängigkeitsbäume um beim Tauschen von Positionen von Teilen eines Satzes, die Abhängigkeiten behalten zu können. Es werden dadurch also Sätze in kleinere Sätze aufgeteilt. “Sentence Morphing” manipuliert die Struktur eines Satzes durch beispielsweise zufälligem Tauschen von Worten.

4 Transformer

Das Transformer Modell und der gleichzeitig erschienene “Attention-mechanism” hat die Forschung von Large language models und andere Bereiche des maschinellen Lernens entscheidend geprägt. 2017 hat eine Gruppe von Forschern bei Google Brain das Paper “Attention Is All You Need.” veröffentlicht und noch bis heute sind Teile in diesen An-sätzen in State-of-the-art Lösungen enthalten source <https://arxiv.org/pdf/1706.03762>

Die veröffentlichte Architektur verfolgt der Idee der “encoder-decoder” Struktur. Der Encoder dieser Architektur weißt pro Iteration einer Sequenz von Eingaben als symbolische Repräsentationen, ein Symbol einer kontinuierlichen Repräsentationen zu. Das Modell ist auto-regressiv, dadurch jedes generierte Symbol in der nächsten Iteration als weitere Eingabe genutzt.

BildHier 1

4.1 Attention

Transformer nutzen eine Funktion für das Summieren von gewichteten Summen, die “Attention Funktion” genannt wird. Diese Funktion weißt dementsprechend eine sogenannte “Query” und ein Satz von Schlüssel-Wert Paaren einem Ausgangswert zu. Die Gewichte, die für die Verrechnung genutzt werden repräsentieren die “Kompatibilität” der query gegenüber des entsprechenden Schlüssels. Dafür wird eine “compatibility function” genutzt.

5 Überblick der Ansätze

Der derzeitige Stand der Forschung zum Thema bildbasierten Klassifikationen von Dokumenten ist sehr vielseitig und verfolgt viele unterschiedliche Ansätze und Technologien. Einige Ansätze verfolgen Grundideen, die sich sehr der allgemeinen Bildklassifikation ähneln. Andere Ansätze vertiefen sich in Eigenschaften die in Dokumenten speziell auftauchen. Bei diesen Ansätzen wird beispielsweise der erkannte Text für die Klassifizierung genutzt, oder die Anordnung der Elemente in einem Dokument. Einige dieser Ansätze wurden im Verlauf der Forschung erweitert, effizienter gestaltet oder mit anderen Ansätzen kombiniert.

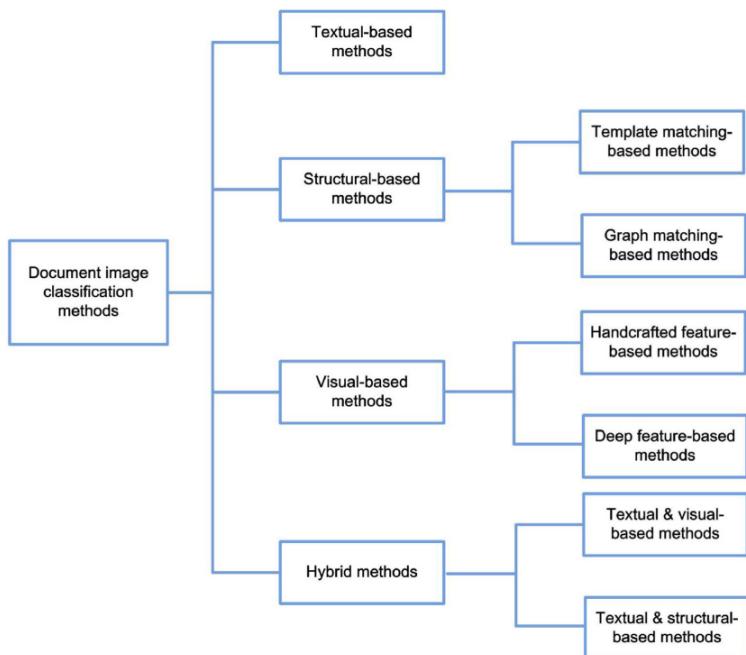


Abbildung 3: Überblick der Ansätze

Quelle: Document image classification: Progress over two decades

Die Ansätze lassen sich in unterschiedliche Methoden aufteilen: Textbasierende Methoden, Strukturbasierende Methoden, visuell basierende Methoden und hybride Methoden source Document image classification: Progress over two decades . Im Rahmen dieser Arbeit werden diese Methoden untersucht und Beispiele betrachtet. Textbasierende Methoden beziehen sich lediglich auf den Text, der in Dokumenten verfügbar ist. Es wird OCR genutzt um diesen Text zu extrahieren. Damit wird die bildbasierende Klassifikation zu einer textbasierende Klassifikation übertragen. Darauf aufbauend werden Lösungen für eine Textklassifikation genutzt. Die große Schwäche dieses Ansatzes ist,

dass OCR Verfahren noch immer sehr fehleranfällig sind. Ist die Klassifikation lediglich auf Text basierend leidet auch die Qualität der Klassifikation unter der OCR Schwäche. Strukturbasierende Methoden bezieht sich auf die Extrahierung und Analyse der Struktur des Dokuments. Bei Template basierenden Methoden werden Vorlagen genutzt und verglichen wie sehr die zu klassifizierenden Dokumente ähnlich zu den Vorlagen sind. Es werden häufig mehr als eine Vorlage für jede einzelne Klasse genutzt. Das Entwickeln dieser Vorlagen kann eine sehr aufwendige Aufgabe sein und erfordert tiefes Wissen über die zu klassifizierenden Objekte. Da diese Methode selbst erstellte Vorlagen benötigt, muss das Wissen vorhanden sein, wie die Features am genauesten in einer Vorlage repräsentiert werden.

Bei Graph basierenden Methoden werden die Dokumente in Graphen dargestellt source <https://www.scitepress.org/Papers/2024/122632/122632.pdf> . Es werden Regionen, die miteinander in Relation stehen, identifiziert und werden als Knotenpunkte in einem Graphen dargestellt. Diese Regionen werden Superpixel genannt source <https://dl.acm.org/doi/10.1145/3652509>.

Die visuell basierende Methode funktionieren ähnlich der Strukturbasierende Methoden bezüglich dem Aspekt, dass Pixelwerte genutzt werden um die Klassifikation umzusetzen. Der Unterschied zwischen diesen Methoden ist, dass visuell basierende Methoden das Dokument grob und indirekt beschreiben.

Die visuell basierende Methoden werden in zwei Kategorien unterteilt: In handgemachten feature basierenden Methoden und Deep feature basierenden Methoden. Eines der Beispiele von handgemachten feature basierenden Methoden ist die Nutzung von sogenannten “local descriptor”, die beispielsweise ein Vektor von Bildpixeln sein können source <https://www.robo>. Diese Vektoren können bestimmte Eigenschaften der Bilder widerspiegeln, wie zum Beispiel Farben, Texturen oder Kanten. In der Klassifizierung werden diese Vektoren dann genutzt um Ähnlichkeit zwischen Klassen und den tatsächlichen Bildern festzustellen.

Bei Deep feature basierende Methoden werden die Merkmale automatisch erlernt. CNN ist ein Beispiel dafür, da die Filter durch Berechnungen am Ende eines Trainingsdurchlaufs korrigiert werden und somit Merkmale bestimmt werden.

Textuelle Ansätze können auch mit visuelle oder strukturellen Ansätzen kombiniert werden. L. Liu, Z. Wang, T. Qiu et al. behaupten, dass diese hybriden Ansätze gewöhnlich eine state-of-the-art Klassifikationsleistung erzielen source Document image classification: Progress over two decades. Grund dafür sei, dass mehrere unterschiedliche Arten von Merkmalen genutzt werden.

6 Metriken für Evaluationen von Klassifikationen

Um unterschiedliche Ansätze miteinander zu vergleichen ist es notwendig bestimmte Aspekte zu bewerten. Es müssen etwa qualitative Aussagen getroffen werden oder quantitative Beobachtungen genutzt werden um diese Aussagen gegenüberzustellen. Im qualitativen Sinne könnten also Aussagen getroffen wie “gut” und “schlecht”, oder “viel” und “wenig”. Diese Herangehensweise birgt allerdings das Risiko subjektive Perspektiven in die Bewertung einfließen zu lassen. Im Gegensatz zu wagen Aussagen im qualitativen Sinne, lassen sich Zahlen miteinander verrechnen um Metriken zu bilden, die dazu dienen können bestimmte Sachverhalte zu repräsentieren. Es werden hauptsächlich die Ergebnisse von Klassifikationen genutzt um diese Metriken zu errechnen. Bis heute gibt es viele unterschiedliche Ansätze auf welche Weise diese Ergebnisse verrechnet werden um Metriken zu erzeugen. Unter den meist genutzten Metriken zählen “accuracy”, “error rate”, “F-beta score” und “Matthews correlation coefficient (MCC)” source <https://arxiv.org/pdf/2209.05355.pdf>. Da erfolgreiche Klassifizierungen das korrekte Zuweisen eines Objektes der zugehörigen Klasse ist, basieren die Metriken zur Bewertung der Klassifikation auf die Korrektheit der Zuweisung. In dem Beispiel einer binären Klassifikation werden die Menge von korrekt positiven (TP), falsch positiven (FP), korrekt negativen (TN) und falsch negativen (FN) hauptsächlich in die Kalkulation von Bewertungsmaßen aufgenommen.

6.1 Wahl der Informationen für Metriken

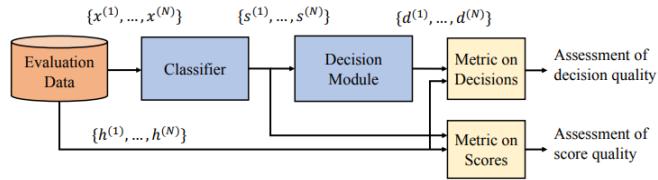


Abbildung 4: Überblick der Ansätze

Quelle: <https://arxiv.org/pdf/2209.05355>

Die oben genannten Informationen wie TP, FP, TN und FN beziehen sich auf die Entscheidungen die am Ende der Klassifizierung erfolgt sind und repräsentiert den Wahrheitsgehalt der korrekten Klassifizierung. Bevor Modelle die Entscheidung getroffen haben, haben gewöhnlicherweise Klassifizierer einen Wert errechnet wie wahrscheinlich ein Objekt jeder verfügbaren Klasse zugehört. Werden nur die finalen Entscheidungen für die Metriken genutzt verfallen also die Informationen welche Klasse ein Objekt beispielsweise am zweitwahrscheinlichsten zugehört.

6.2 Wahl der Metriken

In der Recherche nach modernen Lösungen für bildbasierte Klassifikation für Dokumente ist es essenziell die Informationen der Autoren der Lösungen zu nutzen um sie miteinander zu Vergleichen. Gewöhnlich veröffentlichen Autoren ihren Ansatz in Begleitung von Metriken, die die Qualität repräsentiert. Zu der Zeit dieser Arbeit wird als Metrik häufig die Genauigkeit, beziehungsweise die “Accuracy” genutzt. Die Accuracy ist eine Metrik die den Wahrheitsgehalt der konkreten Entscheidungen darstellt. Die Accuracy stellt das Verhältnis zwischen den korrekten Entscheidungen und allen Entscheidungen dar.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Auch wenn unterschiedliche Metriken unterschiedliche Aussagekräfte haben, bietet es in einer solchen Arbeit an sich an die Metriken zu wenden, die von der Allgemeinheit veröffentlicht und genutzt werden, da die Alternative wäre jede Architektur dessen Metriken fehlen zu implementieren und selbst zu errechnen. Da die Accuracy die Entscheidungswerte nutzt sind also die Werte TP, TN, FP und FN relevant. Einige Literaturen nutzen

auch Metriken wie den sogenannten “F1 score” source <https://www.geeksforgeeks.org/f1-score-in-machine-learning/>, der allerdings keine Parameter nutzt, die nicht auch in der Accuracy vorhanden sind.

7 Daten für das Training

Werden Algorithmen genutzt um automatisiert Muster in Daten zu finden ist es notwendig Daten zu beschaffen um die Algorithmen auf diese Daten anzuwenden. Diese Arbeit untersucht moderne Ansätze und vergleicht diese auf unterschiedliche Aspekte in der bildbasierenden Klassifikation von Dokumenten. Im Bereich des maschinellen Lernens ist es wichtig zu beachten, dass ein Modell nur so gut sein kann, wie die Qualität der Daten sind mit denen die Modelle trainiert werden. Das heißt, dass die Vergleichbarkeit zwischen unterschiedlichen Modellen darunter leiden kann, wenn auch unterschiedliche Datensätze genutzt werden. Außerdem gibt es bereits beschriebene Verfahren wie Pre-Processing, die bei einem Modell ein besseres Resultat erzielen kann. Die Varianz in genutzte Datensätze stellt also eine Herausforderung dar, moderne Ansätze gegenüberzustellen. Eine Lösung dafür kann das eigenständige Antrainieren der Modelle mit einer gleichbleibenden Menge von Datensätzen sein. Diese Lösung ist allerdings sehr Zeitaufwendig, besonders bei weniger gut dokumentierten Modellen. Außerdem ist dies nicht zwingend notwendig, da in einigen Fällen die Varianz in Datensätzen und Pre-Processing Verfahren nicht sehr groß sind. Durch einer persönlichen Einschätzung kann also die Varianz als gering genug bewertet werden, dass die Ergebnisse unterschiedliche Ansätze trotz alledem vergleichbar sind.

Da ein guter und großer Datensatz eine Bedingung für einen Erfolg in Klassifikation im Bereich des maschinellen Lernens, und die eigenständige Beschaffung eine große Herausforderung ist, wird häufig auf öffentlich verfügbare Datensätze zurückgegriffen. Eines dieser häufig genutzten öffentlichen Datensätze ist RVL-CDIP, welches aus 400.000 Bildern bestehen. Diese Daten sind in 16 Klassen unterteilt source <https://paperswithcode.com/dataset/rvl-cdip>. Neben diesem Datensatz gibt es Tobacco-3482, welches auch in einigen Literaturen verwendet wurde, allerdings mit 3482 Bildern auf 10 Klassen verteilt, eine deutlich geringere Menge darstellt.

Da in vielen Literaturen RVL-CDIP genutzt wird, nutzt diese Arbeit diesen Datensatz ebenfalls. Da die gesamte Menge an Bildern nach eigener Einschätzung nicht für den Zweck dieser Arbeit benötigt wird, wird daher auf eine reduzierte Menge dieses Datensatzes zurückgegriffen.

7.1 Aufteilung der Daten

Neben der Wahl und Vorbereitung der Daten spielt noch die Anordnung der Daten eine Rolle. Wie viele Daten für das Training, die Evaluation und das Testen genutzt werden hat eine Auswirkung auf die Qualität der Klassifizierung. Werden mehr Daten des Datensets für die Evaluation genutzt, ist zwar die Evaluation ausführlicher und somit aussagekräftiger, allerdings hat das Modell weniger Beispiele zum Erkennen und Antrainieren von Mustern. Daher gibt es keine klare Aufteilung dieser Datensätze, allerdings gibt es gewisse Aufteilungen die in der Literatur häufiger vorkommen. Beispielsweise taucht die Aufteilung 80% Training, 10% Evaluation und 10% Test häufig auf. Auch 60% Training, 20% Evaluation und 20% Test lässt sich gelegentlich finden, scheint allerdings weniger dominant im Bereich des maschinellen Lernens zu sein, da wertvolle Daten aus dem Training in die Bewertung fließen. Experimente vergleichen diese beiden Aufteilungen die bessere Ergebnisse bei 80-10-10 gefunden haben source <https://medium.com/@itsmeSamrat/splitting-the-data-to-60-20-20-ratio-vs-80-10-10-which-one-is-better-bbc3503830d8>. Dies ist allerdings kein Beweis, dass diese Aufteilung besser ist, da der Kontext der Anwendungsfälle und viele weitere Faktoren eine solche Schlussfolgerung nicht so einfach ermöglichen. Der RVL-CDIP Datensatz ist bereits in einer 80-10-10 Trennung vorbereitet.

8 Analyse des Datensatzes

In diesem Kapitel werden die konkreten Dokumente untersucht. In den vorherigen Kapiteln wurden unterschiedliche Ansätze betrachtet und es wird untersucht ob die Klassifikation der Dokumente Nutzen von den Ansätzen ziehen können. Die Klassifikation dieser Arbeit beschränkt sich auf 5 von den 16 verfügbaren Klassen um den Umfang und Aufwand für die Analyse und das Training angemessen der Arbeit zu

halten. Demnach werden im folgenden Emails, Quittungen, Notizen, Fragebögen und Spezifikationen untersucht.

8.1 Email

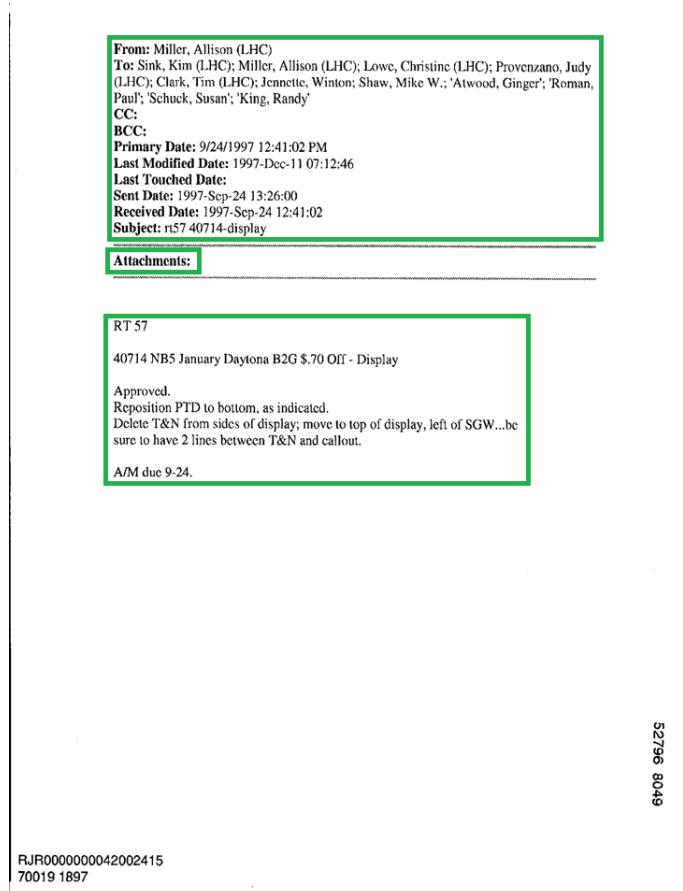


Abbildung 5: Überblick der Ansätze

Quelle: <https://michaelkipp.de/deeplearning/Evaluation.html>

Emails mögen zwar, je nach verwendetem Email Programm eine gewisse Variation der Darstellung haben, verfolgen allerdings sehr Häufig ein Muster, das gleich ist. Zu Beginn werden gewisse Metadaten aufgelistet. Zum Beispiel Absender, Empfänger und Betreff. Je nach Variation kommen noch Daten, Anhänge, Weiterleitungen und andere Informationen hinzu. Darauf folgt dann der Text, der über die Email gesendet wird. Der Text kann deutlich höhere Variationen darstellen, da dieser von Personen geschrieben wird und nicht von Email Software formuliert wird. Dieser Text kann formell geschrieben werden, daher können möglicherweise Anreden oder Redewendungen wiedergefunden werden. Diese Formulierungen variieren allerdings wahrscheinlicher als die gedruckten

Metadaten der Email. Dennoch können im Emailtext Muster gefunden werden, wie dass “Best regards” auf Emails hinweisen.

Emails haben eine sehr starke strukturelle Komposition und sehr häufig Begriffe, die mit der Klasse assoziiert werden können.

8.2 Quittungen

FREEDOM OF INFORMATION AND PRIVACY ACT BRANCH		SERVICE ORDER, INVOICE, AND RECEIPT		
FEDERAL TRADE COMMISSION				
WASHINGTON, D.C. 20580				
NAME John P. Rupp		FIRM/ORGANIZATION Covington & Burling		DATE 4/22/82
ADDRESS (street, city, state, zip code) 1201 Pennsylvania Avenue, N.W., P. O. Box 7566 Washington, D.C. 20044				PHONE
SUBJECT Cigarette Advertising		DESCRIPTION		DOCKET/FILE NO.
PARALEGAL half Aponte				FOIA NO. 82-0121
REPRODUCTION NO. OF COPIES 1	DOCUMENTS	NO. OF PAGES 2470	\$ 296.40	
	first 10.00 free of charge		-\$ 10.00	
SEARCHES				
TELETYPE				
MICROFILM				
COMPUTER				
CERTIFICATION				
		TOTAL AMOUNT DUE (Payable on Receipt)		\$ 286.40
PAYMENT INSTRUCTIONS				
Make check payable to U.S. Treasury. Put FOIA No. (above) on check. Mail to: Division of Budget and Finance Room 766 Federal Trade Commission Washington, D.C. 20580				
03025016				
SEE OTHER SIDE FOR SCHEDULE OF FEES				

Abbildung 6: Überblick der Ansätze

Quelle: <https://michaelkipp.de/deeplearning/Evaluation.html>

Quittungen können eine starke Varianz aufweisen. Quittungen haben zwar auch eine gewisse Menge an Informationen die auf vielen Quittungen zu finden sind, diese sind allerdings weniger häufig ähnlich angeordnet als sie es bei Emails sind. Ein Unterschied zwischen Emails und Quittungen sind unter anderem, dass diese ohne Informationen des Empfängers ausgestellt werden können. Daher muss nicht zwingend ein Name ersichtlich sein. Demnach ist die Struktur und der Inhalt der gedruckten Metadaten nicht zwingend

hinreichend für die Klassifikation. Die Aussagekraft über die Klasse geschieht bei Quittungen allerdings in der Auflistung der erworbenen Produkte. Die Struktur der Quittung besteht hauptsächlich aus untereinander angeordneten Namen, Quantitäten und Preisen. Hierbei ist wichtig anzumerken, dass der textuelle Inhalt der Namen, Quantitäten und Preisen nicht zwingend hinreichen für die Klassifizierung ist, jedoch die Struktur. Demnach ist beispielsweise die Relevanz nicht "15", "Kugelschreiber", "15,00\$", sondern die Informationen, dass diese Regionen nebeneinander stehen. Allerdings können auch Währungssymbole einen Aufschluss auf die Klasse geben.

