HAW HAMBURG

Bachelorarbeit

Tim Pries

Explainable Artificial Intelligence in der Dokumentenklassifikation

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Tim Pries

Explainable Artificial Intelligence in der Dokumentenklassifikation

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 18.09.2024

Tim Pries

Thema der Arbeit

Explainable Artificial Intelligence in der Dokumentenklassifikation

Stichworte

Explainable Artificial Intelligence, Dokumentenklassifikation, Convolutional Neuronal Networks

Kurzzusammenfassung

Künstliche Intelligenz erlangt immer größere Wichtigkeit in verschiedensten Bereichen unseres Lebens. Gerade im Bereich der Texterkennung haben sich in den vergangenen Jahren zahlreiche neue Entwicklungen ergeben. Durch Künstliche Neuronale Netze konnten große Fortschritte erzielt werden. Ziel dieser Arbeit sollte es sein, die CNN Modellen InceptionV3 und VGG 16 mit dem DocILE-Datensatz zu trainieren und mit Hilfe von Explainable AI (XAI) die Entscheidungsfindung der Dokumentenklassifikation näher zu verstehen. Hierfür wurden die verschiedene XAI-Methoden (Grad-CAM, Integrated Gradient sowie die Layerwise Relevante Propagation (LRP) genutzt. Für beide Modelle konnte eine gute Klassifikation der Dokumente insbesondere für solche verzeichnet werden, die häufig im Datensatz vorkamen. Für im Datensatz unterrepräsentiere Dokumentenklassen konnten schlechtere Ergebnisse verzeichnet werden. Dies spiegelte sich ebenfalls in den XAI-Untersuchungen wider. Somit kann für diese Arbeit konkludiert werden, dass die Aussagekraft eines Modells vor allem für die Aspekte als sinnvoll betrachtet werden kann, mit dem das Modell viel trainiert wird. Um also eine sinnvolle Dokumentenklassifikation zu erhalten, sollte ein Trainingsdatensatz ausreichend balanciert sein. Inhalt weiterer Forschung könnte sein, wie auch mit nur begrenzt vorliegenden Trainingsdatensätzen bessere Auswahlkriterien erstellt werden können, um Modelle mit weniger Trainingsdaten effektiver trainieren zu können.

Tim Pries

Title of Thesis

Explainable Artificial Intelligence in document classification

Keywords

Explainable Artificial Intelligence, Dokumentclassification, Convolutional Neuronal Networks

Abstract

Artificial intelligence is becoming increasingly important in various areas of our lives. Numerous new developments have emerged in recent years, particularly in the field of text recognition. Artificial neural networks have enabled great progress to be made. The aim of this work was to train the CNN models InceptionV3 and VGG 16 with the DocILE dataset and to better understand the decision-making process of document classification with the help of Explainable AI (XAI). The various XAI methods (Grad-CAM, Integrated Gradient and Layerwise Relevant Propagation (LRP)) were used for this purpose. For both models, a good classification of the documents was recorded, especially for those that occurred frequently in the data set. Poorer results were recorded for document classes that were underrepresented in the data set. This was also reflected in the XAI studies. It can therefore be concluded for this work that the informative value of a model can be considered meaningful primarily for those aspects with which the model is trained a lot. In order to obtain a meaningful document classification, a training data set should be sufficiently balanced. Further research could focus on how better selection criteria can be created even with limited training data sets in order to train models more effectively with less training data.)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Künstliche Intelligenz (KI) ist eine der bedeutendsten technischen Entwicklungen der letzten Jahrzehnte und beeinflusst zahllose Aspekte des modernen Lebens. Unter dem Begriff künstliche Intelligenz sind Systeme und Maschinen zusammengefasst, die menschliches Handeln nachahmen können. Dies geschieht durch die Übernahme von Fähigkeiten, die bisher menschlicher Intelligenz zugeordnet wurden. Hier seien das Erkennen von Mustern, Lösungen von Problemen, Erkennung von Sprache und insbesondere das Treffen von Entscheidungen genannt. Es wurde bereits vor Jahrzehnten an künstlicher Intelligenz geforscht, jedoch haben erst die gesteigerten Rechenleistungen modernen Computer der letzten Jahre einen großen Fortschritt im Bereich der künstlichen Intelligenz möglich gemacht. Anwendungsmöglichkeiten von künstlicher Intelligenz sind mannigfaltig. Bereits jetzt wird KI genutzt, um in der Industrie Produktionsprozesse zu beschleunigen und zu optimieren, wir begegnen künstlicher Intelligenz im Alltag in Form von Chatbots oder personalisierten Shoppingempfehlungen. Was den verschiedenen KI-Modellen gemein ist, ist, dass sie durch verschiedenste Algorithmen versuchen, Muster zu erkennen und zu deuten und auf dieser Basis möglichst passende Antworten für zukünftige Abfragen liefern, sodass eine Entscheidungsfindung resultiert. Um die Funktionsweise innerhalb der künstlichen Intelligenz besser verstehen zu können und etwaige Schwächen oder Stärken in der Nutzung mit bestimmten Zielsetzungen zu erlangen, wurden weitere Modelle entwickelt, die die zum Teil nicht greifbaren Strukturen innerhalb der KI-Entscheidungsfindung erklärbar zu machen. Diese Möglichkeiten werden als Explainable Artificial Intelligence bezeichnen (XAI). Ziel der hier durchgeführten Arbeit sollte die Untersuchung verschiedener KI-Modelle in Bezug auf die Klassifikation von Geschäftsdokumenten sein, sowie die weitere Untersuchung hinsichtlich der auch durch XAI-Modelle analysierten möglichen Schwächen der Modelle. Hierzu wurde ein großer Datensatz von Dokumenten genutzt und mit den genannten Modellen analysiert.

1.1 Stand der Technik

In den letzten Jahren hat die automatische Klassifizierung von Geschäftsdokumenten mithilfe von neuronalen Netzen erheblich an Bedeutung gewonnen. Die Anwendung von Convolutional Neural Networks (CNNs) ermöglicht es, komplexe Muster und Strukturen in Dokumenten zu erkennen, was für die Unterscheidung verschiedener Dokumententypen entscheidend ist. CNNs haben sich als effektive Werkzeuge für die Bild- und Dokumentenklassifizierung etabliert. [16] legten den Grundstein für die Anwendung von CNNs in der Mustererkennung. Später demonstrierten [12], wie tiefe CNNs die Leistung in Bildklassifizierungsaufgaben erheblich verbessern können. Für die Dokumentenklassifizierung haben [20] gezeigt, dass CNNs effektiv zwischen verschiedenen Dokumententypen unterscheiden können. Das VGG16-Modell, eingeführt von [34], zeichnet sich durch seine Tiefe und die Verwendung kleiner 3x3-Filter aus. Es hat sich in vielen Bildklassifizierungsaufgaben bewährt. Das InceptionV3-Modell von [39] nutzt Inception-Module, um sowohl lokale als auch globale Merkmale zu erfassen. Die Interpretierbarkeit von neuronalen Netzen ist ein wichtiges Forschungsgebiet, insbesondere bei Anwendungen, die Transparenz erfordern. Methoden wie Grad-CAM, Integrated Gradients und Layer-wise Relevance Propagation (LRP) helfen dabei, die Entscheidungsprozesse von CNNs nachvollziehbar zu machen. [32] führten Grad-CAM ein, um visuelle Erklärungen für die Vorhersagen von CNNs zu generieren. Integrated Gradients, vorgestellt von [38], bieten eine axiomatische Grundlage für die Attributionsmethoden in neuronalen Netzen. [22] nutzten Integrated Gradients, um die Entscheidungsfindung von Modellen in Textklassifizierungsaufgaben zu analysieren. LRP, entwickelt von [3], ermöglicht eine feingranulare Analyse der Beitrage einzelner Neuronen zur Modellvorhersage. Diese Methode wurde von [21] weiterentwickelt und in verschiedenen Anwendungsbereichen, einschließlich der Dokumentenklassifizierung, eingesetzt. Die Validierung der Aussagekraft von XAI Methoden ist entscheidend, um sicherzustellen, dass die erzeugten Erklärungen tatsächlich die Entscheidungsgrundlagen der Modelle widerspiegeln. [31] schlugen die Verwendung von regionalen Perturbationen vor, um die Relevanzzuweisungen zu testen. Die Average Output Probability Change (AOPC) wurde dabei als Metrik eingeführt, um die Auswirkungen der Entfernung relevanter Bildbereiche auf die Modellvorhersagen zu quantifizieren. Der DocILE-Benchmark, vorgestellt von [36], bietet einen umfangreichen Datensatz für die Forschung im Bereich der Dokumentenklassifizierung und -extraktion. Er umfasst über eine Million Dokumente und konzentriert sich auf die Lokalisierung und Extraktion von Schlüsselinformationen. Die Kombination von tiefen CNNs wie VGG 16 und InceptionV3 mit XAI Methoden ermöglicht es, leistungsfähige und interpretierbare Mo-

delle für die Dokumentenklassifizierung zu entwickeln. Die Anpassung dieser Modelle an spezifische Aufgaben und Datensätze sowie die Validierung ihrer Erklärbarkeit sind entscheidend für den Fortschritt in diesem Forschungsbereich.

2 Theoretischer Hintergrund

2.1 Modelle

2.1.1 Künstliche Neuronale Netze

Künstliche Neuronale Netze (KNN) sind vom Aufbau und der Funktionsweise biologischer neuronaler Netzwerke im menschlichen Gehirn inspiriert. Sie bestehen im Wesentlichen aus miteinander verbundenen künstlichen Neuronen, die in mehreren Schichten (Layer) organisiert sind [27, S. 78-79].
Typischerweise unterscheidet man innerhalb der Schichten zwischen einer Eingangsschicht (Input Layer), einer oder mehreren verborgenen Schichten (Hidden Layer) und einer Ausgangsschicht (Output Layer) (Abbildung 2.1). Der Input Layer nimmt die Eingabedaten (Input Values) direkt auf und leitet sie an den nachfolgenden Layer weiter. Die Hidden Layers führen die eigentlichen Berechnungen durch, indem sie die eingehenden Signale aus der vorherigen Schicht verarbeiten. Der Output Layer liefert das Ergebnis dieser Berechnungen und stellt die Ausgabe des Netzwerks dar. Das KNN ist dabei Layer weise vollständig vernetzt, das heißt jedes Neuron einer Schicht ist mit jedem Neuron der vorhergehenden Schicht verbunden, wobei die Ausgangssignale der Neuronen der vorangehenden Schicht als Eingabewerte für die Neuronen der aktuellen Schicht dienen.[2]

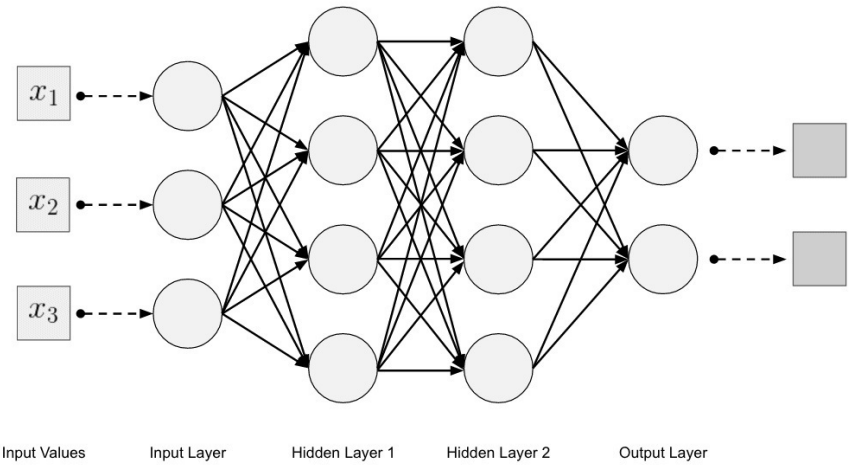

Abbildung 2.1: Aufbau eines KNNs mit zwei Hidden Layers. [27, S. 105]

2.1.1.1 Künstliches Neuron

Ein künstliches Neuron besteht aus mehreren Eingaben, denen ihr jeweiliges Gewicht zugeordnet ist, einer Summenfunktion und der Aktivierungsfunktion (Abbildung 2.2). Das Eingabesignal x_j ist die Information des vorangegangen künstlichen Neurons oder stammt aus den Eingabedaten, dieses Signal wird mit dem Gewicht w_j multipliziert und durch die Summenfunktion addiert. Zusätzlich kann dem Resultat ein Bias b hinzugefügt werden, dadurch kann das Neuron auch bei niedrig anliegenden Eingabesignalen aktiviert werden was zu einem besseren Lernergebnis des KNNs führt.

$$z = \sum_{j=1}^{d} w_j \cdot x_j + b \tag{2.1}$$

Dieses Ergebnis wird dann als Argument an die Aktivierungsfunktion $\phi(z)$ übergeben. Die Aktivierungsfunktion führt eine nichtlineare Transformation durch und bestimmt so den Ausgabewert y des künstlichen Neurons. [2, S. 6]

$$y = \phi(z) = \phi\left(\sum_{j=1}^{n} w_j \cdot x_j + b\right) \tag{2.2}$$

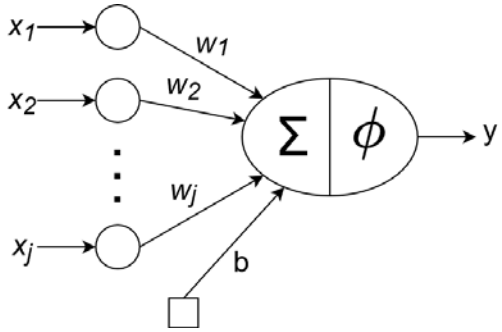

Abbildung 2.2: Aufbau eines künstlichen Neuron mit Bias. (adaptiert nach [2, 5])

2.1.1.2 Aktivierungsfunktion

Rectified Linear Unit

Die Rectified Linear Unit (ReLU) ist eine weit verbreitete Aktivierungsfunktion, die insbesondere in Hidden Layern von KNNs für visuelle Aufgaben eingesetzt wird [23]. Die ReLU-Aktivierungsfunktion setzt negative Eingaben auf Null und lässt positive Werte unverändert, wodurch sie eine stückweise lineare und nicht-lineare Funktion darstellt, die effizient berechnet werden kann [2, S. 13,17][9, S. 189].

$$\phi(x) = max(0, x) = \begin{cases} x & \text{wenn } x \geq 0 \\ 0 & \text{sonst} \end{cases} \tag{2.3}$$

SoftMax

In KNNs wird zur Ausgabenklassifizierung typischerweise die Softmax-Aktivierungsfunktion im Output Layer angewendet, dabei werden die Ausgabewerte der vorherigen Schicht in eine Wahrscheinlichkeitsverteilung der Klassen umgewandelt. Für jeden Ausgabewert v_i einer Klasse i wird eine Wahrscheinlichkeit berechnet. Dabei stellt k die Anzahl der Klassen dar.

$$\phi(\overline{v})_i = \frac{e^{v_i}}{\sum_{j=1}^{k} e^{v_j}} \quad \forall i \in \{1, \ldots, k\} \tag{2.4}$$

Die Softmax-Funktion gibt dabei an, wie wahrscheinlich es ist, dass die Eingabe zur Klasse i in Abhängigkeit der anderen Klassen gehört [2, S. 14,17,19] [9, S. 180-184].

2.1.1.3 Overfitting und Underfitting

Eine Überanpassung (Overfitting) tritt auf, wenn ein Modell sich den Trainingsdaten zu genau anpasst. Dabei erlernt das Modell nicht nur die Muster im Datensatz, sondern auch dessen Rauschen und Schwankungen. Das führt dazu, dass das Modell sehr gut innerhalb der Trainingsdaten abschneidet, jedoch bei unbekannten Daten versagt und somit keine generalisierten Vorhersagen treffen kann. Der Trainingsdatensatz stellt dabei nur einen Ausschnitt aus der realen Welt dar, sodass ein überangepasstes Modell nicht flexibel genug ist um auf unbekannte Daten verlässlich reagieren zu können.

Die Unteranpassung (Underfitting) beschreibt das Gegenteil. Hierbei findet das Modell nicht die grundlegenden Muster im Trainingsdatensatz, dies führt zu einer schwachen Vorhersage sowohl auf den Trainingsdaten als auch auf unbekannte Daten [27, 51] [2, S. 25,26].

2.1.1.4 Optimierungsfunktion

Optimierungsfunktionen werden beim Training neuronaler Netze eingesetzt, um die Gewichtungen der Neuronen anzupassen. Ziel ist es, eine Kostenfunktion zu minimieren, die die Differenz zwischen der Vorhersage des Modells und den tatsächlichen Ergebnissen beschreibt. Die Funktion gibt außerdem an, wie weit das Modell vom optimalen Zustand entfernt ist.

Der Gradientenabstieg (Gradient Descent) ist eine der am häufigsten verwendeten Methoden. Ein Gradient beschreibt die Ableitung der Kostenfunktion und zeigt in Richtung des stärksten Anstiegs der Funktion. Der Gradientenabstieg passt dabei die Modellparameter in negativer Richtung des Gradienten an, also in Richtung des stärksten Abfalls. Die Parameter werden dabei schrittweise verändert, bis das Minimum der Kostenfunktion gefunden wird. Hierfür wird der Gradient der Kostenfunktion über alle Trainingsdaten berechnet. Die Schrittgröße wird durch der

Lernrate bestimmt. Eine gut gewählte Lernrate ist entscheidend für die Effizenz des Trainings. Eine zu kleine Lernrate führt zu einer langsamen Konvergenz und erhöht die Wahrscheinlichkeit, in einem lokalen Minimum stecken zu bleiben. Eine zu große Lernrate hingegen kann zu einem oszillierenden Verhalten führen, wodurch das globale Minimum nicht gefunden wird [27, S. 58-60]. Die Topographie der Kostenfunktion kann mehrere Minima aufweisen, sogenannte Täler. Das tiefste Tal wird als globales Minimum bezeichnet und stellt die optimale Kostenfunktion dar. Lokale Minima sind Täler, die höher liegen und weiter vom globalen Minimum entfernt sind. Geriet ein Gradientenabstieg in ein lokales Minumum, kann er darin stecken bleiben. [9, S. 80-84]

Der Lernprozess kann durch den stochastischen Gradientenabstieg (Stochastic Gradient Descent, SGD) beschleunigt werden. Dabei wird zur Berechnung des Gradienten, im Gegensatz zum klassischen Gradientenabstieg, nur eine zufällige Teilmenge der Daten (Batch) verwendet. Dadurch wird die Berechnungszeit pro Schritt erheblich reduziert. da der Gradient nicht über die gesamten Trainingsdaten, sondern nur über einen Teil berechnet wird.
Darüber hinaus wird die Lernrate während des Trainings schrittweise angepasst. Zu Beginn wird eine relativ hohe Lernrate verwendet, um schnelle Fortschritte zu erzielen. Im Verlauf des Trainings wird die Lernrate sukzessive verringert, um eine stabile Konvergenz zu ermöglichen. Die Berechnungszeit pro Schritt nimmt aufgrund der Batches bei steigender Trainingsdatenmenge nicht zu [9, S. 290-292].

2.1.1.5 Training

Künstliche Neuronale Netze erlernen die Zusammenhänge zwischen Eingabedaten und den gewünschten Ausgaben durch iterative Anpassung der Gewichte. Dieser Lernprozess wird als Training bezeichnet. Dafür werden Trainingsdaten benötigt die je nach Bereitstellung für unterschiedliche Lernverfahren verwendet werden können. Beim überwachten Lernen (Supervised Learning) werden die Eingangsdaten mit den korrekten Ausgaben (Label) verknüpft, hierdurch lernt das Netz bestimmte Eingaben mit bestimmten Ausgaben zu assoziieren. Diese Methode eignet sich besonders gut für Klassifikationsaufgaben. Beim unüberwachten Lernen (Unsupervised Learning) hingegen erhalten die Trainingsdaten keine Labels. Das Ziel ist es, durch eine sinnvolle Gruppierung der Daten eine Struktur in den Daten

zu entdecken. Das halbüberwachte Lernen (Semi-supervised Learning) kombiniert Elemente aus den vorherigen Verfahren. Die bereitgestellten Trainingsdaten enthalten teilweise Labels, das Netz lernt sowohl aus Trainingsdaten mit und ohne Label, wodurch die Vorhersagegenauigkeit verbessert wird. Beim verstärktenden Lernen (Reinforcement Learning) interagiert das KNN mit einer Umgebung und erhält für sein Aktionen Belohnungen oder Strafen, wodurch es lernt, welche Aktionen zum besten Ergebnis führen. Das Netz wählt beim aktiven Lernen (Active Learning) die Daten selbst aus, die es für das Lernen benötigt. Es konzentriert sich dabei auf die Datenpunkte, die ihm die meisten neuen Erkenntnisse liefern. [18, S. 7-8]

2.1.2 Convolutional Neural Networks

Convolutionale Neuronale Netzwerke (CNN) bezeichnen eine spezielle Architektur künstlicher neuronaler Netzwerke, die stark von der Funktionsweise der biologischen Sehrinde inspiriert ist und sich besonders gut geeignet für die Verarbeitung von mehrdimensionalen Daten wie Bildern zeigt [7]. Im Gegensatz zu herkömmlichen KNNs integrieren CNNs eine oder mehrere Faltungsschichten (Convolutional Layers), um lokale Merkmale (Features) zu ermitteln. Die Pooling Schicht (Pooling Layer) reduziert die räumliche Dimension des Convolutional Layers. Durch die Kombination von Convolutional- und Pooling Layern werden zunehmend komplexere Features extrahiert, wobei üblicherweise die ersten Layer einfache Strukturen erkennen und spätere Layer komplexere Formen und Texturen identifizieren. Bevor die extrahierten Features von den vollständig verbundenen Schichten (Fully Connceted Layers, FCs) weiterverarbeitet und klassifiziert werden können, müssen sie durch den Flatten Layer in eine eindimensionale Vektorform transformiert werden (Abbildung 2.3) [37][25, S. 546,547].

