HAW HAMBURG

BACHELOR THESIS
Paul Meyer

Evaluierung ausgewählter xAI-Methoden zur Nachvollziehbarkeit von Diskriminator-Entscheidungen in GANs

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Paul Meyer

Evaluierung ausgewählter xAI-Methoden zur Nachvollziehbarkeit von Diskriminator-Entscheidungen in GANs

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

Betreuende Prüferin: Prof. Dr. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Thomas Clemen

Eingereicht am: 30. Mai 2024

Paul Meyer

Thema der Arbeit

Evaluierung ausgewählter xAI-Methoden zur Nachvollziehbarkeit von Diskriminator-Entscheidungen in GANs

Stichworte

xAI, Nachvollziehbarkeit, Erklärbarkeit, Interpretierbarkeit, GAN, Diskriminator, Generator

Kurzzusammenfassung

Generative Adversarial Nets (GAN) haben in den letzten Jahren an Popularität gewonnen, in der Forschung und auch im kommerziellen Einsatz. GANs können genutzt werden, um künstlich Bilder, Text oder Ton zu erzeugen. Weiterhin eignen sie sich dazu, Trainingsdatensätze künstlich zu erweitern und die Auflösung von Bildern zu verbessern. In der jüngeren Vergangenheit hat außerdem das Thema Nachvollziehbarkeit algorithmischer Entscheidungen an Bedeutung gewonnen. Zum einen getrieben durch die Forschung, aber auch durch Gesetzgeber und Regulierungsbehörden. Der Ruf nach Nachvollziehbarkeit betrifft daher auch Künstliche Intelligenz und Anwendungen wie GANs. Diese Arbeit untersucht wie GANs für Nutzende besser nachvollziehbar gemacht werden können. Dazu werden die Ausgaben eines Teilmodells des GAN (Diskriminator) mit Methoden zur Nachvollziehbarkeit von Künstlicher Intelligenz analysiert. Außerdem wird überprüft, wie verständlich die Erklärungen dieser Methoden für Nutzende eines GAN sind. Dafür wurde ein Proof of Concept zum Vergleich der Methoden entwickelt, die ein GAN analysieren, welches künstliche Bilder generiert.

Paul Meyer

Title of Thesis

Evaluation of selected xAI methods for the traceability of discriminator decisions in GANs

Keywords

xAI, traceability, explainability, interpretability, GAN, discriminator, generator

Abstract

Generative Adversarial Nets (GAN) have gained popularity in both research and industry. GANs are used to generate artificial images, texts, or audio waves. GANs are either able to expand training datasets or improve the resolution of images. The topic of traceability of algorithmic decisions has also gained importance in the recent past. This development is driven by researchers and either by legislative authorities or regulators. The need for traceability applies to Artificial Intelligence and generative models like GANs. This thesis examinates how the traceability of GANs can be improved for any user. To do so the outcomes of a GANs discriminative model (discriminator) are analyzed with xAI methods. Additionally, it is examined how comprehensible the explanations provided by the xAI methods are for any user. Therefore, a proof of concept has been developed to compare the xAI methods. Within the proof of concept, the xAI methods analyze a GAN which generates artificial images.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Informatik ist auf komplexe Weise in die Dynamik globaler Herausforderungen eingebunden. Theoretische Konzepte zur Lösung von erkannten Problemen existieren oft deutlich früher als ihre praktische Umsetzung. Dies gilt insbesondere für künstliche Intelligenz (KI). Grundlagen dieser Technologie wurden schon in den 60er Jahren des Vergangenen Jahrhunderts erarbeitet (Graziani u. a. (2023)). Die Idee das neuronen-basierte Arbeiten des menschlichen Gehirns nachzuahmen stammt schon aus den 1940er Jahren (McCulloch und Pitts (1943)). Sogenannte neuronale Netze gelten als Standardtechnologie, um menschliche Fähigkeiten wie Kreativität, Schreiben und Sprechen nachzuahmen (Graziani u. a. (2023)). Durch den technischen Fortschritt und die zunehmende Digitalisierung seit Anfang der 2000er Jahre ist es möglich, die einst theoretischen Konzepte zur künstlichen Intelligenz in die Praxis umzusetzen.
Künstliche Intelligenz benötigt zwei Dinge - Rechenleistung und Trainingsdaten (Frochte (2019), S.28). Ersteres ist in Form hochperformanter GPUs oder TPUs und schneller, günstiger Speichermedien verfügbar. Letzteres durch die rasant zunehmende digitale Vernetzung von Menschen und Maschine. Es stehen immense Datenmengen zu Verfügung, mit denen künstliche neuronale Netze trainiert werden können (Wittpahl (2019), S.37). Die Verfügbarkeit einer Vielzahl von Open-Source-Datensammlungen, sowie KI-spezifischer Frameworks für Programmiersprachen sind eine weitere Säule der KI. So ist es inzwischen möglich, sich mit generativen Modellen wie ChatGPT zu unterhalten oder sich auf Basis von Texteingaben Bilder generieren zu lassen. Besonders die Erzeugung von Bildern ist auch für KI keine einfach zu lösende Aufgabe.