8.3 Notizen

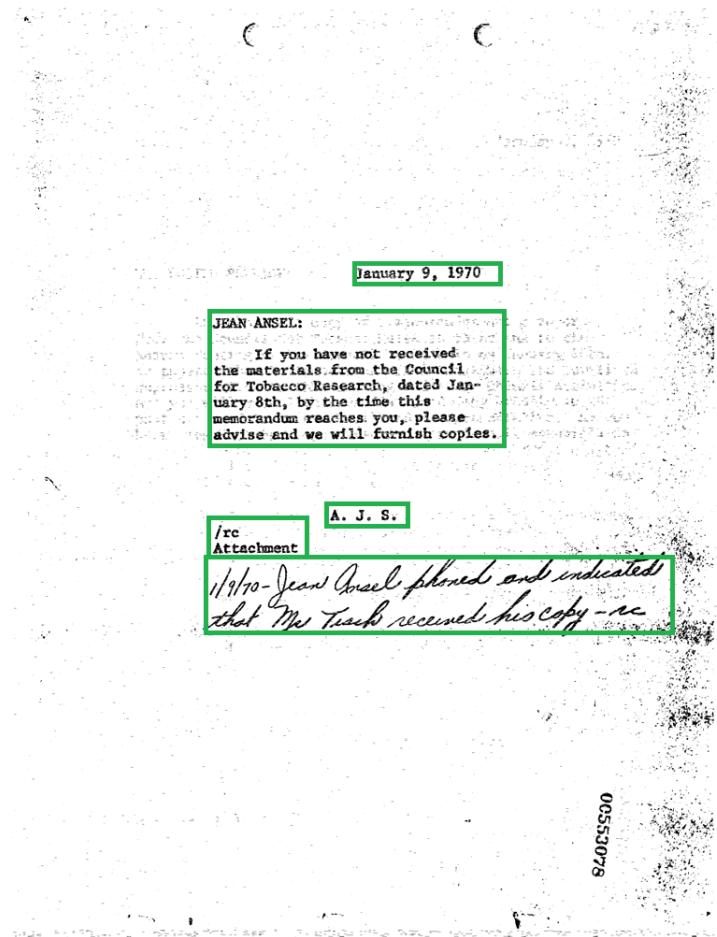


Abbildung 7: Überblick der Ansätze

Quelle: <https://michaelkipp.de/deeplearning/Evaluation.html>

Notizen sind von den 5 Klassen die unstrukturiertesten Daten. Notizen verfolgen selten Muster, die sich bei allen Notizen erkennen lassen. Allerdings ist die Wahrscheinlichkeit Handschrift in Notizen zu finden höher als bei beispielsweise Quittungen und Emails,

was ein Merkmal darstellt. Notizen lassen sich deutlich besser visuell erkennen als strukturell, da es weniger erkennbare Strukturen gibt. Dafür gibt es beispielsweise handschriftliche Charakteristiken wie Schreibschrift oder visuelle Eigenschaften wie schwache Tinte beim Schreiben oder Korrekturen.

8.4 Fragebögen

LEGGETT & LUSTIG RESEARCH, INC.
101 Park Avenue
New York, New York 10017
Telephone No. (212) 725-1146/2109

J-R-592
February, 1972

THIS QUESTIONNAIRE IS NOT TO BE ADMINISTERED TO ANYONE UNDER 21 YEARS OF AGE.

SPIRIT CIGARETTE PRINT AD TEST
SCREENING QUESTIONNAIRE

INTRODUCTION: Good morning/later evening, I am from Leggett & Lustig, a national opinion and research company, are doing some research on people's reactions to various consumer products. Could you spare a few minutes to answer some questions? Your opinions will be greatly appreciated.

A. Do you, or anyone in your household, work for an advertising agency, marketing research company or a firm engaged in the manufacture, sale, or distribution of cigarettes?
(TERMINATE, RECORD ON TALLY SHEET)
(ASK Q. B) Yes.....1
No.....2

B. Into which of the following age categories do you fall? (READ CHOICES;
CIRCLE ONE NUMBER)

STERLING, MILLER OR TALLY SHEET	18-24.....1
	25-34.....2
	35-44.....3
	55 or older.....4

C. Do you currently smoke at least four (4) packs of cigarettes a week?
(ASK Q. D)
(TERMINATE, RECORD ON TALLY SHEET)
Yes.....1
No.....2

D. Is your regular brand of cigarette filtered or unfiltered?
(ASK Q. E)
Yes.....1
No.....2

E. What size do you usually smoke? (READ CHOICES)

Regular Size (Less than 85mm)	1
King Size (85mm-95mm)	2
Long (100mm)	3
Extra Long (110mm)	4

F. (SHOW RESPONDENT CARD 1.) Which of these brands is your regular brand of cigarette, that is, the brand of cigarettes you smoke most often? (RECORD BELOW BY CIRCLING ONE NUMBER)

HIFI (UNDER 15 MG & OVER)	FULL FLAVOR (15 MG & OVER)
Benson & Hedges Lights.....7-1	Multifilter.....8-1
Camel Lights.....2	Benson & Hedges.....10-1
Carlton.....3	Camel Filters.....1
Decade.....4	Chesterfield.....1
Doral.....5	Pall Mall Lights.....4
Fest.....6	Parliament.....5
Kent.....7	Reefer.....5
Kent Golden Lights.....8	Reefer.....6
L & M Lights.....9	Tareyton Lights.....8
Lucky Strike.....10	Old Gold.....9
Marlboro Lights.....11	Pall Mall.....10
Merito.....12	Raleigh Filters.....10
Vintage.....13	Tareyton.....11
Viceroy Extra Mild.....14	Villard.....12
Winston Lights.....15	Winston.....11-1

RESPONDENT MUST SMOKES ONE OF THE PRE-LISTED BRANDS TO QUALIFY. IF NOT,
TERMINATE AND RECORD ON TALLY SHEET.

G. (CIRCLE QUOTA GROUP FOR THIS RESPONDENT.)

Male Hi Fi.....1	12+
Female Hi Fi.....2	
Male Full Flavor.....3	
Female Full Flavor.....4	

IF THE FILM IMAGE IS LESS CLEAR
AT THIS POSITION, IT IS DUE TO THE
LIMIT OF THE DOCUMENT BEING FILMED.

67730 2908

Abbildung 8: Überblick der Ansätze

Quelle: <https://michaelkipp.de/deeplearning/Evaluation.html>

Fragebögen können ebenso wie Quittungen je nach Aussteller variieren. Es gibt auch hier zwar einige Metadaten die vorkommen können wie ein Datum oder ein Betreff, diese müssen allerdings nicht vorhanden sein. Fragebögen weisen eine gewisse strukturelle Ähnlichkeit zu Quittungen auf. Diese ist das benachbarte Anreihen und Regionen. Wie es bei Quittungen beispielsweise Name, Quantität und Preis ist, sind es bei Fragebögen Fragen und Antworten. Demnach zieht die Klassifikation auch bei Fragebögen einen Vorteil aus struktureller Betrachtung, sowohl als auch die textuelle. Die textuelle

Betrachtung ist wichtig, da Antworten häufig durch “Ja” oder “Nein” formuliert werden, oder durch quantifizierende oder qualifizierende Aussagen wie “sehr” und “wenig” oder “stark” und “schwach”.

8.5 Spezifikationen

SMOKING TOBACCO PRODUCTS		
	Coumarin (ppm)	Deer Tongue (% by Weight of Finished Product)
<u>Domestic</u>		
<u>ATC</u>		
HALF AND HALF PALADIN BOURBON BLEND	725 725 725	0.4 0.4 None
<u>Brown & Williamson</u>		
Sir Walter Raleigh	40*	None
<u>Liggett Group</u>		
Velvet	100	None
<u>U.S. Tobacco</u> <u>(for Philip Morris)</u>		
Bond Street	220	0.1 Approx.
<u>R. J. Reynolds</u>		
Carter Hall	150	0.2 Approx.
Maderia Gold	425	0.2 Approx.
Prince Albert	None	None
Royal Comfort	1,000	None
<u>Sutliff</u>		
Mixture No. 79	125	None
<u>Lane Limited</u>		
Captain Black	3,500	None
<u>John Middleton</u>		
Cherry Blend	None	None
<u>Imports</u>		
<u>Douwe Egberts</u>		
Amphora Regular	None	None
<u>Swedish Tobacco Co.</u>		
Borkum Riff	625	None
ATX02 0076228		

Abbildung 9: Überblick der Ansätze

Quelle: <https://michaelkipp.de/deeplearning/Evaluation.html>

Auch Spezifikationen folgen dem Muster einer Aufzählung von benachbarten Informationen wie Namen und Quantitäten. Wie bei Quittungen und Fragebögen, die eine Ähnlich Struktur darstellen, variieren Spezifikationen je nach Aussteller. Eine der wenigen Unterschiede zwischen Spezifikationen und Quittungen sind allerdings, dass es sich seltener um Kosten oder Preise handelt. Demnach haben auch Spezifikationen eine konkrete Struktur, allerdings ist die Betrachtung des Inhalts von hoher Relevanz um diese voneinander zu unterscheiden.

8.6 Datensatz im Bezug zur Wahl der Ansätze

In diesem Datensatz befinden sich Klassen die stark durch den textuellen Inhalt repräsentiert werden wie Fragebögen. Die Erkennung von Worten wie “Ja” und “Nein” haben einen großen Einfluss auf die Klassifizierung. Außerdem gibt es Klassen die eine sehr starke strukturelle Charakterisierung haben wie Emails. Es gibt Klassen die eine starke Struktur und häufig wiederkehrende Symbole verwenden wie Quittungen, die sehr häufig Auflistungen verfolgen und Zahlen, sowie Währungssymbole. Im Gegensatz dazu haben Notizen allerdings weder Strukturen die stark erkennbar sind, noch Wörter die in Muster erkennbar sind. Notizen können jedoch visuelle Wiedererkennungsmerkmale aufweisen. Daraus folgend sind drei Punkte sehr relevant für eine erfolgreiche Klassifizierung:

Betrachtung der Struktur des Dokuments
Betrachtung der Texte, beziehungsweise Zeichen im Dokument
Betrachtung des Erscheinungsbildes des Dokuments

9 Abwägung der Ansätze

9.1 Deep Learning als genereller Ansatz für Klassifikationen

Maschinelles Lernen und Deep learning haben große Erfolge im Bereich der Klassifikation über die letzten Jahre erzielt. Große und komplexe Probleme ließen sich durch das Anlernen von Modellen effizient lösen. Durch die Natur des maschinellen Lernens ließen sich die Probleme auch mit einer reduzierten Menge an benötigtem Expertenwissen lösen, da die Modelle mit dem Aussetzen von Beispielsdaten ein Optimum automatisch annähern. Es gibt viele unterschiedliche Architekturen im Bereich des maschinellen Lernens. Alle Architekturen haben ihre Stärken und Schwächen und somit bestimmte Anwendungsbereiche. Beispielsweise lässt sich durch Reinforcement learning die Steuerungen in Rennspielen optimieren um die bestmöglichen Zeiten zu erreichen source <https://www.youtube.com/watch?v=Dw3BZ6OUNTERSTRICH8LY>.



Abbildung 10: Darstellung mehrere Reinforcement Learning Iterationen in einem Rennspiel

Quelle: <https://www.youtube.com/watch?v=Dw3BZ6OUNTERSTRICH8LY>

Beispiele wie diese erwecken den Eindruck dass maschinelles Lernen die beste Lösung für jedes Optimierungsproblem darstellt. Zusätzlich zeigt der derzeitige Stand der Forschung des maschinellen Lernens eine State-of-the-Art Leistung nach der anderen im Bereich von Klassifizierungen, was diese Annahme unterstützt. Auch wenn diese Entwicklungen signifikant sind, sind sie nicht eindeutige Beweise, dass maschinelles Lernen der einzige richtige Ansatz ist. David Wolpert hat durch das sogenannte “No Free Lunch” Theorem interpretiert, dass für jeden beliebigen Lernalgorithmus A und B gleich viele Situationen existieren in denen der Lernalgorithmus A besser als B ist und umgekehrt <https://arxiv.org/pdf/2202.04513.pdf>. Daher ist es wichtig die Ansätze zu verstehen um nachzuvollziehen welche Fälle für welchen Algorithmus vorteilhaft sind. Beispielsweise haben L. Grinsztajn et al. ein Muster entdeckt in dem zu sehen ist, dass “Tree-based models” dazu tendieren bessere Ergebnisse als Deep learning Ansätze zu erzielen wenn es um Daten in Tabellenformat geht source <https://openreview.net/pdf?id=Fp7phQszn>. Erklärt wird dieses Phänomen durch die Natur der Daten im Tabellenformat. Diese Daten können irreguläre Muster haben und Features mit geringen Informationsgehalt, was den Erfolg mit Deep Learning Ansätzen hindern kann.

9.2 Deep Learning als Ansatz für Bildklassifikationen

Wie im letzten Kapitel beschrieben scheint Deep Learning nicht zwingend der beste Ansatz für tabellarische Daten zu sein. Daher beschäftigt sich dieses Kapitel mit dem

Fall des Datentyps dieser Arbeit, Bilddateien. Han Xiao et al. haben mehrere unterschiedliche Klassifizierungsmethoden genutzt um Bilder aus dem Fashion-MNIST source <https://paperswithcode.com/sota/image-classification-on-fashion-mnist> Datensatz zu klassifizieren source <https://www.researchgate.net/publication/342229617ssification>. Die Resultate werden dann in der folgenden Tabelle mit DARTS source <https://www.researchgate.net/publication/342229617ssification> verglichen, was exemplarische Deep Learning Lösungen darstellen.

Tablehier 1

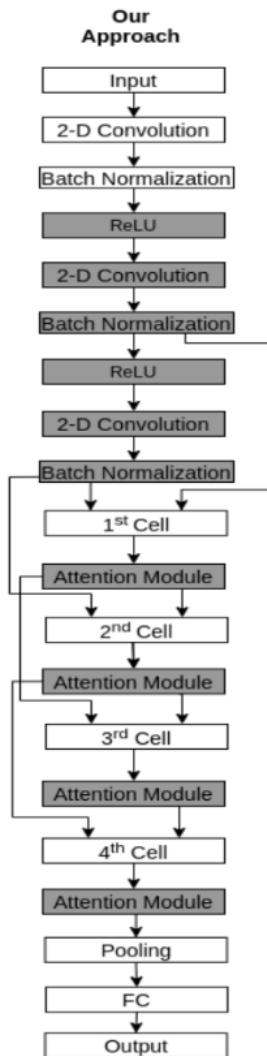


Abbildung 11: DARTS Architektur

Quelle: <https://www.researchgate.net/publication/342229617ssification>

Wie in Tabelle 1 zu sehen erzielen die Deep Learning Lösungen eine deutlich bessere Genauigkeit als Klassifizierer. Diese Arbeit handelt allerdings von Dokumentenbildklassifikationen. Tabelle 1 betrachtet die Deep Learning und nicht Deep Learning Ansätze für Bildern von Kleidung. Bei Bildern von Dokumenten kommt die Komponente hinzu,

dass auch weitere Informationen in Form von Text hinzukommen. Wie in der Übersicht der Ansätze zu sehen gibt es speziell für Text und Struktur Deep Learning Ansätze die diese Eigenschaften zu nutzen macht. Daher lässt sich argumentieren, dass auch für Dokumentenbildklassifikation Deep Learning ein guter Ansatz sein wird.

Besonders CNNs und Transformatoren haben sich im Bereich der bildbasierten Klassifikation von Dokumenten durchgesetzt. Maschinelles Lernen beinhaltet allerdings noch viele weitere unterschiedliche Ansätze, wie Reinforcement Learning source <https://www.geeksforgeeks.org/what-is-reinforcement-learning/>, Schwarmintelligenzen und evolutionäre Algorithmen. Auch wenn diese Ansätze weniger verbreitet in dem Feld sind, finden sie auch Anwendung für Bildklassifikation. Das Modell DEL-EPSI source <https://www.sciencedirect.com/science/article/pii/S1877050923021270> nutzt beispielsweise evolutionäre Programmierung source <https://www.sciencedirect.com/topics/computer-science/evolutionary-programming>. und Schwarmintelligenzen source <https://www.sciencedirect.com/topics/intelligence>. DEL-EPSI erreicht auf dem CIFAR-10 Datensatz eine gute Accuracy von 96,7% source <https://www.sciencedirect.com/science/article/pii/S1877050923021270>, wird allerdings von moderneren Ansätzen wie Transformatoren überschattet, wie ViT-H/14, welcher eine Accuracy von 99,5% auf den gleichen Datensatz erreicht source <https://arxiv.org/pdf/2010.11929v2.pdf>.

10 Wahl eines Ansatzes

In dieser Arbeit wird ein moderner Ansatz gewählt, der eine gute Genauigkeit im Kontext der bildbasierten Klassifikation von Dokumenten erzielt, vertretbare Trainingskosten in Form von benötigter Rechenleistung benötigt und eine akzeptable Einarbeitung in Form von Verfügbarkeit des Codes und Dokumentation bietet. Basierend auf Literaturen der Forschung lässt sich schließen, dass die Genauigkeit der Klassifizierung steigt, wenn mehrere unterschiedliche Arten von Informationen aus den Daten extrahiert werden und gemeinsam für die Klassifizierung genutzt werden. Konkret wird genannt, dass es eine signifikante Interesse an hybriden Ansätzen gibt und einen konsequenten Mehrwert für kleine und große Datensätze bietet source https://link.springer.com/chapter/10.1007/978-3-030-43823-4_35. Eine Studie demonstriert, dass eine Fusion von semantischen Informationen von OCR mit den

visuellen Informationen des Bildes eine Leistungssteigerung im Vergleich zu visuellen Ansätzen erzielt source <https://ieeexplore.ieee.org/abstract/document/8977998>. Außerdem haben M. Viana et al. eine 7% Leistungssteigerung präsentiert, wenn die durch CNN generierten visuellen Features mit Text embeddings kombiniert werden source <https://link.springer.com/chapter/10.1007/978-3-030-02284-64>.

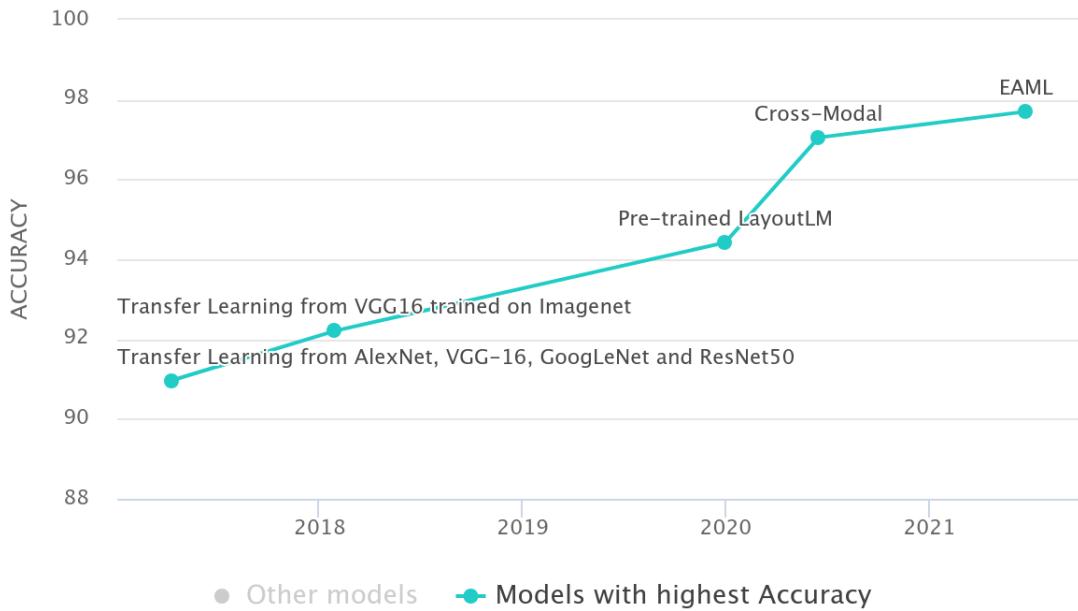


Abbildung 12: DARTS Architektur

Quelle: <https://paperswithcode.com/sota/document-image-classification-on-rvl-cdip>

Abbildung ??? stellt eine Grafik, der derzeitigen State-of-the-Art Ansätze relativ der Genauigkeit und Zeitpunkte der Evaluationen dar. Die Ansätze beziehen sich auf bildbasierten Klassifikation von Dokumenten auf den Datensatz RVL-CDIP. Diese Grafik deckt nicht alle Ansätze ab, aber repräsentiert den Trend, dass multimodale Ansätze (LayoutLM, Cross-Modal, EAML) eine bessere Genauigkeit als Ansätze erzielen die lediglich eine Art von Daten verwenden (VGG16, ResNet50, AlexNet). Bei der Analyse der State-of-the-Art Ansätze ist es auch relevant die Größe der Modelle zu betrachten. Auch wenn beispielsweise LayoutLM in diesem Kontext eine bessere Genauigkeit erzielt, hat das LayoutLMv3 Large Modell 368 Millionen Parameter source <https://arxiv.org/pdf/2204.08387.pdf>, wobei VGG16 138 Millionen Parameter hat source <https://builtin.com/machine-learning/vgg16>. Soll in einem praktischen Fall ein Modell für eine Lösung gewählt werden ist also auch die Menge der Parameter wichtig, da diese den Trainingsaufwand darstellen. Mit der Entwicklung von performanteren und

günstigeren Komponenten, beispielsweise in Form von Grafikkarten, ist ein höherer Trainingsaufwand jedoch immer einfacher, beziehungsweise günstiger zu erreichen. Soll ein Ansatz für eine Lösung zur bildbasierten Klassifikation von Dokumenten gewählt werden ist außerdem relevant, dass der Quellcode für das Training und gegebenenfalls des Testens öffentlich verfügbar und gut dokumentiert ist.