2.1.2.1 Convolutional Layer

Der Convolutional Layer ermöglicht die Extraktion von lokalen Features aus einem Bild. Dazu wird ein kleiner, gewichteter Filter (Kernel) über die Eingabedaten geschoben, wobei an jeder Position das Elementweise Produkt zwischen Kernel und

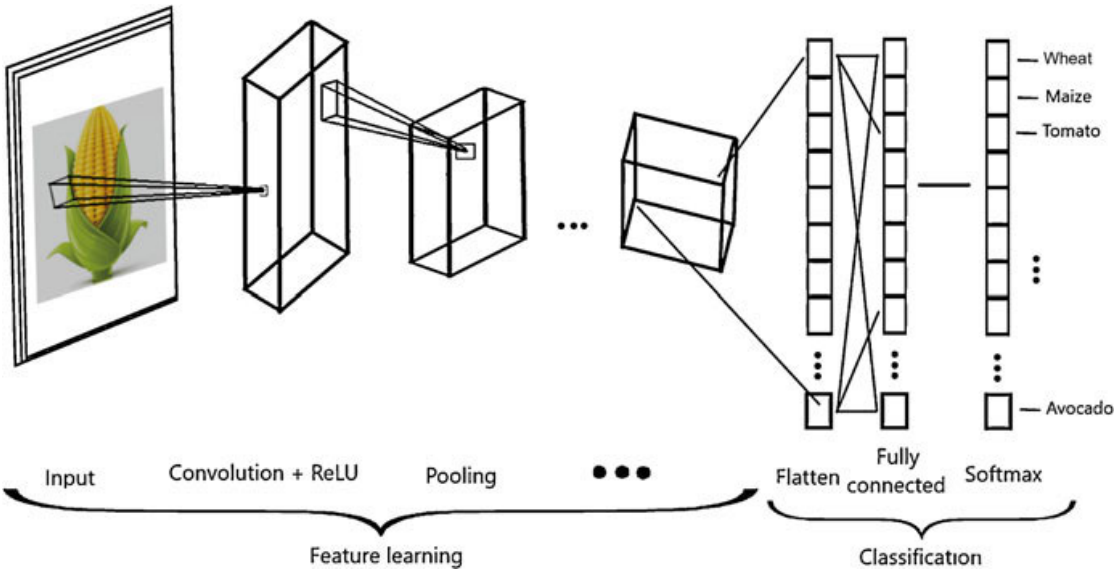

Abbildung 2.3: Beispiel CNN mit Convolutional Layer inkl. ReLU Aktivierungsfunktion, Pooling Layer, Flatten Layer und zwei FCLs[25, S. 547].

dem entsprechenden Bildausschnitt berechnet und summiert wird. Diese Operation wird als Faltung bezeichnet [5, S. 258].

Die Wahl der Kernelgröße bestimmt somit die räumliche Ausdehnung der erfassten Features. Diese sind quadratisch und üblicherweise zwischen 2x2 und 4x4 Pixel groß. Die Schrittgröße (Stride) des Kernels bestimmt, um wie viele Pixel der Kernel je Faltung weiter geschoben wird. Je größer der Stride, desto geringer ist die Auflösung und die Rechenzeit [27, S. 264]. Nach jeder Faltung wird eine nichtlineare Aktivierungsfunktion, wie beispielsweise die ReLU-Funktion, angewendet. Diese führt zu einer sparsameren Repräsentation der Daten und verbessert die Lernfähigkeit des Netzwerks. Das Ergebnis einer Faltung ist eine Merkmalskarte (Feature Map). Jede Feature Map stellt einen Kanal dar, der die Intensität eines bestimmten Features an den jeweiligen Positionen des Bildes enthält [5, S. 258] [42]. Eine beispielhafte Faltungsoperation ins in Abbildung 2.4 dargestellt.

Ein Bild kann mehr als einen Kanal besitzen, so besteht beispielsweise ein RGB-Farbbild aus drei Kanälen. Für jeden Kanal wird mindestens ein Kernel verwendet, dabei besitzt jeder seine eigenen Gewichte und extrahiert spezifische Features aus dem entsprechenden Kanal. Die Anwendung mehrerer Kernel führt zur Erzeugung mehrerer Merkmalskarten (Feature Maps), welche zusammen einen Tensor bilden.

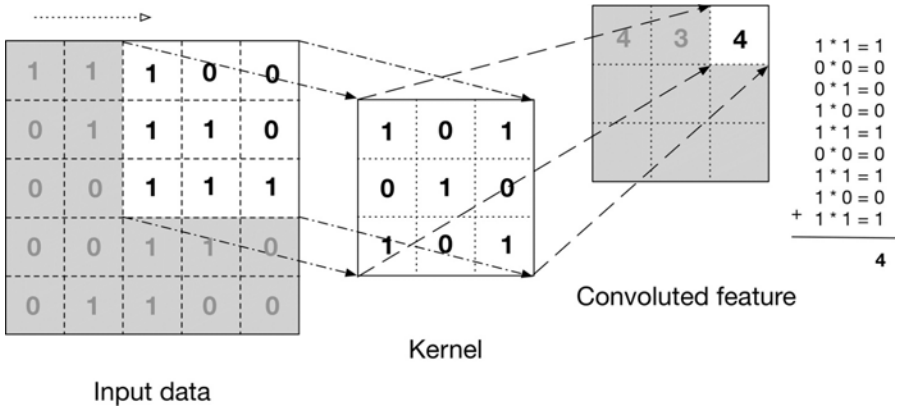

Abbildung 2.4: Beispiel einer Faltungsoperation [27, S. 251]

Die Höhe und Breite des entstehenden Tensors ist im Vergleich zum Eingangsbild typischerweise kleiner, besitzt jedoch eine größere Tiefe. Diese Tiefendimension repräsentiert die verschiedenen extrahierten Features. [37].

2.1.2.2 Pooling Layer

Obwohl das Bild durch den Convolutional Layer verkleinert wird, kann die hohe Anzahl an Parametern in tiefen CNNs die Wahrscheinlichkeit des Overfittings erhöhen. Um dieses Risiko zu reduzieren und die Rechenkomplexität zu verringern, wird nach einem Convolutional Layer üblicherweise ein Pooling Layer eingesetzt. Dadurch reduziert sich die Höhe und Breite der Feature Maps, während die Position der wichtigsten Features erhalten bleibt, ohne dass sich die Tiefendimension verändert. Als häufigste Pooling Methode wird das sogenannte Max-Pooling eingesetzt. Hierbei wird innerhalb eines üblichen 2x2 oder 3x3 Pixel großen Filterbereichs der maximale Wert ausgewählt[27, S. 266,267]. Eine weitere Möglichkeit des Poolings stellt das Average Pooling dar, hierbei wird der arithmetischer Mittelwert des ausgewählten Filterbereichs ausgegeben [25]. Zusätzlich gibt es das Global Average Pooling, was eine alternative Methode zum Flatten Layer darstellt. Hierbei wird für jede Klasse der Klassifikationsaufgabe der Durchschnitt aller Werte der Feature Map berechnet. Dadurch ist die Verbindung zwischen den Feature Maps und den jeweiligen Klassen ähnlicher als bei der herkömmlichen Methode [17].

2.1.2.3 Fully Connected Layer

Eine oder mehrere Fully Connected Layer bilden in CNNs in der Regel die letzten Layer. Diese wandeln die hochdimensionalen Features aus den vorherigen Flatten Layer in eine Klassifikation um. In FCs ist jedes künstliche Neuron mit jedem künstlichen Neuron des vorherigen Layers verbunden. Das ermöglicht dem Netzwerk, komplexere nichtlineare Beziehungen zwischen den Eingangsfeatures zu lernen. Durch die Hinzunahme weitere FCs kann diese Fähigkeit gesteigert werden, da zusätzliche Layer die Modellierung von noch komplexeren Zusammenhängen erlauben. Diese inneren FCs verwenden üblicherweise die ReLU funktion. Das führt jedoch zu einer deutlich größeren Anzahl an Parametern, was einerseits die Modellkapazität aber andererseits auch das Risiko des Overfittings erhöht. Der letzte FC fungiert als Output Layer des CNN. Jedes künstliche Neuron in diesem Layer repräsentiert eine Klasse. Die Ausgabe des Neurons entspricht der Wahrscheinlichkeit, mit der das Eingabebild dieser Klasse zugeordnet wird. Zur Berechnung dieser Wahrscheinlichkeitsverteilung wird typischerweise die Softmax Funktion verwendet. [2, S. 327,328] [33]

2.1.2.4 VGG 16

Das VGG 16 Modell ist ein CNN, das 2014 von der Visual Geometry Group der University of Oxford für die ImageNet Large Scale Visual Recognition Challenge (ILSVRC) entwickelt wurde. Das Modell wurde mit dem ImageNet-Datensatz trainiert mit dem Ziel, Bilder in 1000 Klassen zu klassifizieren. Es verarbeitet RGB-Bilder mit der Größe von 224x224 Pixeln und umfasst insgesamt 16 Layer, die in sechs Blöcken organisiert sind. Die ersten fünf Blöcke enthalten insgesamt zehn Convolutional Layer, die jeweils eine Kernelgröße von 3x3 Pixeln haben und eine ReLU-Aktivierungsfunktion nutzen. Jeder dieser Blöcke wird durch ein Pooling Layer abgeschlossen, der die Max-Pooling Methode verwendet. Im letzten Block befinden sind drei FCs. Der letzte Layer enthält 1000 Neuronen, um die 1000 Klassen abbilden zu können. Die Klassifikation erfolgt durch die Softmax Aktivierungsfunktion (Abbildung 2.5) [34].

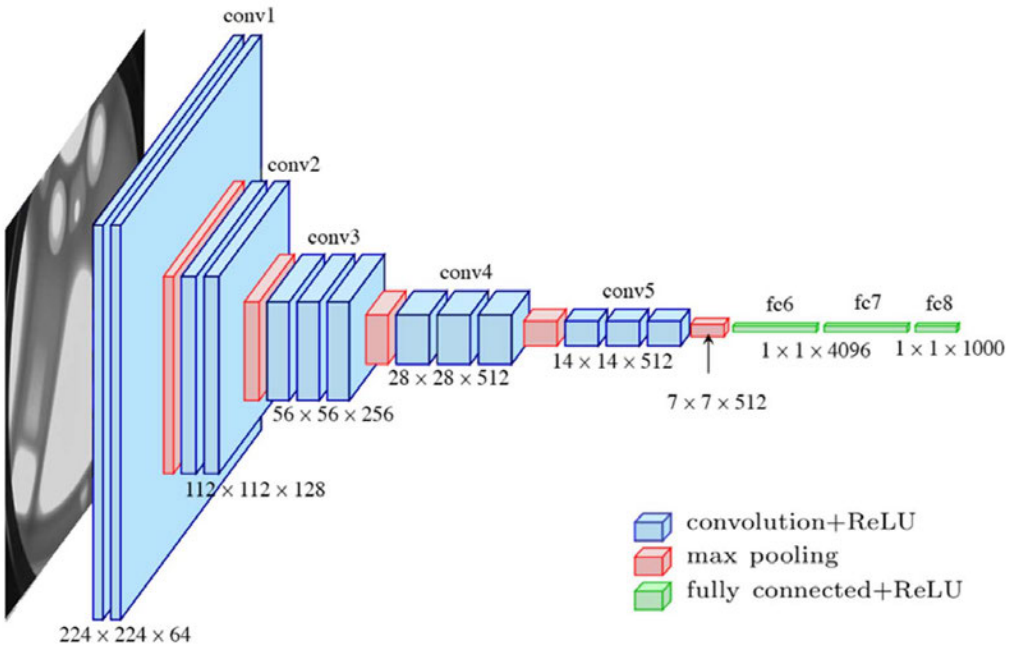

Abbildung 2.5: Standard VGG-16 Architektur nach [34] [8]

2.1.2.5 InceptionV3

InceptionV3 wurde 2015 von Google Forschern vorgestellt. Das CNN kann Bilder in 1000 Objektklassen klassifizieren, dazu wurde es auf dem ImageNet-Datensatz trainiert. Als Eingabedaten werden Farbbilder mit 299x299 Pixeln verwendet. Durch die 42 Layer tiefe Architektur und die namensgebenden Inception-Module wird eine ressourceneffiziente Feature Extraktion ermöglicht. Das Modell ist neben den Input- und Output in drei Inception-Module (A, B, C), Grid Size Reduction und einem Hilfsklassifikator (Auxiliary Classifier) Blöcken aufgeteilt (Abbildung 2.6). Die Module führen parallel Faltungen mit unterschiedlicher Kernelgröße aus, um sowohl kleine als auch große Merkmale zu ermitteln. Durch die Faktorisierung großer Kernels in kleinere wird der Rechenaufwand deutlich reduziert. Anschließend werden die Ergebnisse der einzelnen Faltungen konkateniert. Grid Size Reduction verkleinert die Feature Maps. Der Auxiliary Classifier wird als Regulierer verwendet, um die Konvergenz zu verbessern. [39]

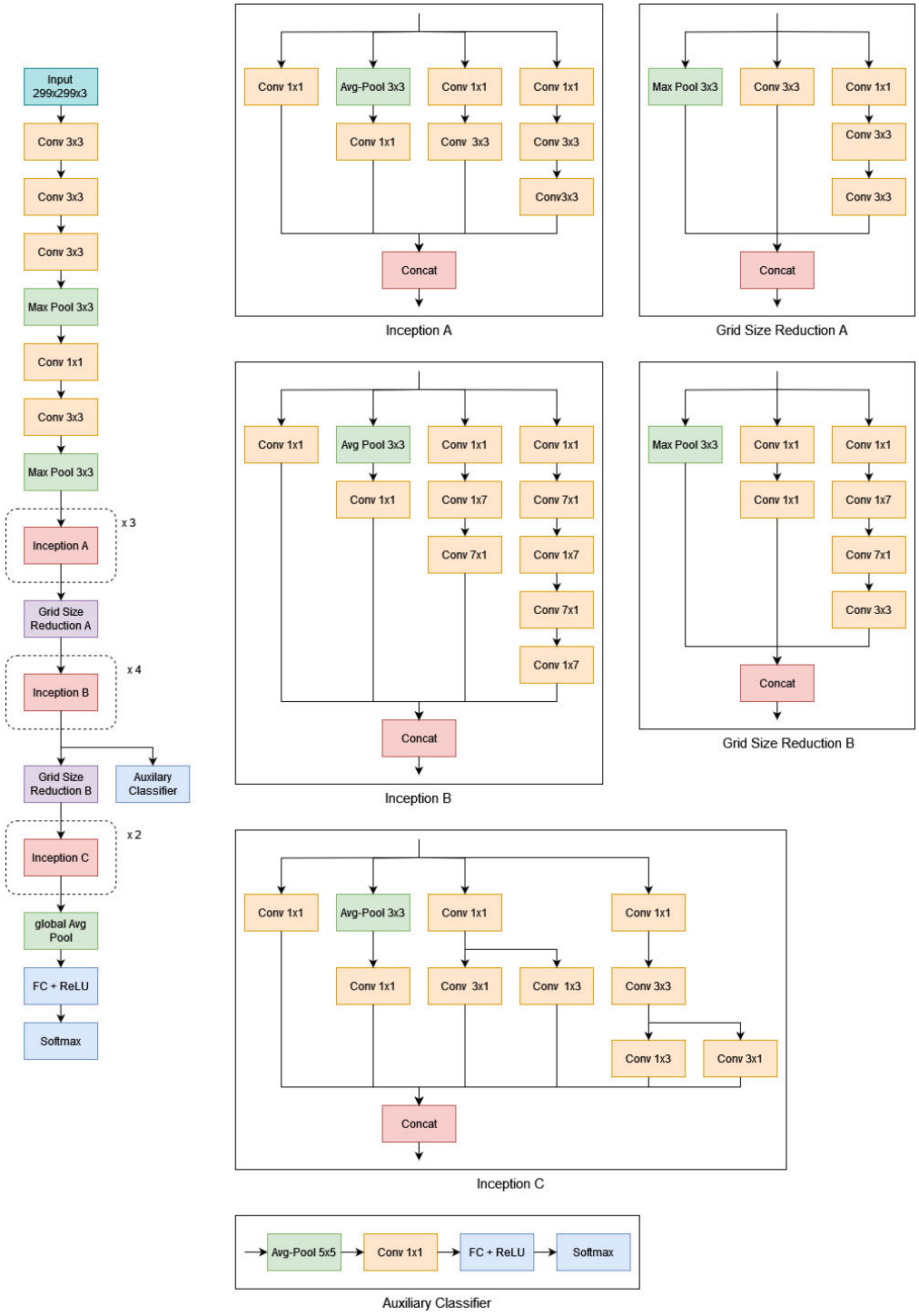

Abbildung 2.6: InceptionV3 Architektur (Adaptiert nach [28])

2.2 Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) ist ein Sammelbegriff für eine Vielzahl von Methoden, deren Aufgabe es ist, die undurchsichtigen Entscheidungsfindungsprozesse von künstlicher Intelligenzen transparenter zu machen. Obwohl häufig die Begriffe Erklärbarkeit (Explainability) und Interpretierbarkeit (Interpretability) als Synonym verwendet werden, gibt es wichtige Unterscheide in ihrer Bedeutung [19]. Während Explainability das allgemeine Verständnis des komplexen Modellfunktionalität zum Ziel hat, bezieht sich die Interpretability auf die Erklärung spezifischer Vorhersagen oder Entscheidungen eines Modells. Anhand dieser Erklärungen kann XAI das Vertrauen in die Entscheidungsfindung von künstlicher Intelligenz stärken und die Entwicklung neuer Modelle fördern [30, S. 3-6].

XAI Methoden lassen sich in der Regel anhand dreier Hauptkriterien kategorisieren. Erstens wird zwischen einer individuellen Vorhersagen (local, local explanation) und dem gesamten Verhalten eines Modells (global, global explanation) unterschieden [11]. Zweitens wird nach dem Zeitpunkt der Erklärung differenziert. Bei Modellbasierten erklärenden (Model-Based) Modellen entsteht die Erklärung direkt während der Vorhersage, gegensätzlich dazu erfolgt die Erklärung bei der Post-Hoc Methode nachträglich. Als letzten wird zwischen einer Modellspezifischen oder Modell-agnostisch Methoden unterschieden[40].

2.2.1 Local Explanation

Local Explanation liefert basierend auf einer bestimmten Eingabe eine detaillierte Interpretation für einzelne Entscheidungen oder Vorhersagen eines Modells. Dadurch lässt sich feststellen, welche speziellen Merkmale die Entscheidung beeinflussen. Am häufigsten werden diese Methoden für neuronale Netze verwendet, sie können jedoch aufgrund ihrer Modellagnostik auch für andere Modelle eingesetzt werden [1].

2.2.2 Global Explaination

Das Ziel von Global Explanation ist es, das allgemeine Entscheidungsverhalten eines Modells zu verstehen. Dabei werden die Merkmale oder Konzepte unabhängig von der Eingabe identifiziert, die beim Vorhersagen oder Klassifizieren im Allgemeinen für das Modell besonders relevant sind [1].

2.2.3 Post-Hoc

Der Post-Hoc Ansatz wird typischerweise auf komplexe Modelle des maschinellen Lernens angewendet, wie beispielsweise auf KNNs. Dazu liegt häufig ein bereits trainiertes Modell vor, bei dem das Innenleben des Modell nicht bekannt ist oder verändert wurde. Die dazugehörige Erklärung erfolgt nachträglich, nachdem die Vorhersage des Modells bereits erstellt wurde. Diesen Vorgang kann als eine Form des Reverse-Engineering betrachten werden. Allerdings muss die so entstandene Erklärung nicht den tatsächlichen Entscheidungsprozess des Modells widerspiegeln [6] [1].

2.2.4 Model-Based

Model-basierte Erklärungen werden häufig für traditionelle Modelle verwendet, die eine Balance zwischen Einfachheit und Komplexität aufweisen, wie beispielsweise Regressionsmodelle. Diese Modelle ermöglichen es, die Beziehung zwischen Eingabedaten und Ausgaben nachvollziehbar darzustellen. Da die Erklärbarkeit in der Regel bei hoher Komplexität und abnimmt, eignen sie sich gut für Anwendungen mit einer begrenzten Anzahl von Features [40].

2.2.5 Modell-spezifisch

Modellspezifische Methoden sind auf bestimmte Modellarchitekturen beschränkt. Sie verwenden oftmals spezifische Eigenschaften eines Modells, wie beispielsweise den Convolutional Layer von CNNs. Ein Nachteil modellspezifischer Methoden ist, dass sie nicht auf andere Modelltypen übertragbar und somit die Anwendbarkeit eingeschränkt wird [1].