2014, erschien ein Paper, welches einen neuen Ansatz für generative Modelle vorschlug. Ian Goodfellow und sein Team von der Universität Montréal entwickelten ein Framework, welches zwei Modelle beinhaltet. Diese arbeiten kompetitiv gegeneinander, um so ein besonders gutes Ergebnis zu erzielen. Dies wird durch sequenziell paralleles Training der Modelle erreicht (Goodfellow u. a. (2014)). Aktuell scheint es aber kaum möglich, den Schaffensprozess generativer künstlicher Intelligenz nachzuvollziehen.

1.1 Problemstellung

In der Praxis ist es für den Endnutzenden oft nicht klar, ob und inwiefern man mit künstlicher Intelligenz interagiert. Außerdem kann oft nicht nachvollzogen werden, warum ein bestimmtes Ergebnis durch eine KI erzeugt wurde (Graziani u. a. (2023)). Daher beschäftigt sich die Forschung damit, wie KI bzw. die zugrundeliegenden Technologien besser verstanden werden können.
Ein Ziel ist es daher, die Ergebnisse von KI für den Nutzenden nachvollziehbar zu machen. Wenn die künstliche Intelligenz von hochdimensionalen Eingaben (z.B. Grafiken) (Goodfellow u. a. (2014)) auf eine Klassifikation abbildet (z.B. ist Katze/ist keine Katze), gibt es verschiedene Möglichkeiten diesen Prozess nachvollziehbar zu machen (Ribeiro u. a. (2016), Lundberg und Lee (2017)). Wird hingegen von niedrigdimensionalen Eingaben (z.B. zeichne eine Katze) in einen hochdimensionalen Raum abgebildet (z.B. Katzenbild), ist dies eine ungleich komplexere Aufgabe für die künstliche Intelligenz und das zugrundeliegende Modell. Zudem ist es schwieriger diesen Prozess für Data Scientists oder sogar Personen ohne Domänenkompetenz verständlich zu machen. Bisher gibt es hierfür keine standardisierten Lösungen. Warum eine generative KI zu einer Eingabe eine entsprechende Ausgabe geliefert hat, ist nicht nachvollziehbar. Eine generative KI erklärt nicht, wie sie zu einem Ergebnis gekommen ist. Daher gibt es in der Forschung die Bestrebung dies zu ändern. Dabei sollen vor allem Möglichkeiten betrachtet werden, die den internen Generierungsprozess verständlicher machen. Dies kann dann als Grundlage für eine Anpassung des Trainingsprozesses genutzt werden, was besonders für die Entwickler des Modells relevant ist. Die Ergebnisse sind also in erster Linie für Data Scientists wichtig und weniger für die Allgemeinheit, also Personen ohne Domänenkompetenz. Trotzdem sollen Potenziale für ein allgemeines Schaffen von Verständnis berücksichtigt werden.

Auf das 2014 von Goodfellow et al. erstmals vorgeschlagenen Modell namens *GAN (Generative Adversarial Network)* bezieht sich die Fragestellung, weil diese Modelle eine sehr breite Anwendung in der Praxis finden. GANs können neben dem Generieren von Inhalten genutzt werden, um künstliche Trainingsdatensätze zu ergänzen, die Auflösung von Bildern zu verbessern oder zur Bildbearbeitung verwendet werden (Chakraborty u. a. (2024)). Teil eines GAN ist ein Modell welches echte von künstlich generierte Inhalten unterscheidet (vgl. 2.4 und 2.4.2). Diesem Modell kommt eine große Bedeutung im Rahmen des Trainings- und Generierungsprozesses von GANs zu. Der Schwerpunkt dieser Arbeit liegt daher auf der Frage:
'Welche xAI-Methoden sind geeignet, um die Entscheidungen dieses Teilmodells (Diskri-

minator) nachzuvollziehen?'.
Dafür werden zunächst die grundlegenden Konzepte zur Nachvollziehbarkeit von KI betrachtet. Im Detail gilt es potenzielle Lösungen zu erarbeiten und im Rahmen eines Proof of Concept zu überprüfen und gleichzeitig die Grenzen der erarbeiteten Lösungen und des Proof of Concept zu beurteilen.

1.2 Zielsetzung

Ziel dieser Arbeit ist es Möglichkeiten zu finden den Generierungsprozess von GANs besser zu verstehen. Daher werden besonders die Entscheidungen des so genannten Diskriminators betrachtet (vgl. 2.4.2). So können Forschende Trainingsprozesse besser verstehen und im Bedarfsfall zielorientiert handeln. Zum anderen können perspektivisch Möglichkeiten erörtert werden, den Generierungsprozess allgemein für Nutzende verständlich zu machen. Im Zuge dessen kann die Güte der Verständlichkeit eingeordnet und bewertet werden. Dies ist relevant, da die von der EU angestoßene Einführung der DSGVO (GDPR) den Nutzenden ein *Recht auf Erklärung von algorithmischen Entscheidungen* einräumt (Goodman und Flaxman (2017)).