Wie in Abbild ??? zu sehen führen EAML, Cross-Modal und LayoutLM mit deren Genauigkeiten auf den Datensatz RVL-CDIP. EAML und Cross-Modal haben allerdings keine öffentlich zugängliche Implementation oder sind unzureichend dokumentiert. Für LayoutLM gibt es öffentliche Implementationen und sind ausreichend dokumentiert. Angesichts der guten Genauigkeit und ausführlicher Dokumentation wird LayoutLM als exemplarische Implementation genutzt um eine effiziente bildbasierte Klassifikation von Dokumenten zu präsentieren.

10.1 LayoutLM

Die erste Veröffentlichung des Modells LayoutLM geschah 2019 und hat den Ansatz verfolgt, visuelle Informationen zusätzlich des Textes zu nutzen um Dokumente zu klassifizieren, was zu dem Zeitpunkt der Veröffentlichung noch weit verbreitet war source <https://arxiv.org/pdf/1912.13318>. LayoutLM nutzt das BERT Modell, was ein Attentionbasierender bidirektonaler Sprachmodellierungsansatz ist. Für das Training der Sprachrepräsentation werden die Aufgaben “Masked Language Modeling(MLM)” und “Next Sentence Prediction(NSP)” genutzt. Bei dem Training werden durch MLM zufällig Wörter maskiert und lässt das Modell diese Wörter erraten source <https://www.techtarget.com/searchenterpriseai/definition/masked-language-models-MLMs>. NSP lässt das Modell zwei vollständige Sätze sehen und soll eine Aussage darüber treffen ob diese Sätze aufeinander folgen oder nicht.

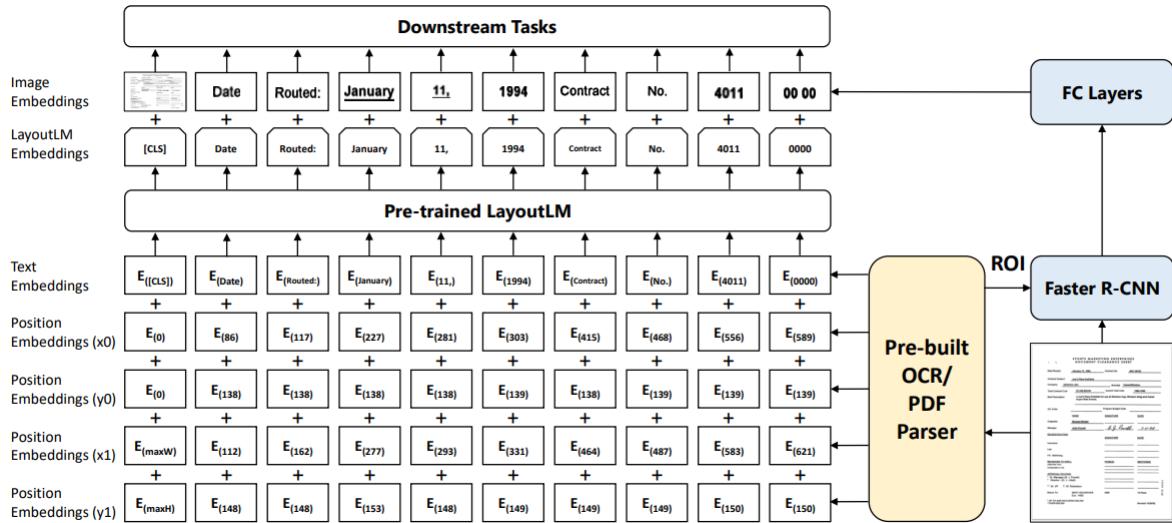


Abbildung 13: DARTS Architektur

Quelle: <https://arxiv.org/pdf/1912.13318>

LayoutLM nutzt ein Image Embedding um die Bild Features zu repräsentieren und ein 2D Position Embedding um die Position der Elemente des Dokuments zu repräsentieren. Das LayoutLM Model selbst würde über ein sogenanntes “Masked Visual-Language Model(MVLM)” und über “Multi-Label Document Classification(MDC)” antrainiert. MVLM ist von MLM inspiriert und verfolgt das Ziel Informationen zu maskieren und das Modell die originalen Informationen erraten zu lassen. MVLM maskiert allerdings nicht nur Text, wie MLM, sondern zufällig 2D Position Embeddings und Text Embeddings. Konsequent könnte das also heißen, dass das Modell Text Embeddings erhalten kann und erraten muss wo auf dem Dokument diese sich befinden. Durch MDC wird beim Training auf die Klassifizierung von Dokumenten mit mehreren unterschiedlichen Labels zurückgegriffen. Das Pre-Training von LayoutLM wurde mit dem Datensatz IIT-CDIP umgesetzt. Für das Pre-Training wurde Tesseract genutzt um die Embeddings zu erzeugen.

Modality	Model	Precision	Recall	F1	#Parameters
Text only	BERT _{BASE}	0.5469	0.671	0.6026	110M
	RoBERTa _{BASE}	0.6349	0.6975	0.6648	125M
	BERT _{LARGE}	0.6113	0.7085	0.6563	340M
	RoBERTa _{LARGE}	0.678	0.7391	0.7072	355M
Text + Layout MVLM	LayoutLM _{BASE} (500K, 6 epochs)	0.665	0.7355	0.6985	113M
	LayoutLM _{BASE} (1M, 6 epochs)	0.6909	0.7735	0.7299	113M
	LayoutLM _{BASE} (2M, 6 epochs)	0.7377	0.782	0.7592	113M
	LayoutLM _{BASE} (11M, 2 epochs)	0.7597	0.8155	0.7866	113M
Text + Layout MVLM+MDC	LayoutLM _{BASE} (1M, 6 epochs)	0.7076	0.7695	0.7372	113M
	LayoutLM _{BASE} (11M, 1 epoch)	0.7194	0.7780	0.7475	113M
Text + Layout MVLM	LayoutLM _{LARGE} (1M, 6 epochs)	0.7171	0.805	0.7585	343M
	LayoutLM _{LARGE} (11M, 1 epoch)	0.7536	0.806	0.7789	343M
Text + Layout + Image MVLM	LayoutLM _{BASE} (1M, 6 epochs)	0.7101	0.7815	0.7441	160M
	LayoutLM _{BASE} (11M, 2 epochs)	0.7677	0.8195	0.7927	160M

Abbildung 14: DARTS Architektur

Quelle: <https://arxiv.org/pdf/1912.13318>

Wie in Tabelle 2 zu sehen resultiert die Nutzung von MVLM in einer Erhöhung der Genauigkeit des Modells. Die Autoren haben LayoutLM für “Form understanding”, “Receipt Understanding” und “Document Image classification” genutzt.

10.2 LayoutLMv3

Die Autoren der ersten Version von LayoutLM haben dargestellt, dass deren Modell bereits damalige State-of-the-Art Lösungen in deren Aufgaben überholt hat. Zu dem Zeitpunkt dieser Arbeit ist die aktuelle Version von LayoutLM “LayoutLMv3”. LayoutLMv3 nutzt weiterhin Text und Image Embeddings, haben allerdings bemerkbare Unterschiede zu der ersten Version source <https://arxiv.org/pdf/2204.08387>. Auch die Architektur ist stark angepasst worden. Viele dieser Änderungen beziehen sich auf den Transformer, der sich nun nicht in der BERT Architektur befindet, wie in der ersten Version.

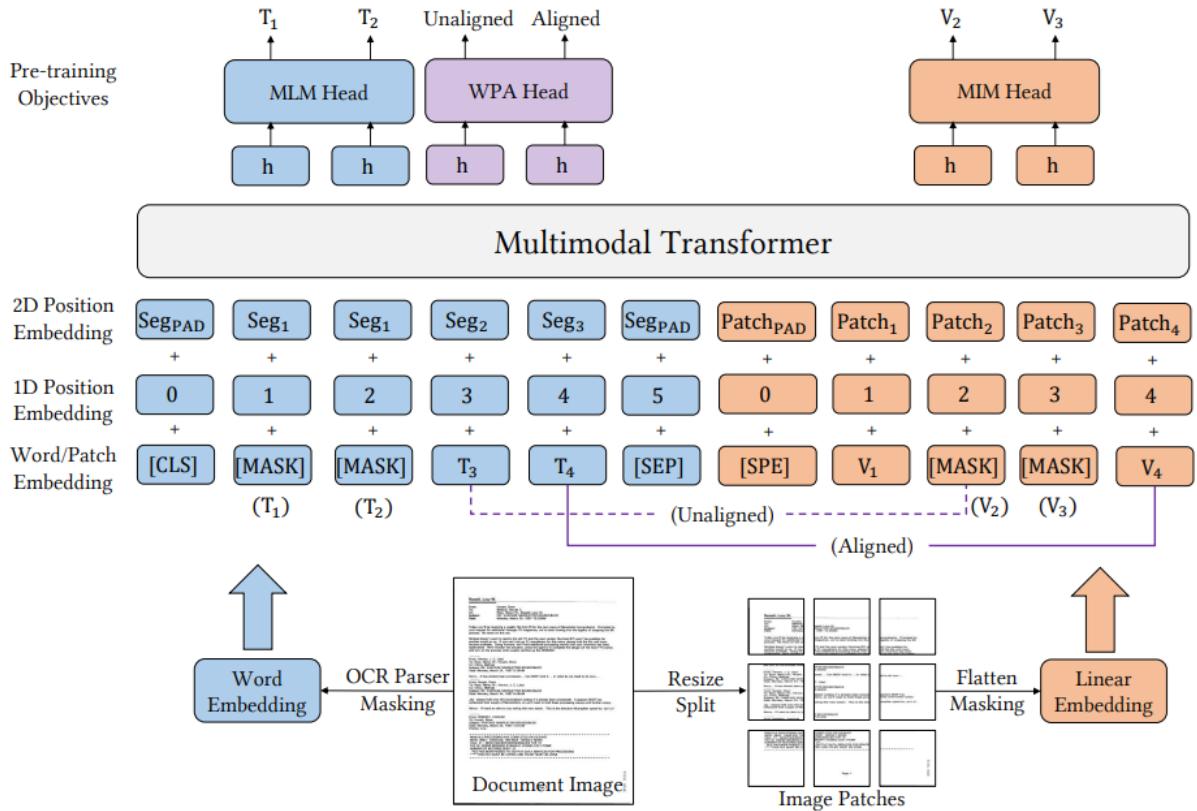


Abbildung 15: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

Der textuelle Kontext für die Text Embeddings werden weiterhin mit einem OCR Toolkit extrahiert. Die Word Embeddings werden allerdings nicht mehr mit einer BERT Architektur extrahiert, sondern mit einem vor trainierten RoBERTa Modell. in der dritten Version wird zusätzlich zu einem zweidimensionalen Position Embedding ein eindimensionales Position Embedding genutzt. Dieses 1D Position Embedding repräsentiert lediglich den Index der Tokens in der Textsequenz. Das 2D Position Embedding repräsentiert weiterhin die Koordinaten der Objekte auf der Bilddatei, gepaart mit der Höhe und Breite. Ein großer Unterschied zwischen den ersten beiden und der dritten Version von LayoutLM, ist die Änderung von Wort Repräsentationen zu Segment Repräsentationen. Die Autoren behaupten, dass mehrere Wörter eine gemeinsame Bedeutung haben und gruppieren diese daher in einzelnen Segmenten source <https://arxiv.org/pdf/2204.08387>.

Auch die Architektur für die Extrahierung von Image Embeddings hat sich bemerkbar verändert. Da die Autoren ein Leistungsproblem an dem R-CNN in der Vorgängerversi-

on bemerkt haben, werden für die Extrahierung für die visuellen Features, von “Vision Transformer” source [https://arxiv.org/pdf/2010.11929](https://arxiv.org/pdf/2010.11929.pdf) und “Vision-and-Language Transformer” source [https://arxiv.org/pdf/2102.03334](https://arxiv.org/pdf/2102.03334.pdf) inspirierte, lineare Projektionen genutzt. In dieser linearen Projektionsschicht werden die Bilder zunächst in Segmente zerlegt. Dann werden die Bildinformationen durch ein neuronales Netzwerk gesendet, das lediglich lineare Operationen verwendet, also auf gewöhnliche Aktivierungsfunktionen von ReLU verzichtet source <https://www.geeksforgeeks.org/what-is-a-projection-layer-in-the-context-of-neural-networks/>. Ziel dieser Schicht ist die Reduzierung der Dimension der Eingabevektoren. Umgesetzt wird dies durch eine Gewichtematrix die vorher angelernt wurde source <https://medium.com/@b.terryjack/deep-learning-the-transformer-9ae5e9c5a190>. An diese Repräsentationen werden eindimensionale positionale Embeddings beigefügt. Die Text und Image Embeddings werden danach in einen multimodalen Transformer gesendet. Die Autoren von LayoutLMv3 haben in der Veröffentlichung die Funktionsweise des multimodalen Transformer nicht genau wiedergegeben. Multimodale Transformer von vergleichbaren Modellen nutzten eine Verkettung von “Self-Attention Layers” innerhalb eines Embedding Typen, gefolgt von sogenannten “Co-Attention Layer”, die Attention auf den jeweilig anderen Embedding Typen errechnen. Ein Beispiel dafür ist der multimodale Transformer von ViLBERT source <https://proceedings.neurips.cc/paperfiles/paper/2019/file/c74d97b01eae257e44aa9d5bade97baf-Paper.pdf>

Für die Pre-training Aufgaben wird das bereits bekannte Masked Language Modeling (MLM) genutzt. Zusätzlich zu der Vorgängerversion wird Masked Image Modeling (MIM) genutzt, was Analog zu MLM die Maskierung von ungefähr 40% der Bild Patches erzeugt. Die dritte Pre-Training Aufgabe ist Word-Patch Alignment (WPA). WPA wird genutzt um zu erraten, ob ein Bild Patch, das mit einem Text korrespondiert, maskiert ist oder nicht.

Model	Parameters	Modality	Image Embedding	FUNSD F1↑	CORD F1↑	RVL-CDIP Accuracy↑	DocVQA ANLS↑
BERT _{BASE} [9]	110M	T	None	60.26	89.68	89.81	63.72
RoBERTa _{BASE} [36]	125M	T	None	66.48	93.54	90.06	66.42
BROS _{BASE} [17]	110M	T+L	None	83.05	95.73	-	-
LiLT _{BASE} [50]	-	T+L	None	88.41	96.07	95.68*	-
LayoutLM _{BASE} [54]	160M	T+L+I (R)	ResNet-101 (fine-tune)	79.27	-	94.42	-
SelfDoc [31]	-	T+L+I (R)	ResNeXt-101	83.36	-	92.81	-
UDoc [14]	272M	T+L+I (R)	ResNet-50	87.93	98.94†	95.05	-
TILT _{BASE} [40]	230M	T+L+I (R)	U-Net	-	95.11	95.25	83.92‡
XYLayoutLM _{BASE} [15]	-	T+L+I (G)	ResNeXt-101	83.35	-	-	-
LayoutLMv2 _{BASE} [56]	200M	T+L+I (G)	ResNeXt101-FPN	82.76	94.95	95.25	78.08
DocFormer _{BASE} [2]	183M	T+L+I (G)	ResNet-50	83.34	96.33	96.17	-
LayoutLMv3_{BASE} (Ours)	133M	T+L+I (P)	Linear	90.29	96.56	95.44	78.76
BERT _{LARGE} [9]	340M	T	None	65.63	90.25	89.92	67.45
RoBERTa _{LARGE} [36]	355M	T	None	70.72	93.80	90.11	69.52
LayoutLM _{LARGE} [54]	343M	T+L	None	77.89	-	91.90	-
BROS _{LARGE} [17]	340M	T+L	None	84.52	97.40	-	-
StructuralLM _{LARGE} [28]	355M	T+L	None	85.14	-	96.08	83.94‡
FormNet [25]	217M	T+L	None	84.69	-	-	-
FormNet [25]	345M	T+L	None	-	97.28	-	-
TILT _{LARGE} [40]	780M	T+L+I (R)	U-Net	-	96.33	95.52	87.05‡
LayoutLMv2 _{LARGE} [56]	426M	T+L+I (G)	ResNeXt101-FPN	84.20	96.01	95.64	83.48
DocFormer _{LARGE} [2]	536M	T+L+I (G)	ResNet-50	84.55	96.99	95.50	-
LayoutLMv3_{LARGE} (Ours)	368M	T+L+I (P)	Linear	92.08	97.46	95.93	83.37

Abbildung 16: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

Tabelle 3 Stellt ein Vergleich mit anderen Modellen dar, die für auf die Datensätze CORD, FUNSD, RVL-CDIP oder DOCVQA antrainiert wurden. Zwei Informationen der, von den Autoren veröffentlichten Tabelle, sind bemerkenswert für die Entscheidung der Wahl des Ansatzes für das exemplarische Antrainieren eines Models. Zunächst erzielt LayoutLMv3-Base eine bessere Genauigkeit als DocFormer-Base auf drei von vier Datensätzen. DocFormer ist der Vergangenheit eine sehr erfolgreiche Lösung im Bereich bildbasierten Klassifikation von Dokumenten gewesen. Daher ist das Erreichen einer höheren Genauigkeit eindrucksvoll. Das zweite Argument ist, dass LayoutLMv3 außerdem eine deutlich geringere Menge an Parametern benötigt, was unter anderem auf das verzichten von CNNs zurück zuschließen ist. LayoutLMv3 hat im Vergleich zu anderen State-of-the-Art Lösung eine sehr gute Genauigkeit und eine beeindruckend geringe Menge an Parametern, was die signifikanten Gründe dafür sind, dass diese Arbeit die dritte Version von LayoutLM nutzt

11 Prognostizierung des Ergebnisses

Die Veröffentlichungen der Forschungsergebnisse der Autoren ist eine gute Orientierung, was bei der exemplarischen Implementation von LayoutLMv3 Base zu erwarten ist. Die Genauigkeit der Klassifizierung auf den Datensatz RVL-CDIP beträgt laut Autoren 95,44% source <https://arxiv.org/pdf/2204.08387.pdf>. Es ist allerdings ein schlechteres Ergebnis in dem Experiment dieser Arbeit, im Vergleich der Experimente der Autoren, aus unterschiedlichen Gründen zu erwarten. In der Veröffentlichung von LayoutLMv3 wurde als Extrahierung der Text Embeddings lediglich von einem “off-the-shelf OCR toolkit” gesprochen source <https://arxiv.org/pdf/2204.08387.pdf>. Daher ist es möglich, dass in diesem Experiment eine OCR Lösung genutzt wird, die Text embeddings einer schlechteren Qualität erzeugt.

Es ist zu vermuten, dass die Autoren gezielte OCR Lösungen ausgewählt haben um die Genauigkeit zu optimieren. Dies ist in dieser Arbeit nicht der Fall. Ein weiterer Grund für die Prognostizierung einer geringeren Genauigkeit in diesem Experiment ist die Nutzung eines reduzierten Datensatzes. Wie in den Grundlagen erwähnt, steht die Qualität der Klassifizierungen eines Models in einer direkten Relation zu der Menge und Qualität des Datensatzes, das dem Model zur Verfügung gestellt wird. Da dieses Experiment lediglich zur Veranschaulichung einer bildbasierten Klassifikation von Dokumenten dient und 400.000 Trainingsdokumente dafür, aus eigener Einschätzung, nicht notwendig sind, wird ein reduzierter Datensatz genutzt.