2.2.6 Modell-agnostisch

Modellagnostischen Methoden sind unabhängig von der Wahl des Modells anwendbar. Die Erklärung basiert dabei ausschließlich auf den Ein- und Ausgabedaten des Modells, ohne auf dessen Mechanismen einzugehen. Da durch ist die Vorhersage vollständig von der Erklärung entkoppelt. Aufgrund dieser Eigenschaft sind modellagnostischen Verfahren in der Regel Post-Hoc Methoden [40].

2.2.7 Saliency Map

Bei Salienzkarten (Saliency Map) werden die wichtigen Regionen in einem Bild hervorgehoben. Diese Regionen erhalten aufgrund ihrer visuellen Features eine erhöhte Aufmerksamkeit [13]. Typischerweise werden Saliency Maps zur Identifikation und Visualisierung von visuell auffälligen Bildbereichen in Neuronalen Netzen verwendet. Bei der Visualisierung werden die Regionen, die ausschlaggebend für die Klassifikation sind, farblich hervorgehoben. Umgangssprachlich wird dies auch als Heatmap bezeichnet [29].

2.2.8 GRAD-CAM

Die gradientengewichtete Klassenaktivierungskarte (Gradient-weighted Class Activation Map, Grad-CAM) eine Post-Hoc, Local Exaplanation Methode, die modellspezifisch für CNNs entwickelt wurde. Grad-CAM generiert anhand einer Zielklasse c eine Heatmap $L^c_{Grad-CAM}$, die die wichtigsten Regionen in einem Bild hervorhebt. Grad-CAM kann theoretisch auf jeden Convolutional Layer angewendet werden, wird jedoch üblicherweise wegen den komplexesten Feature Maps im letzten Convolutional Layer verwendet.

Für jede Feature Map A^k des letzten Convolutional Layers wird der Gradient des Klassenscores y^c der Zielklasse c bestimmt. Die Gradienten werden über die gesamte Feature Map, mit Breite m und Höhe n, berechnet. Dabei werden die einzelnen Neuronen der Feature Map durch die Indizes i und j repräsentiert. Diese Gradienten führen zu einer Gewichtung α^c_k, welche eine partielle Linearisierung von

A^k darstellt und somit die Bedeutung der jeweiligen Feature Map für die Zielklasse c erfasst.

$$\alpha_k^c = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{\partial y_c}{\partial A_{i,j}^k} \tag{2.5}$$

Anschließend wird die Gewichtung α_k^c mit den Aktivierungen A^k der Feature Map multipliziert. Durch das aufsummieren des Ergebnisses wird die Heatmap erstellt, wobei hohe Werte auf eine starke Aktivierung in den entsprechenden Bildregionen hinweisen. Auf die resultierende Heatmap wird die ReLU Funktion angewendet. Hierdurch werden negative Werte entfernt und ausschließlich positive Aktivierungen bleiben erhalten.

$$L_{Grad-CAM}^c = ReLU\left(\sum_k \alpha_k^c A^k\right) \tag{2.6}$$

Die erzeugte Heatmap hat die gleiche Größe wie die Feature Maps des gewählten Convolutional Layers, sie muss daher noch auf die ursprüngliche Bildgröße hochskaliert werden [32].

2.2.9 Integrated Gradients

Integrierte Gradienten (Integrated Gradients, IG) berechnen die Relevanz der Eingabefeatures zur Erstellung einer Heatmap. Das Verfahren basiert dabei auf zwei grundlegenden Axiome.
Das erste Axiom ist die Sensitivität. Es bestimmt eine Attribution, die angibt, ob ein Feature relevant ist oder nicht. Unterscheiden sich ein Input x und eine neutrale Eingabe, genannt Basiswert, x' durch das Features x_i und der Vorhersage $f(x) \neq f(x')$, dann erhält x_i eine non-zero Attribution und ist damit ein relevantes Feature.
Das zweite Axiom ist die Implementierungsinvarianz. Es besagt, das die Attribution gleich bleibt, wenn der Output von zwei unterschiedlich implementierten Netzen F_1 und F_2 für alle Inputs identisch ist.

Gegeben sei ein neuronales Netz $F : R^n \rightarrow [0, 1]$, mit einem Input $x \in R^n$ und einem Basiswert $x' \in R^n$. Die Gradienten werden entlang einer geraden Linie vom Basiswert x' zur Eingabe x berechnet. Dabei bewegt sich x' schrittweise auf x zu, wobei α eine Interpolationskonstante ist, die zwischen 0 und 1 variiert. α bestimmt dabei den Grad der Annäherung von x' auf x. Anschließend werden die Gradienten entlang dieser Linie integriert. Als Basiswert x' wird in der Regel eine Zahl nahe Null gewählt, der im Falle von Bildern einem schwarzes Bild entspricht.

$$IntegratedGrads_i(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \tag{2.7}$$

Integrated Gradients stellt damit die Relevanz eines Features i dar, wobei ein höherer Wert auf einen hohen Einfluss des Features hinweist.

Des Weiteren wird das Axiom der Vollständigkeit erfüllt, in dem die Summe aller Attributionen gleich der Differenz zwischen dem Output $F(x)$ für den Input x und dem Output $F(x')$ für den Basiswert x' ist.

$$\sum_{i=1}^{n} IntegratedGrads_i(x) = F(x) - F(x') \tag{2.8}$$

Für eine effizientere Berechnung des Integrated Gradients kann das Integral durch die Riemannsche Summen approximiert werden. Dabei wird das Integral durch eine Summe der Gradienten an diskreten Punkten entlang der Linie zwischen x' und x ersetzt. Die Anzahl der Punkte entspricht der Schrittzahl m in der Approximation. [38]

$$IntegratedGrads_i i^{approx}(x) = (x_i - x'_i) \times \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m}}{\times}(x - x))\partial x_i \times \frac{1}{m} \tag{2.9}$$

2.2.10 Layerwise Relevance Propagation

Die Layerwise Relevance Propagation (LRP) erstellt Saliency Maps, indem die Relevanz der einzelnen Neuronen eines KNNs für eine spezifische Klassifikation

berechnet wird. Dabei wird die Relevanz schrittweise rückwärts für jeden Layer, beginnend beim Output Layer bis hin zum Input Layer berechnet. LRP basiert auf einem Erhaltungssatz, der besagt, dass die Relevanz eines Neurons vollständig an den vorhergehenden Layer verteilt werden muss, ohne das dabei Relevanz verloren geht. Dies wird als Conservation Property bezeichnet.

Bei zwei verbundenen Neuronen j und k wird die Relevanz R_k des Neurons k auf die Neuronen des vorhergehenden Layers verteilt. Somit ergibt sich die Relevanz R_j des Neurons j aus der Summe aller Neuronen k, gewichtet durch z_{jk} und R_k. Dabei gibt z_{jk} an, wie stark das Neuron j das Neuron k relevant beeinflusst hat. Der Term z_{jk} im Nennenrsorgt dafür, dass die Conservation Property bestehen bleibt. Gleichung 2.10

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k \tag{2.10}$$

Das bedeutet auf Layer-Ebene, dass die Relevanz eines Layers vollständig auf die Neuronen des vorherigen Layers übertragen wird, sodass die Gesamtrelevanz in allen Layern unverändert bleibt Gleichung 2.11 Schematisch ist dies in Abbildung 2.7 dargestellt.

$$\sum_j R_j = \sum_k R_k \tag{2.11}$$

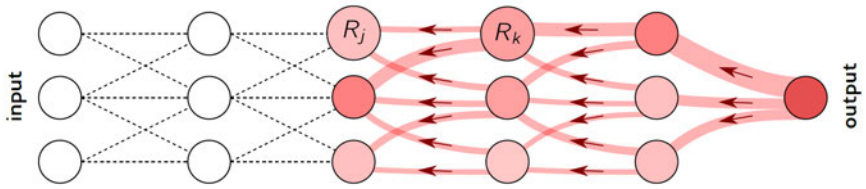

Abbildung 2.7: Struktur von LRP [21])

Für die Berechnung der Relevanz zwischen den verbunden Neuronen gibt es verschiedene LRP-Regeln. Die Basisregel (LRP-0) orientiert sich an der Berechnung der Neuronenaktivierung eines KNNs (Gleichung 2.1). Die Aktivierung eines Neurons a_k wird durch die Summe der aller Aktivierten Neuronen a_j des vorherigen

Layer, deren Gewichtung w_{jk} und einen Bias berechnet, wobei der Bias als zusätzliche Komponente a_0 mit der Gewichtung w_{0k} dargestellt wird.

$$a_k = \sum_{0,j} a_j w_{jk} \tag{2.12}$$

Die LRP-0 Regel verwendet diese Aktivierungsberechnung (Gleichung 2.12), um die Relevanz R_k eines Neurons k zu bestimmen, die eine Relevanz für die Neuronen j des vorherigen Layers haben. Die Verteilung der Relevanz basiert dabei auf der Aktivierung a_j und der Gewichtung w_{jk} zwischen den Layern j und k, siehe Gleichung 2.13 [3] [21].

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k \tag{2.13}$$

2.3 Metriken zur Bewertung

Neuronale Netze legen in der Regel den Schwerpunkt auf eine möglichst genaue Vorhersage. Diese Ergebnisse von verschiedenen Modellen unterscheiden sich jedoch oft, obwohl mit demselben Datensatz trainiert wurde. Um diese Unterschiede objektiv bewerten zu können, werden verschiedene Metriken verwendet.

2.3.1 Confusion Matrix

Die Confusion Matrix (Konfusionsmatrix) ist eine Tabelle, die die Leistung eines Klassifikationsmodells zeigt. Das geschieht, indem die tatsächliche Klasse mit der vorhergesagten verglichen wird. Die Zeilen repräsentieren die korrekten Klassen und die Spalten die vorhergesagten. Auf der Hauptdiagonalen befinden sich die korrekten Klassifikationen (True Positiv, TP), diese verläuft von oben links nach unten rechts. Jede Zelle gibt somit an, wie häufig Objekte richtig klassifiziert wurde. Alle Zellen außerhalb der Hauptdiagonalen zeigen Fehler an. Liegt eine Zelle in der Zeile und nicht auf der Hauptdiagonalen, so ist diese ein False Negative (FN). Dieses Objekt wurden fälschlicherweise einer anderen Klasse zugeordnet. Befindet sich eine Zelle in einer Spalte, aber nicht auf der Hauptdiagonalen, so

handelt es sich um ein False Positve (FP). Diese Zellen zeigen, wie oft Objekte fälschlicherweise der vorhergesagten Klasse zugeordnet wurden. Zellen ohne Wert die sich außerhalb der Hauptdiagonalen befinden sind True Negative (TN) [10]. Beispielhaft ist dies in Abbildung 2.8 dargestellt.

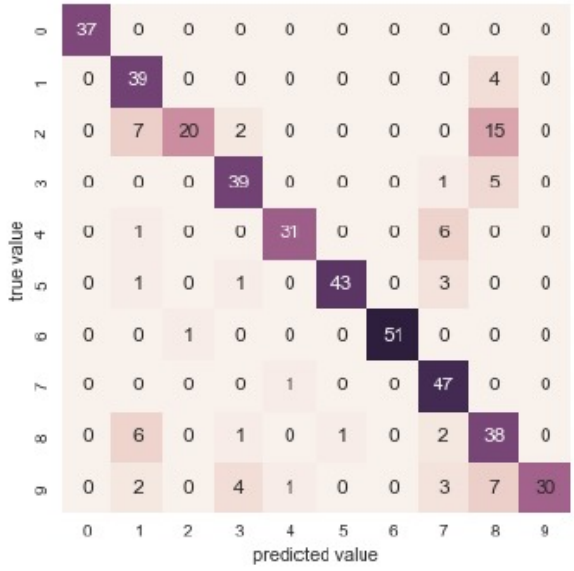

Abbildung 2.8: Beispielhafte Confusion Matrix mit zehn Klassen [41].

2.3.2 Accuracy

Die Genauigkeit (Accuracy) ist eine Metrik, die angibt, wie hoch der Anteil korrekt klassifizierter Bewertungen im Vergleich zur gesamten Anzahl der Ausgaben ist. Dabei werden die True Positives und True Negatives im Zähler zusammen gerechnet - diese Werte wurden korrekt klassifiziert und werden im Vergleich zu allen Elemente berechnet. Die Gleichung ist in Gleichung 2.14 dargestellt

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

Die Accuracy allein sagt oft wenig aus. Der Wert kann hoch sein, selbst wenn das Modell viele True Negatives enthält [10].

2.3.3 Precision

Die Präzision (Precision) gibt den Anteil der korrekt vorhergesagten Elemente einer Klasse an. Dazu werden die True Positives durch die gesamte Anzahl einer Spalte geteilt. Diese Metrik sagt aus wie zuverlässig der Klassifikator ist [10]. Die Gleichung ist in Gleichung 2.15 dargestellt.

$$Precision = \frac{TP}{TP + FP} \tag{2.15}$$

2.3.4 Recall

Der Recall misst hingen welcher Anteil der Elemente einer Klasse korrekt zugeordnet wurden. Dabei werden sowohl True Positive als auch die False Negatives berücksichtigt[10]. Die Gleichung hierzu ist in Gleichung 2.16 dargestellt.

$$Recall = \frac{TP}{TP + NF} \tag{2.16}$$

2.3.5 F1Score

Der F1-Score kombiniert Precision und Recall zu einer einzigen Metrik. Er wird als das harmonische Mittel dieser beiden Werte berechnet [10]. Die Gleichung hierzu ist in Gleichung 2.17 dargestellt.

$$F1 - Score = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall} \right) \tag{2.17}$$

Mit dem F1-Score wird das Leistungsmaß eines Klassifikators für eine bestimmte Klasse bestimmt.

2.3.6 Region Perturbation

Bei der Methode der regionalen Perturbation werden die für die Klassifikation relevanten Bildregionen schrittweise entfernt, um die Qualität der verwendeten Heatmap zu quantifizieren. Eine gute Heatmap sollte dabei die wichtigsten Regionen korrekt identifizieren. Das Ziel ist es, zu ermitteln, wie stark die Klassifikationswahrscheinlichkeit sinkt, wenn die am deutlichsten hervorgehobenen Bereiche der Heatmap stückweise entfernt werden. Dazu wird die Heatmap in ein Raster unterteilt. Durch die verwendete Most Relevant First (MoRF) Methode wird die jeweils am relevantesten erscheinende Bildregion, basierend auf der Heatmap, zuerst entfernt. Eine gute Heatmap führt zu einem schnellen Leistungsabfall, wenn die relevanten Regionen durch MoRF zuerst entfernt werden [31].

2.3.6.1 AOPC

Die Area Over the Perturbation Curve (AOPC) misst die Qualität einer Heatmap, indem sie misst, wie stark die Klassifikationsleistung des neuronalen Netzwerks bei der schrittweisen Bildstörung abnimmt. Der AOPC Wert wird als Durchschnitt der Differenzen zwischen der Klassifikationsausgabe des Modells für das ungestörte Bild und der Ausgabe nach jedem Störschritt berechnet. Dabei ist L die Anzahl der Stufen der Pixelstörung, $f(x)$ die Ausgabewahrscheinlichkeit für die Klasse mit der höchsten Wahrscheinlichkeit, und x das ungestörte Bild. Nach k Stufen der Störung ergibt sich das gestörte Bild $x^{(k)}_{MoRF}$. Die Differenz $f(x^{(0)}_{MoRF}) - f(x^{(k)}_{MoRF})$ gibt an, wie stark sich die Vorhersage nach jeder Störung verändert. Um den AOPC zu berechnen, werden die Differenzen für alle Stufen der Störung summiert und durch $L+1$ geteilt. $p(x)$ bezeichnet dabei den Mittelwert aller Bilder des Datensatzes (s. Gleichung 2.18)

$$AOPC = \frac{1}{L+1}\left(\sum_{k=0}^{L} f(x^{(0)}_{MoRF}) - f(x^{(k)}_{MoRF})\right)_{p(x)} \tag{2.18}$$

Ein höherer AOPC Wert gibt an, dass ein Modell empfindlich auf die Entfernung relevanter Pixel reagiert, wodurch die hohe Bedeutung der entfernten Bildregionen bestätigt wird [31].

2.4 PyTorch

PyTorch ist eine Open-Source-Bibliothek für Python zur Entwicklung und dem Trainieren von neuronalen Netzen. Die Kernkomponenten sind in hauptsächlich in C++ implementiert, was eine hohe Effizienz gewährleistet. Durch die Python-typische Syntax ist Einarbeitung intuitiv und benutzerfreundlich. Ein Trainingsprozess wird in einer Schleife implementiert, die dabei die Eingabedaten durchläuft, die Vorhersagen trifft, den Fehler durch eine definierte Verlustfunktion berechnet und anschließend die Gewichtungen durch einen Optimierungsalgorithmus anpasst. Da durch wird eine hohe Kontrolle über den Trainingsablauf geboten. PyTorch unterstützt sowohl CPU als auch GPU-basierte Berechnung. Letzteres kann durch die parallelen Verarbeitung besonders schnell rechenintensive Aufgaben absolvieren. Dies führt zu einer Beschleunigung des Trainingsprozesses bei großen Datensätzen oder komplexen Modellen [26] [24].

3 Methoden

3.1 Datenverarbeitung

3.1.1 DocILE Datensatz

Der Document Information Localization and Extraction (DocILE) Benchmark stellt den größten Datensatz für Geschäftsdokumente bereit. Der Fokus liegt auf den Aufgaben der Lokalisierung und Extraktion von Schlüsselinformationen (Key Information Localization and Extraction, KILE) sowie der Erkennung von Posten in Geschäftsunterlagen (Line Item Recognition, LIR). Insgesamt umfasst der Datensatz 1.039.147 Dokumente, die in verschiedene Kategorien unterteilt sind. Von den Dokumenten sind 106.680 gelabelt, wobei 6.680 kommentierte Geschäftsdokumente aus öffentlichen Quellen stammen und 100.000 Dokumente wurden anhand von realen Layouts synthetisch generiert wurden. Die verbleibenden 932.467 Dokumente sind ungelabelt und wurden ebenfalls aus öffentliche Quellen bezogen Tabelle 3.1 [36].

Tabelle 3.1: DocILE dataset — the three subsets [36]

	annotated	**synthetic**	**unlabeled**
documents	6 680	100 000	932 467
pages	8 715	100 000	3.4M
layout clusters	1 152	100	*Unknown*
pages per doc.	1-3	1	1-884

Ein markantes Merkmal des gelabelten Datensatzes ist die ungleiche Verteilung der Klassen. Der Datensatz enthält neun unterschiedliche Klassen, wobei die Mehrheit der Dokumente auf nur die Klassen *tex_ invoice* (67,84%) und *order* (26,10%) fällt.

Die restlichen sieben Klassen machen lediglich 6.06% des gesamten Datensatzes aus, was eine deutliche Klassenunbalance zeigt (Abbildung 3.1).

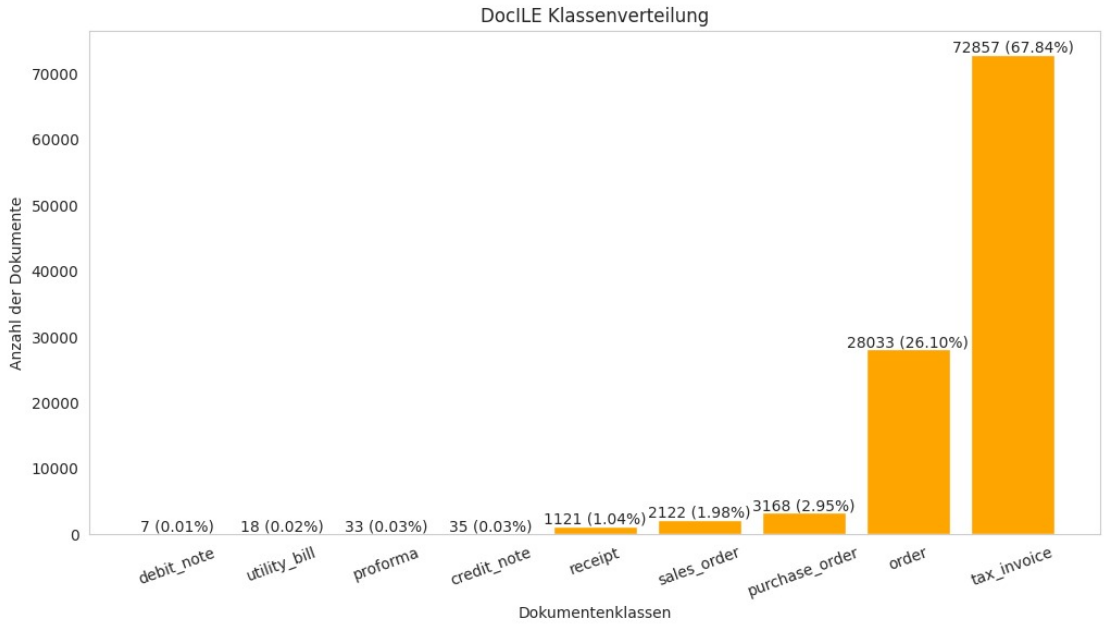

Abbildung 3.1: Verteilung der Dokumentenklassen im DocILE Datensatz

Die gelabelten echten Dokumente des DocILE-Datensatzes sind ausschließlich in englischer Sprache verfasst und umfassen maximal drei Seiten. Ein Großteil dieser Dokumente stammt aus dem 20. Jahrhundert und weist daher typische Merkmale älterer Geschäftsdokumente auf. Die synthetisch generierten Dokumenten wurden im HTML-Format erstellt und anschließend in *Portable Document Format* (PDF) umgewandelt [36].