1.3 Aufbau der Arbeit

Der verbleibende Teil dieser Arbeit ist wie folgt strukturiert: In Kapitel 2 werden zentrale Begriffe wie Künstliche Intelligenz und Maschinelles Lernen erklärt und voneinander abgegrenzt. Dies ist wichtig da nicht alle Begriffe konsistent definiert sind und nicht immer mit derselben Bedeutung verwendet werden. Daher wird festgelegt, welche Begriffe im weiteren Verlauf der Arbeit verwendet werden. Ebenfalls werden einige Begriffe im Englischen eingeführt, da es keine adäquaten deutschen Übersetzungen gibt. Weiterhin werden für die Arbeit relevante Grundlagen aus den Bereichen Nachvollziehbarkeit von KI und GANs beschrieben.
Kapitel 3 beschreibt das Vorgehen zur praktischen und technischen Umsetzung des Proof of Concept (PoC).
In Kapitel 4 werden die theoretischen und praktischen Ergebnisse vorgestellt. In Kapitel 5 werden die Ergebnisse diskutiert. Kapitel 6 schließt diese Arbeit mit einer Zusammenfassung und einem Ausblick in die Zukunft ab.

2 Grundlagen

Dieses Kapitel stellt Informationen zum theoretischen Hintergrund von Künstlicher Intelligenz (vgl. 2.1) und Maschinellem Lernen (vgl. 2.2) bereit. In 2.3 werden gängige Ansätze zum Erklären und Verstehen von KI und ML-Algorithmen beschrieben. In 2.4 wird die Funktionsweise von GANs erklärt und in 2.5 die eines cGAN.

2.1 Künstliche Intelligenz (KI)

'Künstliche Intelligenz ist die Fähigkeit einer Maschine, menschliche Fähigkeiten wie logisches Denken, Lernen, Planen und Kreativität zu imitieren.'
- EU-Parlament (2023)

'Unter künstlicher Intelligenz (KI) verstehen wir Technologien, die menschliche Fähigkeiten im Sehen, Hören, Analysieren, Entscheiden und Handeln ergänzen und stärken.'
- Microsoft (2020)

'Künstliche Intelligenz ist die Fähigkeit eines Computers oder computergesteuerten Roboters, Aufgaben zu lösen, die normalerweise von intelligenten Wesen erledigt werden.'
- Copeland (2019)

Künstliche Intelligenz wird von Institutionen, Unternehmen oder in Lehrbüchern auf verschiedene, aber oft ähnliche Art definiert. Grundsätzlich implizieren obige und andere Definitionen, dass die KI auf eine nicht exakt bestimmte Art (logisch) denken kann. So sollen die geistigen Fähigkeiten intelligenter Lebewesen auch Maschinen verfügbar gemacht werden.

Wegen unterschiedlicher Definitionen von KI existieren diverse Überschneidungen in den Definitionen der Begriffe die mit KI assoziiert werden. Dies betrifft vor allem Methoden zur Nachvollziehbarkeit von KI. Daher werden diese Begriffe für die vorliegende Arbeit

vereinheitlicht und sind entsprechend der vorgenommenen Abgrenzungen zu verstehen (vgl. 2.3). Da viele Begriffe keine exakte deutsche Übersetzung haben, wird ebenso festgelegt, in welcher Sprache die Begriffe ab hier verwendet werden. Dies dient der Vereinheitlichung und dem Lesefluss, da deutsche Übersetzungen oft wenig intuitiv sind (z.B. Deep Learning dt. tiefes Lernen).

Es gibt zwei Zweige der KI: *Symbolische KI* (engl. Symbolic AI) und *Maschinelles Lernen* (Graziani u. a. (2023)). Symbolische KI spielt vor allem im Bereich automatischer, intelligenter Agenten eine Rolle und arbeitet auf Basis einer deterministischen Wissensbasis (Norvig und Russell (2002)).

ML basiert auf dem Lernen aus Daten und der dadurch gesammelten Erfahrung (Frochte (2019), S.13). Anwendungen, die ML implementieren werden im Folgenden als Modelle bezeichnet. ML und insbesondere Deep Learning (DL) Modelle sind außerordentlich leistungsfähig und können immer komplexere Aufgaben bewältigen (Frochte (2019), S.25). Die vorliegende Arbeit befasst sich ausschließlich mit diesem Teilgebiet der KI. Wenn im Folgenden von KI die Rede ist, meint dies immer KI-Systeme die ML implementieren. Besonders komplexere Aufgaben werden mit Deep Learning implementierenden Modellen gelöst (Frochte (2019), S.27, Paaß und Hecker (2020). S.6).

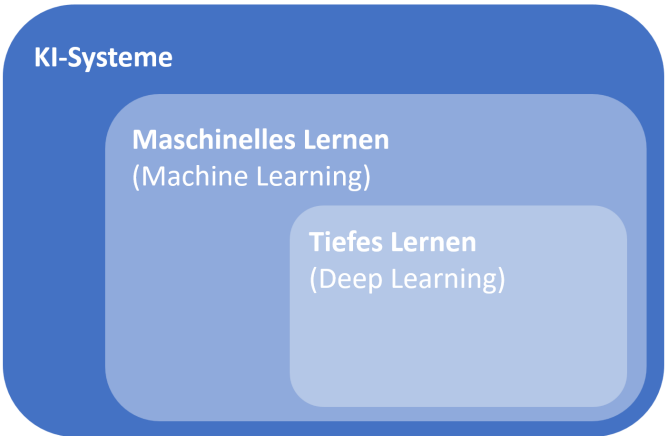

Abbildung 2.1: KI-Systeme, Maschinelles Lernen und Deep-Learning im Venn-Diagramm (nach Graziani u. a. (2023))

Die angesprochene zunehmende Komplexität der Modelle bietet eine große Chance immer komplexere Probleme mit Hilfe von Machine Learning zu lösen. Dadurch sind die Modelle 'undurchsichtiger' geworden und die Erzeugung der Ergebnisse kann kaum oder gar nicht

nachvollzogen werden. Solche undurchsichtigen Modelle, die schwer nachvollziehbare Methoden wie DL implementieren, bezeichnet man als Black-Box-Modelle (Rudin (2019), Arrieta u. a. (2019), Graziani u. a. (2023)).