Zwar wurde bei dem öffentlich zugänglichen reduzierten Datensatz hauptsächlich Daten entfernt die erwarteter weise zu einer schlechteren Klassifikation beitragen, also eine Form von Pre-Processing ist, allerdings ist die Relation zwischen der Menge der entfernten Daten und dem Gewinn durch Pre-Processing vermutlich nicht ausgeglichen. Mit anderen Worten schadet die Entfernung trotz Pre-Processing dennoch die Genauigkeit. Es ist dennoch zu erwarten, dass die Kombination des multimodalen Ansatzes und Stärken der Transformatoren und Attention-Mechanismen ausreichen um ein Modell anzutrainieren, dass auf den Datensatz RVL-CDIP eine bessere Genauigkeit erzeugt als reine CNN Lösungen.

12 Praktische Umsetzung

12.1 Vorbereitung der Entwicklungsumgebung

In dieser Arbeit sollen State-of-the-art Ansätze Implementiert werden und die Ergebnisse miteinander Analysiert und strukturell verglichen werden. Um das umsetzen zu können muss zunächst eine Entwicklungsumgebung auf dem lokalen Rechner aufgesetzt werden. Die derzeitige Forschung in vielen Bereichen des maschinellen Lernens nutzt Python als Programmiersprache, da Prototypen schnell damit umgesetzt werden können und die Möglichkeit von nativen Beifügen von anderen, performanteren Sprachen wie C, nicht ausgeschlossen ist. Eine Vielzahl an öffentlich zugänglichen Implementationen sind daher in Python geschrieben. Als Entwicklungsumgebung wird Visual Studio Code genutzt, da es nicht nur in der Lage ist Python Entwicklung zu ermöglichen, sondern auch sogenannte Python “Notebooks” darzustellen. Diese Notebooks dienen als strukturierte Segmentierung gesamter Python Programme in einzelne “Zellen”, welche unabhängig voneinander ausgeführt werden können. Damit beispielsweise eine Zelle eine Methode einer anderen aufrufen kann wird eine “Umgebung” benötigt.

12.2 Beschaffung der Dokumentenbilder

Wie bereits genannt wird das Beispielsmodel mit dem RVL-CDIP Datensatz antrainiert. Allerdings ist der gesamte Datensatz, bestehend aus 400.000 Dokumenten, nicht erforderlich für das Experiment, daher wird auf einen reduzierten Datensatz von 25.000 Dokumenten zurückgegriffen. Dieser Datensatz ist über die Plattform Kaggle verfügbar source <https://www.kaggle.com/datasets/vaclavpechtor/rvl-cdip-small-200>. Dort lässt sich der gesamte Datensatz herunterladen, welcher bereits in alle 16 Klassen unterteilt ist. Somit sind die Trainings- und Validierungsdaten vorhanden. Da allerdings noch getestet werden soll, werden die Validierungsdaten in Validierungs- und Testdaten geteilt.

13 Implementation von LayoutLMv3

LayoutLMv3 ist ein Modell das eine gute Genauigkeit in bildbasierter Klassifikation von Dokumenten, zu dem Zeitpunkt der Veröffentlichung, und noch immer erzielt. Daher

existieren mehrere unterschiedliche, öffentliche Implementation im Internet. Dies bietet die Möglichkeit mit vergleichsweise moderatem Aufwand Programme zu schreiben, die LayoutLMv3 nutzt um eine Klassifikation nach eigenem Bedarf ausführt. Als Grundlage nimmt diese Arbeit einen öffentlichen Quellcode, der als Einstieg in die Architektur dient source <https://www.mlexpert.io/blog/document-classification-with-layoutlmv3> und wird im Sinne des Experimentes angepasst.

13.1 Prüfung des CUDA Toolkits

Wie bereits beschrieben wird eine Grafikkarte für die Berechnung des Modells genutzt. Um zu überprüfen welche CUDA Version installiert ist und um die aktiven Prozesse auf der Grafikkarte zu überprüfen lässt sich der Befehl “nvidia-smi” nutzen.

```
Thu Nov  7 14:05:44 2024
+-----+
| NVIDIA-SMI 560.94          Driver Version: 560.94        CUDA Version: 12.6 |
|-----+-----+-----+
| GPU  Name        Driver-Model | Bus-Id      Disp.A | Volatile Uncorr. ECC | | |
| Fan  Temp  Perf  Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
| |                               |             |            | MIG M. |
+-----+-----+-----+
| 0  NVIDIA GeForce RTX 2060    WDDM   | 00000000:0A:00.0  On  |           N/A |
| 0%   49C   P8          19W / 170W | 746MiB / 6144MiB | 6%     Default |
|                                         |                  |           N/A |
+-----+-----+-----+
+-----+
| Processes:
| GPU  GI  CI          PID  Type  Process name          GPU Memory |
| ID   ID              ID   ID               Usage          |
+-----+
| 0  N/A  N/A          1332  C+G  ...nt.CBS_cw5n1h2txyewy\SearchHost.exe  N/A |
| 0  N/A  N/A          2252  C+G  ...2txyewy\StartMenuExperienceHost.exe  N/A |
| 0  N/A  N/A          5008  C+G  ...oogle\Chrome\Application\chrome.exe  N/A |
| 0  N/A  N/A          6376  C+G  ...Programs\Microsoft VS Code\Code.exe  N/A |
| 0  N/A  N/A          7920  C+G  ...siveControlPanel\SystemSettings.exe  N/A |
+-----+
```

Abbildung 17: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

13.2 Installation der Abhängigkeiten und Bibliotheken

Da sich viele Umsetzungen des maschinellen Lernens in gewissen Aspekten ähneln gibt es Bibliotheken die diese Anwendungsfälle bereits implementiert haben und abdecken. Diese werden mit dem Paketmanager pip installiert.

```
!pip install -qqq transformers==4.27.2 --progress-bar off
!pip install -qqq pytorch-lightning==1.9.4 --progress-bar off
!pip install -qqq torchmetrics==0.11.4 --progress-bar off
!pip install -qqq easyocr==1.6.2 --progress-bar off
!pip install -qqq Pillow==9.4.0 --progress-bar off
!pip install -qqq tensorboardX==2.5.1 --progress-bar off
```

```
!pip install -qqq huggingface_hub==0.11.1 --progress-bar off
```

Die Installation durch pip erfolgen durch Konsolenbefehle. Um dies in einem Python Notebook umzusetzen wird das Ausrufezeichen am Anfang jedes Konsolenbefehls gesetzt. Durch die pip install Operationen werden die folgenden Pakete installiert. Die Parameter “-qqq” und “–progress-bar off” sind lediglich Optionen für die Reduzierung der Ausgabe. Jede Installation beinhaltet eine konkrete Versionsnummer, um die Kompatibilität mit der Python Version und den Versionen der anderen Pakete zu gewährleisten.

Transformers ist eine, von Hugging Face zur Verfügung gestellte Sammlung vortrainierter Modelle source <https://pypi.org/project/transformers/>, wovon in dieser Implementation Nutzen gemacht wird. Pytorch-lightning ist ein Deep Learning Framework source <https://lightning.ai/docs/pytorch/stable/>, welches eine Vielzahl von Werkzeuge bietet Modellobjekte zu erzeugen, anzuwenden und zu manipulieren. Torchmetrics knüpft an pytorch-lightning an und bietet eine Sammlung an Metrikimplementierungen source <https://lightning.ai/docs/torchmetrics/stable/>.

Da LayoutLMv3 Textembeddings nutzt, muss ein OCR Modul genutzt werden. Es gibt eine Menge an Lösungen die genutzt werden können. Der Author der Implementation hat sich für EasyOCR entschieden, was ein OCR Modul mit über 80 unterstützten Sprachen ist source <https://github.com/JaidedAI/EasyOCR>. EasyOCR ist eine, in hauptsächlich Python entwickelte, Bibliothek und wird zu dem Zeitpunkt dieser Arbeit noch regelmäßig aktualisiert, mit dem letzten Aktualisierungsdatum am 24. September 2024. Da in dieser Arbeit keine Optimierung der Wahl der OCR Module geschicht, wird für das Experiment EasyOCR ebenfalls genutzt. Pillow ist eine Bibliothek die Bildverarbeitungsprozesse zur Verfügung stellt <https://pypi.org/project/pillow/>. Da für die Verwendung des Modells Bilder manipuliert und normiert werden müssen, ist ein solches Paket notwendig. TensorboardX ist ein Werkzeugkasten, der genutzt wird um relevante Metriken im Verlauf des Trainings und Testens zu visualisieren source <https://clear.ml/docs/latest/docs/integrations/tensorboardx/>. Die Plattform Huggingface wird genutzt um die Gewichte des trainierten Modells zu speichern und für das Testen zur Verfügung zu stellen. Um die Plattform nutzen zu können wird die Bibliothek huggingfacehub installiert source <https://huggingface.co/docs/huggingfacehub/en/installation>.

```

from transformers import LayoutLMv3FeatureExtractor, LayoutLMv3TokenizerFast, LayoutLMv3Processor, LayoutLMv3ForSequenceClassification
from tqdm import tqdm
import torch
from torch.utils.data import Dataset, DataLoader
import pytorch_lightning as pl
from pytorch_lightning.callbacks import ModelCheckpoint
from PIL import Image, ImageDraw, ImageFont
import numpy as np
import imgkit
import easyocr
import torchvision.transforms as T
from pathlib import Path
import matplotlib.pyplot as plt
import os
from typing import List
import json
from torchmetrics import Accuracy
from huggingface_hub import notebook_login
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

%matplotlib inline
pl.seed_everything(42)

```

Die gesamten Pakete werden importiert, oder lediglich die benötigten Funktionen der Pakete. Es werden wichtige Funktionen von LayoutLMv3 importiert und Datenstrukturen wie Datasets oder DataLoader. ModelCheckpoints sind Datenstrukturen, welche die Gewichte in einem Neuronalen Netzwerk darstellen. Des weiteren wird aus torchmetrics die Accuracy als Bewertungsmetrik importiert und aus sklearn.metrics die Funktion für das Erzeugen einer Verwechslungsmatrix und dessen Darstellung. Außerdem wird pytorch lightning genutzt um einen “Seed” zu definieren und zur Verfügung zu stellen. Dieser Seed wird genutzt um pseudo Zufälle zu erzeugen. Diese Zufälle werden beispielsweise genutzt um Datensätze zu mischen, damit es keine bestimmte Reihenfolge gibt in welcher das Modell die Trainingsdaten erhält.

13.3 Implementation des OCR Moduls

```

image_paths_train = sorted(list(Path("../DocumentDatasetMoreReducedLayoutLM/train").glob("*/*.tif")))
print(image_paths_train)
image = Image.open(image_paths_train[1]).convert("RGB")
width, height = image.size
Image

```

Bevor das OCR Modul importiert wird muss Zugang auf die Bildressourcen erzeugt und geprüft werden. Dafür wird ein relativer Dateipfad in eine Variable gespeichert und auf .tif Daten gefiltert. .tif ist der Dateityp der Datensets. image_path_train beeinhaltet nun eine Liste aller Dateipfade der Trainingsdaten, was durch print(image_paths_train) geprüft wird.

```
[WindowsPath( ' .. / DocumentDatasetMoreReducedLayoutLM / train / email / 0011834351. tif ' ),
```

Folgend wird ein konkretes Bildobjekt erzeugt und dargestellt um einen exemplarischen OCR Prozess mitzuverfolgen und zu prüfen.

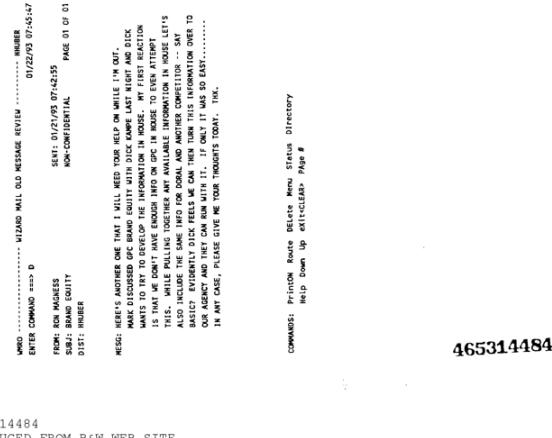


Abbildung 18: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

Dieses Bild stellt nun das originale Format dar. Die Darstellung und Formate wie Höhe und Breite sind die exakte Version wie die Dokumente in dem Datenset erhältlich sind.

```
reader = easyocr.Reader(['en'])
```

```
%%time
image_path = image_paths_train[1]
ocr_result = reader.readtext(str(image_path))
```

Eine Instanz von EasyOCR wird erzeugt und die Zielsprache englisch wird festgelegt. Folgend wird ein Dateipfad aus der Liste der Dateipfade selektiert und in image_path gespeichert. Dieser wird an die OCR Instanz für die Extrahierung gesendet und antwortet

mit einer Liste welche die “bounding box”, also eine Repräsentation der Orientierung auf dem Bild, den erkannten Text und das sogenannte “confident level” beinhaltet. Das confident Level repräsentiert die Wahrscheinlichkeit mit welchem, laut dem OCR Modul, der erkannte Text richtig ist.

```
ocr_result [66]
```

```
([[534, 857], [652, 857], [652, 889], [534, 889]],  
'465314484',  
0.9994514825704863)
```

Diese Ausgabe zeigt die 4 Koordinaten der Orientierung, den erkannten Text “465314484” und die hohe Wahrscheinlichkeit der Korrektheit von 0,99%. Die Bounding Box wird durch EasyOCR als Repräsentation der vier Ecken dargestellt, die den erkannten Text umranden. Dieses Format ist allerdings redundant, da beispielsweise die linken Ecken übereinander liegen und somit den gleichen X Wert besitzen. Um diese Redundanz aufzulösen wird eine Hilfsmethode genutzt.

```
def create_bounding_box(bbox_data):  
    xs = []  
    ys = []  
    for x, y in bbox_data:  
        xs.append(x)  
        ys.append(y)  
  
    left = int(min(xs))  
    top = int(min(ys))  
    right = int(max(xs))  
    bottom = int(max(ys))  
  
    return [left, top, right, bottom]
```

Um den OCR Prozess zu überprüfen wird das selbe Bild mit angewandten OCR dargestellt.

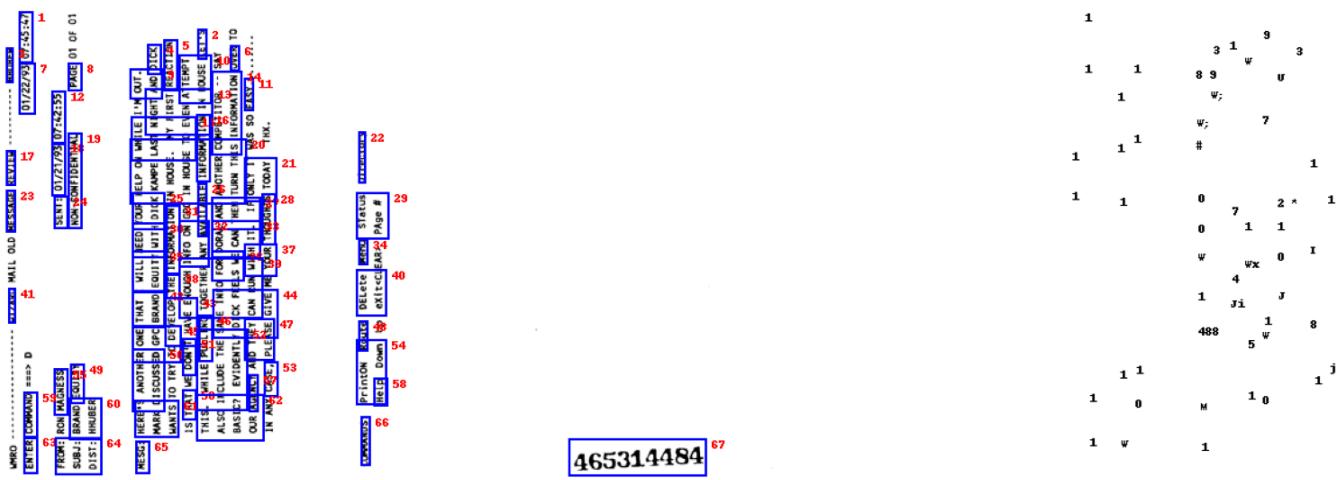


Abbildung 19: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

Das Bild befindet sich auf der linken Seite mit angewandten, durchnummerierten bounding boxes. Auf der rechten Seite ist der erkannte Text an der erkannten Stelle sichtbar. Auf den ersten Blick wird eine Schwäche erkannt. Der Großteil des Textes ist 90 Grad rotiert und somit sehr schlecht, bis gar nicht erkannt. Dennoch separieren die bounding boxes verhältnismäßig gut die Segmente. Die horizontalen Texte werden deutlich besser erkannt, wobei allerdings beispielsweise die Box 68 (unten links im Bild) die 4 nicht im eigentlichen Segment erkennt.

13.4 Pre Processing

```
feature_extractor = LayoutLMv3FeatureExtractor(apply_ocr=False)
tokenizer = LayoutLMv3TokenizerFast.from_pretrained("microsoft/layoutlmv3-base")
processor = LayoutLMv3Processor(feature_extractor, tokenizer)
```

Für das Pre Processing wird der Feature Extractor, Tokenizer und Processor initialisiert. Dafür werden die Transformer Imports von Huggingface genutzt. Da OCR in einem vorherigen Schritt durchgeführt wird, erhält der Feature Extractor den Parameter false für apply ocr. Da der Tokenizer antrainiert werden muss, wird ein Tokenizer genutzt, der bereits von Microsoft vor trainiert wurde. Um die Daten für das Modell vorzubereiten wird nun der Processor des LayoutLMv3 Modells benutzt.

```
encoding = processor(
    image,
    words,
    boxes=boxes,
    max_length=512,
    padding="max_length",
    truncation=True,
    return_tensors="pt"
)
```

Als Übergabeparameter werden die Bilddateien und die Liste der Wörter, die mit den einzelnen Bildern korrespondieren übergeben. Anschließend werden die bounding boxes übergeben und die maximale Länge wird auf 512 Tokens gesetzt. Mit dem Padding wird festgelegt, dass wenn weniger Tokens als 512 existieren, leere Tokens aufgefüllt werden

source <https://huggingface.co/docs/transformers/v4.46.2/en/modeldoc/layoutlmv3transformers.LayoutLMv>

Die truncation besagt, dass Eingaben länger als 512 abgeschnitten werden. Abschließend wird die Ausgabe als Datentyp PyTorch-Tensor definiert. Folgende Ausgabe präsentiert demnach das erforderliche Format für das Modell:

```
print(f """
input_ids: {list(encoding["input_ids"].squeeze().shape)}
word_boxes: {list(encoding["bbox"].squeeze().shape)}
image_data: {list(encoding["pixel_values"].squeeze().shape)}
image_size: {image.size}
""")
```

```
input_ids: [512]
word_boxes: [512, 4]
image_data: [3, 224, 224]
image_size: (762, 1000)
```

13.5 Datensatzobjekt

Um dem Modellobjekt die Daten zur Verfügung zu stellen muss eine Klasse definiert werden, welche die Länge des Datensets wiedergeben kann und die einzelnen Daten,

welche in dem Format “Input IDs”, “Attention Mask”, “Bounding Box”, “Pixel Values” und “Labels” wiedergegeben werden. Dafür wird der Klasse sowohl der Pfad der Daten zur Verfügung gestellt, als auch der bereits definierte Processor.

```
class DocumentClassificationDataset(Dataset):

    def __init__(self, image_paths, processor):
        self.image_paths = image_paths
        self.processor = processor

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, item):

        image_path = self.image_paths[item]
        json_path = image_path.with_suffix('.json')
        with json_path.open("r") as f:
            ocr_result = json.load(f)

        with Image.open(image_path).convert("RGB") as image:

            width, height = image.size
            width_scale = 1000 / width
            height_scale = 1000 / height

            words = []
            boxes = []
            for row in ocr_result:
                boxes.append(scale_bounding_box(row["bounding_box"], width_scale, height_scale))
                words.append(row["word"])

            encoding = self.processor(
                image,
                words,
                boxes=boxes,
                max_length=512,
                padding="max_length",
                truncation=True,
                return_tensors="pt"
            )

        label = DOCUMENT_CLASSES.index(image_path.parent.name)

    return dict(
        input_ids=encoding["input_ids"].flatten(),
        attention_mask=encoding["attention_mask"].flatten(),
        bbox=encoding["bbox"].flatten(end_dim=1),
        pixel_values=encoding["pixel_values"].flatten(end_dim=1),
        labels=torch.tensor(label, dtype=torch.long)
    )
```

Diese Klasse wird genutzt um die Training-, Validation- und Testdatensets zu definieren.

```
train_dataset = DocumentClassificationDataset(image_paths_train, processor)
val_dataset = DocumentClassificationDataset(image_paths_val, processor)
test_dataset = DocumentClassificationDataset(image_paths_test, processor)
```

Die Datasets sind folglich für die Pytorch DataLoader vorbereitet, welche als Übergabe-parameter ein Datasetobjekt erwarten, sowie die Batchgröße und einen Bool Wert für das Mischen des Datensets.

```

train_data_loader = DataLoader(
    train_dataset,
    batch_size=1,
    shuffle=True
)

val_data_loader = DataLoader(
    val_dataset,
    batch_size=1,
    shuffle=False
)

test_data_loader = DataLoader(
    test_dataset,
    batch_size=1,
    shuffle=False
)

```

Die Batchgröße hat einen starken Einfluss auf die Dauer des Trainings und ist abhängig der genutzten Hardware. Mehrere Iterationen des Trainings haben ergeben, dass eine Batchgröße von 1 das Training, auf dem System welches für dieses Experiment genutzt wird, ausreichend beschleunigt. Der anfangs definierte Seed wird hier genutzt um die Trainingsdaten zu mischen. Die Validierung und das Testen kann ohne negative Effekte auf ungemischten Datensätzen erfolgen.