Die Annotation eines Dokuments wird als JSON-Datei bereitgestellt. Die Datei enthält vier zentrale Felder. Das erste Feld, *field_extractions*, beschreibt die extrahierten Textzeilen aus dem Dokument. Das zweite Feld, *line_item_extractions*, enthält den Inhalt der Tabellenzellen. Im dritte Feld, *line_item_header*, sind die Kopfzeilen der Tabllen gespeichert. Alle drei Felder beinhalten zusätzlich Informationen zum Typ, der Seitenzahl und der Positions, die durch eine Bounding Box angegeben wird. Das vierte Feld, *metadata*, enthält Metadaten des Dokuments wie die Sprache, die Quelle sowie den *document_type* (Listing 3.1) [35]. Letzterer ist besonders wichtig die Klassifizierungsaufgabe im Rahmen des Supervised Learning.

Listing 3.1: Beispiel eines Dokument der Klasse *order* im JSON-Format [35]


```
{
    "field_extractions":[...],
    "line_item_extractions":[...],
    "line_item_headers":[...],
    "metadata":{
        "cluster_id":944,
        "currency":"usd",
        "document_type":"order",
        "language":"eng",
        "original_filename":"00ee49a7-b144-4a11-bb26-8695c0dc567f",
        "page_count":1,
        "page_sizes_at_200dpi":[[1700,2200]],
        "page_to_table_grid":{ ... },
        "source":"pif"
    }
}
```


3.1.2 Datenvorverarbeitung

Die Datenvorverarbeitung wurde innerhalb der Arbeitsgruppe vor Durchführung der Versuche etabliert. Für die Klassifizierungsaufgaben werden ausschließlich die gelabelten Daten des DocILE Datensatz verwendet. Von den 6.680 echten Geschäftsdokumenten enthalten 1.321 Dokumente keine informationen des *document_type*, wodurch diese nicht klassifiziert werden können. So das alle synthetischen und nur 5.359 echte Dokumenten verwendet werden.

Da die Dokumente als PDF-Datei vorliegen, welches von den verwendeten Modellen nicht verarbeitet werden kann, müssen diese zunächst in einseitige Bilder im JPG-Format konvertiert werden. Das erfolgt mittels *pdf2image* [4]. Um die Seitenzahl nachvollziehbar zu machen wird diese als Appendix im Dateinamen festgehalten. Des Weiteren erfordern die Modelle eine spezifische Ordnerstruktur, in der die Bilder nach Klassen organisiert sind. Diese Struktur sieht folgendermaßen aus: Abbildung 3.2.

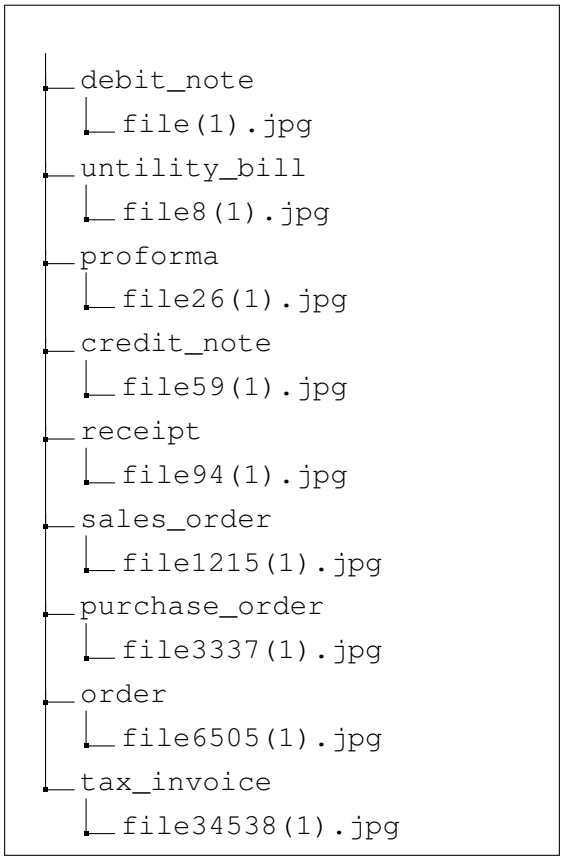

Abbildung 3.2: Ordnerstruktur nach Klassen

Um das großen Ungleichgewicht des Datensatzes auszugleichen, wird dieser durch die zufällige Wahl jedes zehnte *tex_ invoice* und jedes fünfte *order* angepasst. Dadurch verbleiben 7285 *tex_ invoice* und 5606 *order*. Der beschnittene Datensatz wird in Trainings- , Validierungs- und Testdatensatz aufgeteilt, zu einem Verhältnis von 702010.

Die normale Eingangsgröße von einem InceptionV3 Modells beträgt 299x299 Pixel und die eines VGG-16 Modell lediglich nur 244x244, die geringe Auflösung und das Quadratischen Formats eignen sich hingegen schlecht um Geschäftsdokumente zu Klassifizieren, weshalb sie das Seitenformat und eine Lesbarkeit beibehalten sollten. Die Dokumente haben nach dem umwandeln eine Größe von 2339 x 1654 Pixel. Da diese Auflösung unnötig groß erscheint, werden die Dokumente an den Seiten ungefähr auf 1122x846 Pixel halbiert, was für eine Pixel Minimierung von 75.46% sorgt. Nach der Verkleinerung sind die Inhalte noch sehr gut lesbar und die kommende Verarbeitungszeit der Modelle wird erheblich reduziert.

3.1.3 Modellanpassungen für die Dokumentenklassifizierung

Convolutional Neural Networks haben sich in den letzten Jahren als äußerst performant für Klassifikationsaufgaben erwiesen [15]. In diesem Zusammenhang stechen besonders das VGG 16 und das InceptionV3 Modell hervor, welche speziell für diese Aufgaben entwickelt wurden [34] [39].

Diese beiden Modelle wurden gewählt, um verschiedene Ansätze der Feature-Extraktionen zu vergleichen. Das VGG 16 Modell ist aufgrund siner tiefe Netzstruktur besonders gut darin, hierarchische Features lernen, was sich als vorteilhaft für das erkennen komplexer Features und Strukturen erweist [34]. Im Gegensatz dazu verwendet das InceptionV3 Modell sogenannte Inception Module, wodurch sowohl feine als auch großflächige Muster effektiv erkannt werden können [39].

Da die ursprünglichen Architekturen für allgemeine Bilderkennung mit deutlich kleineren Eingabedimensionen und einer höheren Anzahl von Klassen konzipiert wurden, muss sowohl das VGG16 als auch das InceptionV3 Modell für die Klassifizierung von Geschäftsdokumenten angepasst werden.

3.1.3.1 VGG 16

Das VGG 16 Modell besteht aus 16 gewichteten Layern, die hauptsächlich auf Convolutional Layer mit kleinen 3x3-Filter max-pooling Layern basiert. Dadurch werden tiefe Features in den Bildern extrahiert.

Für die Klassifizierung von Geschäftsdokumenten wurde der Input Layer so angepasst, dass er RGB-Bilder der Größe 1122x846 Pixel verarbeiten kann. Die Standard Eingaberöße des VGG 16 Modells hat eine Auflösung von 224x224 Pixeln. Um den hohen Informationsgehalt der Geschäftsdokumente nicht zu verlieren, wurde die Eingabe vergrößert.

Der Output Layer wurde ebenfalls angepassst, um die neun Dokumentenklassen korrekt abbilden zu können. Der Ursprünglichen auf 1000 Klassen ausgelegte Output Layer wurde dafür auf neun reduziert, während der Fully Connected Layer mit 4096 Neuronen unverändert bleibt.

Für das Training des Modells wurde der vorbereiteter Trainingsdatensatz mit 13.576 Dokumenten verwendet. Wegen der hohen Auflösung der Bilder wurde eine lange Trainingszeit erwartet, weshalb die Anzahl der Epochen auf fünf beschränkt wurde. Nach jeder Epoche wird das Modells mithilfe eines Validierungsdatensatzes getestet. Dabei werden Metriken wieAccuracy, Loss verwendet um die Performance des Modells zu evaluieren. Die Traingingszeit betrug trotz der geringen Anzahl an Epochen 154 Minuten. Mithilfe des Testdatensatzes wurde auf den trainierten Modell eine Konfusions Matrix erstellt, sowohl wurde für jede Klasse der F1-Score, die Precision und der Recall ermittelt.

3.1.3.2 InceptionV3

Das InceptionV3 Modell wurde ursprünglich für die Klassifizierung von Bildern mit einer Eingabegröße von 299x299 Pixeln entwickelt und enthält mehrere Inception-Module. Innerhalb dieser Module werden parallel mehrere Convolution Layer unterschiedlicher Filtergrößen angewenden, um verschiedenartige Merkmale zu extrahieren.

Wie auch beim VGG 16 Modell wurde der Input Layer so modifiziert, dass RGB-Bilder mit einer Größe von 1122x846 Pixeln verarbeitet werden können. Da der Orginale Output Layer von InceptionV3 einen Output von 1000 Klassen hat, wurde dieser ebenfalls so angepasst, dass er die neun Klassen von Geschaftsdokumenten korrekt abbilden kann. Die geschah indem die Anzahl der Neuroen im letzten Fully Connected Layer auf die Anzahl der Dokumentenklassen reduziert wurde, während der vorherige Layer mit 2048 Neuronen unverändet blieb.

Auch für das InceptionV3 Modell wurder der selbe vorverarbeitete Trainingsdatensatz wie für das VGG 16 Modell verwendet. Obwohl InceptionV3 effizienter trainiert, führte die hohe Bildauflösung zu einer langer Traingszeit, weshalb die Anzahl der Epochen ebenfalls auf fünf festgelegt wurde. Das training wurde innerhalb von 111 Minuten absolviert. Die Modelleistung wurde ebenfalls nach jeder Epoch mit Accuracy und Loss bewertet. Nach dem Training wurden, wie beim VGG 16 Modell, die Konfusions Matrix, der F1-Score, die Precision und der Recall auf Basis des Trainingsdatensatzes erstellt.

3.1.4 Anwendung von XAI

Um die Vorhersagen der verwendeten Modelle interpretierbar zu machen wurden Grad-CAM, Integrated Gradient und LRP implementiert. Diese Methoden wurden mithilfe *Captum-Bibliothek* von *Facebook* umgesetzt. Es wurden zufällig ausgewählte Dokumente der jeweiligen Klasse verwendet.

3.1.4.1 Captum Framework

Captum ist eine Open-Source-Bibliothek die für die Interpretierbarkeit von PyTorch-Modell entwickelt wurde. Dabei bietet Captum eine einheitliche Schnittstelle zur Anwendung verschiedener XAI-Methoden, wie Grad-CAM, Integrated Gradients und LRP [14].

3.1.4.2 Grad-CAM

Grad-CAM visualisiert die Bereiche von einem Bild, die am stärksten zur Vorhersage eines Modells beitragen. Die Methoden nutzt die Gradienten zur Vorhersage des letzten Convolutional Layer, um eine Heatmap zu erstellen. Dabei wurde Grad-CAM in Captum durch den *LayerGradCam-Wrapper* implementiert. Der Gradient der Zielklasse wurde über die letzte Schicht berechnet, beim VGG 16 ist die der letzte Convolutional Layer, beim InceptionV3 das letzte Inception Modul. Durch das übereinaderlegen der erzeugten Heatmaps auf die ursprünglichen Eingabebilder wurden die relevante Bereiche hervorgehoben.

3.1.4.3 Integrated Gradients

Integrated Gradients berechnet die relevanten Eingabeattribute einer speziellen Vorhersagen. Dafür wird der Gradient über einen einem Pfad zwischen der Eingabe und einem Baseline Bild bewegt integriert. Das Ergebnis zeigt die wichtigsten Pixel, die zur Klassifikationsentscheidung beigetragen haben. Diese Methode wurde durch den *IntegratedGradients-Wrapper* von Captum implementiert. Standardmäßig werden 50 Schritte zwischen der Baseline und der Eingabe gewählt. Aufgrund des hohen GPU-Speicherbedarfs bei den hochauflösenden Dokumentenbildern wurde die Anzahl der Schritte auf 12 reduziert. Für jede Berechnung wurde die entsprechende Zielklasse angegeben.

3.1.4.4 LRP

LRP wird verwendet, um die Relevanz jedes Neurons in Bezug auf eine spezielle Vorhersage zu bestimmen. Dabei wird rückwärts durch das Netz propagiert und bei jedem Schritt die Relevanz der Eingaben berechnet. Dieses Vorgehen wird bis zum Input Layer wiederholt, sodass eine detaillierte Rückverfolgung der Relevanzen möglich ist. Die Implementierung erfolgte über den *LRP-Wrapper* von Captum. Hier wird nur die Zielklasse angegeben und die resultierernden Relevanzen als Heatmaps visualisiert.

3.1.5 Bewertung der XAI-Methoden

Zur Bewertung der Aussagekraft der erzeugten Heatmaps wurde das Verfahren der regionalen Perturbation verwendet. Die Bildregionen wurden in ein Raster unterteilt, und die Klassifikantionsleistung nach jeder Störung wurde mithilfe von AOPC zu quantifiziert.

3.1.5.1 Implementierung der Perturbation

Für die Implementierung der Perturbation wurde ein Raster mit einer Seitenlänge von 25 Pixeln gewählt. Im ursprünglichen Paper wurde ein Raster mit 9x9 Pixeln verwendet. Da jedoch die Dokumente in diesem Experiment eine Auflösung von 1122x846 Pixeln haben, würde die Verwendung kleinerer Raster zu einer deutlich längeren Verarbeitungszeit führen. Ein 25 Pixelraster bietet dabei eine angemessene Größe um Zeilen und Zellen in den Geschäftsdokumenten zu Markieren. Während der Perturbation werden die ersten 500 relevanten Zellen des Rasters entfernt, die Reihenfolge wird dabei durch die Wichtigkeit der Regionen bestimmt. Häufig werden nur die ersten 100 relevanten Zellen verwendet, da die Features von Geschaftsdokumenten in der Regel weiter verteilt liegen, wurden die ersten 500 releventen Zellen entfernt. Anschließend wir der AOPC berechnet und der Kurvenverlauf visualisiert.

4 Resultate

In dieser Arbeit wurden das VGG 16 und das InceptionV3 Modell darauf trainiert, Geschäftsdokumente aus dem DocILE-Datensatz in neun verschiedene Klassen zu kategorisieren. Ziel der Experimente war es, die Leistung dieser Modelle zu bewerten und durch die XAI Methoden Grad-CAM, Integrated Gradients und LRP eine nachvollziehbare Erklärungen für ihre Entscheidungen zu erzeugen.

Zusätzlich wurde eine Analyse der Modellrobustheit durch eine regionale Perturbation durchgeführt. Dabei werden die wichtige Bildbereiche, basierend auf den XAI Methoden, schrittweise verändert wurden. Die Effekte dieser Veränderungen auf die Klassifikationsgenauigkeit wurden mit Hilfe der AOPC-Metrik evaluiert.

4.1 Modellperformance

4.1.1 VGG 16 Trainingsergebnisse

Im Verlauf des Trainings des VGG 16 Modells stieg die Trainingsgenauigkeit kontinuierlich von 93,93% in der ersten Epoche auf 99,05% in der letzten Epoche. Diese stetige Verbesserung zeigt, dass das Modell während des Trainings immer besser darin wurde, die Trainingsdaten korrekt zu klassifizieren. Es zeigten sich Schwankungen in der Validierungsgenauigkeit. Ab der dritten Epoche stabilisierte sich die Validierungsgenauigkeit bei etwa 98,63% und erreichte schließlich einen Höchstwert von 99,07% in der fünften Epoche. Dies ist grafisch in Abbildung 4.1 dargestellt.

Zu Beginn des Trainings zeigte sich ein Loss von 0,19, welcher nach der fünften Epoche auf 0,034 sank. In Bezug auf den Validierungsloss sowie die Validierungsgenauigkeit zeigten sich in der zweiten Epoche Schwankungen, diese stabilisierten

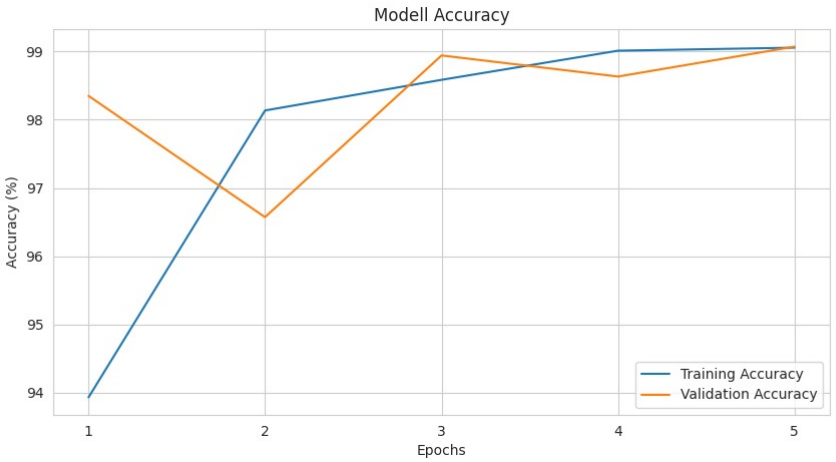

Abbildung 4.1: VGG 16 Modellgenauigkeit nach 5 Epochen

sich nach der dritten Epoche. Zum Abschluss des Trainings nach fünf Epochen zeigte sich ein Validierungsloss von 0,031. Die Ergebnisse sind in Abbildung 4.2 dargestellt.

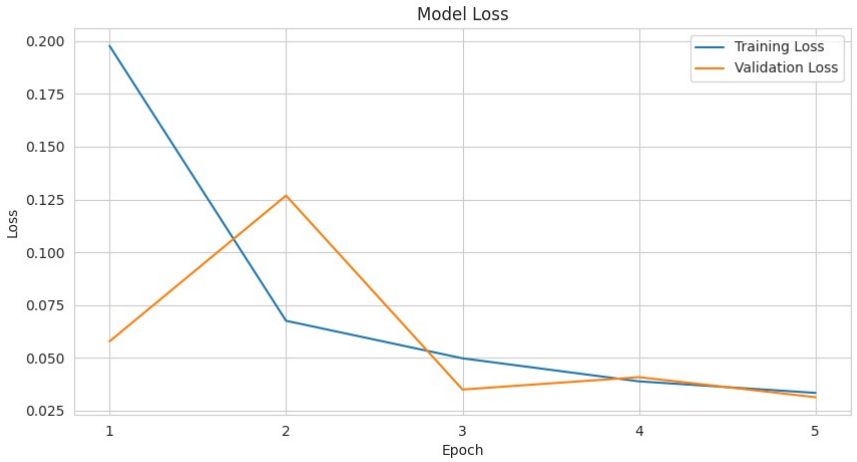

Abbildung 4.2: VGG 16 Modell Loss nach 5 Epochen

4.1.2 VGG 16 Performance

Die Leistung des VGG16-Modells wurde durch die Analyse der Konfusionsmatrix auf Basis der Trainingsdaten beurteilt. Ein Blick auf die Hauptdiagonale der Ma-

trix zeigt, dass die dargestellten Werte häufig von der Hauptdiagonalen abweichen. Sichtbar ist, dass die Klassen*proforma*, *credit_ note*, *utility_ bill* und *debit_ note* eine geringere Genauigkeit aufweisen, die weiteren Klassen zeigen eine im Vergleich höhere Genauigkeit. Dies ist in Abbildung 4.3 grafisch aufgezeichnet.