Begriffsklärung - *Nachvollziehbarkeit*		
Erklärbare KI	Explainable AI	xAI
Interpretierbare KI	Interpretable AI	- - -
Verantwortungsvolle KI	Responsible AI	RAI

Tabelle 2.1: Methoden mit dem Ziel, Black-Box-Modelle verständlich zu machen.

Die wesentlichen Begriffe der Methoden zur Lösung dieses Verständnisproblems, fasst die Tabelle 2.1 zusammen.
Im Folgenden werden ausschließlich die englischen Begriffe aus Tabelle 2.1 verwendet. Als Oberbegriff der in Tabelle 2.1 eingeführten Begriffe wird *Nachvollziehbarkeit* verwendet. Die Methoden ermöglichen es, die Generierung der Ergebnisse von Modellen nachzuvollziehen. Modelle subsumiert dabei Systeme die ML und ggf. DL implementieren und wird im Folgenden als Oberbegriff verwendet.

2.2 Maschinelles Lernen (ML)

Es werden drei Lernalgorithmen des Machine Learnings unterschieden, um Wissen aus Daten zu extrahieren (Frochte (2019), S.20):

- Supervised Learnring (Überwachtes Lernen)
- Unsupervised Learning (Unüberwachtes Lernen)
- Reinforcement Learning (Bestärkendes Lernen)

Alle drei Lernalgorithmen haben das Ziel, Muster in strukturierten oder unstrukturierten Datensätzen zu erkennen, bzw. ein optimales Verhalten in einer definierten Umgebung zu ermitteln. Die Strategien und Bedingungen der Lernalgorithmen unterscheidet sich entsprechend.
Supervised Learning ist ein Lernalgorithmus, der meist für Regressions- oder Klassifikationsprobleme eingesetzt wird. Beim Lösen von Klassifikationsproblemen wird aus einer Menge von Eingabedaten in zwei oder mehrere Klassen abgebildet (z.B. Katzenfoto ja/nein). Bei einem Regressionsproblem wird aus einer Menge von Eingabedaten

in eine stetige Zielmenge abgebildet, beispielsweise von Datensätzen über Personen auf deren zu erwartendes Gehalt. In beiden Fällen erfolgt die Abbildung mit einer Funktion, deren Parameter im Rahmen des Trainings angepasst werden ('gelernt werden'). Dafür benötigt man Datensätze in denen der Wert, auf den abgebildet wird ein Label hat. Zum Beispiel ob ein Katzenfoto vorliegt oder das jeweilige Gehalt der Personen. Man spricht von Supervised Learning, da dem Lernalgorithmus im Trainingsprozess bekannt ist, welche konkreten Werte die Zielmenge haben kann. Nach umfangreichem Training ist der Lernalgorithmus in der Lage, Datensätze ohne Label in die Zielmenge abzubilden (Frochte (2019), S.21).

Im Gegensatz dazu sind beim **Unsupervised Learning** die eben beschriebenen Label nicht vorhanden. Üblicherweise löst dieser Lernalgorithmus Probleme bei denen eigenständig Klassen gebildet werden müssen. Die Elemente eines Datensatzes werden dann diesen Klassen zugeordnet. Das Ziel ist es, eigenständig Muster und Gruppen in den Daten zu identifizieren (Frochte (2019), S.22).

Reinforcement Learning wird hauptsächlich für lernende Software-Agenten angewendet, die sich in einer definierten Umgebung möglichst optimal verhalten sollen (Li (2018)). Der lernende Agent führt Aktionen aus und erhält dafür positives oder negatives Feedback. Ziel des Agenten ist es, eine Strategie zu entwickeln, so dass er sein positives Feedback maximiert (Frochte (2019), S.23).

Grundsätzlich können alle drei Lernalgorithmen mit Deep Learning realisieren werden, müssen es aber nicht zwangsläufig (Li (2018)). In 2.4 wird gezeigt welche dieser Lernalgorithmen für GANs benötigt werden.

2.3 ML-Algorithmen verstehen

Die Ergebnisse einer KI nachvollziehen zu können, ist für die verschiedensten Anwendungsbereiche relevant (Graziani u. a. (2023), Miller (2023)). Es existieren unterschiedliche Ansätze mit unterschiedlichen Ansprüchen an die Güte und die Zielgruppe der Erklärungen (Saeed und Omlin (2023)). Dieser Abschnitt fasst die gängigen Möglichkeiten zur Nachvollziehbarkeit von KI und ihre jeweiligen Ziele zusammen.