13.6 Training

Um das Modell antrainieren zu können muss zunächst eine Modellinstanz erzeugt werden. Dafür wird das vortrainierte Modell von Microsoft genutzt. Als Parameter für “num_labels” wird 2 genutzt um ein binäres Problem festzulegen.

```
model = LayoutLMv3ForSequenceClassification.from_pretrained("microsoft/layoutlmv3-base", num_labels=2)
```

Die Korrektheit der erkannten Klassen werden geprüft. Als Datenstruktur werden Ordner verwendet die als Ordnernamen die Klassen beinhalten. In den Ordner werden alle Bilddateien mit den zugehörigen Klassen abgelegt.

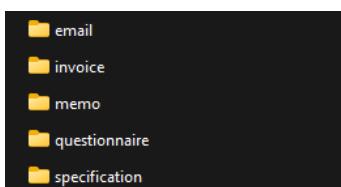


Abbildung 20: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

```

DOCUMENT_CLASSES = sorted(list(map(lambda p: p.name, Path("../DocumentDatasetMoreReducedLayoutLM/train").glob("*"))))
DOCUMENT_CLASSES

```

Folgende Ausgabe zeigt die Korrekte Auflistung der erkannten Klassen

```
[ 'email', 'invoice', 'memo', 'questionnaire', 'specification' ]
```

Um die vortrainierten Gewichte in ein Modell zu laden muss zunächst die Struktur des Modell Moduls definiert werden. Dafür wird die ModelModule Klasse definiert und erbt von der, von Pytorch Lightning zur Verfügung gestellten Klasse “LightningModule”. Es werden zunächst Objekte definiert, welche die Zuweisungen der Enumeratoren repräsentieren, sowie die Definitionen der Accuracy Metriken.

Die Methode “Forward” wird von dem Pytorch Lightning Modul für die Prognose der Klassen, beziehungsweise die Schlussfolgerung source <https://pytorch-lightning.readthedocs.io/en/1.6.0/start>

Die Training- und Validationstep Methoden definieren die Schleife des Training und Validationsvorganges. Abschließend wird der Optimizer konfiguriert.

```
class ModelModule(pl.LightningModule):
    def __init__(self, n_classes:int):
        super().__init__()
        self.model = LayoutLMv3ForSequenceClassification.from_pretrained(
            "microsoft/layoutlmv3-base",
            num_labels=n_classes
        )
        self.model.config.id2label = {k: v for k, v in enumerate(DOCUMENT_CLASSES)}
        self.model.config.label2id = {v: k for k, v in enumerate(DOCUMENT_CLASSES)}
        self.train_accuracy = Accuracy(task="multiclass", num_classes=n_classes)
        self.val_accuracy = Accuracy(task="multiclass", num_classes=n_classes)

    def forward(self, input_ids, attention_mask, bbox, pixel_values, labels=None):
        return self.model(
            input_ids,
            attention_mask=attention_mask,
            bbox=bbox,
            pixel_values=pixel_values,
            labels=labels
        )

    def training_step(self, batch, batch_idx):
        input_ids = batch["input_ids"]
        attention_mask = batch["attention_mask"]
        bbox = batch["bbox"]
        pixel_values = batch["pixel_values"]
        labels = batch["labels"]
        output = self(input_ids, attention_mask, bbox, pixel_values, labels)
        self.log("train_loss", output.loss)
        self.log("train_acc", self.train_accuracy(output.logits, labels), on_step=True, on_epoch=True)
        return output.loss

    def validation_step(self, batch, batch_idx):
        input_ids = batch["input_ids"]
        attention_mask = batch["attention_mask"]
        bbox = batch["bbox"]
        pixel_values = batch["pixel_values"]
        labels = batch["labels"]
        output = self(input_ids, attention_mask, bbox, pixel_values, labels)
        self.log("val_loss", output.loss)
        self.log("val_acc", self.val_accuracy(output.logits, labels), on_step=False, on_epoch=True)
        return output.loss
```

```

def configure_optimizers(self):
    optimizer = torch.optim.Adam(self.model.parameters(), lr=0.00001) #1e-5
    return Optimizer

```

Damit die Schlussfolgerung im Bezug auf das Lightning Modul erfolgen kann, muss das Modell mit allen relevanten Informationen übergeben werden. Daher gibt die Forward Methode das Modell Objekt wieder, welche selbst die Input ID, Attention Mask, Bounding Box, Pixel Value und Label als Parameter besitzt.

Die Training- und Validationstep Methode benötigt diese Parameter auch, aber da diese Prozesse nicht nur auf einzelne Daten pro Zyklus angewendet werden, sondern auf sogenannte “Batches”, werden Batches übergeben und nicht die Parameter selber. Darauf folgend wird die Verlustfunktion errechnet, als Log ausgegeben und der aufrufenden Funktion wiedergegeben. Diese aufrufende Funktion befindet sich im “Trainer”, der im Folgenden Code zu finden ist. Dieser Trainer stellt eine Abstraktion für den Trainingsprozess dar. Als Optimierung wird “Adam” genutzt, dieser ist Teil der torch Bibliothek. Aus dieser wird Adam aufgerufen und mit den definierten Parametern, zusätzlich der Lernrate von 0.00001 übergeben. Anschließend wird der Optimierer aus der Funktion configure_Optimizers übergeben.

Nach dem Training muss das Modell mit allen Gewichten gespeichert werden um es für Tests zu nutzen. Daher wird die Variable model_checkpoint definiert mit den Parametern, die definieren welches Modell aus dem Trainingsprozess gespeichert werden soll.

```

model_checkpoint = ModelCheckpoint(
    filename="{epoch}-{step}-{val_loss:.4f}", save_last=True, save_top_k=3, monitor="val_loss", mode="min"
)

```

Nun sind alle Vorbereitungen für das Training abgeschlossen. Der vorher definierte Trainer wird jetzt mit der Methode “.fit” genutzt um mittels Modell Modul und den Dataloader die Gewichte anzutrainieren.

```

trainer.fit(model_module, train_data_loader, val_data_loader)

```

Die Ausgabe stellt das Modell dar mit allen Parametern und Parametergrößen.

	Name	Type	Params
0	model	LayoutLMv3ForSequenceClassification	125 M
1	train_accuracy	MulticlassAccuracy	0
2	val_accuracy	MulticlassAccuracy	0

125 M	Trainable params
0	Non-trainable params
125 M	Total params
251.843	Total estimated model params size (MB)

Der gesamte Trainingsprozess wird mit Fortschrittsbalken als Ausgabe visualisiert. Dort ist der derzeitige Fortschritt zu sehen, die durchschnittliche Trainingsgeschwindigkeit und der aktuelle Verlust.

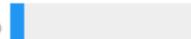
Epoch 0: 7%  834/11123 [02:37<32:18, 5.31it/s, loss=1.02, v_num=74]

Abbildung 21: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

Auf jede Epoche folgt eine Validierung. Diese errechnet ebenfalls den Verlust, aktualisiert die Gewichte aber nicht. Es dient zur Übersicht der Leistung des Modells während dem Training.

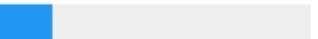
Validation DataLoader 0: 19%  477/2478 [00:37<02:38, 12.64it/s]

Abbildung 22: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

Da als Abbruchkondition eine maximale Anzahl an Epochen definiert wurde stoppt das Training nach genau 30 Epochen.

`'Trainer.fit' stopped: 'max_epochs=30' reached.`

Metriken während des Trainings

TensorBoard, die Bibliothek die für die Visualisierung genutzt wird, zeigt die definierten Metriken während des Trainings und der Evaluation.

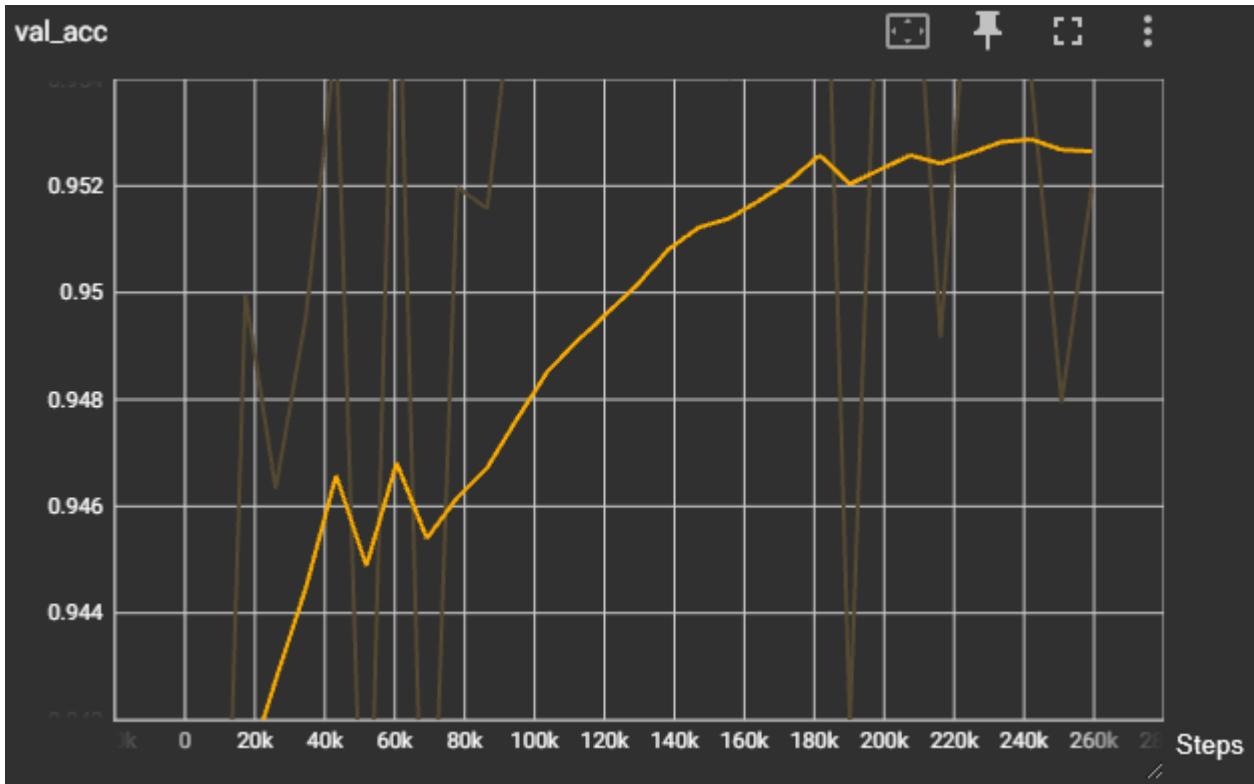


Abbildung 23: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

Abbildung ??? zeigt einen Graph, der die Genauigkeit der Klassifizierung auf das Validierungsdataset auf der Y-Achse und die durchgeföhrten Schritte des Trainingsprozesses auf der X-Achse anzeigt. Der dunklere Graph stellt die exakten Werte der Genauigkeit dar. Der helle Graph ist eine geglättete Variante der selben Werte, welche der Darstellbarkeit dient.

Mit einigen Ausnahmen zu beginn des Trainingsprozesses ist eine stetige Steigerung der Genauigkeit zu bemerken. Ab dem Schritt 180.000 liegt die Validierungsgenauigkeit bei etwas über 95,2%. Ab ungefähr diesem Punkt das Phänomen des “Overfitting” zu sehen. Obwohl der Trainingsprozess weiterhin durchgeführt wird und sich weiterhin die Genauigkeit ändert, gibt es keine Tendenz der Verbesserung zu sehen und stagniert auf einem ungefähren Niveau. Es ist daher zu vermuten, dass dieses Niveau die “Barriere” darstellt, über das die Genauigkeit nicht mehr kommt.

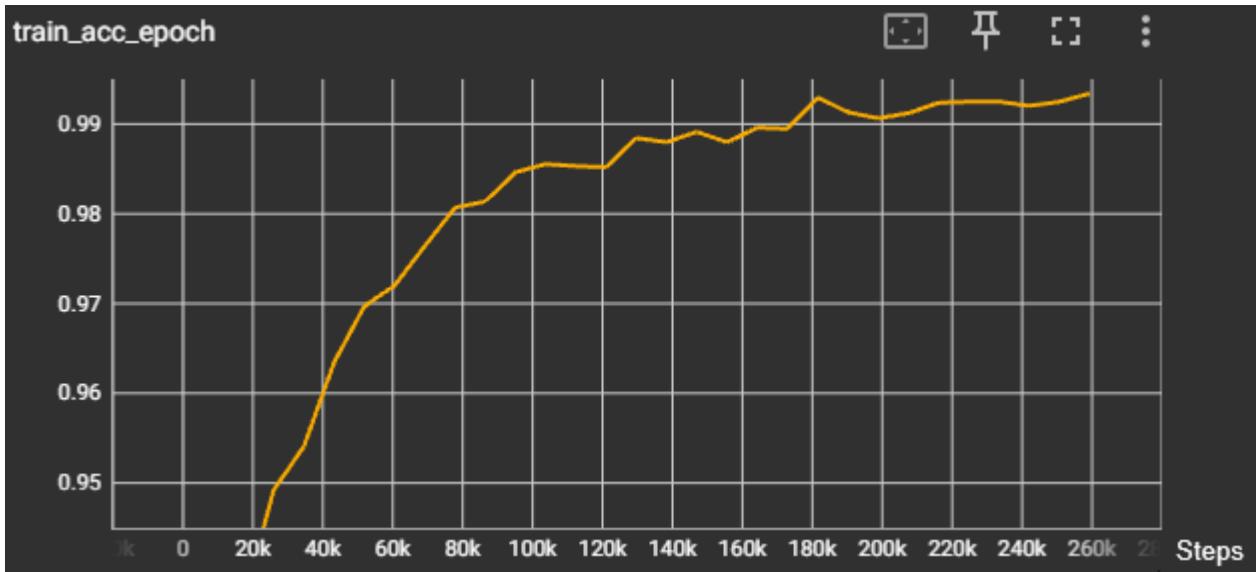


Abbildung 24: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

Abbildung ??? stellt stattdessen den Durchschnitt jeder Genauigkeit der einzelnen Epochen auf das Trainingsdatenset, auf den selben Schritten dar. Da es sich bereits um Durchschnitte handelt und dadurch starke Sprünge die Darstellbarkeit nicht beeinflusst wurde hier auch keine Glättung angewendet.

Ähnlich zur Abbildung ??? gibt es eine starke Tendenz der Steigerung der Genauigkeit, flacht danach allerdings stark ab, bis zu einem Punkt der Stagnierung. Dieser Punkt scheint hier bei ungefähr 99% zu sein. Im Vergleich zu den Validierungsdaten ist die Genauigkeit auf den Trainingsdaten deutlich besser. Das ist auch zu erwarten, da das Modell, bis auf das Pre-Training das vorher stattgefunden hat, ausschließlich mit den Daten trainiert wird, dessen Genauigkeit in dieser Abbildung dargestellt wird. Eine gute Genauigkeit auf das Trainingsdatenset ist allerdings keineswegs ein Indikator für gute Generalisierung des Modells, sondern bloß ein Indikator dafür, dass das Modell das Datenset, das für das Training genutzt wird, gut klassifizieren kann.

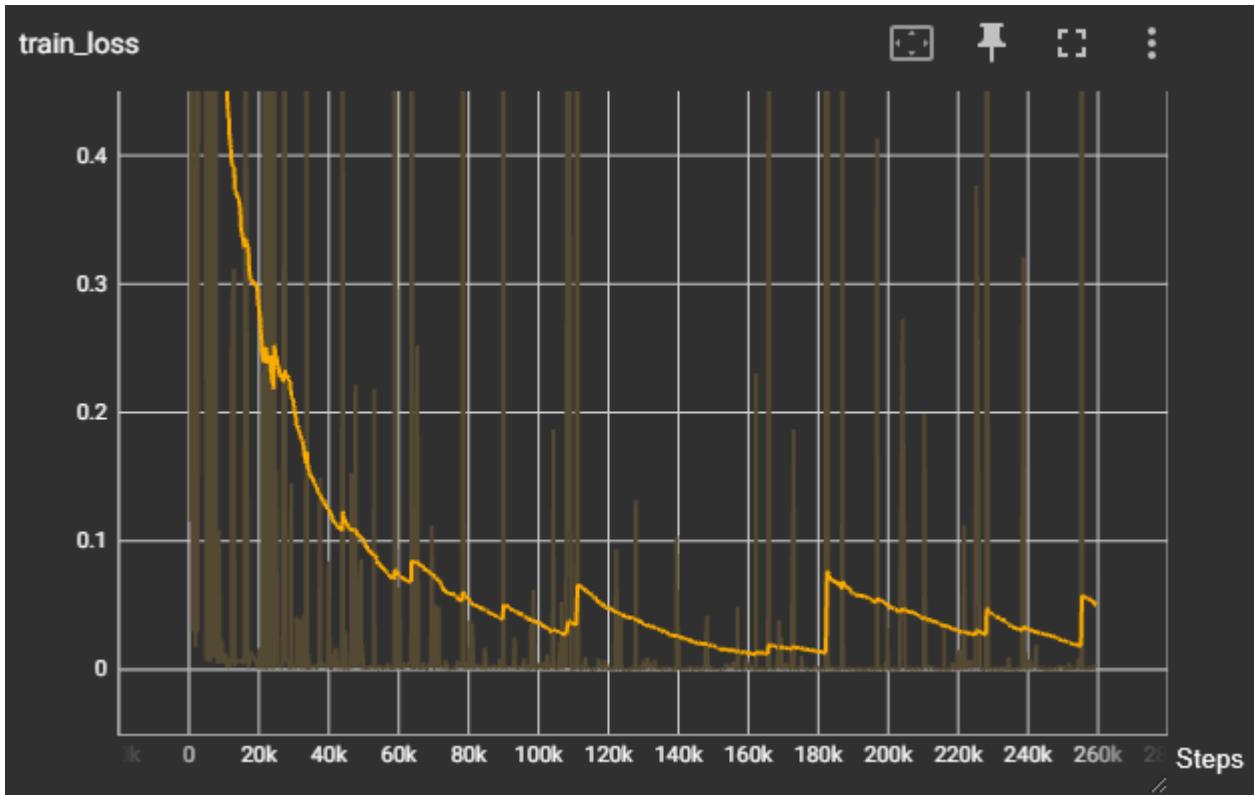


Abbildung 25: LayoutLMv3 Architektur

Quelle: <https://arxiv.org/pdf/2204.08387>

Abbildung ??? zeigt den errechneten Wert der Verlustfunktion, ebenfalls in Relation der durchgeföhrten Schritte des Trainingsprozesses. Da während des Trainings starke Fehler auftreten können, sind viele Sprünge in den Rohdaten zu finden, daher wird auch dieser Graph, wie in Abbildung ???, geglättet. Der geglättete Graph stellt eine erwartete Tendenz dar. Der Verlust sinkt schnell zu Beginn des Trainings und flacht darauf hin stark ab, bis er Graph einen Intervall erreicht, der nicht verlassen wird. Dieser Intervall ist hier ungefähr 0.8 bis 0.2.

13.7 Testen

Nachdem das Model antrainiert wurde werden die Testdaten klassifiziert um anhand dessen die Leistung des Modells zu überprüfen. Dafür wird eine weitere Hilfsmethode definiert. Die Methode predict_document_image soll das Label von einzelnen Bildern zurückgeben.

```
def predict_document_image(
    image_path: Path,
    model: LayoutLMv3ForSequenceClassification,
    processor: LayoutLMv3Processor):
```

```

json_path = image_path.with_suffix(".json")
with json_path.open("r") as f:
    ocr_result = json.load(f)

    with Image.open(image_path).convert("RGB") as image:

        width, height = image.size
        width_scale = 1000 / width
        height_scale = 1000 / height

        words = []
        boxes = []
        for row in ocr_result:
            boxes.append(
                scale_bounding_box(
                    row["bounding_box"],
                    width_scale,
                    height_scale
                )
            )
        words.append(row["word"])

encoding = processor(
    image,
    words,
    boxes=boxes,
    max_length=512,
    padding="max_length",
    truncation=True,
    return_tensors="pt"
)

with torch.inference_mode():
    output = model(
        input_ids=encoding["input_ids"].to(DEVICE),
        attention_mask=encoding["attention_mask"].to(DEVICE),
        bbox=encoding["bbox"].to(DEVICE),
        pixel_values=encoding["pixel_values"].to(DEVICE)
    )

predicted_class = output.logits.argmax()
return model.config.id2label[predicted_class.item()]

```

Ein Großteil der Definitionen sind ähnlich zu dem verwendeten Code für das Training.