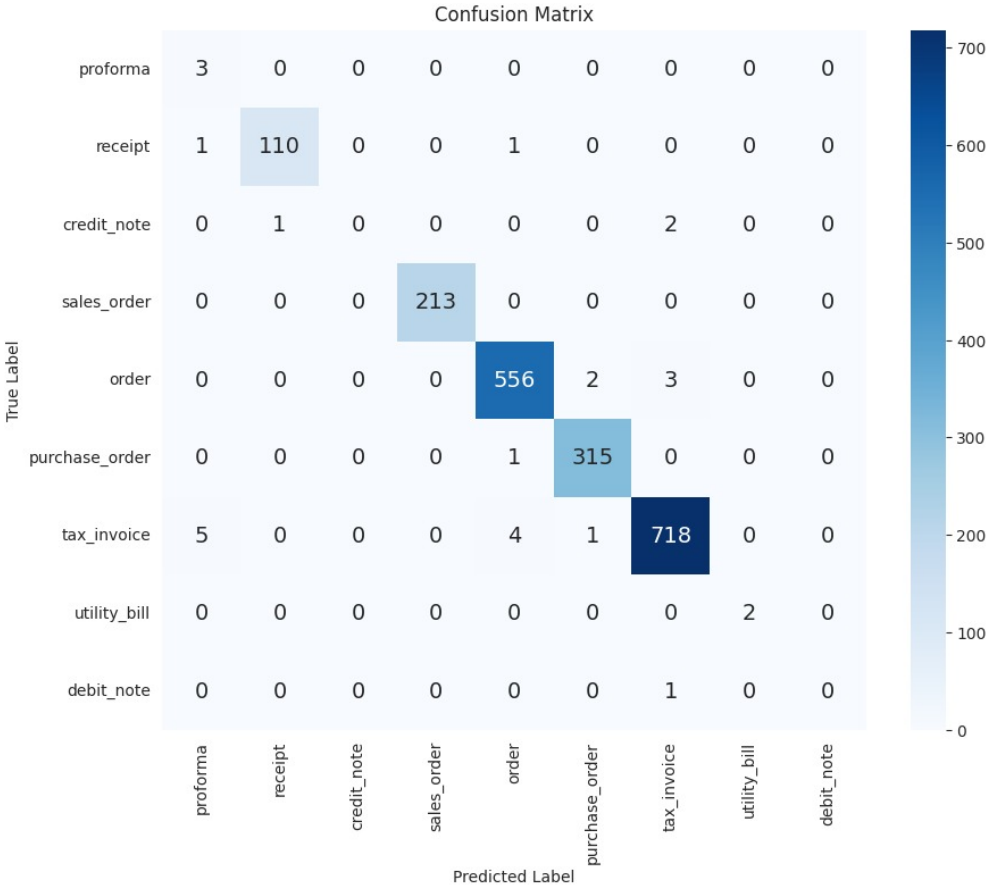

Abbildung 4.3: Confusions Matrix vom vom trainierten VGG 16

Basierend auf der Konfusionsmatrix wurde die Performance der einzelnen Klassen ermittelt. Die Klassen *receipt*, *sales_ order*, *order*, *purchase_ order* und *tax_ invoice* zeigten eine sehr hohe Performance. Besonders hervorzuheben ist, dass die zwei *utility_ bill* Dokumente korrekt klassifiziert wurden. Die Dokumente *utility_ bill* wurden bei einer Ausgangssumme von zwei Dokumenten korrekt klassifiziert. Die Dokumente *credit_ note* und *debit_ note* wurden jeweils falsch klassifiziert. Weiterhin ist zu beachten, dass einige Dokumente anderer Klassen fälschlicherweise als *proforma* klassifiziert wurden. Die Perfomance wird in Abbildung 4.4 darge-

stellt. Im Gegensatz dazu wurde die Klassifikation der *credit_note* und *debit_note* Dokumente durch das Modell jeweils falsch durchgeführt, was zu einer schlechten Performance dieser Klassen führte. Auch wenn alle *proforma* Dokumente korrekt klassifiziert wurden, zeigt sich insgesamt eine schwächere Performance für diese Klasse. Ursächlich für die als schlechter bewertete Perfomance ist, dass einige Dokumente anderer Klassen fälschlicherweise als *proforma* Dokumente klassifiziert wurden.

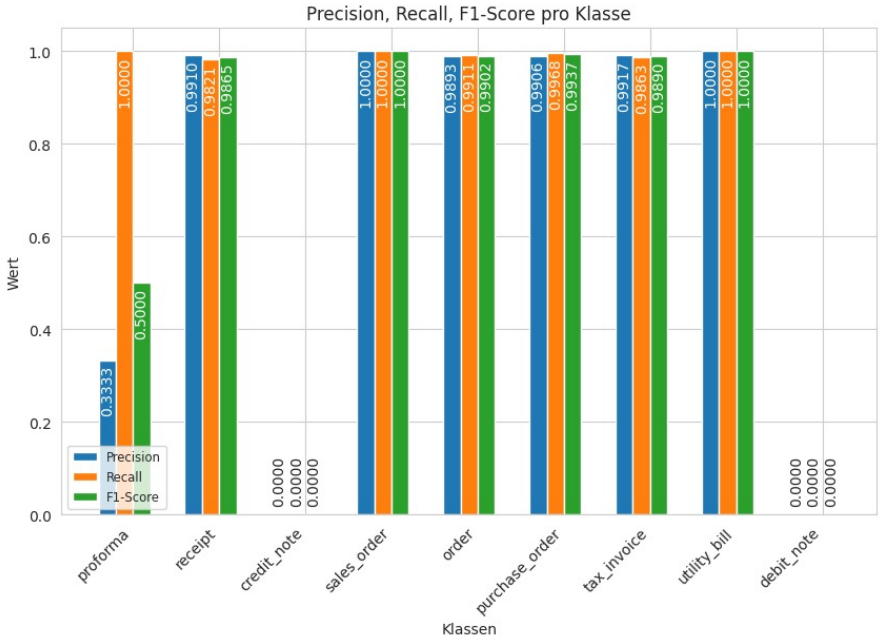

Abbildung 4.4: VGG 16 Performance

4.1.3 InceptionV3 Trainingsergebnisse

Das InceptionV3-Modell zeigte im Verlauf des Trainings eine hohe Performance. Bereits nach der ersten Epoche erreichte das Modell eine Genauigkeit von 98,53%, die mit einem leichten Rückgang in der vierten Epoche auf schließlich 99,31% nach der fünften Epoche anstieg. Die Validierungsgenauigkeit verlief insgesamt stabil und zeigte keine signifikanten Schwankungen. Sie stieg von 99,05% in der ersten Epoche auf 99,56% in der letzten Epoche an. Die Ergebnisse sind grafisch in Abbildung 4.5 dargestellt.

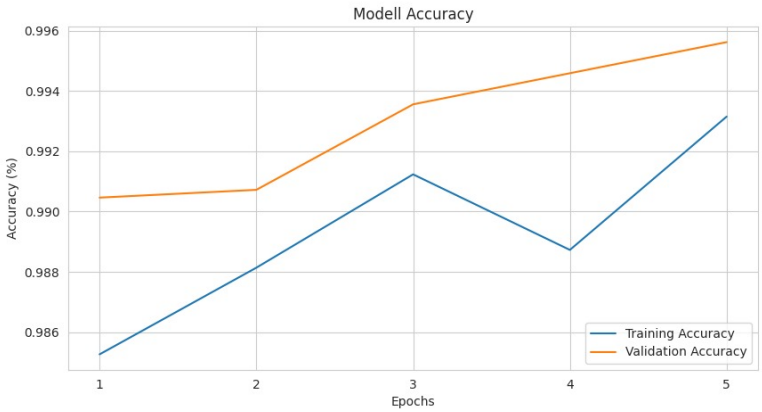

Abbildung 4.5: InceptionV3 Modellgenauigkeit nach 5 Epochen

Der Loss-Wert entwickelte sich ähnlich. Nach der ersten Epoche lag der Trainings-Loss bei 0,057, sank jedoch kontinuierlich bis auf 0,025 in der fünften Epoche ab. Der Validierungs-Loss betrug zu Beginn 0,033 und reduzierte sich ebenfalls, nach der letzten Epoche betrug er nur noch 0,015 Die Werte sind in Abbildung 4.6 dargestellt.

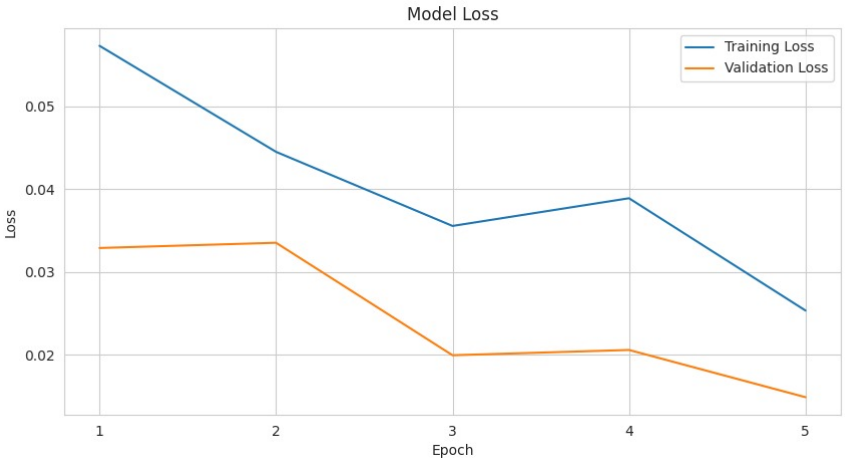

Abbildung 4.6: InceptionV3 Modell Loss nach 5 Epochen

4.1.4 InceptionV3 Performance

Die Performance des InceptionV3-Modells wurde anhand der Konfusionsmatrix evaluiert, die auf den Ergebnissen des Trainingsdatensatzes basiert. Es zeigte sich hohe Perfomancewerte für die Klassen *receipt*, *sales_order*, *order*, *purchase_order* und *tax_invoice*. Weiterhin wurde die Klasse *credit_note* korrekt klassifiziert. Die Klassen *proforma* und *debit_note* wurden falsch zugeordnet. Die Ergebnisse sind in der Abbildung Abbildung 4.7 zu sehen.

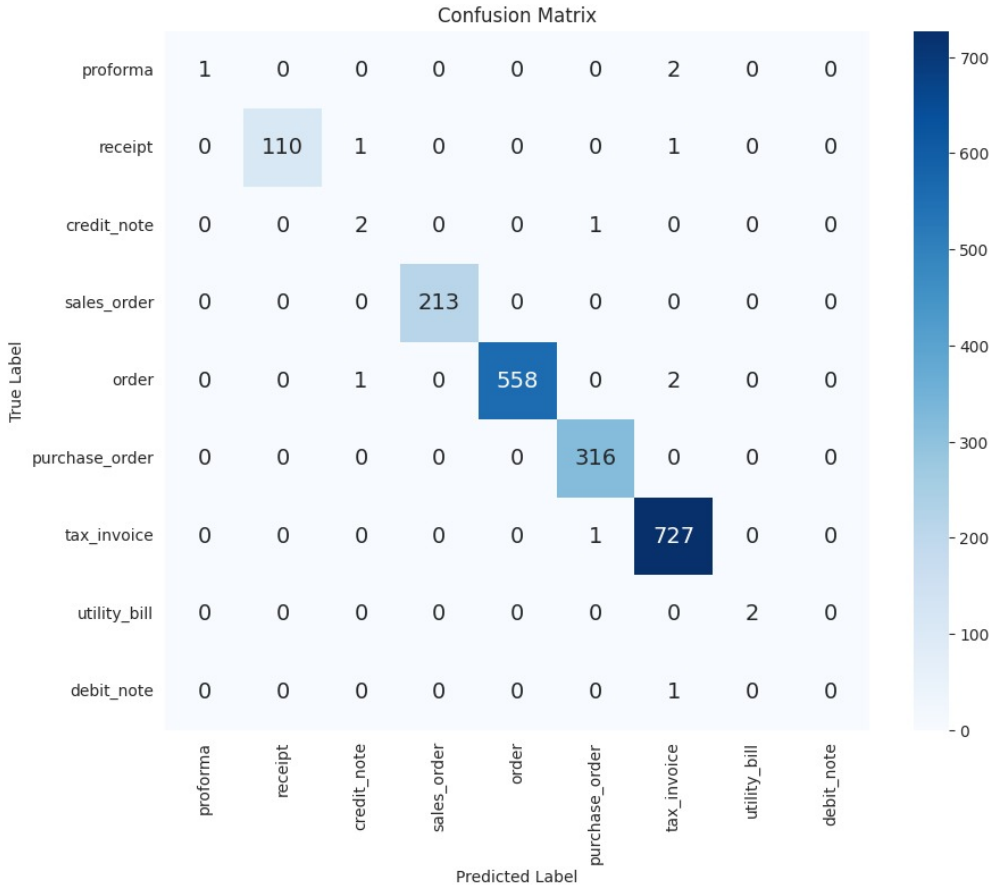

Abbildung 4.7: Confusions Matrix vom trainierten InceptionV3 Modell

Die detaillierte Analyse der Klassenperformance zeigt, dass neben den zum Großteil richtig klassifizierten Klassen *receipt*, *sales_order*, *order*, *purchase_order* und *tax_invoice* auch die Klasse *credit_note* eine hohe Genauigkeit aufweist. Der Klasse *proforma* wurden keine falschen Dokumente zugeordnet, allerdings wurden zwei

von drei *proforma* Dokumenten falsch klassifiziert, was den niedrigen Recall Wert für diese Klasse erklärt. Die Klasse *debit_note* wurde ebenfalls einer falschen zugeordnet, was zu einer insgesamt schlechten Performance führte. Grafisch ist dies in Abbildung 4.8 dargestellt.

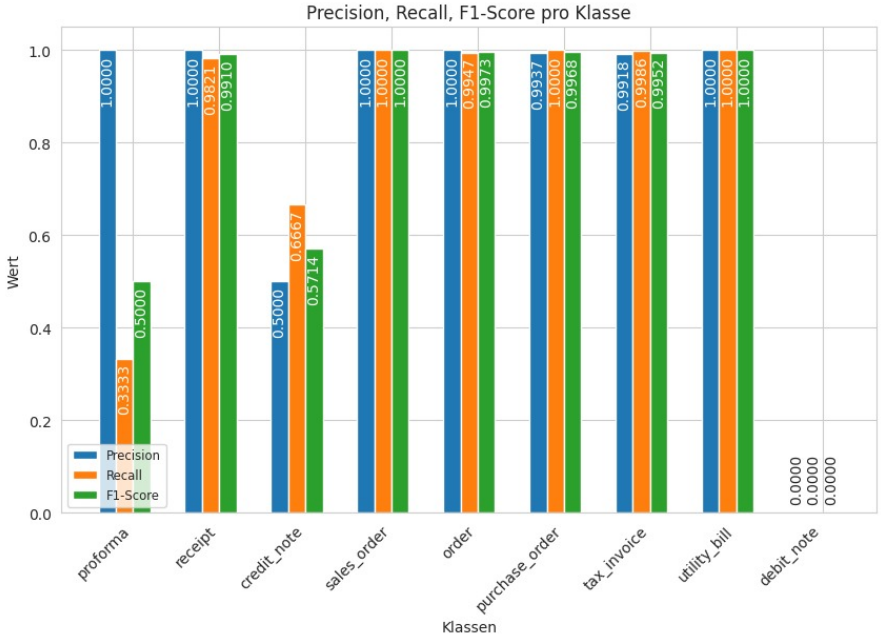

Abbildung 4.8: InceptionV3 Performance

4.2 Ergebnisse der XAI-Methoden

In diesem Abschnitt werden die Ergebnisse der XAI Methoden präsentiert, die auf die VGG 16 und InceptionV3 Modelle angewendet wurden. Ziel der Untersuchung war es, die von den Modellen verwendeten Bildbereiche zu visualisieren und deren Relevanz für die Klassifikationsentscheidung zu bewerten. Dies wurde durch die Anwendung von Grad-CAM, Integrated Gradients und LRP erreicht. In Abbildung 4.9 sind zufällig ausgewählte Geschäftsdokumente dargestellt.

4.2.1 Grad-CAM

Das VGG 16 Modell zeigt bei der Anwendung von Grad-CAM eine Tendenz, breite und umgebende Bildbereiche als relevant zu markieren (Abbildung 4.10). In den Klassen *sales_ order*, *receipt*, *utility_ bill*, *credit_ note*, *purchase_ order* und *debit_ note* werden große, zusammenhängende Flächen hervorgehoben. Auffällige Details, wie die Textblöcke in *receipt*, werden dagegen nicht als relevant für die Klassifikation angesehen.

InceptionV3 zeigte in der Grad-CAM Heatmap eine stärkere Fokussierung auf detaillierte Bildbereiche, wie Abbildung 4.11 zeigt. Besonders in den Klassen *receipt*, *purchase_ order*, *sales_ order* und *tax_ invoice* werden klar abgegrenzte und oft kleinere Regionen als relevant hervorgehoben. Bei den Klassifikationen von *order*, *utility_ bill*, *credit_ note* und *debit_ note* zeigt InceptionV3 wiederum ein anderes Verhalten, indem es konträr zu den anderen Klassen agiert und spezifische Details nicht so stark betont. Stattdessen identifiziert das Modell in diesen Fällen eher breitere Bereiche als relevant.

4.2.2 Integrated Gradients

Das VGG16 Modell zeigt eine diffusere Verteilung der relevanten Bildbereiche, wie in Abbildung 4.12 dargestellt ist. Besonders bei den Klassen *order*, *credit_ note*, *debit_ note* und *utility_ bill* sind die Relevanzbereiche weniger klar abgegrenzt und verteilen sich auf größere Teile des Bildes.In einigen Klassen, wie *sales_ order* und *receipt*, zeigt das VGG 16 Modell jedoch eine klarere Fokussierung auf bestimmte kleine relevante Bildbereiche.

Das InceptionV3-Modell zeigt klarer abgegrenzte Relevanzbereiche in den Integrated Gradients Heatmaps, wie in Abbildung 4.13 dargestellt. Besonders in den Klassen *receipt*, *purchase_ order* und *tax_ invoice* sind klar abgegrenzte, helle Regionen zu erkennen.

4.2.3 LRP

Bei dem VGG 16 Modell zeigen die LRP Heatmaps insgesamt sehr geringe Relevanzwerte über fast alle Klassen hinweg an. Lediglich bei den Klassen *sales_ order* und *credit_ note* ist eine leichte erkennbare Relevanz zu erkennen, diese sind schwach und diffus. Bei den anderen Klassen ist kaum eine nennenswerte Relevanz sichtbar. Grafisch ist dies in Abbildung 4.14 dargestellt.

Die LRP Heatmaps für InceptionV3 zeigen eine schwache Relevanzdarstellung in den meisten Klassen, wie in Abbildung 4.15 dargestellt. Einzig bei *sales_ order* sind deutlichere Relevanzbereiche zu erkennen. In den Klassen *receipt*, *purchase_ order* und *credit_ note* werden nur schwach ausgeprägte Bereiche leicht hervorgehoben.

4.3 Perturbation und AOPC

In diesem Abschnitt werden die Ergebnisse der regionalen Perturbation und der AOPC-Methode vorgestellt. Die Perturbationstechnik wurde angewendet, um sukzessive die am stärksten relevanten Bereiche der Heatmaps zu entfernen und die Auswirkungen auf die Modellvorhersagen zu analysieren. Mit AOPC wurde die Sensitivität des Modells gegenüber diesen Störungen quantifiziert.

4.3.1 Grad-CAM Perturbation und AOPC

Bei dem VGG 16 Modell beträgt der AOPC Wert der Klassen *receipt*, *credit_ note*, *sales_ order* und *purchase_ order* nahezu 0. Auch die Klassen *tax_ invoice* und *debit_ note* weisen sehr niedrige AOPC Werte auf (0,01). Die höchste Sensitivität zeigt sich in der Klasse *proforma* (AOPC 0,22), *utility_ bill* (0,19) und *order* (0,18). Grafisch ist dies in Abbildung 4.16 dargestellt.

Bei dem InteptionV3 Modell ist sind hohe AOPC Werte zu sehen (Abbildung 4.17). Die Klassen *receipt*, *sales_ order*, *order*, *purchase_ order* und *tax_ invoice* von dem liegen die AOPC-Werte bei 0,60 oder höher. Die Klassen *proforma*, *credit_ note*, *utility_ bill* und *debit_ note* weisen hingegen schwächere AOPC Werte von 0,22 bis

zu 0,01 auf. Zu sehen ist, das beide Modelle für die Klasse *debit_ note* einen sehr niedrigen wert von 0.01 haben.

4.3.2 Integrated Gradients Perturbation und AOPC

Durch die Anwendung von Integrated Gradients auf das VGG 16 Modell ist eine größere Streuung der AOPC Werte über die verschiedenen Klassen hinweg zu beobachten (Abbildung 4.18). Besonders in den Klassen *utility_ bill* (AOPC 0,86), *tax_ invoice* (0,82), *receipt* (0,80) und *sales_ order* (0,72) weisen einen Hochen AOPC Wert auf. Für die Klasse *proforma* beträgt der AOPC Wert 0,42. Der AOPC Wert für *order* (0,29) ist relativ niedrig. Ein interessanter Punkt ist die Klasse *credit_ note*, bei der ein negativer AOPC-Wert von -0,02 auftritt.

Das InceptionV3 Modell zeigt insgesamt hohe AOPC Werte, insbesondere in den Klassen *purchase_ order* (0,90), *sales_ order* (0,89) und *receipt* (0,88) (Abbildung 4.19). Die Klasse *order* hat ebenfalls einen hohe AOPC Werte von 0.82. Jedoch zeigen die niedrigen Werte von *utility_ bill* (0.05) und *debit_ note* (0.02) auch, dass das Modell bei diesen Klassen Probleme hat. Auch die Klassen *proforma* (0,18), *credit_ note* (0,14) und *tax_ invoice* (0,29) zeigen moderate AOPC Werte.

4.3.3 LRP Perturbation und AOPC

Die AOPC Werte des VGG 16 Modells sind insgesamt, durch LRP identifizierten Bildbereiche, niedrig (Abbildung 4.20). Der höchste Wert ist in der Klasse *utilit_ bill* (0,63) zu erkennen. In den Klassen *proforma* (0,51) und *receipt* (0,40) haben ebenfalls moderate AOPC Werte. Für die Klasse *sales_ order* beträgt der AOPC Wert 0,00. Des Weiteren habe die Klassen *credit_ note* (-0,01) und *debit_ note* (-0,02) negative Werte.