In einigen Arbeiten werden die Begriffe *Erklärbarkeit* und *Interpretierbarkeit* synonym verwendet (Miller (2019)), in der Mehrzahl aber nicht (Rudin (2019), Arrieta u. a. (2019),

Palacio u. a. (2021)). Da diese Begriffe Methoden mit unterschiedlichen Ansätzen (Explainable AI vgl. 2.3.1 und Interpretable AI vgl. 2.3.2) subsumieren, sollten diese nicht synonym verwendet werden (Graziani u. a. (2023)). Daher werden diese Begriffe in der vorliegenden Arbeit den folgenden Definitionen entsprechend verwendet.

2.3.1 Explainable AI

Explainable AI beschreibt Methoden und Frameworks, um die Ausgaben von KI-Systemen nachvollziehen zu können. Mit Hilfe der Ein- und Ausgabedaten des Originalmodells wird ein Surrogatmodell trainiert, um die Ausgabedaten mit den Eingabedaten zu erklären (vgl. LIME (Ribeiro u. a. (2016)) und SHAP (Lundberg und Lee (2017))). Die dabei gewährten Einblicke sind primär technisch und hauptsächlich für Data Scientists verständlich (Rudin (2019)). Solche Methoden erklären, welche Attribute eines Datensatzes, welchen Einfluss auf die Ausgaben haben, die ein neuronales Netz generiert hat.
Ein Beispiel hierfür wäre SHAP. Diese Methode berechnet auf Basis eines eigens trainierten Surrogatmodells die Einflusswerte auf das Ergebnis.
Explainable AI greift daher nie in den Generierungs- und Trainingsprozess des eigentlichen Modells ein. Außerdem kennen diese Methoden keine Implementationsdetails des Basismodells und erhalten auch keine Informationen aus dessen Generierungsprozess. Daher wird dieser Ansatz als *Post-Hoc-Erklärung* bezeichnet (Molnar u. a. (2020)). Der Ansatz kann lokal für einen Datenpunkt (Ribeiro u. a. (2016)) oder global für das gesamte Modell angewendet werden (Lundberg und Lee (2017)).

2.3.2 Interpretable AI

Interpretable AI bezeichnet Modelle, die entweder intrinsisch erklärend sind oder im Rahmen des Trainings parallel die Ausgabegenerierung dokumentieren (Chang u. a. (2022)). Es bedarf also keines Surrogatmodells. Interpretable AI ist insbesondere nicht als Post-Hoc-Ansatz zu verstehen.
Lineare oder logistische Regression sind Beispiele für Interpretable AI, da jede Ausgabe direkt einer Eingabe zugeordnet werden kann (Mahya und Fürnkranz (2023)). Auch sollen Modelle nicht nur für Data Scientists nachvollziehbar sein, sondern für eine größere Nutzendengruppe mit und ohne spezifischem Fachwissen (Chang u. a. (2022), Graziani u. a. (2023)).

2.3.3 Responsible AI

Responsible AI beschreibt Modelle, die nicht nur ihre Ausgaben rechtfertigen und dem Nutzenden nachvollziehbar machen, sondern es werden auch ethische Aspekte berücksichtigt. Im Fokus steht der Nutzende, dem garantiert werden soll, dass Ausgaben fair, verlässlich, nicht-diskriminierend und konsistent sind. Außerdem wird sozialen und rechtlichen Normen entsprochen (Arrieta u. a. (2019)). Insbesondere sollen diese Methoden große KI-Systeme, die jeder nutzen kann, nachvollziehbar machen. Das Ziel wäre, dass ein Large Language Model (z.B. ChatGPT) erklärt wie es eine Ausgabe generiert hat. Es rechtfertigt, wie fair, zuverlässig und ethisch korrekt vorgegangen wurde (Arrieta u. a. (2019)).

2.4 Generative Adverserial Networks (GAN)

Das von Goodfellow u.a. 2014 vorgeschlagene Modell zum Generieren von Inhalten hat sich in den vergangenen Jahren etabliert und wurde stetig weiterentwickelt (Salimans u. a. (2016), Radford u. a. (2016), Chakraborty u. a. (2024)).
Um den Generierungsprozess dieser Modelle nachvollziehen zu können, werden in 2.4.1 die Grundidee und der Aufbau eines GAN erklärt. Anschließend werden die zentralen Komponenten eines GAN, der *Diskriminator* und der *Generator*, erklärt.
In 2.4.4 wird der Trainingsprozess eines GAN beschrieben. Die Funktionsweise des speziellen GAN-Typs cGAN wird in 2.5 erläutert.

2.4.1 Idee

Das Besondere an einem GAN ist seine Architektur. Goodfellow u. a. (2014) schlugen kein neuartiges Netz oder eine revolutionäre Aktivierungsfunktion vor, sondern ein neues strukturelles Konzept. Dies betrifft die Gesamtarchitektur und den Trainingsprozess. Ein GAN besteht aus zwei Komponenten, dem Generator (G) und dem Diskriminator (D) (Goodfellow u. a. (2014)). Der Generator G generiert künstliche Instanzen einer definierten Domäne z.B. Grafiken, Text oder Musik. Der Diskriminator D hat die Aufgabe von G künstlich erzeugte Domäneninstanzen von echten Instanzen zu unterscheiden, also z.B generierte Katzenbilder von Fotos real existierender Katzen (Rashid (2020), S.62). G und D stehen in einem Wettbewerb, in dem G versucht künstliche Domäneninstanzen

zu generieren, die D nicht von echten unterscheiden kann. D versucht die vorgelegten Instanzen nach generierten und echten zu unterscheiden (Creswell u. a. (2018)).