Eine Besonderheit in dieser Hilfsfunktion ist die Nutzung von “`torch.inference_mode()`”,

was einen Modus darstellt, in welchem Berechnungen die für die reine Klassifizierung

nicht benötigt werden nicht getätigter werden source https://pytorch.org/docs/stable/generated/torch.autograd.inference_mode.html

Ein Vorteil dafür ist, dass wenn ein Gradient nicht berechnet wird, der Prozess schneller durchlaufen kann.

Diese Hilfsfunktion wird dann auf alle Bilder der Testdaten angewandt und es werden zwei Listen erstellt. In den Listen befinden sich die tatsächlichen Labels und die Labels, die das Modell ermittelt hat.

14 Anhang

14.0.1 Optimierungsprobleme

Eine Optimierung ist die Suche unter allen Alternativen in einem Zielkriterium, nach einer besten Lösung für ein Problem [2]. Solche Probleme lassen sich durch Optimierungsmodelle darstellen. Optimierungsmodelle bestehen aus Daten, einer zulässigen Menge, einer Zielfunktion und Entscheidungsvariablen [3]. Die Rahmenbedingungen des Anwendungsfalls werden durch die Daten quantifiziert.

Beispielsweise werden sie durch Kosten in einem Anwendungsfall dargestellt. Daten sind unveränderliche Größen die während der Optimierung fest bleiben. Entscheidungsvariablen sind die Werte, die während des Optimierungsprozesses verändert werden. Die optimalen Werte der Entscheidungsvariablen werden gesucht. Die Entscheidungsvariablen müssen sich in einem zulässigen Bereich befinden. Dieser Bereich wird als Menge definiert und kann für einzelne, oder alle Variablen gelten. Optimierungsprobleme können zum Beispiel durch Minimierungsprobleme oder Maximierungsprobleme dargestellt werden. Das könnte in einem Maximierungsproblem bedeuten, dass die Summe der Daten multipliziert mit den Entscheidungsvariablen größtmöglich errechnet werden soll, im Rahmen der Bedingungen.

Ein Gängiges Beispiel für ein Maximierungsproblem ist das "Leeren des Münzfachs", also beim Kauf eines Produkts, soll der Kaufpreis durch so viele Münzen wie möglich erreicht werden. Der Wert der Münzen wird durch die Daten beschrieben, die gewählte Menge der Münzen durch die Entscheidungsvariablen und die verfügbare Menge der Münzen durch die zulässige Menge.

Jedes Minimierungsproblem lässt sich als äquivalentes Maximierungsproblem definieren, da man auch die negative Zielfunktion maximieren kann, anstatt die Zielfunktion zu minimieren.

14.0.2 Gradienten

Mathematische Funktionen lassen sich nach Variablen ableiten. Die Errechnung eines Gradienten ist die erste Ableitung einer Funktion. Besteht die Funktion aus mehreren Variablen entsteht ein Gradient aus einem Vektor der partiellen Ableitungen nach allen Variablen.

Gradienten haben die Eigenschaft, dass sie in die Richtung des steilsten Anstiegs zeigen. Lässt man sich also beispielsweise die Zielfunktion eines Optimierungsproblems visuell darstellen, zeigt ein Gradient an einem beliebigen Punkt, senkrecht seiner Höhenlinie. Da der Gradient nützliche Informationen über Steigungen hat, können Gradienten als Werkzeug genutzt werden um Optimierungsprobleme zu lösen. Das numerische Verfahren der Nutzung des Gradienten zum Ermitteln eines Minimums wird Gradientenabstiegsverfahren genannt. Bei diesem Verfahren wird ein beliebiger Startpunkt genommen und die negative Gradientenrichtung wird verfolgt bis der Tiefpunkt gefunden wurde. In der Praxis können Gradientenabstiegsverfahren durch sogenannte Hyperparameter modifiziert werden. Beispielsweise lässt sich die Distanz, die der Gradient gefolgt wird verändern. Es können auch Abbruchsbedingungen definiert werden, die das Verfahren stoppen. Beispielsweise kann eine bestimmte Dauer gesetzt werden, damit in der Praxis überflüssige Rechenzeit gespart wird.

Das Gradientenabstiegsverfahren ist eine gängige Methode für das Maschinelle Lernen, da dort eine “Verlustfunktion” definiert wird und anhand der Funktion ein Minimum gesucht wird.

14.0.3 Binäre Klassifikationsprobleme

Das oben genannte Beispiel, die Klassifizierung zwischen Hund und Katze ist ein Binäres Klassifikationsproblem, da es zwei Klassen besitzt. Binäre Klassifikationsprobleme lassen sich häufig als “Ja, nein” Probleme definieren. In der Medizin können solche Modelle genutzt werden um Bilder von beispielsweise Muttermalen als “gesund” oder “verdächtig” zu klassifizieren.

14.0.4 Multi-Klassen Klassifikationsprobleme

Die Gemeinsamkeit zwischen dem binären und dem multi-Klassen Klassifikationsproblem ist dass eine Entscheidung für eine Klasse getroffen werden soll. Der Unterschied liegt darin, dass Multi-Klassen Klassifikationsprobleme mehr als zwei Klassen untersucht. So wird aus dem Beispiel der Hund-Katze Klassifizierung ein Multi-Klassen Problem, wenn der Klassifikator auch die Entscheidung treffen muss, ob es sich bei dem Tier um ein Hase oder eine Maus handeln könnte. In der Theorie kann sich ein solches Klassifikationsproblem mit beliebig vielen Klassen beschäftigen, allerdings steigt die Komplexität der Modelle mit jeder zusätzlichen Klasse [5]. Das bedeutet in der Praxis, dass Klassifikatoren mehr Parameter und mehr Daten benötigen um eine breit gefächerte Klassifizierung vornehmen zu können.

14.0.5 Multi-Label Klassifikationsprobleme

Die Gemeinsamkeit zwischen dem Multi-Klassen und Multi-Label Klassifikationsproblem ist dass mehr als zwei Entscheidungsmöglichkeiten gibt. Der Unterschied liegt darin, dass das der Klassifikator des Multi-Label Klassifikationsproblem mehr als eine Entscheidung treffen kann. Der Klassifikator entscheidet sich als nicht zwingend nur für eine Klasse, sondern gegebenenfalls für mehrere.

Forschungsarbeiten der Medizin erweitern das Problem der Klassifizierung der Muttermale [6]. Dort werden Muttermale nicht nur auf “gesund” oder “verdächtig” untersucht, sondern auch auf die Konkrete Art des Verdachts, was die Menge der Klassen über zwei heraus erweitert. Außerdem können die Daten der Muttermale mehr als einen Verdacht gleichzeitig aufweisen. Es ist aber weiterhin möglich dass der Klassifikator nur für eine Klasse entscheidet.

14.0.6 Künstliche Neuronen

Künstliche Neuronen sind Objekte, die innerhalb eines künstlichen neuronalen Netzes (KNN) die selbe Funktionalität ausüben, aber unterschiedliche variable Werte annehmen können [10]. KNNs können unterschiedlich viele Neuronen beinhalten. Deep Learning zeigt sich normalerweise dadurch aus viele Schichten von Neuronen zu beinhalten.

Jedes einzelne Neuron ist in der Lage einen Wert zu erhalten und diesen in einer

Funktion zu einem Ausgangswert zu verrechnen. Die Funktion der Neuronen ist aus dem biologischen Vorbild nachgestellt. Es werden alle eingehenden Werte x mit jedem entsprechenden Gewicht w , wobei die Werte das elektrische Signal repräsentieren. Diese Summe wird dann mit einem sogenannten Bias b summiert. Menschliche Neuronen haben die Eigenschaft nach einem gewissen Schwellwert des elektrischen Signals zu "feuern". Diese Eigenschaft wird in der künstlichen Repräsentation durch einer Aktivierungsfunktion f dargestellt [13].

14.0.7 Aktivierungsfunktion

Viele Probleme des maschinellen Lernens sind nicht lineare Probleme. Um ein Modell nachzustellen, dass nicht lineare Probleme löst, müssen nicht lineare Aktivierungsfunktionen genutzt werden. Es können unterschiedliche Aktivierungsfunktionen gewählt werden. Häufig genutzt werden die Sigmoid und die Rectified Linear Unit (ReLU) [14]. ReLU ist eine simple Funktion die für positive Werte den Wert unverändert weiter gibt und für negative Werte eine Null. Wie bereits bekannt wird bei einem Training regelmäßig ein Gradient des Modells errechnet. Die Aktivierungsfunktion hat einen starken Einfluss auf den Gradienten. Es gibt unterschiedliche Phänomene, wie den "exploding gradient" [15] und den "vanishing gradient". In beiden Fällen entsteht eine Problematik mit der Nutzung eines Gradienten, da er dort Werte annimmt, die schwierig zu nutzen sind. Vereinfacht lassen sich diese Phänomene erklären, dass der exploding gradient große Werte annimmt und der vanishing gradient kleine. Diese Probleme treten vor allem dann auf, wenn die Modelle "tief" werden, also eine große Menge von Schichten besitzt.

Die ReLU Funktion löst das Problem der rechnerischen Komplexität der Sigmoid und Tanh Funktion.

Der Nachteil der ReLU Funktion ist der vanishing gradient bei negativen Werten. Es gibt viele Variationen der ReLU Funktion und eine davon ist die Leaky Rectified Linear Unit (LReLU), welche positive Wert wie die ReLU behandelt, negative Werte aber nicht auf 0, sondern den Eingang mit einer kleinen Konstante multipliziert. Diese Variation soll den vanishing gradient entgegenwirken, bringt allerdings die Herausforderung eine Konstante zu finden, die gut zu dem Anwendungsfall passt.

Künstliche neuronale Netze bestehen aus unterschiedlichen Schichten, die unterschiedliche Zwecke erfüllen. Eine Schicht besteht in diesem Kontext aus einer Menge aus Neuronen, die mit anderen Neuronen verbunden werden kann. “Fully-Connected Layer” sind Schichten, bei welcher jedes Neuron einer Schicht mit Neuronen der Nachbarschichten verbunden sind [16]. Fully-Connected Layer werden typischerweise im Bereich des maschinellen Lernen genutzt. Es gibt unterschiedliche Arten von Schichten. Es gibt “Input Layer”, “Hidden Layer” und “Output Layer”. Der Input Layer dient als “Startpunkt” des Modells. Diese Schicht schickt die Eingabe an die nachfolgende Schicht. Die Eingabe ist in der Regel eine Repräsentation einer Datei, über die das Modell eine “Lösung” finden soll. Als Lösung ist hier zum Beispiel eine Klassifizierung gemeint.

Wenn ein Modell physische Informationen verarbeiten soll (zum Beispiel ein Dokument oder gesprochenes Wort), muss das physische Objekt als digitalen Zahlenwert, oder eine Kombination von Zahlenwerten übersetzt werden, da Neuronen reale Werte als Eingabe nehmen. Eine solche Übersetzung könnte eine Bitrepräsentation eines digitalen Fotos eines Dokuments sein. Die letzte Schicht eines künstlichen neuronalen Netztes wird “Output Layer” genannt. Der Output Layer stellt das “Ergebnis” des Modells dar. Ein Neuron eines Output Layer in einem Modell für ein Klassifikationsproblem könnte also die Wahrscheinlichkeit für die Zugehörigkeit einer Klasse bedeuten. Daher haben Modelle für Klassifikationen typischerweise so viele Neuronen in einem Output Layer wie die Anzahl der Klassen.

Hidden layer sind alle Schichten die sich zwischen des Input und Output Layers befinden. Diese Schichten werden unter anderem “versteckt” genannt, weil im Gegensatz der Eingabe und Ausgabe Schicht keine Verbindung zur Außenwelt besteht.

Einige Eigenschaften wie die Vollständige Verbindung zwischen benachbarten Schichten können angepasst werden um die Leistung der Modelle zu verbessern. Es können also Verbindungen zwischen Schichten gelöscht werden, um die Genauigkeit zu verbessern [17]. Es gibt außerdem Anpassungen wie “Skip Connections”, die den Zweck erfüllen Werte an Schichten vorbei zu tragen, diese also zu “Überspringen” [18].

14.0.8 Pre-Processing

Viele Maschine Learning Modelle werden genutzt um Probleme in der “echten Welt” zu lösen, also nehmen diese Modelle physische Daten, in digitaler Repräsentation entgegen und lösen ein definiertes Problem. Neben dem bereits genannten Beispiel über Digitalisierung von Bilddateien, gibt es auch Beispiele über andere Arten von Daten. Es gibt Architekturen wie Natural Language Processing (NLP) die sich mit Text beschäftigen [19]. Der Aspekt, dass physische Informationen digitalisiert werden müssen ist ähnlich zu dem Beispiel der Bildverarbeitung, allerdings ist die Methode der Digitalisierung weniger intuitiv. Bei Bildern bietet es sich an, die RGB kodierte Repräsentation für das Modell zu nutzen, da die Modelle mittels sogenannten Filtern über eine solche Repräsentation die Features erkennen können. Moderne Lösungen zu NLP greifen allerdings nicht auf die typische Kodierung wie beispielsweise ASCII-Code zurück um die Daten zu Repräsentieren.

Tokenization “Tokenization”, im Kontext von NLP, ist der Prozess der Übersetzung von menschlich lesbaren Sätzen in eine Sequenz von bestimmten Tokens die für statistische Modelle genutzt werden können.

Tokenization ist einer der ersten Schritte eines natural language processing Modells [20]. In der typischen originalen Repräsentation eines Textes, ist ein Text lediglich eine lange Liste von Charakteren, die je nach Kodierung anders übersetzt werden. Eine Tokenization könnte also eine oder mehrere dieser Charaktere in einer anderen Darstellung als unterschiedliche Tokens repräsentieren. Als Beispiel könnte das Wort “Hallo” als 5 Charaktere in einem digitalen Dokument dargestellt werden, aber durch ein Tokenization Prozess als genau ein Token.

Tokenization ist allerdings nicht zwingend der erste Prozess der in der Datenverarbeitung geschieht. Digitale Texte haben häufig Informationen wie Textgröße, Textart oder ähnliches. Außerdem können Dokumente auch Informationen wie Bilder oder andere Formatierungen haben. NLP Modelle befassen sich häufig mit den Sätzen allein, ohne diese Darstellungsinformationen. Die Modelle, die in dieser Arbeit untersucht werden, zeigen jedoch, dass diese Informationen sehr wohl von Relevanz von Klassifizierung sein können. Daher sollten diese Informationen, je nach genutztes Modell, vorzugsweise nicht

gelöscht werden.

Transformationen Im Verlauf dieser Arbeit wird die Methode der “Faltung” vorgestellt, die sich in der Forschung der Bilderklassifizierung der letzten Jahren bewährt hat. Typischerweise erwarten solche Modelle eine feste Skalierung von Höhe und Breite der Bilddateien. Eine Lösung könnte sein nur Daten zu nutzen die diese Erwartung erfüllt, was die Menge der zu nutzenden Daten stark begrenzen kann. Daher ist eine häufige Lösung die zu nutzenden Daten in der gewünschte Form zu transformieren. Transformationen können zum Beispiel Translationen, Skalierungen, Dehnungen, Scherungen oder Rotationen sein [21]. Isometrische Transformationen ändern die Position und Orientierung eines Objekts und behält die Form und Größe [22]. Solche Transformationen werden in einem späteren Thema, der “Data Augmentation” aufgegriffen.

Es werden also nicht-isometrische Transformationen genutzt um die gewünschte Form von Bildern zu erreichen. Beispielsweise Streckung oder Stauchung. Solche Transformationen riskieren allerdings auch Verlust von Informationen, da dadurch Pixel verloren gehen können.

14.0.9 Data Augmentation

Der Genauigkeit der Modelle steht häufig in Relation der Menge der Trainingsdaten. Je mehr unterschiedliche und qualitative Daten ein Klassifizierungsmodell zur Verfügung hat, desto besser wird es im Durchschnitt generalisieren können [23]. Diese Menge an guten Daten aufbringen zu können, kann allerdings häufig eine große Herausforderung sein.

Data Augmentation ist eine Reihe von Techniken die vorhandene Daten nimmt und sie mit bestimmten Methoden manipuliert. Diese Manipulation hat den Nutzen, dass aus vorhandenen Daten, eine Menge von “künstliche Daten” erzeugt werden. Die Idee ist es, eine Datei zu nehmen, diese in einer strukturierten Art und Weise zu verändern, sodass die Eigenschaften die ihre Klasse repräsentiert, wenig verändert wird, aber dennoch ein neues und eigenständiges Objekt darstellt. Bilder als Dateien haben eine Struktur von räumlichen Verhältnissen der Pixel zueinander und einen bestimmten Farbwert jeder einzelnen Pixel. Bilder können Features beinhalten, welche sich durch bestimmte

räumliche Verhältnisse, in Kombination der Farbwerte der Pixel darstellen. Die Data Augmentation muss sich bei Bilddateien also mit der strukturierten Manipulation der Anordnung und Farbwerte der Pixel befassen.

Value-based Image Transformation Value-based Image Transformation befasst sich mit der Manipulation der Farbwerte. “Pixel Erasing” maskiert die Farbinformationen eines Bereiches der Bilddatei, wobei “Photometric Transformation” die Farbwerte verändert, aber essentielle Informationen enthalten bleiben.

Pixel Erasing Pixel Erasing ist das maskieren von Pixel in bestimmten Bereichen. Das Maskieren kann das setzen eines bestimmten Wertes in dem Bereich sein, wie zum Beispiel das vollständige Schwarzen. Maskieren kann aber auch durch das Erzeugen eines Rausches umgesetzt werden. Diese Methode kann auf die Eingabe direkt angewendet werden, mit pseudo zufälligen Rechtecken, die den Bereich definieren auf dem die Technik angewendet werden soll. Pixel Erasing kann auch auf gezielte Bereiche angewendet werden, nachdem mehrere Informationen aus dem Bild extrahiert wurden. Ein Beispiel ist das “entfernen” von sehr unklaren Pixeln [24]. Sollte es möglich sein, vor der Klassifizierung zu erkennen, dass ein Bild über Informationen verfügt, welche nicht hilfreich, oder sogar schädlich für die Klassifizierung sein könnten, können diese Bereiche gezielt entfernt werden.

Image Cropping Image Cropping bezeichnet die Methode alle Informationen, bis auf einen ausgewählten Bereich, zu entfernen. Diese Methode kann für eine Vielzahl von Vorteile genutzt werden. Abhängig von den Trainingsdaten und der Rechenleistung die für das Training genutzt werden kann, kann das Entfernen von Information das Training beschleunigen. Die Zeit die für das Laden und Verarbeiten der Daten benötigt wird, wird geringer wenn die Menge an Information einer Datei geringer ist. Außerdem verringert es den benötigten Speicher. Ähnlich wie die Pixel Erasing Methode, kann Image Cropping genutzt werden um Informationen die vermutlich wenig hilfreich für die Klassifikation entfernt werden. Im Gegensatz zum Pixel Erasing, löscht Image Cropping die Informationen, anstatt diese zu maskieren.

Image Cropping kann allerdings auch in einer Form genutzt werden, dass ein Bild

segmentiert wird, also anstatt einen Bereich zu wählen und den Rest zu löschen, werden mehrere Bereiche gewählt und in mehrere Bilder zerlegt [25]. Beschäftigt sich das Modell mit teilweise normierten Daten, wie zum Beispiel Dokumente, kann Expertenwissen genutzt werden um die Parameter der Segmentierung festzulegen. Zum Beispiel können Dokumente in der Höhe segmentiert werden, um dem Modell die Möglichkeit zu geben sie auf die Kopfzeile zu fokussieren, ohne im gleichen Schritt den Rest des Dokumentes betrachten zu müssen. Die simpleren Formen von Image Cropping sind “random cropping”, also das zufällige Wählen der Trenninformationen, und das manuelle Selektieren dieser Informationen. “Attention-based” und “Aesthetics-based” Ansätze haben eine tiefere Komplexität.

Attention-based Ansatz Der Begriff “Attention” wird im Verlauf dieser Arbeit tiefgreifende Relevanz haben und in einem späterem Kapitel ausführlich aufgegriffen. Um einen Überblick über Attention-Based Ansätze zu verschaffen, genügt es vorerst, die Bedeutung des Wortes “Attention”, auf Deutsch “Aufmerksamkeit”, nachzuvollziehen. Die Aufmerksamkeit wird definiert als Konzentration auf ein Objekt [26]. Der Ansatz, die Idee der Aufmerksamkeit in solche Probleme zu Nutzen ist, die Fragestellung zu verfolgen, auf welche Teile eines Bildes die meiste Aufmerksamkeit gelenkt werden soll. Umgesetzt wird dies durch einen sogenannten “Attention Score”. Regionen die den wichtigsten Informationsgehalt besitzen, haben einen hohen Attention Score. Die Regionen erhalten im übertragenen Sinne also die “höchste Konzentration”. In solchen Prozessen können “Regions of interests” (ROI) durch unterschiedliche Informationen, wie Größe und Position der Region auf dem Bild, erzeugt werden. Diese ROI dienen dann als Hilfsmittel, die Parameter eines Image Croppings zu errechnen.

Der Begriff Attention wird dort auch in interaktiven Image Cropping Methoden verwendet. Der gaze-based Approach nutzt Augenbewegungen eines tatsächlichen Beobachters eines Bildes um Konzentration eines Menschen auf einem Bild. Die Idee ist, dass ein Mensch seine Augen auf Stellen richtet, die wichtig zum “Verstehen” sind und daher einen hohen Attention Score verdienen.