Das InceptionV3-Modell zeigt im Vergleich eine höhere Sensitivität gegenüber den durch LRP identifizierten relevanten Bereichen (Abbildung 4.21). Besonders in den Klassen *receipt* (0,85), *sales_ order* (0,86), *purchase_ order* (0,86) und *order* (0,76) sind die AOPC Werte hoch, was auf eine präzise Identifikation relevanter Merkmale hindeutet. Interessanterweise zeigt die Klasse *credit_ note* mit 0,36 eine bessere Sensitivität im Vergleich zum VGG 16 Modell. In der Klassen *proforma* (0,23)

zeigt das InceptionV3 Modell eine moderate Sensitivität gegenüber der Perturbation, was auf eine geringere Abhängigkeit von den identifizierten Bildbereichen hindeutet. Der AOPC Wert von 0,02 der Klasse *debit_note* ist ebenfalls niedrig.

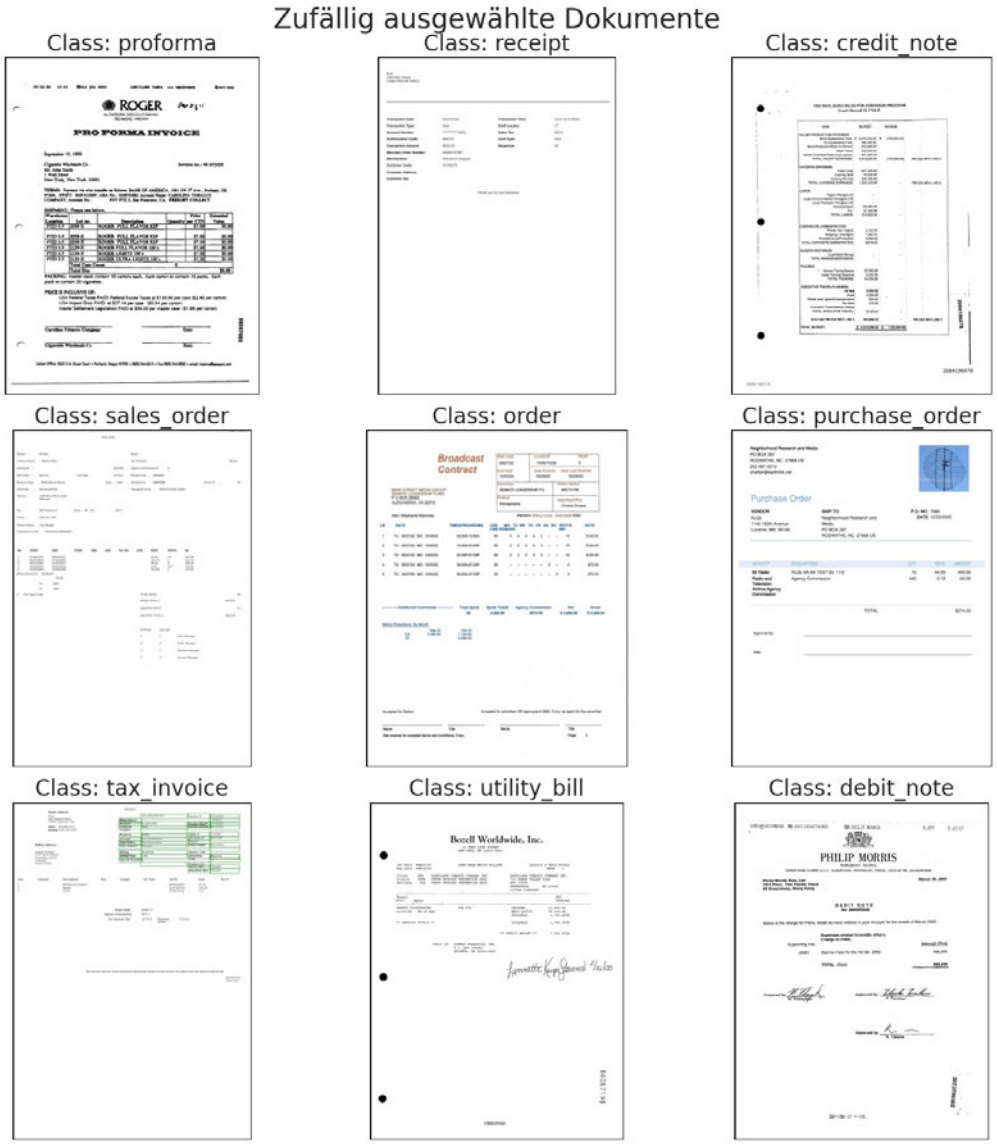

Abbildung 4.9: Zufällige Auswahl von Geschäftsdokumenten pro Klasse

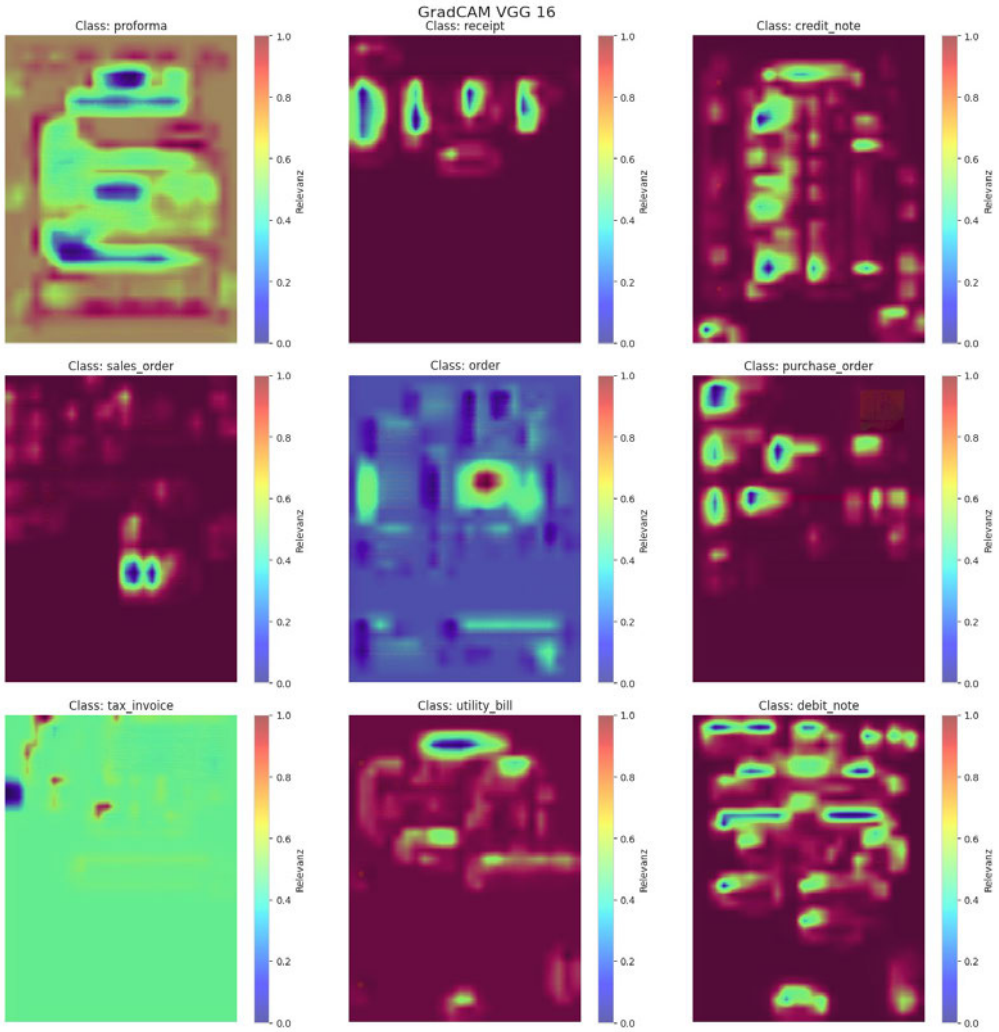

Abbildung 4.10: VGG 16 Grad-CAM Heatmap

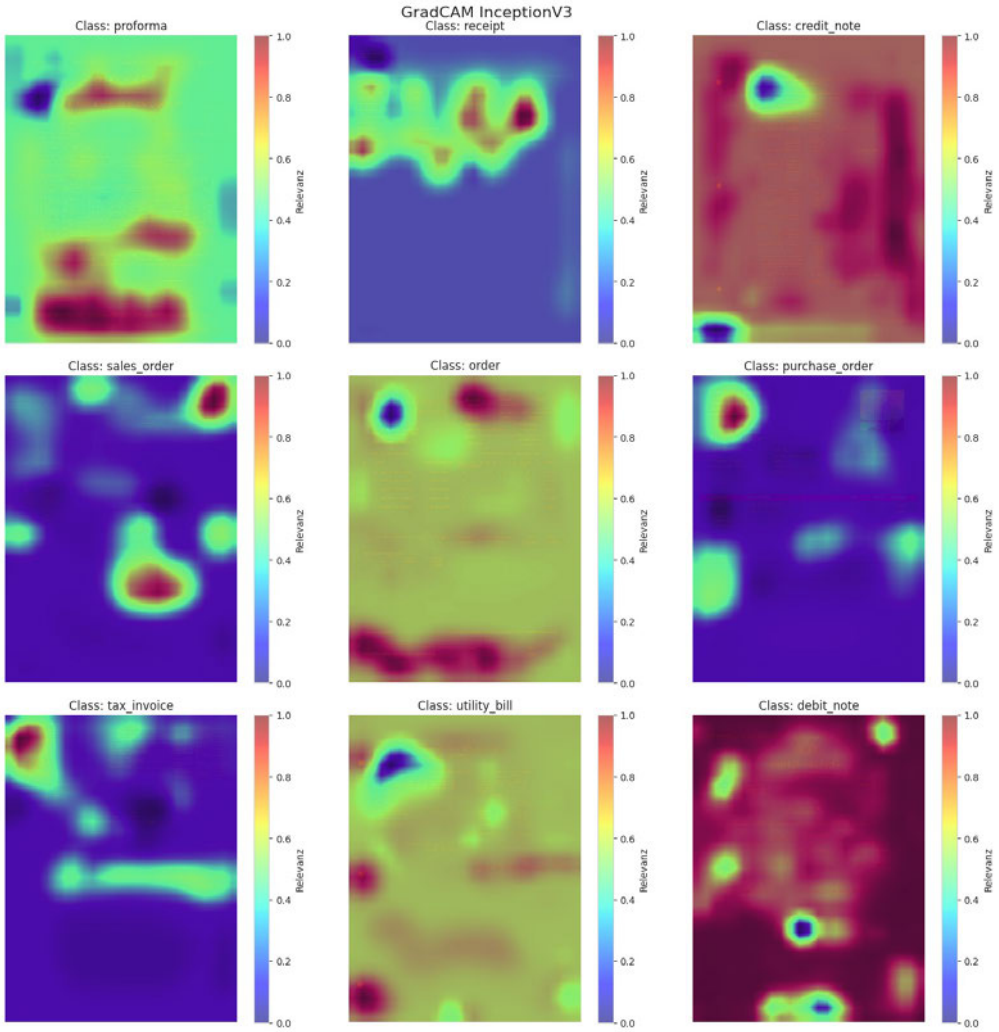

Abbildung 4.11: InceptionV3 Grad-CAM Heatmap

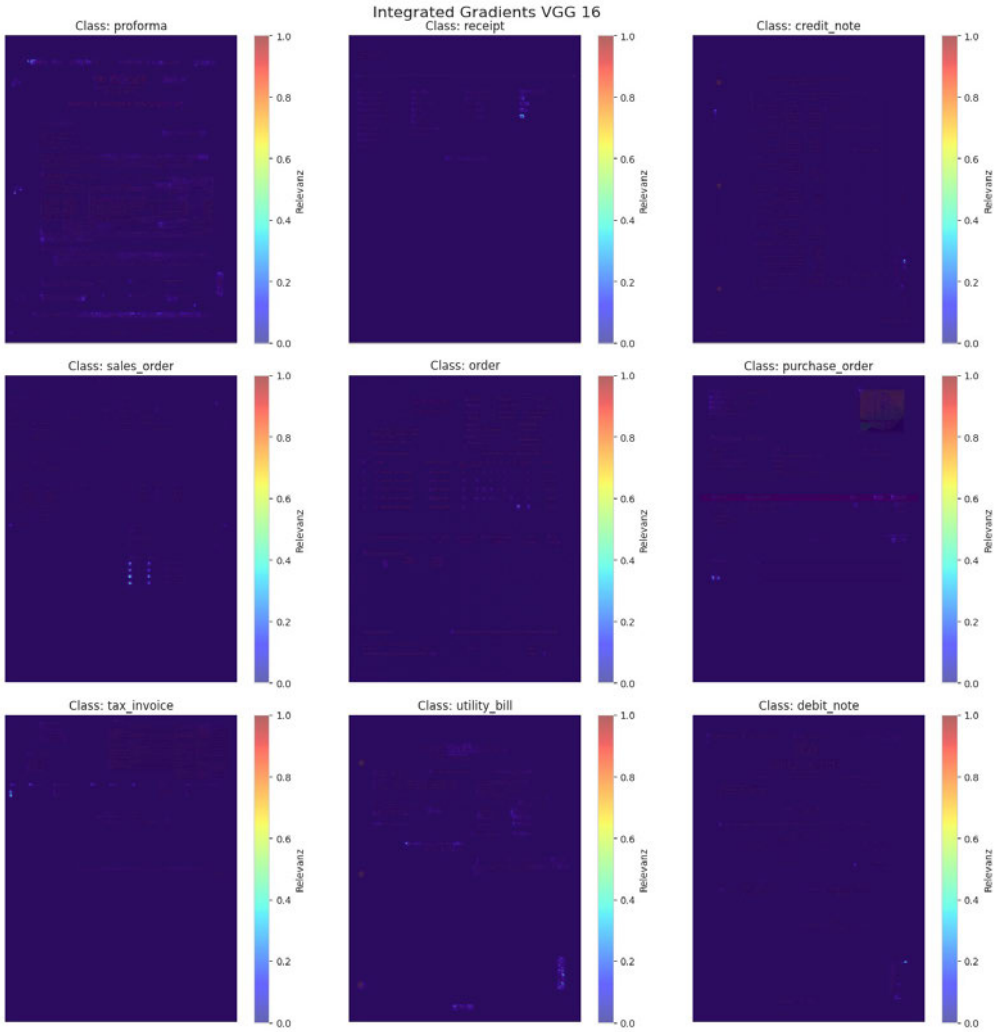

Abbildung 4.12: VGG 16 Integrated Gradients Heatmap

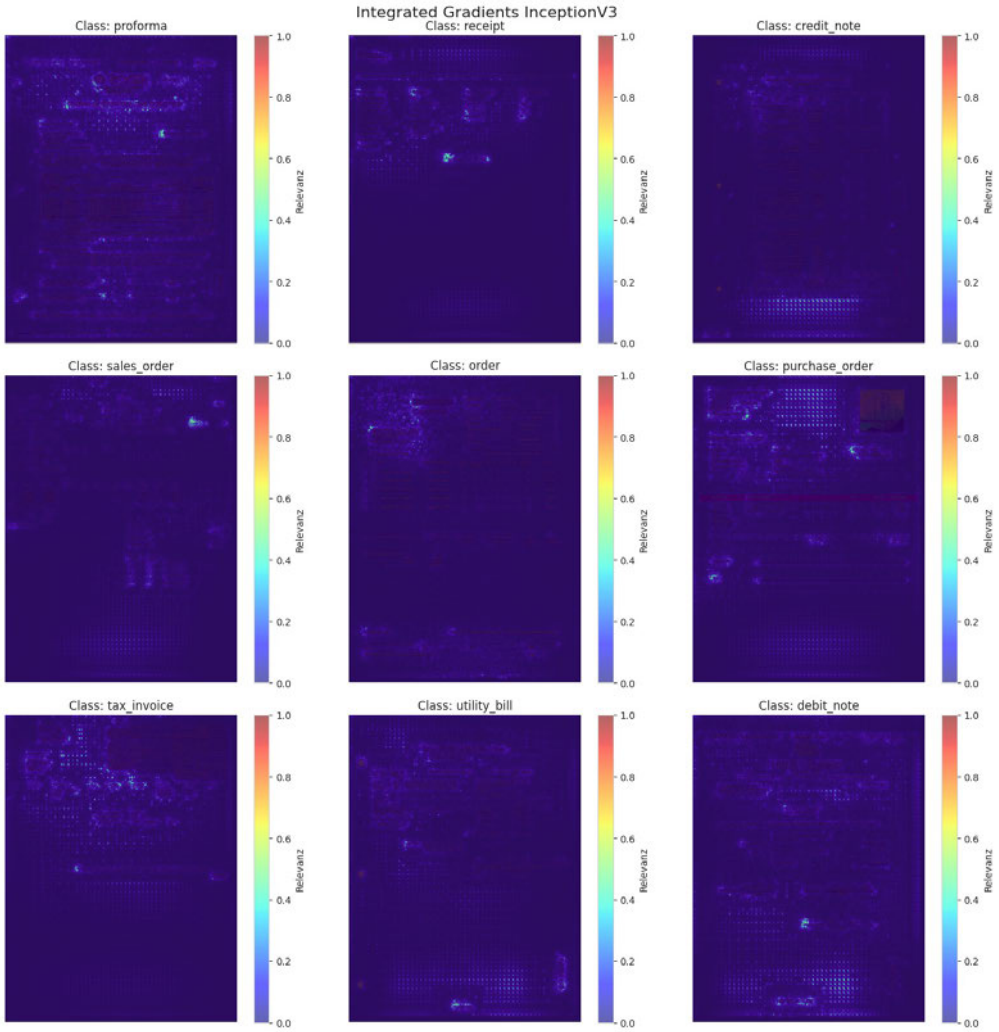

Abbildung 4.13: InceptionV3 Integrated Gradients Heatmap

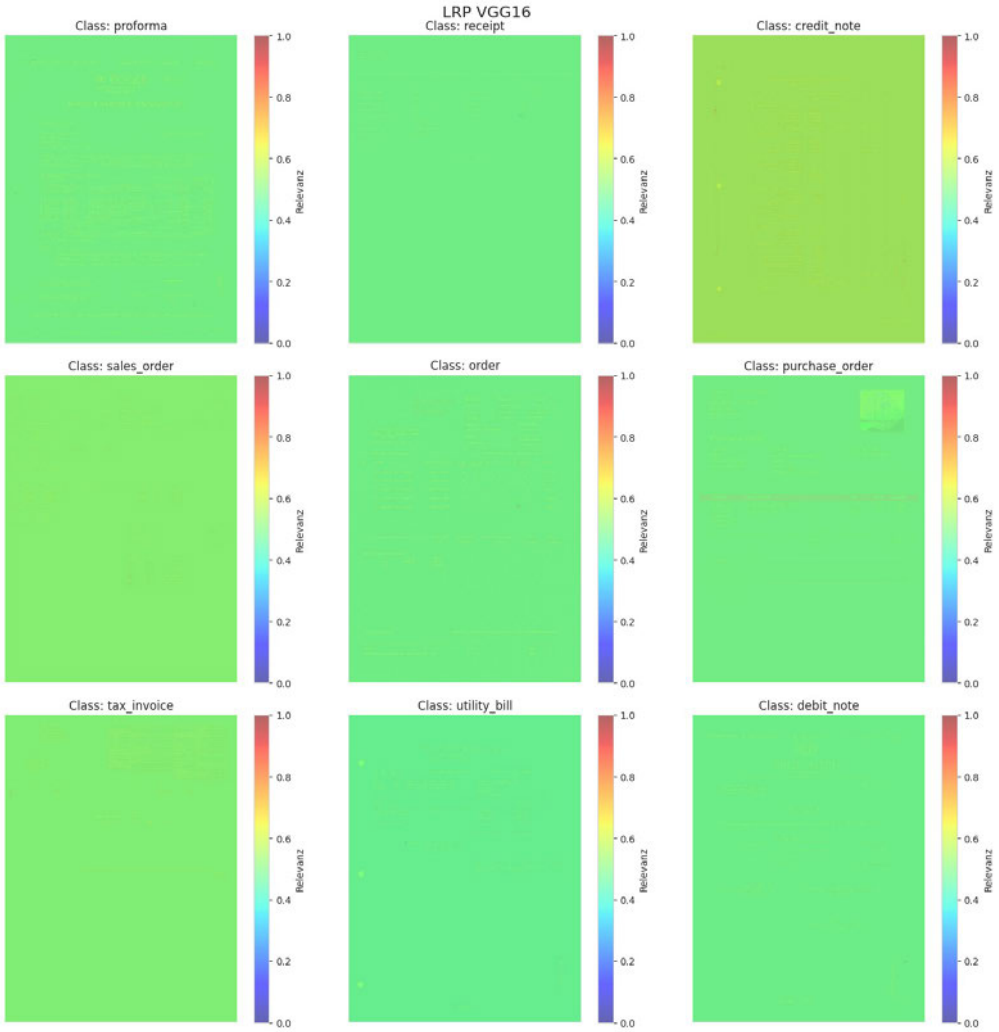

Abbildung 4.14: VGG 16 LRP Heatmap

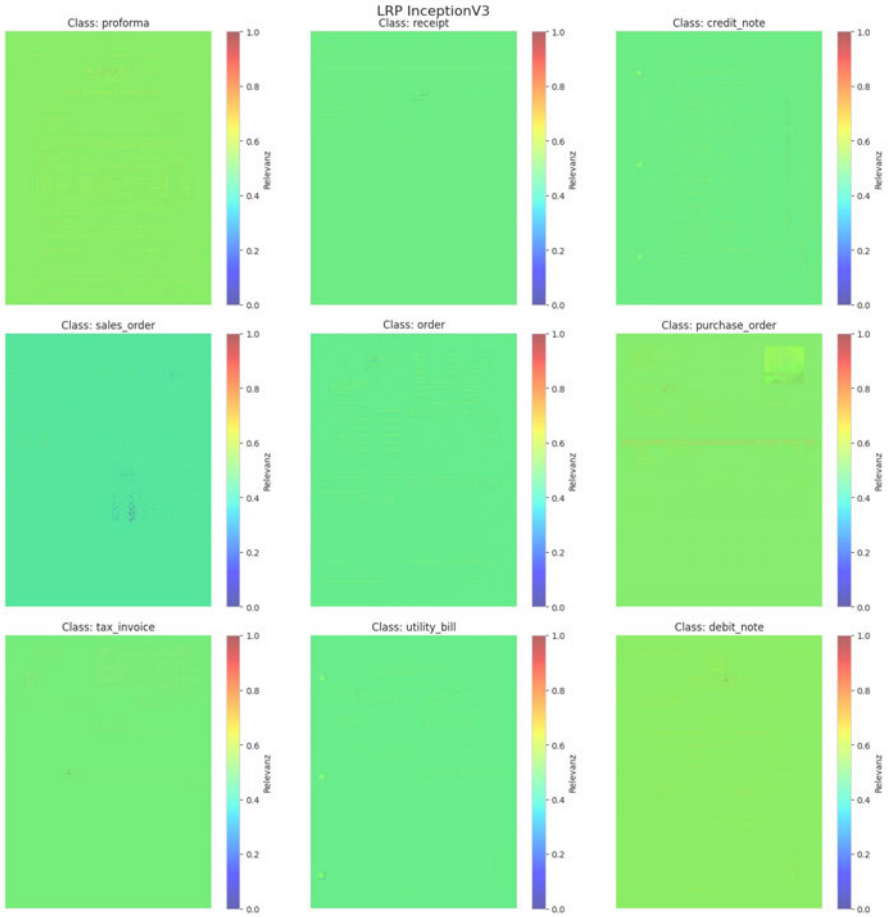

Abbildung 4.15: InceptionV3 LRP Heatmap

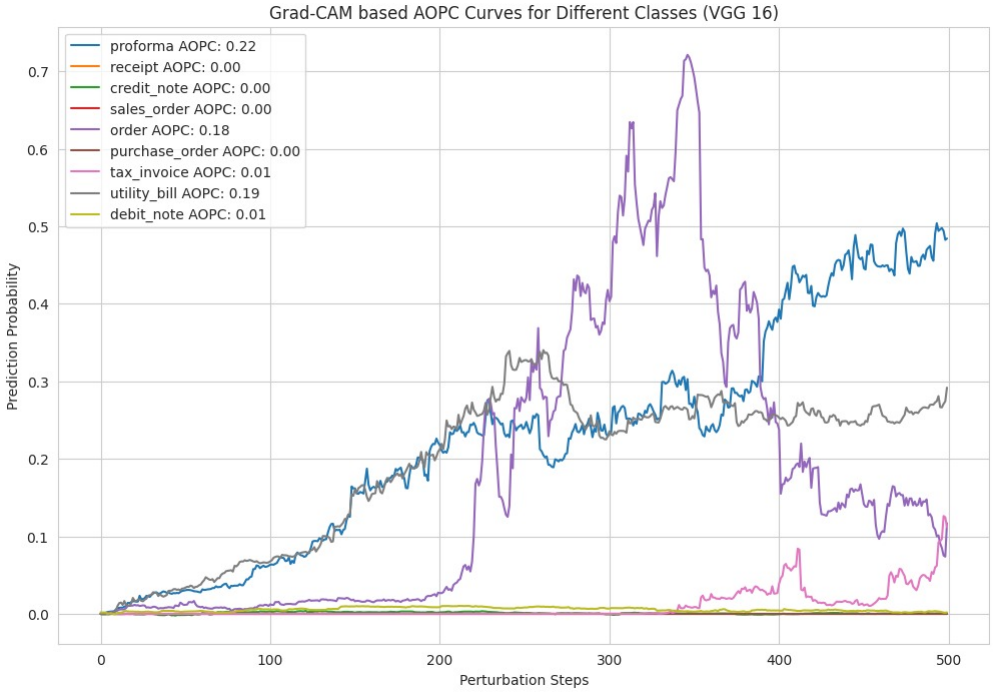

Abbildung 4.16: AOPC von Grad-CAM für das VGG 16 Modell

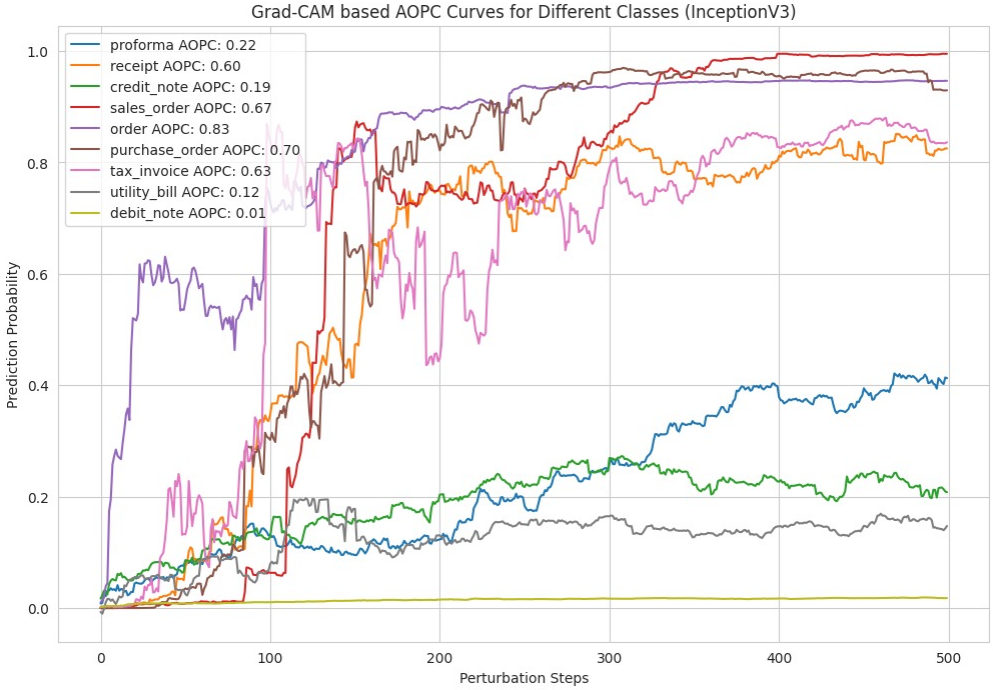

Abbildung 4.17: AOPC von Grad-CAM für das InceptionV3 Modell

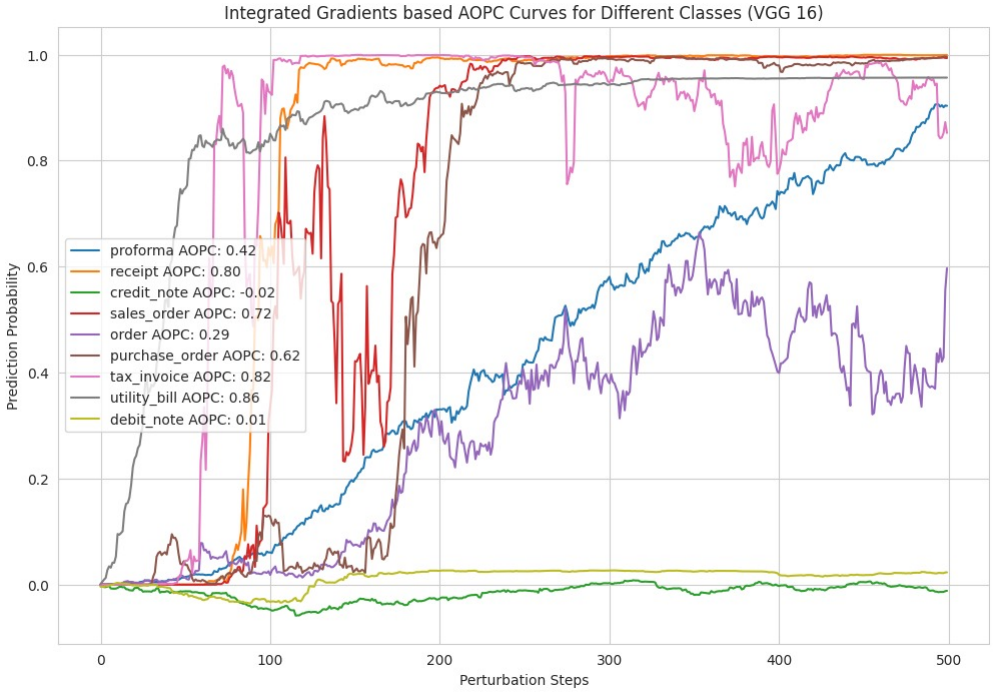

Abbildung 4.18: AOPC von Integrated Gradients für das VGG 16 Modell

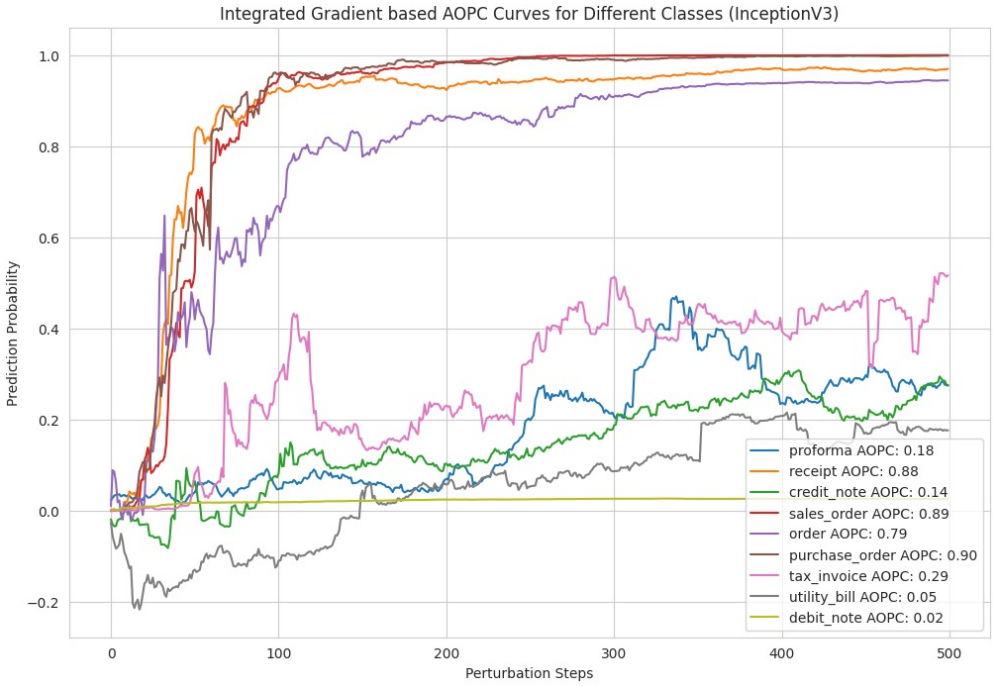

Abbildung 4.19: AOPC von Integrated Gradients für das InceptionV3 Modell

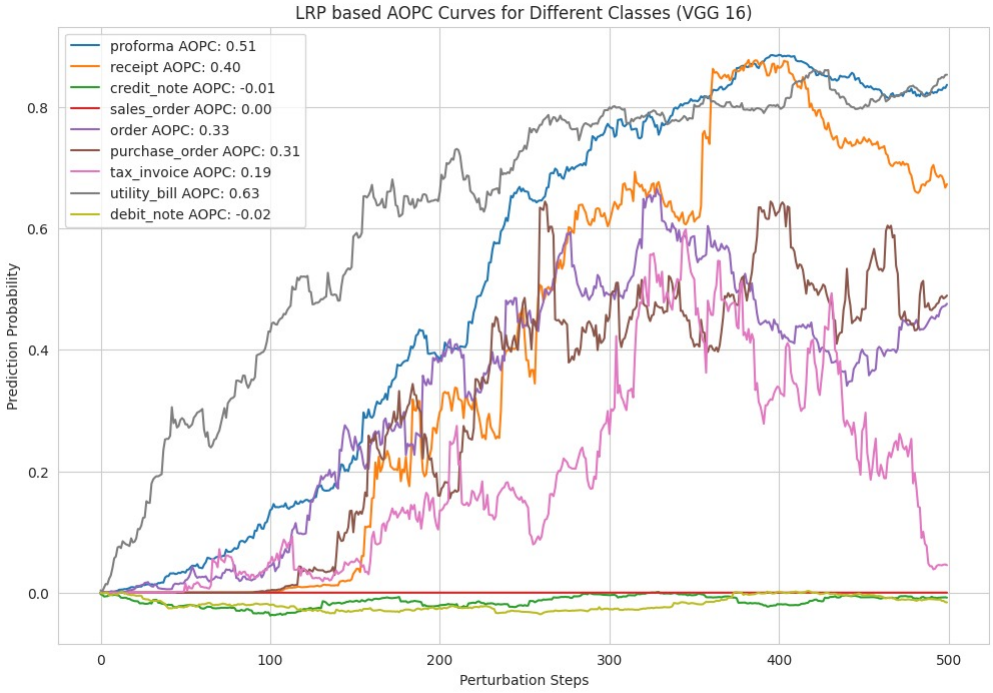

Abbildung 4.20: AOPC von LRP für das VGG 16 Modell

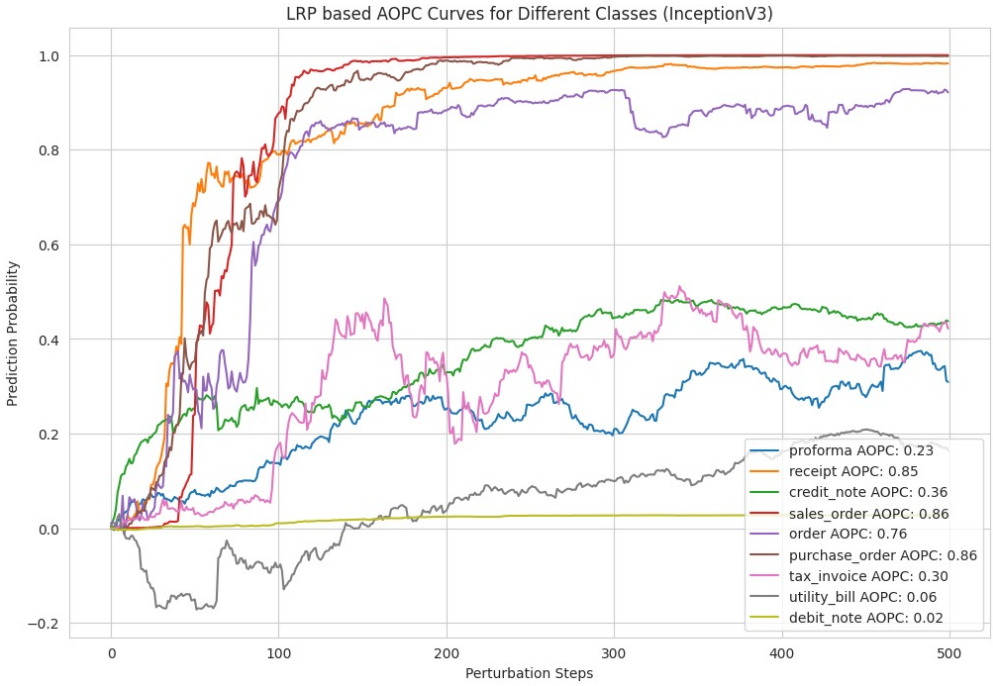

Abbildung 4.21: AOPC von LRP für das InceptionV3 Modell

5 Diskussion

In der Analyse der VGG 16 Trainingsergebnisse zeigte sich eine gesteigerte Trainingsgenauigkeit, welche bis auf 99,05% anstieg. Es zeigten sich jedoch gleichzeitig in der Validierungsgenaugikeit zwischenzeitige Schwankungen. Die Schwankungen in der Validierungsgenauigkeit sind nicht ungewöhnlich, könnten jedoch auf ein mögliches Overfitting hindeuten. Schließlich zeigte sich jedoch der Höchstwert der Validierungsgenaugikeit bei 99,07%, was als solide Konvergenz des Modells gewertet werden kann. Zu Beginn des Trainings war der Loss mit 0,19 relativ hoch, woran man erkennt, dass das Modell zu diesem Zeitpunkt noch viele Fehler gemacht hat. Der Loss verbesserte sich stetig, sodass er nach der fünften Epoche auf 0,034 gesunken war. Auch der Validierungsloss schwankte, ähnlich wie die Validierungsgenauigkeit, in den zweiten Epochen, konnte sich nach der dritten Epoche