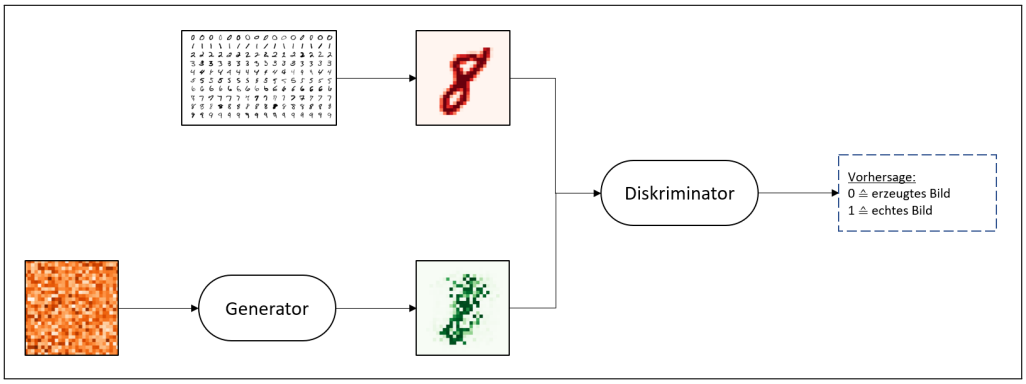

Abbildung 2.2: Datenfluss und Aufbau eines GAN (nach Steinwendner und Schwaiger (2020), S.322)

D und G sind in der Regel vielschichtige neuronale Netze, können aber prinzipiell differenzierbare Funktionen beliebiger Gestalt sein (Goodfellow u. a. (2014)). Diese Vielschichtigkeit impliziert, dass GANs Modelle sind, die *Deep Learning* implementieren. Das Training von G und D ist kompliziert und muss aufeinander abgestimmt sein (Mescheder u. a. (2018)). Es erfolgt aber mit Trainigsdaten ohne konkretes Label. Obwohl der Diskriminator mit den Labeln echte/unechte Instanz arbeitet, lassen sich GANs in die in 2.2 beschriebene Kategorie *Unsupervised Learning* einordnen (Creswell u. a. (2018)). Man spricht im Kontext von GANs auch von *Unsupervised Deep Learning* (vgl. Abbildung 2.3).

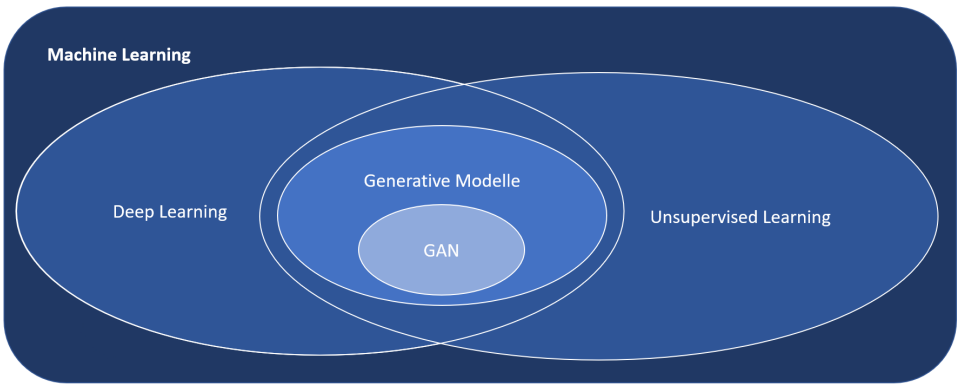

Abbildung 2.3: GANs implementieren Deep Unsupervised Learning (eigene Abbildung)

Damit ein GAN gute Ergebnisse erzielt, müssen G und D simultan trainiert werden (Goodfellow u. a. (2014)). Das heißt beide müssen 'gemeinsam lernen'. Eine Komponente darf der anderen nicht zu weit im Training voraus sein. Ist D zu gut darin künstlich generierte Instanzen zu erkennen, wird G es nie schaffen Instanzen zu generieren die D als echt klassifiziert. Ist D nicht gut darin generierte Instanzen zu erkennen, generiert G keine Instanzen höherer Qualität, da das aktuelle niedrige Niveau reicht, um D zu täuschen (Rashid (2020), S.91). Das Hauptziel qualitativ hochwertige Grafiken oder andere Inhalte künstlich zu generieren würde nicht erreicht werden (Nagisetty u. a. (2022)).

2.4.2 Diskriminator (D)

Der Diskriminator D ist ein neuronales Netz, welches die Aufgabe hat künstlich generierte Domäneninstanzen von echten zu unterscheiden. D löst ein binäres Klassifikationsproblem und bildet von einer hochdimensionalen Eingabe auf eine Zahl ab. In Formel 2.1 werden die hochdimensionalen Eingaben $x \in R^{|x|}$ durch D in das Intervall $(0, 1)$ abgebildet, wobei x eine Instanz aus dem x-dimensionalen Raum möglicher Eingaben ist (Creswell u. a. (2018)).

$$D : D(x) \rightarrow (0, 1) \tag{2.1}$$

Eine Ausgabe nahe Eins bedeutet, dass der Diskriminator mit hoher Wahrscheinlichkeit eine echte Domäneninstanz vorgelegt bekommen hat. Eine Ausgabe nahe Null bedeutet, dass D hat mit hoher Wahrscheinlichkeit eine generierte Domäneninstanz vorgelegt bekommen hat. Je besser G arbeitet, desto schwieriger wird es für D diese Klassifikation korrekt vorzunehmen (Goodfellow u. a. (2014)).