Aesthetics-based Ansatz In Aesthetics-based Ansätzen wird das Bild in Regionen unterteilt die einer gewissen Klasse zugehörig sein kann, welche nicht zwingend die

Klassen eines Klassifikationsproblems sind, wie zum Beispiel “Hintergrund”. Es können zum Beispiel “Vordergrund” Regionen erkannt werden und mit Image Cropping separiert werden. Ein Bild kann also Beispielsweise in mehrere Teile separiert werden und auf alle teile eine “Qualitätsklassifikation” angewendet werden [27]. Diese Klassifikation ergibt dann einen “Qualitätswert”, der repräsentiert wie “wertvoll’ dieses Segment ist. Abhängig der Werte können dann Segmente gelöscht werden, die einen geringen Wert aufweisen und dadurch die Vermutung entsteht, dass sie nicht nützlich für das Modell sind.

Change-based Ansatz Der Change-based Ansatz fokussiert sich auf die Unterschiede zwischen dem originalen Bild und dem Bild das nach einem Image Cropping entsteht. Dieser Ansatz hat das Ziel Image Cropping als Prozess in einem Modell zu optimieren source <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=6618974> . In dem Change-based Ansatz wird ein automatischer Cropping Prozess kontinuierlich optimiert. Es wird die Qualität des Bildes vor und nach dem Cropping verglichen und diese Ergebnisse werden genutzt um die Parameter der Image Cropping Prozesses anzupassen.

Geometric Transformation Während Image Cropping sich damit befasst hat Bilder zu segmentieren und Bereiche zu löschen, befasst sich die Geometric Transformation mit dem Manipulieren der räumlichen Verhältnisse zwischen den Pixeln, während der Wert der Pixel identisch bleibt [23]. Durch solche Transformationen können “neue” Daten aus bereits existierenden Daten erzeugt werden. Für ein trainierendes Modell kann ein gespiegeltes Bild als eigenständiges, neues Bild im Vergleich des nicht gespiegelten originalen Bild gesehen werden.

Affine Transformationen Transformationen wie “Translation”, “Rotation”, “Shearing” und “Flipping” manipulieren das Bild, während die Verhältnisse von Distanzen der Pixel erhalten bleiben. Translation bewegt das gesamte Bild in eine Richtung. Ist wird also die selbe Manipulation der X und Y Positionen auf jedes Pixel gleich angewandt. Solche Manipulationen können hilfreich sein, dem Modell keine räumliche Relation zu gewissen Features anlernen zu lassen. Ein Beispiel könnte sein, dass wenn Bilder von Autos nach Marken Klassifiziert werden müssen und die Bilder die BMWs enthalten, das Auto auf der rechten Seite des Bildes haben, es nicht anlernt, dass wenn ein Auto

auf der rechten Seite steht, es ein BMW ist. Eine Rotation besitzt einen festen Punkt und dreht das Bild um einen bestimmten Winkel. Image Shearing ist das “Schieben” eines Bildes parallel zu einer bestimmten Linie. Je weiter entfernt die Parallele von der festen Linie ist, desto weiter wird diese in die zu schiebende Richtung bewegt [28]. Image Flipping ist die Spiegelung eines Bilder der X-Achse, Y-Achse oder beider Achsen.

Non-affine Transformation Die Non-affine Transformation manipuliert die räumlichen Verhältnisse zwischen den Pixeln zueinander mehr als bei der affine Transformation. Zum Beispiel kann durch perspective Transformation aus existierenden, bereits mit Label versehenden Bildern, neue Bilder mit einer neuen Perspektive erzeugt werden. Damit ist gemeint, dass das gleiche Bild zu transformiert wird, als ob es von einem anderen Winkel fotografiert wurde source <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=8943416>. Ähnlich der Qualitätsklassifikation des Aesthetics-based Ansatzes, kann außerdem “Image stretching” genutzt werden. Auch beim Image stretching wird eine Bewertung für Relevanz ausgeführt, es werden irrelevante Teile allerdings nicht entfernt, wie beim Image Cropping, aber verkleinert und relevante Teile vergrößert. Diese Anpassungen der Größen können durch Stauchungen und Streckungen entstehen. “PatchShuffle” ist ein Ansatz bei dem Bilder oder Feature Maps in viele kleine “mini-Batches” aufgeteilt werden. Innerhalb einiger dieser mini-Batches werden die Pixel zufälliger miteinander getauscht source <https://arxiv.org/pdf/1707.07103.pdf>.

Multiple Image Augmentation Diese Augmentation errechnet ein Summe aus zwei zufällig erwählten Bildern. Der Farbwert jeder Pixel wird aus beiden Bildern summiert und erzeugt ein neues Bild. Es gibt mehrere unterschiedliche Ausführungen dieser Operationen. AugMix wendet beispielsweise drei Augmentationen auf ein Bild an und kombiniert diese dann um ein neues Bild zu erzeugen

Data Augmentation für Text Daten Da es Dokumentenbilderklassifizierungsverfahren gibt die außerdem den Inhalt der Wörter für die Klassifizierung nutzt stellt sich analog zu der Data Augmentation für Bilder auch die Frage ob man ähnliche Augmentationen auch mit dem Text durchführen kann

Value-based Text Transformation In dieser Form von Transformation werden einzelne Worte manipuliert. Bei der “Invariant Replacement” Transformation werden Worte zufällig ausgewählt und es wird ein Wort gesucht, welches eine ähnliche Bedeutung besitzt, aber ein anderes Wort als das originale Wort ist. Dieses Wort wird dann dadurch ersetzt.

Bei dem “Token Addition” Verfahren werden beispielsweise “Junk Tokens” generiert, welche keine eigene Bedeutung repräsentieren und nur als Rauschen dienen.

Structure-based Text Transformation

14.1 Encoder

Die veröffentlichte Architektur hat einen Stapel von identischen Schichten. Jede Schicht hat zwei Unterschichten. Die erste Unterschicht ist ein sogenannter “Multi-head self-attention mechanism” und die zweite ein einfaches “fully connected feed-forward network”. Jede Unterschicht verfügt über eine “Skip connection” die an der Multi-Head Attention, oder dem Feed Forward Network vorbei in eine Layer Normalization endet. Zusammengefasst ergibt sich der Ausgang jeder Unterschicht aus folgendem Ausdruck

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

wo x die Eingabe und $\text{Sublayer}(x)$ die Funktion der Unterschicht ist.

14.2 Decoder

Der Decoder besteht teilweise aus den gleichen Elementen wie der Encoder. Zusätzlich besitzt der Decoder einen Multi-Head Attention Layer, der als Eingabe die Ausgabe des Encoders nimmt. Die Self-Attention Unterschicht des Decoders wird maskiert, um sicherzustellen, dass die Zuweisungen der kontinuierlichen Repräsentationen, also die “Vorhersagen” nur auf den Eingaben der vorherigen Iterationen basieren. Daher wird durch diese Maskierung ein Offset von einer Iteration erzeugt.

14.2.1 Scaled Dot-Product Attention

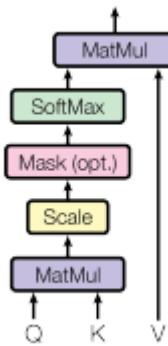


Abbildung 26: Scaled Dot-Product Attention

Quelle: <https://arxiv.org/pdf/1706.03762>

Ein Beispiel einer einzelne Attention function wird “Scaled Dot-Product Attention” genannt. Als Eingänge werden queries und keys genutzt die eine Dimension dk haben. Die values werden in einer Dimension von dv als dritten Eingang genutzt. Es wird das Skalarprodukt der query mit allen keys errechnet und durch die Wurzel der Dimension dk geteilt. Auf dem Ergebnis wird dann eine Softmax Funktion angewendet. Die Softmax Funktion ergibt dann das Gewicht was für die Values genutzt werden. Die Attention Funktion wird folgendermaßen formuliert:

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

Q, K und V stellen Matritzen von Queries, Keys und Values dar. Verglichen zur Scaled Dot-Product Attention Funktion existiert die Dot-Product Attention Funktion, die mit der Ausnahme der fehlenden Skalierung 1/wurzel dk identisch ist. Die sogenannte Aditive Attention Funktion nutzt ein Feed-Forward Network mit einem Hidden Layer.

14.2.2 Multi-Head Attention

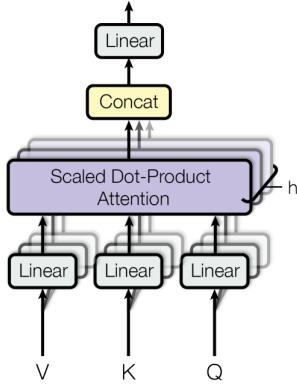


Abbildung 27: Multi-Head Attention

Quelle: <https://arxiv.org/pdf/1706.03762>

Die Attention Funktion kann bei einer Multi-Head Attention auch parallel auf unterschiedlichen Schichten angewendet werden. Die Eingaben werden mit der Attention Funktion auf jeder der unterschiedlichen Schichten verrechnet, wobei jede Schicht unterschiedlich lernen kann. Außerdem wird mit jede query, key und value Matrix ein Skalarprodukt mit Gewichtematrizen erzeugt. Alle Ausgänge der Schichten werden dann zusammengerfügt und abschließend als Skalarprodukt mit einer Gewichtematrix W hoch o dargestellt.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, QW_i^K, QW_i^V)$$

14.3 Template basierende Methoden

Bei Template basierende Methoden werden Vorlagen genutzt um die Klasse der Bilder zu ermitteln. Diese Vorlagen haben den Nutzen die Struktur der Klassen zu repräsentieren. Ein Ansatz ist die Ermittlung der Linien in einem Dokument source <https://dl.acm.org/doi/pdf/10.1145/335603.335611>. Linien sind hilfreiche Komponenten um Regionen in Dokumenten zu separieren und die Zerteilung von Dokumenten in strukturellen Regionen ist die Grundidee der strukturbasierenden Methoden. Es werden Komponenten identifiziert und logisch gruppiert. Demnach werden unterschiedliche Komponente in eine Gruppe platziert die sich in strukturell geschlossen Regionen

befinden. Das könnte zum Beispiel bedeuten, dass diese Komponenten von einer durchgehenden Linie umrandet sind. Diese Regionen werden dann gruppiert in bestimmten Datenmodellen repräsentiert und mit den Vorlagen verglichen.

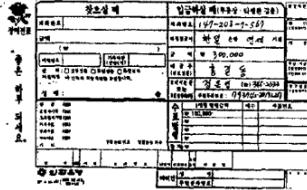


Abbildung 28: Überblick der Ansätze

Quelle: <https://dl.acm.org/doi/pdf/10.1145/335603.335611>

Demnach können viele Information mit der einzigen Nutzung von erkannten Linien extrahiert werden. Was den Vorteil hat, dass viele Informationen mit wenig Rechenleistung ermittelt werden. Der Nachteil an diesem Ansatz ist, dass Klassen existieren müssen, für die es Vorlagen gibt die diese sehr gut repräsentieren. Wird also eine Klassifikation von Rechnungen vorausgesetzt, könnte die Struktur von Händler zu Händler unterschiedlich sein. Das macht diesen Ansatz sinnvoll wenn die Struktur fest vorgeschrieben ist, aber undynamisch, wenn die Strukturen innerhalb einer Klasse Varianz aufweisen.

14.4 Graph basierende Methoden

Bei den Graph basierenden Methoden werden strukturelle Komponenten, ähnlich wie bei den Template basierende Methoden, identifiziert und in ein Datenmodell repräsentiert, das Datenmodell wird in diesem Fall als Graph umgesetzt. In diesem Graph stellen die Knoten die Art der Komponente dar. Die Art könnte eine Überschrift, ein Bild oder ähnliches sein. Die Konten zwischen den Knoten beschreiben die räumlichen Verhältnisse zwischen den Knoten dar source <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=1047980>.

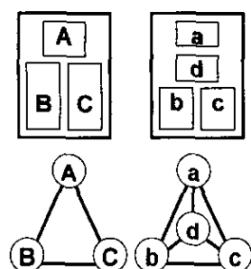


Abbildung 29: Überblick der Ansätze

Quelle: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=1047980>

Graphen haben den Vorteil, dass sie sehr stark die Struktur der Dokumente repräsentieren können source Document image classification: Progress over two decades. Allerdings ist die Prämisse, dass die Segmentierung des Dokuments gut umgesetzt wurde, damit der Graph das Dokument gut repräsentiert. In der Realität ist die Segmentierung noch fehleranfällig, daher leidet auch die Aussagekraft des Graphs darunter. Der Vergleich von Graphen untereinander, der auch “Graph matching” genannt wird, ist gewöhnlich rechnerisch aufwendig. Daher gibt es Ansätze die Graphen in Vektoren repräsentieren source <https://www.sciencedirect.com/science/article/abs/pii/S003132031000542X>.

14.5 Handgemachte feature basierenden Methoden

Es gibt mehrere unterschiedliche Ansätze der Handgemachte feature basierenden Methoden. Eines dieser Ansätze ist die Nutzung von Local Descriptors, die auch untereinander unterschiedliche Variationen haben können. Diese Local Descriptors werden generell für sogenannte “Keypoint detection” und “Keypoint description” genutzt. Keypoints sind in diesem Kontext Elemente in einem Bild, das gut für eine Klassifikation genutzt werden. Dieser Ansatz hat Erfolg in Klassifikation für natürliche Bilder gefunden und “objekt tracking” source Document image classification: Progress over two decades. Es gibt Ansätze die Local Descriptors nutzen um Klassifikation für Dokumente zu betreiben. Der Ansatz von Viet Phuong et al. nutzt diesen Ansatz primär für die Nutzung der Logos, die auf Dokumenten gefunden werden, für die Klassifikation source <https://projet.liris.cnrs.fr/imagine/pub/proceedings/ICPR-2012/media/files/2535.pdf>.

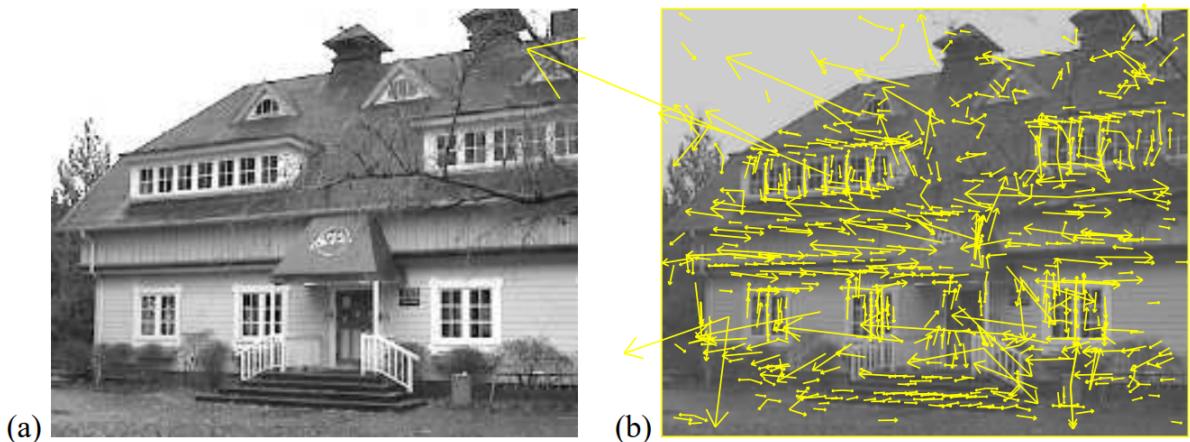


Abbildung 30: Überblick der Ansätze

Quelle: <https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf>

14.6 Deep feature basierende Methode

Bei Deep feature basierenden Methoden ist die Grundidee, dass durch ein Modell die Eigenschaften der Dokumente eigenständig und automatisch erkannt werden. Diese Eigenschaften werden dann genutzt um Dokumente in den gleichen Klassen zu klassifizieren. Der Begriff “deep” in Deep feature kommt von der Tiefe der Modelle, also die Anzahl der Schichten die genutzt werden. Es gibt bis zu diesem Zeitpunkt viele unterschiedliche Ansätze zum Thema Deep learning. CNNs haben allerdings eine besondere Bedeutung im Kontext der Bildklassifizierung. Faltungen haben sich sehr passend ergeben, da dadurch die Dimension der Eingabedaten stark reduziert werden und wichtige Informationen wie beispielsweise Kanten oder ähnliches erhalten bleiben. Durch die Reduzierung der Dimension lässt sich letztendlich ein fully connected Layer für die Klassifikation betreiben, welche eine angemessene Menge an Neuronen besitzt. 2012 haben CNNs deutlich mehr Aufmerksamkeit erhalten als AlexNet die Visual Recognition Challenge 2012 gewonnen hat source <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Ein großer Vorteil dieser Ansätze ist, dass nicht zwingend Expertenwissen über die Dokumente erforderlich ist. Es muss also nicht zwingend bekannt sein was Klassen voneinander unterscheiden, da das Training von Deep feature basierenden Methoden diese Arbeit übernimmt. Expertenwissen kann hilfreich sein, aber nicht notwendig.

14.7 Text und visuell basierende Methoden

Die Idee der hybriden Methoden ist das Nutzen von den Vorteilen mehrerer Ansätze. Zum Zeitpunkt dieser Arbeit sind Text und visuell basierende Methoden noch Teil der Modelle mit einer vergleichsweise guten Genauigkeit. Souhail Bakkali et al. haben einen Ansatz präsentiert, der zwei Kanäle für die Klassifikation von Dokumenten nutzt.

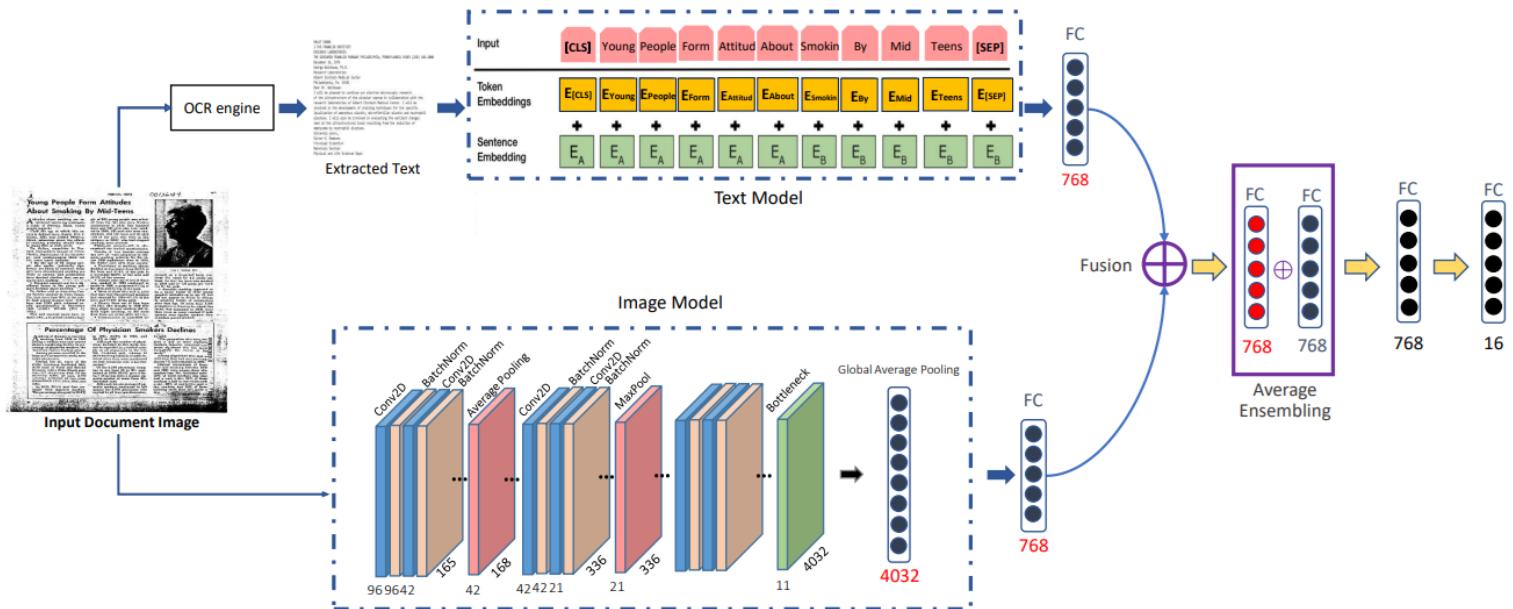


Figure 2. The proposed cross-modal deep network

Abbildung 31: Text und visuell basierendes Modell

Quelle: <https://openaccess.thecvf.com/content>

source <https://openaccess.thecvf.com/contentCVPRW2020/papers/w34> Ein Kanal verarbeitet Text, der über eine OCR Engine extrahiert wurde und der andere Kanal verwendet ein Convolutional Neural Network. Abgesehen von der Verwendung bereits existierender Architekturen entsteht eine neue Herausforderung, wie die Resultate der beiden Ansätze „Fusioniert“ werden. Das Textmodell und das Bildmodell übermitteln wertvolle Informationen am Ausgang und es gibt mehrere Möglichkeiten wie diese Informationen genutzt werden um eine konkrete Aussage am Ende des gesamten Modells zu treffen.