wieder stabilisieren. Am Ende des Trainings betrug der Validierungsloss 0,031. Somit war davon auszugehen das Modell in der Lage ist auch die Trainingsdaten effektiv verarbeiten zu können Es zeigten sich in der Perfomanceanalyse des VGG 16 Modells, dass die Trainingsdaten in der Konfusionsmatrix häufig von der Hauptdiagonale abwichen, sodass interpretiert werden kann, dass trotz Anpassungen im Datensatz eine deutliche Imbalance in den Klassenergebnissen erkennbar ist. Dies betrifft insbesondere die Klassen *proforma*, *credit_ note*, *utility_ bill* und *debit_ note*, die im Trainingsdatensatz stark unterrepräsentiert waren. Diese Ungleichverteilung spiegelt sich auch in der Konfusionsmatrix wider, in der diese Klassen eine geringere Genauigkeit aufweisen. Im Gegensatz dazu schneiden die anderen Klassen deutlich besser ab und zeigen insgesamt eine zufriedenstellende Leistung. Die Konfusionsmatrix bezüglich der Klassen *receipt*, *sales_ order*, *order*, *purchase_ order* und *tax_ invoice* zeigte eine hohe Performance, sodass eine zuverlässige Klassifikation angenommen werden kann. Es wurden zum Teil Dokumente richtig klassifiziert, wie bspw. die *utility_ bill*, allerdings kann diese Leistung aufgrund der geringen Anzahl von Testdokumente auch zufälliger Natur sein.

In der Analyse von Inception V3 zeigte sich eine hohe Genauigkeit, es zeigten sich weiterhin kaum Schwankungen in der Validierungsgenaugikeit, sodass ingesamt eine starke Validierungsgenaugikeit angenommen werden kann. Dies wird weiter unterstützt, da sich eine stabile Reduktion des Validierungs-Losses zeigte. Obwohl das Modell insgesamt gute Ergebnisse erzielt, könnte es bei einem Training über mehrere Epochen möglicherweise noch bessere Ergebnisse erzielen, jedoch besteht auch das Risiko des Overfittings. Ähnlich wie beim VGG16-Modell schnitten stark repräsentierte Klassen sehr gut ab. Besonders positiv zu bewerten ist ist, dass die Klasse *credit_ note* korrekt klassifiziert wurde, was beim VGG16-Modell nicht der Fall war. Auf der anderen Seite zeigte Inception V3 Schwächen bei der Klasse *proforma*, die teilweise fehlerhaft klassifiziert wurde, und *debit_ note*, die ebenfalls falsch zugeordnet wurde. Die Klassenperformancanalyse für Inception V3 zeigte eine hohe Genauigkeit für *receipt*, *sales_ order*, *order*, *purchase_ order* und *tax_ invoice* auch die Klasse *credit_ note*. Letztere war von VGG 16 nicht korrekt klassifiziert worden. Aufgrund der kleinen Testmenge könnte diese Leistung jedoch auch zufällig sein. Ebenfalls bemerkenswert ist die gute Performance der Klasse *utility_ bill*, obwohl auch hier nur eine kleine Testmenge gab. Interessant ist, dass das Inception V3-Modell keine Testbilder fälschlicherweise der Klasse *proforma* zugeordnet hat, was zu einer hohen Präzision für diese Klasse führte. VGG 16 hatte der Klasse fälschlicherweise Testbilder zugeordnet.