2.4.3 Generator (G)

Der Generator G ist ebenfalls ein neuronales Netz, mit der Aufgabe künstliche Domäneninstanzen zu generieren. G löst dabei ein ungleich schwierigeres Problem, da er eine niedrigdimensionale Eingabe auf eine hochdimensionale Ausgabe abbilden muss (Mescheder u. a. (2018)). Formeln 2.2 macht dieses deutlich: G bildet von einem Element des Raumes der Eingabedaten $z \in R^{|z|}$ nach $R^{|x|}$ ab. $R^{|z|}$ wird auch als *Latent Space* (dt. Latenter Raum) bezeichnet und ist eine komprimierte Darstellung des Zielraumes, der hier die Basis für den Generierungsprozess bildet (Creswell u. a. (2018)). $R^{|x|}$ ist der

Raum der Ausgaben (Zielraum) des Generators und entspricht dem Raum der Eingaben von D (vgl. 2.4.2).

$$G : G(z) \rightarrow R^{|x|} \tag{2.2}$$

Die Eingabe $z \in R^{|z|}$ kann dabei eine Zahl, ein Vektor oder ein Bild niedriger Auflösung repräsentieren. Wichtig ist, dass diese Eingaben randomisiert unterschiedlich sind, da ein fertig trainiertes neuronales Netz für gleiche Eingaben immer dieselben Ausgaben generiert (Rashid (2020), S.95). Daher sind randomisierte Eingaben die Basis für den Generierungsprozess und sichern zu, dass unterschiedliche Ausgaben generiert werden (Rashid (2020), S.98). In Abbildung 2.2 wird die randomisierte Eingabe als zufällige Bitmap dargestellt (orange).
G ist dann ausreichend trainiert, wenn D für beliebige Eingaben $D(x) = 0.5$ ausgibt (Goodfellow u. a. (2014), Creswell u. a. (2018)).
Hervorzuheben ist, dass G sich keine Trainingsdaten merkt und daraus Domäneninstanzen generiert (Rashid (2020), S.175). G bildet von der randomisierten Eingabe $z \in R^{|z|}$ auf eine Instanz ab, deren Wahrscheinlichkeitsverteilung möglichst der der Trainingsdaten entsprechen soll. G lernt diese Wahrscheinlichkeitsverteilung möglichst exakt nachzubilden. Die Art der Instanz (Bild, Text, Ton) ist für G dabei nicht relevant (Goodfellow u. a. (2014)).

2.4.4 Training eines GAN

GANs werden zyklisch trainiert. G und D sind neuronale Netze und passen die Gewichte ihrer Neuronen im Rahmen des Trainings an. Dazu werden folgende Schritte durchgeführt, die dem Standardtrainingsverfahren für neuronale Netze entsprechen:

1. Die Differenz zwischen der gewünschten Ausgabe des neuronalen Netzes und der tatsächlichen Ausgabe wird berechnet. Diese als Verlust (engl. Loss) bezeichnete Größe, kann z.B als *Mittlerer quadratischer Fehler* (MSE) oder *Binäre Kreuzentropie* (BCE) berechnet werden (Rashid (2020), S.167).

2. Mit dem *Gradientenabstiegsverfahren* wird berechnet, wie die Gewichte des Neuronalen Netzes angepasst werden müssen, damit der Verlust über das gesamte neuronale Netz kleiner wird (Nesterov (2013)). Das Verfahren nutzt dafür den *Backpropagation-Algorithmus*. Dieser passt mit Hilfe der Kettenregel Gewichte so

an, dass sich der Verlust verringert. Dabei wird eine Kombination aus Gewichten bestimmt, die den Gesamtverlust bestmöglich minimiert (Steinwendner und Schwaiger (2020), S.151).

Das Ziel ist es qualitativ hochwertige Instanzen einer Domäne zu generieren. Daher müssen G und D in einer bestimmten Trainingsschleife trainiert werden (Rashid (2020), S.80). Aus folgenden drei Schritten besteht das Training eines GAN:

1. D erhält eine echte Domäneninstanz und die Information, dass diese als echte Instanz klassifiziert werden muss mit einem Ausgabewert nahe 1. Mit Hilfe der zuvor beschriebenen zwei Schritte des Trainingsverfahrens für neuronale Netze, werden die Gewichte von D aktualisiert.

2. D erhält eine Ausgabe von G. Diese soll als unechte Domäneninstanz mit einem Wert nahe 0 klassifiziert werden. Die Gewichte von D werden aktualisiert. Da G in diesem Schritt aktiv wird, aber nicht trainiert werden soll, muss die Aktualisierung der Gewichte von G in diesem Schritt aktiv unterbunden werden.

3. G generiert eine Ausgabe mit der Vorgabe, dass D diese als echte Domäneninstanz klassifizieren soll. Je nach Feedback von D werden die Gewichte von G aktualisiert. Das Feedback von D an G ist der in D berechnete Verlust für die Eingabe von G.