Ritesh Sarkhel et al. beschreiben einen Ansatz für die Klassifizierung heterogene visuell reiche Dokumente source <https://www.ijcai.org/proceedings/2019/0466.pdf>. Dieser Ansatz nutzt ein sogenanntes „spatial pyramid model“, was für eine Extrahierung

von unterschiedbaren “feature descriptors” genutzt wird. Dieses Model nutzt eine Hierarchie, die gewöhnlich in Dokumenten genutzt wird. Mit Hierarchie ist im Kontext von Dokumenten beispielsweise gemeint, dass eine Zeile Teil eines Absatzes ist und ein Wort Teil einer Zeile ist. Die Hierarchie besteht in dieser Abstrahierung aus 5 Teilen: Die Seite, Spalten, Absätze, Zeilen und Worte.

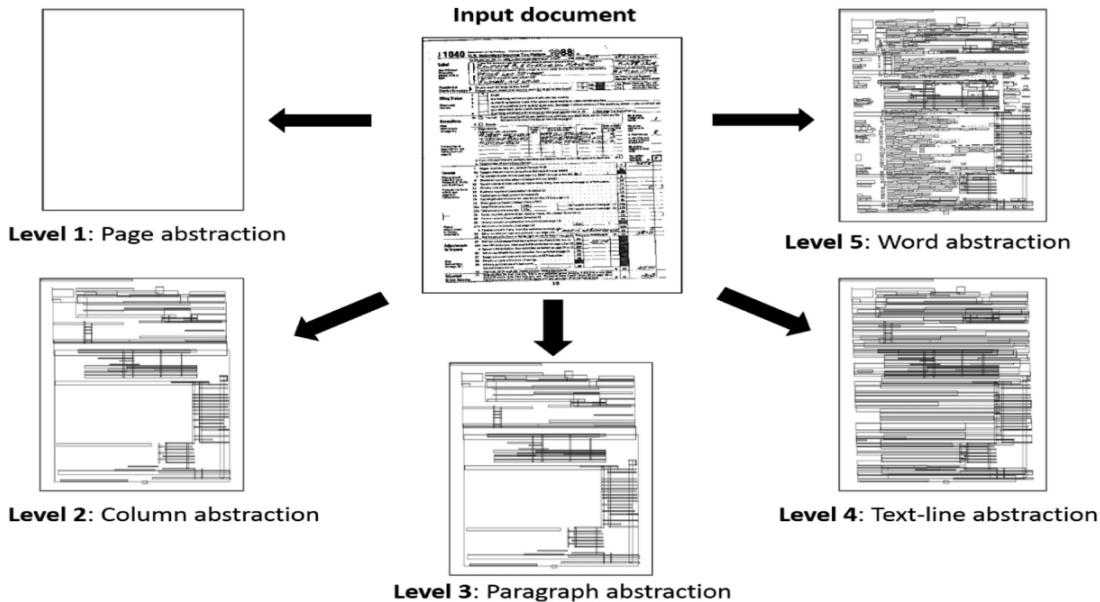


Abbildung 32: Überblick der Ansätze

Quelle: <https://www.ijcai.org/proceedings/2019/0466.pdf>

Dieses hierarchisches Model wird bei einem Dokument D durch Tupel (A_{indexD} , T_{indexD}) wo $AD = \text{Vereinigung von } a_{indexi}$ als Satz von atomaren Element a_{indexi} und T_{indexD} als Repräsentation von der visuellen Organisation dargestellt wird. T_{indexD} stellt also die Baumstruktur der graphischen Elemente dar. Der Wurzelknoten des Baums stellt das gesamte Dokument und die Unterknoten die strukturellen Komponenten. Die “Blätter” der Baumstruktur repräsentieren die atomaren Elemente des Dokumentes, also beispielsweise die einzelnen Wörter. Diese Knoten werden dann durch ein Tupel $n_{indexi} = (x_{indexi}, y_{indexi}, h_{indexi}, w_{indexi}, t_{indexi})$ dargestellt, wo x_{indexi} und y_{indexi} die Koordinaten der Kante Oben-Links, h_{indexi} und w_{indexi} die Höhe und Breite und t_{indexi} der Text der Komponente ist.

Die Autoren nutzen ein sogenanntes “Koordinationsnetzwerk”, das sie “LadderNet” nennen. Mittels dieses Netzwerkes identifiziert das Model die “optimale Pyramidenstufe”. Es werden alle feature descriptors ausgewählt die mit der optimalen Pyramidenstufe

korrespondieren und mit denen wird über ein “Multi-column convolutional network” die Klassifizierung betrieben.

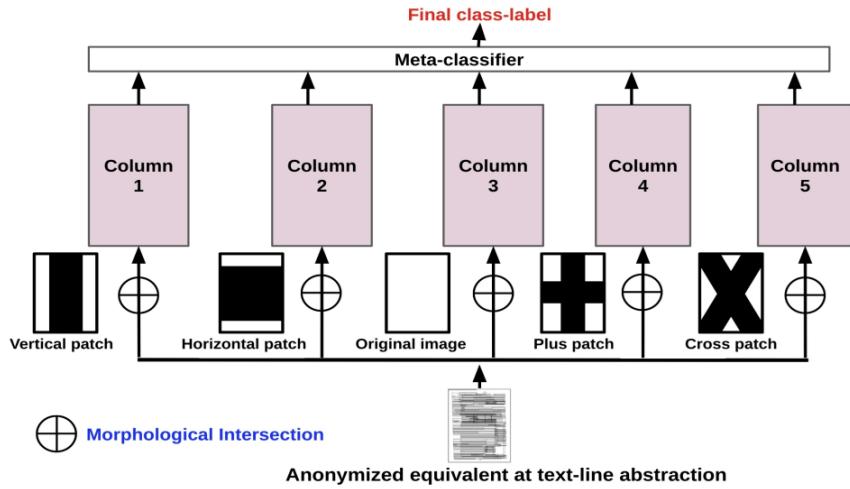


Abbildung 33: Überblick der Ansätze

Quelle: <https://www.ijcai.org/proceedings/2019/0466.pdf>

Abbildung ????? stellt dar wie diese Ansätze verwendet werden um das finale Klassenlabel zu finden. Das Dokument wird durch das LadderNet geschickt um die optimale stufe zu ermitteln und die damit entsprechende Abstrahierung des Dokumentes wird für die Klassifizierung genutzt.

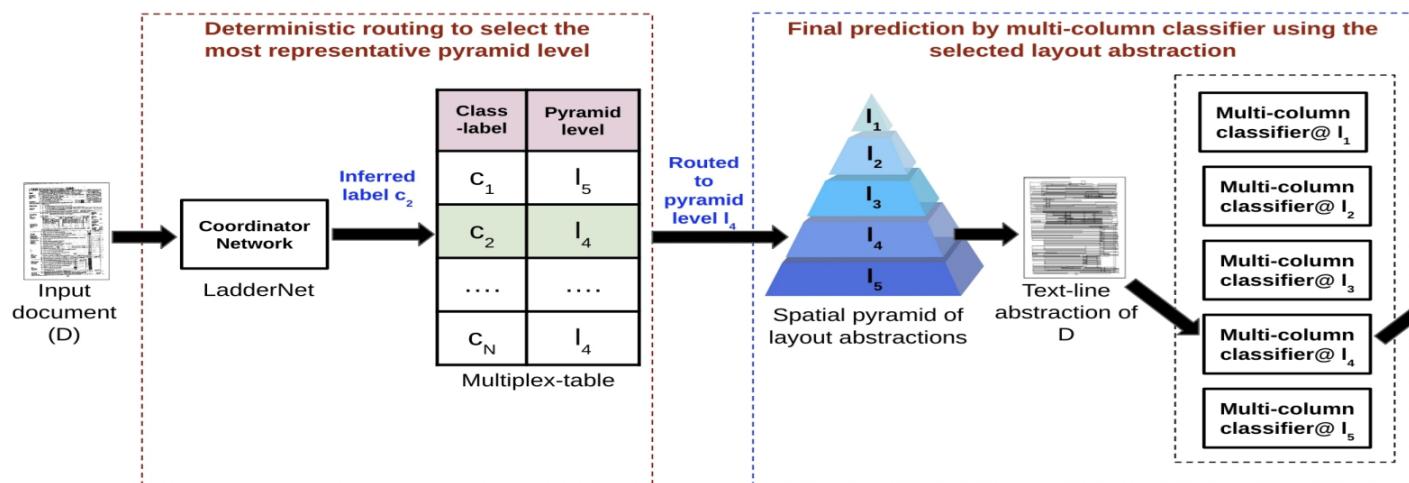


Abbildung 34: Überblick der Ansätze

Quelle: <https://www.ijcai.org/proceedings/2019/0466.pdf>

14.8 Text und strukturbasierende Methoden

Bei Text und strukturbasierende Methoden werden Texte und die Anordnung von strukturellen Komponenten zur Klassifizierung genutzt. Der Unterschied zu Text und visuell basierende Methoden ist, dass bei der Struktur lediglich relevant ist wo eine Komponente liegt und nicht wie diese aussieht. Pixelinformationen werden bei einer Struktur relevant um eine Komponente zu identifizieren, diese Informationen werden aber nicht für die Klassifikation selbst genutzt. Einer der wenigen bekannten Ansätze ist LayoutLM. LayoutLM nutzt das Sprachmodell Bert und erweitert das mit zwei Embeddings. Ein zweidimensionales Position Embedding und ein Bild Embedding. Da beide Formen der Daten sich informativ gut ergänzen haben multimodale Ansätze eine gute Auswirkung auf die Genauigkeit, hat allerdings negative Auswirkungen auf die benötigte Rechenleistung.

14.9 Erwartete Kosten

In der “Entscheidungstheorie” wird der Begriff “Erwartete Kosten”(EC) aufgegriffen source <https://arxiv.org/pdf/2209.05355.pdf>. Die erwarteten Kosten sind eine zu minimierende Funktion, die die erzeugten Kosten einer Fehlentscheidung repräsentieren. Der Begriff “Kosten” ist in dem Kontext ein Platzhalter für Aufwand in einer Beliebigen Form. Ein Beispiel ist, dass ein falsch negatives Ergebnis eines tatsächlichen Tumors eines Bildes im medizinischen Kontext eine schwerere Auswirkung hätte als ein falsch positives Ergebnis.

Die Klassen in welche die Objekte klassifiziert werden sollen und der Zweck haben also einen großen Einfluss auf die erwarteten Kosten. Da es sich in dieser Arbeit um die Klassifizierung von Dokumenten handelt, müsste der Anwendungsfall ein sehr sensibler sein, damit die erwarteten Kosten einen beachtlichen Wert erreichen.

14.10 Verwechslungsmatrix

Metriken wie Accuracy und F1 Score sind Werte, die auf einem schnellen Blick eine Einschätzung über die Qualität der Lösung bieten, allerdings Informationen über “Schwächen” in der Klassifikation nicht darstellen. Beispielsweise könnte es relevant sein zu sehen welche Klassen sich schwierig korrekt klassifizieren lassen. Eine Verwechslungs-

matrix (CM) ist eine nützliche Darstellung um zu repräsentieren, welche Daten mit welchen Klassen „verwechselt“ werden. In dieser zweidimensionalen Matrix werden auf einer Achse die korrekten Label dargestellt und auf der anderen Achse die vorhergesagten Label. Somit stellt eine CM eines gut generalisierendes Model eine Diagonale dar, dessen Farbschema einen hohen Wert repräsentiert, da die Diagonale die TP darstellt.

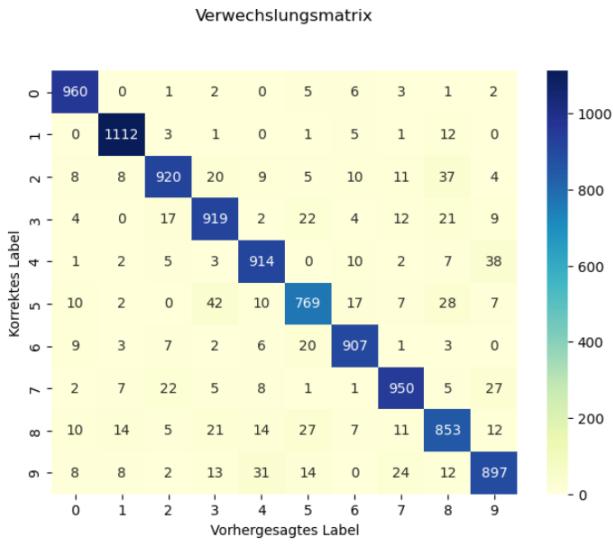


Abbildung 35: Überblick der Ansätze

Quelle: <https://michaelkipp.de/deeplearning/Evaluation.html>

14.11 Conda

Conda ist ein Open Source Package Management System, was zusätzlich als Environment Management System genutzt werden kann. Conda wurde ursprünglich für Python entwickelt. Abgesehen von dem Vorteil, Notebooks nutzen zu können ist Conda für diese Arbeit sehr hilfreich, da unterschiedliche Umgebungen, unterschiedliche Pakete laden können.

Diese Arbeit beschäftigt sich mit aktuellen Lösungen zur Document Image Klassifizierung, aber auch mit etwas älteren. Dementsprechend kommt aus auch dazu, dass Implementationen unterschiedlicher Modelle, auch unterschiedliche Versionen von Paketen nutzen. Um zu vermeiden, dass je nach Implementation bestimmte Pakete up- oder downgraded werden müssen, können Conda Umgebungen konfiguriert und frei gewechselt werden.

14.12 CUDA

Um Trainingsprozesse für maschinelles Lernen durchzuführen bedarf es eine sehr große Menge an Berechnungen. Grund dafür liegt in der Natur der Architekturen. Beispielsweise besteht ein Neuronales Netzwerk aus vielen Neuronen die korrigiert werden müssen, CNNs aus vielen Pixelwerten die berechnet werden müssen, oder Transformatoren aus vielen embeddings die berechnet werden müssen. Um diese Aufgabe zu bewältigen bietet es sich an eine Grafikkarte zu verwenden, da diese dazu in der Lage sind viele Berechnungen parallel umzusetzen. Grafikkarten werden allerdings gewöhnlicherweise von Grafiktreibern gesteuert, was es uns nicht einfach erlaubt eigenen Code darauf laufen zu lassen wie beispielsweise Python kompilierter Maschinencode auf einer CPU. Daher hat NVIDIA eine Berechnungsplattform namens CUDA entwickelt source <https://developer.nvidia.com/cuda-zone>. CUDA unterstützt Sprachen wie C, C++ und Python, was es einfach ermöglicht GPU gesteuerten Programmcode für maschinelles Lernen zu entwickeln. Dafür nutzt die praktische Umsetzung dieser Arbeit das CUDA Toolkit, was sogenannte “GPU-accelerated libraries” nutzt, sowie einen Compiler, Entwicklungs Tools, und eine CUDA Laufzeit. Bei der Wahl der Toolkits ist es allerdings wichtig genau die eigene Hardware zu kennen, da die Toolkits genau auf diese abgestimmt sein müssen.

15 Literaturverzeichnis

Literatur

- [1] R. T. Kneusel, *Practical deep learning: A Python-based introduction*, first edition ed. San Francisco: No Starch Press, 2021.
- [2] O. Stein, *Grundzüge der Globalen Optimierung*, 2nd ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021. [Online]. Available: <http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1867093>
- [3] N. Sudermann-Merx, *Einführung in Optimierungsmodelle: Mit Beispielen und Real-World-Anwendungen in Python*, 1st ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2023. [Online]. Available: <https://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-3127730>
- [4] Lamarr Institute for Machine Learning and Artificial Intelligence, “Klassifikation im maschinellen lernen » lamarr-blog,” 2022. [Online]. Available: <https://lamarr-institute.org/de/blog/ml-klassifikation/>
- [5] C. M. Bishop, *Pattern recognition and machine learning*, 13th ed., ser. Information science and statistics. New York: Springer, 2009. [Online]. Available: <https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>
- [6] M. Goyal, T. Knackstedt, S. Yan, and S. Hassanpour, “Artificial intelligence-based image classification methods for diagnosis of skin cancer: Challenges and opportunities,” *Computers in Biology and Medicine*, vol. 127, p. 104065, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482520303966>
- [7] Jamshed Memon, Maira Sami, Rizwan Ahmed Khan, “Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr),” 20.09.2024.
- [8] S. Mori, C. Y. Suen, and K. Yamamoto, “Historical review of ocr research and development,” *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1029–1058, 1992.

- [9] Prof. Dr. Günter W. Maier, “Definition: Lernen,” *Springer Fachmedien Wiesbaden GmbH*, 14.02.2018. [Online]. Available: <https://wirtschaftslexikon.gabler.de/definition/lernen-41169>
- [10] S. Ackerman, Ed., *Discovering the Brain*. National Academies Press (US), 1992.
- [11] Dr. Alan Woodruff, “What is a neuron?” 2016. [Online]. Available: <https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron>
- [12] “Das gehirn,” 20.09.2024. [Online]. Available: <https://www.mpg.de/gehirn>
- [13] S. Mallick, “Activation functions in deep learning – a complete overview,” *Satya Mallick*, 30.10.2017. [Online]. Available: <https://learnopencv.com/understanding-activation-functions-in-deep-learning/>
- [14] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation functions in deep learning: A comprehensive survey and benchmark.” [Online]. Available: <https://arxiv.org/pdf/2109.14545>
- [15] G. Philipp, D. Song, and J. G. Carbonell, “The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions.” [Online]. Available: <https://arxiv.org/pdf/1712.05577>
- [16] A. Ram and C. C. Reyes-Aldasoro, “The relationship between fully connected layers and number of classes for the analysis of retinal images.” [Online]. Available: <https://arxiv.org/pdf/2004.03624>
- [17] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series classification: a review,” *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019. [Online]. Available: <https://arxiv.org/pdf/1809.04356>
- [18] Guoping Xu, Xiaxia Wang, Xinglong Wu, Xuesong Leng, Yongchao Xu, “Development of skip connection in deep neural networks for computer vision and medical image analysis: A survey.” [Online]. Available: <https://arxiv.org/html/2405.01725v1>

- [19] C. W. Schmidt, V. Reddy, H. Zhang, A. Alameddine, O. Uzan, Y. Pinter, and C. Tanner, “Tokenization is more than compression.” [Online]. Available: [https://arxiv.org/pdf/2402.18376](https://arxiv.org/pdf/2402.18376.pdf)
- [20] H. van Halteren, *Syntactic Wordclass Tagging*, ser. Text, Speech and Language Technology Ser. Dordrecht: Springer Netherlands, 1999, vol. v.9. [Online]. Available: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6824924>
- [21] Andre Vratislavsky, “Geometrische transformationen,” 20.09.2024. [Online]. Available: <https://userpage.fu-berlin.de/~vratisla/Bildverarbeitung/GeoTrans/GeoTrans.html>
- [22] study.com, “Isometry in geometry | definition, types & dilation | study.com,” 20.09.2024. [Online]. Available: <https://study.com/learn/lesson/isometry-geometry-overview-types.html>
- [23] Z. Wang, P. Wang, K. Liu, P. Wang, Y. Fu, C.-T. Lu, C. C. Aggarwal, J. Pei, and Y. Zhou, “A comprehensive survey on data augmentation.” [Online]. Available: [https://arxiv.org/pdf/2405.09591](https://arxiv.org/pdf/2405.09591.pdf)
- [24] Yan Fang, Feng Zhu, Bowen Cheng, Luoqi Liu, Yao Zhao, and Yunchao Wei, “Locating noise is halfway denoising for semi-supervised segmentation.” [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/papers/Fang_Locating_Noise_is_Halfway_Denoising_for_Semi-Supervised_Segmentation_ICCV_2023_paper.pdf
- [25] Y. Xu, Y. Xu, T. Lv, L. Cui, F. Wei, G. Wang, Y. Lu, D. Florencio, C. Zhang, W. Che, M. Zhang, and L. Zhou, “Layoutlmv2: Multi-modal pre-training for visually-rich document understanding.” [Online]. Available: [https://arxiv.org/pdf/2012.14740v4](https://arxiv.org/pdf/2012.14740v4.pdf)
- [26] “Aufmerksamkeit.” [Online]. Available: <https://www.dwds.de/wb/Aufmerksamkeit>
- [27] M. Nishiyama, T. Okabe, Y. Sato, and I. Sato, “Sensation-based photo cropping,” in *MM’09*, W. Gao, Y. Rui, A. Hanjalic, C. Xu, E. Steinbach, A. El Saddik, and M. Zhou, Eds. New York, NY: ACM Press, 2009, pp. 669–672.

- [28] “Computer graphics shearing,” 20.09.2024. [Online]. Available: <https://www.javatpoint.com/computer-graphics-shearing>
- [29] K. O’Shea and R. Nash, “An introduction to convolutional neural networks.” [Online]. Available: <https://arxiv.org/pdf/1511.08458>
- [30] “Stride (machine learning),” *DeepAI*, 17.05.2019. [Online]. Available: <https://deepai.org/machine-learning-glossary-and-terms/stride>

16 Eigenständigkeitserklärung

“Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Literatur und Hilfsmittel angefertigt habe. Wörtlich übernommene Sätze und Satzteile sind als Zitate belegt, andere Anlehnungen hinsichtlich Aussage und Umfang unter Quellenangabe kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keine Prüfungsbehörde vorgelegen und ist auch noch nicht veröffentlicht.”

Ort, Datum

Unterschrift