Bei der Analyse von Grad-CAM im VGG 16 Modell zeigte sich, dass in Abhängigkeit von der Repräsentation einer Dokumentenklasse verschiedene Flächen innerhalb der Dokumente hervorgehoben wurden. Waren es bei breit repräsentierten Klassen kleinere Bildbereiche, zeigten sich bei weniger repräsentierten Dokumenten große, zusammenhängende Flächen. Dies deutet darauf hin, dass das Modell bei diesen eher unterrepräsentierten Klassen auf globale Merkmale und Kontextinformationen setzt, anstatt sich auf detaillierte Bereiche zu konzentrieren. In stark repräsentierten Klassen wie *order* und *tax_ invoice* werden jedoch kleinere Bildbereiche als wichtig hervorgehoben, was auf eine stärkere Fokussierung auf bestimmte Merkmale hinweist. Inception V3 zeigte jedoch insgesamt spezifischere Fokussierung in der Analyse mit Grad-CAM. Insgesamt zeigt Grad-CAM, dass Inception V3 in der Lage ist, spezifische Details wie Textblöcke oder Symbole gezielt zu identifizieren. Wie im Ergebnisteil beschrieben, zeigt VGG 16 diffuse Verteilung der relevanten Bildbereiche bei der Betrachtung der Integrated Gradients. Dies deutet

darauf hin, dass das VGG 16 Modell Schwierigkeiten hat, spezifische Merkmale genau zu erkennen. Dies könnte sich auf die Genauigkeit der Klassifikation negativ auswirken, besonders bei Klassen, die detaillierte Merkmale erfordern. Für bestimmte Dokumentenklassen zeigte das Modell VGG 16 eine klarere Fokussierung auf bestimmte kleine relevante Bildbereiche. Die Heatmaps dieser Klassen deuten darauf hin, dass das Modell in der Lage ist, die entscheidungsrelevanten Regionen in diesen Dokumenten besser zu erkennen. Inception V3 zeigt im Vergleich zu VGG 16 insbesondere in den Integrated Gradient Heatmaps deutlich abgegrenztere Regionen. Diese Heatmaps deuten darauf hin, dass InceptionV3 präziser auf spezifische Merkmale fokussiert und somit eine höhere Klassifikationsgenauigkeit erzielt. Auch bei den unterrepräsentierten und insgesamt weniger genau klassifizierten Klassen, wie *proforma* und *debit_ note*, zeigt das Modell Inception V3 eine breitere Verteilung von Relevanzwerten, was darauf hindeutet, dass hier mehrere Merkmale in die Entscheidung einfließen. Die LRP Heatmaps sowohl für das VGG 16 als auch für Inception V3 Modell zeigen insgesamt sehr schwache Relevanzausprägungen in den meisten Klassen. Dies könnte bedeuten, dass LRP in diesem speziellen Kontext nicht in der Lage ist, klare und stark ausgeprägte Relevanzbereiche zu identifizieren.

Das VGG 16 Modell weist insgesamt eine geringere Sensitivität gegenüber der Perturbation relevanter Bildbereiche auf, was sich in niedrigen AOPC-Werten in vielen Klassen zeigt. Insbesondere die Klassen *receipt*, *credit_ note*, *sales_ order* und *purchase_ order* erreichen AOPC Werte nahe Null. Dies deutet darauf hin, dass die von Grad-CAM identifizierten relevanten Bereiche keinen großen Einfluss auf die Vorhersagen des Modells haben. Ein möglicher Grund könnte sein, dass VGG 16 eher auf globale Bildmerkmale fokussiert ist und weniger präzise spezifische Bereiche erkennt. Ein bemerkenswertes Verhalten zeigt die Klasse *order*: Nach dem Entfernen von etwa 350 relevanten Bildbereichen flacht die Kurve ab. Dies lässt vermuten, dass die entfernten Bereiche eher störend waren und ihre Eliminierung zu stabileren Vorhersagen führt. InceptionV3 Modell

Im Gegensatz dazu zeigt das InceptionV3-Modell eine deutlich höhere Empfindlichkeit gegenüber der Perturbation relevanter Bildbereiche. Klassen wie *receipt*, *sales_ order* und *purchase_ order* weisen AOPC-Werte von über 0,60 auf, was darauf hinweist, dass diese Bereiche einen signifikanten Einfluss auf die Modellvorhersagen haben. Dies suggeriert, dass InceptionV3 in der Lage ist, detailliertere und rele-

vantere Bildbereiche zu identifizieren, die für die Klassifikation entscheidend sind. Diese Ergebnisse spiegeln die insgesamt bessere Performance von InceptionV3 im Vergleich zu VGG16 wider.

Die AOPC-Ergebnisse von Integrated Gradients zeigen eine breite Streuung für das VGG16-Modell. Hohe AOPC-Werte in Klassen wie *utility_ bill* (0,86), *tax_ invoice* (0,82) und *receipt* (0,80) deuten darauf hin, dass das Modell in diesen Klassen stark auf die von Integrated Gradients identifizierten relevanten Bereiche angewiesen ist. Allerdings weist die Klasse *credit_ note* einen negativen AOPC Wert von -0,02 auf, was bedeutet, dass die Entfernung der als relevant identifizierten Bereiche zu einer Verbesserung der Modellvorhersage führt. Dies könnte darauf hindeuten, dass das Modell in dieser Klasse irreführende Merkmale identifiziert und Schwierigkeiten hat, relevante Bildbereiche korrekt zu erkennen. InceptionV3 Modell

Das InceptionV3 Modell zeigt im Vergleich zu VGG 16 höhere AOPC Werte in den meisten Klassen. Die hohen Werte in *purchase_ order*, *sales_ order* und *receipt* deuten darauf hin, dass das Modell stärker auf die relevanten Bildbereiche angewiesen ist und diese besser erkennt. Allerdings gibt es auch Klassen wie *utility_ bill* und *debit_ note*, in denen InceptionV3 weniger sensitiv auf die Perturbation reagiert. Dies könnte auf Schwierigkeiten bei der Erkennung relevanter Bereiche in diesen Klassen hinweisen, möglicherweise bedingt durch eine geringe Anzahl von Trainingsbeispielen oder die Komplexität der Klassifikation dieser Kategorien.

Die AOPC Ergebnisse von LRP sind beim VGG 16 Modell insgesamt schwächer. In den meisten Klassen zeigt das Modell nur eine geringe Sensitivität gegenüber der Perturbation relevanter Bildbereiche. Lediglich die Klasse *utility_ bill* weist einen höheren AOPC Wert von 0,63 auf. Negative AOPC Werte in Klassen wie *credit_ note* (-0,01) und *debit_ note* (-0,02) deuten darauf hin, dass die Entfernung der vermeintlich relevanten Bereiche zu einer verbesserten Modellvorhersage führen kann. Dies suggeriert, dass VGG 16 Schwierigkeiten hat, mit LRP relevante Bereiche in diesen Klassen korrekt zu identifizieren. InceptionV3 Modell

Das InceptionV3 Modell zeigt auch bei LRP eine höhere Sensitivität gegenüber der Perturbation als VGG 16, insbesondere in den Klassen *receipt* (0,85), *sales_ order* (0,86) und *purchase_ order* (0,86). Diese hohen AOPC Werte deuten darauf hin, dass das Modell stark von den durch LRP identifizierten relevanten Bereichen abhängt. Bemerkenswert ist die Klasse *credit_ note*, die mit einem AOPC Wert von

0,36 eine bessere Sensitivität im InceptionV3 Modell zeigt. Dies legt nahe, dass InceptionV3 in der Lage ist, relevantere Merkmale für diese Klasse zu identifizieren.

6 Zusammenfassung

Zusammenfassend zeigt diese Arbeit, dass das Inception V3-Modell im Vergleich zum VGG 16 Modell eine überlegene Leistung bei der Dokumentenklassifikation erzielt. Während das VGG 16 Modell eine hohe Trainingsgenauigkeit von bis zu 99,05% erreichte, zeigten sich Schwankungen in der Validierungsgenauigkeit, die auf ein mögliches Overfitting hindeuten könnten. Trotz dieser Schwankungen erzielte VGG 16 eine maximale Validierungsgenauigkeit von 99,07%, was auf eine solide Konvergenz des Modells schließen lässt. Allerdings litt das Modell unter einer deutlichen Klassenungleichheit, insbesondere bei den unterrepräsentierten Klassen *proforma*, *credit_note*, *utility_bill* und *debit_note*, was sich in der Konfusionsmatrix widerspiegelte.

Im Gegensatz dazu zeichnete sich das Inception V3-Modell durch eine hohe und stabile Genauigkeit sowie eine konsistente Reduktion des Validierungs-Losses aus, ohne signifikante Schwankungen in der Validierungsgenauigkeit. Besonders hervorzuheben ist die korrekte Klassifikation der Klasse *credit_note*, die beim VGG 16 Modell nicht zufriedenstellend abgebildet wurde. Die Analyse mittels Grad-CAM und Integrated Gradients verdeutlichte, dass Inception V3 spezifischere und relevantere Bildbereiche identifiziert, was zu einer höheren Klassifikationsgenauigkeit beiträgt.

Die Performanzanalysen mittels Konfusionsmatrix, Grad-CAM, Integrated Gradients und Layer-wise Relevance Propagation (LRP) lieferten zusätzliche Einblicke in das Entscheidungsverhalten der Modelle. VGG 16 neigte dazu, diffuse Relevanzbereiche zu identifizieren, was die Erkennung spezifischer Merkmale erschwerte. Im Gegensatz dazu zeigte Inception V3 eine präzisere Fokussierung auf relevante Bildbereiche, was sich in höheren AOPC Werten und einer besseren Sensitivität gegenüber Perturbationen widerspiegelte. Diese Ergebnisse deuten darauf hin, dass

Inception V3 nicht nur eine höhere Klassifikationsgenauigkeit aufweist, sondern auch transparentere und nachvollziehbarere Entscheidungsprozesse bietet.

Trotz der positiven Ergebnisse gibt es auch Limitationen, die beachtet werden müssen. Die starke Abhängigkeit von stark repräsentierten Klassen und die Schwierigkeiten bei der Klassifikation weniger repräsentierter Klassen deuten darauf hin, dass zukünftige Arbeiten verstärkt auf Methoden zur Datenaugmentation oder auf die Implementierung fortschrittlicher Techniken zur Bewältigung von Klassenungleichgewichten setzen sollten. Zudem könnten weitere Erklärungsmodelle eingesetzt werden, um die Entscheidungsprozesse der Modelle noch besser zu verstehen und zu optimieren.

Obwohl beide Modelle ihre jeweiligen Stärken und Schwächen aufweisen, verdeutlichen die Ergebnisse dieser Arbeit, dass Inception V3 eine robustere und präzisere Lösung für die Dokumentenklassifikation darstellt. Insbesondere bei der Handhabung von weniger repräsentierten Klassen und der Resistenz gegenüber Veränderungen in den Eingabedaten bietet Inception V3 eine langfristig relevante und effektive Methode im Vergleich zu den allgemeineren Ansätzen des VGG 16 Modells.

Literaturverzeichnis

[1] Adadi, Amina ; Berrada, Mohammed: Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). In: *IEEE Access* 6 (2018), S. 52138–52160

[2] Aggarwal, Charu C.: *Neural Networks and Deep Learning: A Textbook*. Cham : Springer, 2018. – ISBN 978-3-319-94463-0

[3] Bach, Sebastian ; Binder, Alexander ; Montavon, Grégoire ; Klauschen, Frederick ; Müller, Klaus-Robert ; Samek, Wojciech: On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. In: *PLOS ONE* 10 (2015), 07, Nr. 7, S. 1–46. – URL https://doi.org/10.1371/journal.pone.0130140

[4] Belval, Edouard: *pdf2image*. https://github.com/Belval/pdf2image. 2024. – 13.8.2024

[5] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. Springer New York, NY, 2006 (Information Science and Statistics). – ISBN 978-0-387-31073-2

[6] Danilevsky, Marina ; Qian, Kun ; Aharonov, Ranit ; Katsis, Yannis ; Kawas, Ban ; Sen, Prithviraj: *A Survey of the State of Explainable AI for Natural Language Processing*. 2020. – URL https://arxiv.org/abs/2010.00711

[7] Eickenberg, Michael ; Gramfort, Alexandre ; Varoquaux, Gael ; Thirion, Bertrand: Seeing it all: Convolutional network layers map the function of the human visual system. In: *NeuroImage* 152 (2016), 10

[8] Ferguson, Max ; Ak, Ronay ; Lee, Yung-Tsun T. ; Law, Kincho H.: Automatic localization of casting defects with convolutional neural networks. In:

2017 IEEE International Conference on Big Data (Big Data), 2017, S. 1726–1735

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

[10] Grandini, Margherita ; Bagli, Enrico ; Visani, Giorgio: *Metrics for Multi-Class Classification: an Overview*. 2020. – URL https://arxiv.org/abs/2008.05756

[11] Guidotti, Riccardo ; Monreale, Anna ; Ruggieri, Salvatore ; Turini, Franco ; Giannotti, Fosca ; Pedreschi, Dino: A Survey of Methods for Explaining Black Box Models. In: *ACM Comput. Surv.* 51 (2018), aug, Nr. 5. – URL https://doi.org/10.1145/3236009. – ISSN 0360-0300

[12] Hinton, Geoffrey E. ; Srivastava, Nitish ; Krizhevsky, Alex ; Sutskever, Ilya ; Salakhutdinov, Ruslan R.: *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. – URL https://arxiv.org/abs/1207.0580

[13] Koch, Christof ; Ullman, Shimon: Shifts in selective visual attention: towards the underlying neural circuitry. In: *Human neurobiology* 4 4 (1985), S. 219–27. – URL https://api.semanticscholar.org/CorpusID:45203429

[14] Kokhlikyan, Narine ; Miglani, Vivek ; Martin, Miguel ; Wang, Edward ; Alsallakh, Bilal ; Reynolds, Jonathan ; Melnikov, Alexander ; Kliushkina, Natalia ; Araya, Carlos ; Yan, Siqi ; Reblitz-Richardson, Orion: *Captum: A unified and generic model interpretability library for PyTorch*. 2020

[15] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F. (Hrsg.) ; Burges, C.J. (Hrsg.) ; Bottou, L. (Hrsg.) ; Weinberger, K.Q. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 25, Curran Associates, Inc., 2012. – URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[16] Lecun, Y. ; Bottou, L. ; Bengio, Y. ; Haffner, P.: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324

[17] Lin, Min ; Chen, Qiang ; Yan, Shuicheng: Network In Network. In: *CoRR* abs/1312.4400 (2013). – URL https://api.semanticscholar.org/CorpusID:16636683

[18] Mohri, M. ; Rostamizadeh, A. ; Talwalkar, A.: *Foundations of Machine Learning*. MIT Press, 2012 (Adaptive Computation and Machine Learning series). – ISBN 978-0-262-01825-8

[19] Mohseni, Sina ; Zarei, Niloofar ; Ragan, Eric D.: A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems. In: *ACM Trans. Interact. Intell. Syst.* 11 (2021), sep, Nr. 3–4. – URL https://doi.org/10.1145/3387166. – ISSN 2160-6455

[20] Mondal, Tanmoy ; Ragot, Nicolas ; Ramel, Jean-Yves ; Pal, Umapada: Exemplary Sequence Cardinality: An effective application for word spotting. In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 2015, S. 1146–1150

[21] Montavon, Grégoire ; Binder, Alexander ; Lapuschkin, Sebastian ; Samek, Wojciech ; Müller, Klaus-Robert: *Layer-Wise Relevance Propagation: An Overview*. S. 193–209. In: Samek, Wojciech (Hrsg.) ; Montavon, Grégoire (Hrsg.) ; Vedaldi, Andrea (Hrsg.) ; Hansen, Lars K. (Hrsg.) ; Müller, Klaus-Robert (Hrsg.): *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Cham : Springer International Publishing, 2019. – URL https://doi.org/10.1007/978-3-030-28954-6_10. – ISBN 978-3-030-28954-6

[22] Mudrakarta, Pramod K. ; Taly, Ankur ; Sundararajan, Mukund ; Dhamdhere, Kedar: *Did the Model Understand the Question?* 5 2018. – URL http://arxiv.org/abs/1805.05492

[23] Nair, Vinod ; Hinton, Geoffrey E.: Rectified Linear Units Improve Restricted Boltzmann Machines. In: Fürnkranz, Johannes (Hrsg.) ; Joachims, Thorsten (Hrsg.): *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, Omnipress, 2010,

S. 807–814. – URL https://icml.cc/Conferences/2010/papers/432.pdf

[24] Novac, Ovidiu-Constantin ; Chirodea, Mihai C. ; Novac, Cornelia M. ; Bizon, Nicu ; Oproescu, Mihai ; Stan, Ovidiu P. ; Gordan, Cornelia E.: Analysis of the Application Efficiency of TensorFlow and PyTorch in Convolutional Neural Network. In: *Sensors* 22 (2022), Nr. 22. – URL https://www.mdpi.com/1424-8220/22/22/8872. – ISSN 1424-8220

[25] Osval Antonio Montesinos López, José C.: *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer Cham, 2022. – ISBN 978-3-030-89010-0

[26] Paszke, Adam ; Gross, Sam ; Massa, Francisco ; Lerer, Adam ; Bradbury, James ; Chanan, Gregory ; Killeen, Trevor ; Lin, Zeming ; Gimelshein, Natalia ; Antiga, Luca ; Desmaison, Alban ; Köpf, Andreas ; Yang, Edward ; DeVito, Zach ; Raison, Martin ; Tejani, Alykhan ; Chilamkurthy, Sasank ; Steiner, Benoit ; Fang, Lu ; Bai, Junjie ; Chintala, Soumith: *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. – URL https://arxiv.org/abs/1912.01703

[27] Patterson, Josh ; Gibson, Adam: *Deep Learning: A Practitioner's Approach*. Sebastopol, CA : O'Reilly Media, 2017. – ISBN 978-1-49191-425-0

[28] Rajpal, Ankit ; Sehra, Khushwant ; Bagri, Rashika ; Sikka, Pooja: XAI-FR: Explainable AI-Based Face Recognition Using Deep Neural Networks. In: *Wireless Personal Communications* 129 (2022), 12

[29] Ras, Gabrielle ; Xie, Ning ; Gerven, Marcel van ; Doran, Derek: *Explainable Deep Learning: A Field Guide for the Uninitiated*. 2021. – URL https://arxiv.org/abs/2004.14545

[30] Rothman, D.: *Hands-On Explainable AI (XAI) with Python: Interpret, Visualize, Explain, and Integrate Reliable AI for Fair, Secure, and Trustworthy AI Apps*. Packt Publishing, Limited, 2020 (Expert insight). – URL https://books.google.de/books?id=KhWzzQEACAAJ. – ISBN 9781800208131

[31] Samek, Wojciech ; Binder, Alexander ; Montavon, Grégoire ; Bach, Sebastian ; Müller, Klaus-Robert: *Evaluating the visualization of what a Deep Neural Network has learned*. 2015. – URL https://arxiv.org/abs/1509.06321

[32] Selvaraju, Ramprasaath R. ; Cogswell, Michael ; Das, Abhishek ; Vedantam, Ramakrishna ; Parikh, Devi ; Batra, Dhruv: Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In: *International Journal of Computer Vision* 128 (2019), Oktober, Nr. 2, S. 336–359. – URL http://dx.doi.org/10.1007/s11263-019-01228-7. – ISSN 1573-1405

[33] Shabbeer Basha, SH ; Ram Dubey, Shiv ; Pulabaigari, Viswanath ; Mukherjee, Snehasis: Impact of Fully Connected Layers on Performance of Convolutional Neural Networks for Image Classification. In: *Neurocomputing* (2019). – URL http://arxiv.org/abs/1902.02771

[34] Simonyan, Karen ; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *arXiv preprint arXiv:1409.1556* (2014). – URL https://arxiv.org/abs/1409.1556

[35] Simsa, Stepan ; Stancl, Daniel ; Skalický, Matyáš ; Patel, Yash: *docile*. https://github.com/rossumai/docile. 2023. – 25.6.2024

[36] Šimsa, Štěpán ; Šulc, Milan ; Uřičář, Michal ; Patel, Yash ; Hamdi, Ahmed ; Kocián, Matěj ; Skalický, Matyáš ; Matas, Jiří ; Doucet, Antoine ; Coustaty, Mickaël ; Karatzas, Dimosthenis: *DocILE Benchmark for Document Information Localization and Extraction*. 2023. – URL https://arxiv.org/abs/2302.05658

[37] Skansi, Sandro: *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer Cham, 2018 (Undergraduate Topics in Computer Science). – ISBN 78-3-319-73004-2

[38] Sundararajan, Mukund ; Taly, Ankur ; Yan, Qiqi: *Axiomatic Attribution for Deep Networks*. 2017. – URL https://arxiv.org/abs/1703.01365

[39] Szegedy, Christian ; Vanhoucke, Vincent ; Ioffe, Sergey ; Shlens, Jonathon ; Wojna, Zbigniew: *Rethinking the Inception Architecture for Computer Vision*. 2015. – URL https://arxiv.org/abs/1512.00567

[40] van der Velden, Bas H. ; Kuijf, Hugo J. ; Gilhuijs, Kenneth G. ; Viergever, Max A.: Explainable artificial intelligence (XAI) in deep learning-based medical image analysis. In: *Medical Image Analysis* 79 (2022), S. 102470. – URL https://www.sciencedirect.com/science/article/pii/S1361841522001177. – ISSN 1361-8415

[41] VanderPlas, J.: *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, 2016. – URL https://books.google.de/books?id=6omNDQAAQBAJ. – ISBN 9781491912133

[42] Yamashita, R. ; Nishio, M. ; Do, RKG. ; Togashi, K.: Convolutional neural networks: an overview and application in radiology. In: *Insights into Imaging* 9 (2018), 8

Erklärung zur selbständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort Datum Unterschrift im Original