Das Training ist beendet, wenn $D(x) = 0.5$ ausgibt, da D dann nicht mehr zwischen echten und generierten Instanzen unterscheiden kann. Außerdem muss die Wahrscheinlichkeitsverteilung der von G generierten Instanzen möglichst der der Trainingsdaten entsprechen (Goodfellow u. a. (2014)). Je ähnlicher sich die Wahrscheinlichkeitsverteilungen sind, desto höher ist die Güte der generierten Instanzen.
Das Training eines GAN ist wegen der großen Menge benötigter Trainingsdaten und Rechenleistung sehr ressourcenaufwendig (Nagisetty u. a. (2022)).
Weitere Probleme können sein, dass das Training nicht konvergiert. Das bedeutet D gibt nicht für jede Eingabe $D(x) = 0.5$ aus und das Training wird nie beendet (Mescheder u. a. (2018)). Auch kann das Training zu lange dauern, so dass die Qualität der generierten Instanzen wieder abnimmt, oder nur sehr ähnliche Instanzen generiert werden. Lösungen dafür können das Training in kleineren Chargen sein (Salimans u. a. (2016)) oder eine andere Architektur in Form sogenannter DCGANs (Radford u. a. (2016)).

2.5 Conditional GAN (cGAN)

Eine Sonderform des GAN ist das *Conditional GAN* (cGAN). Ein cGAN kann Domäneninstanzen einer bestimmten Klasse der Domäne erzeugen.
Seien Bilder von Ziffern die Domäne, dann generiert ein cGAN explizit nur eine bestimmte Ziffer, weil die Klassen der Domäne explizit als Parameter an das cGAN übergeben werden können. Ein Standard-GAN hat diese Option nicht, es generiert zufällig Ziffern aus den Klassen der Domäne 0-9.
Für die Realisierung müssen Anpassungen an der Architektur des GAN vorgenommen werden. G und D erhalten eine Extrainformation, bezeichnet als y (Mirza und Osindero (2014)). Die Extrainformation y kann ein Label sein, z.B. im Falle von Ziffern eine 'Drei' oder im Falle von Gesichtern die Information, dass die Haare blond sein sollen. Sind G und D neuronale Netze wird y der ersten Schicht des neuronalen Netzes hinzugefügt (Mirza und Osindero (2014), Chakraborty u. a. (2024)).
Für D gilt, dass es x und y als Eingaben erhält, im Falle von Bildern also das Bild selbst und eine geeignete Repräsentation der Extrainformation. G erhält als Eingabe eine Kombination aus z und y.
Die Formeln 2.1 und 2.2 müssen angepasst werden (Antipov u. a. (2017)):

$$D : D(x, y) \to (0, 1) \qquad G : G(z, y) \to R^{|x|} \tag{2.3}$$

Im Falle eines cGAN, das Ziffern generiert, wäre eine geeignete Darstellung von y ein Vektor der Länge zehn, in den die zu generierende Ziffer per One-Hot-Encoding codiert ist (Rashid (2020), S.156).
Das in 2.4.4 beschriebene Training muss ebenfalls angepasst werden:
Im ersten Schritt der Trainingsschleife erhält D y aus den Trainingsdaten. Für den zweiten und dritten Schritt wird jeweils ein zufälliges y erzeugt und G übergeben.
Nach Abschluss des Trainings kann das cGAN dann Instanzen der einzelnen Klassen der Domäne erzeugen (Mirza und Osindero (2014)).

3 Methoden

Die vorliegende Arbeit wurde auf Basis einer Literaturrecherche durchgeführt, um die theoretischen Grundlagen zur Beantwortung der Forschungsfrage (vgl. 1.1) zu erarbeiten. Während der Recherche wurden Methoden bzw. Frameworks identifiziert, die geeignet sind, die Ausgaben des Diskriminators nachzuvollziehen. Als Oberbegriff für die Methoden und Frameworks wird im Folgenden der Begriff *xAI-Methoden* verwendet, da die Methoden mindestens dem Anspruch von Explainable AI (vgl. 2.3.1) genügen sollen.
Dieses Kapitel beschreibt in 3.1 anhand welcher Kriterien die xAI-Methoden ausgewählt wurden.
In 3.2 wird die Erarbeitung des Proof of Concept (PoC) beschrieben, welches die Möglichkeiten und Grenzen der ausgewählten xAI-Methoden visualisieren und quantifizieren soll.

3.1 Ausgewählte Methoden zur Nachvollziehbarkeit

Auf Basis der Literaturrecherche werden Kriterien erarbeitet, denen die ausgewählten xAI-Methoden für eine potenzielle Eignung genügen müssen. Es werden fachlich und technische Kriterien unterschieden.
Folgende **fachliche Kriterien** wurden auf Basis von Molnar (2022) herangezogen:

- Die xAI-Methode ermöglicht mindestens lokale Nachvollziehbarkeit (vgl. 2.3.1).
- Die xAI-Methode ist inhaltlich in der Lage Daten in Form von Bildern zu verarbeiten und diese als solche wahrzunehmen.
- Die xAI-Methode kann den Beitrag eines Elements einer Instanz zu einem Gesamtergebnis bestimmen. Im Kontext der vorliegenden Arbeit ist das Gesamtergebnis die durch D vorgenommene Klassifikation eines Bildes. Ein Element einer Instanz ist ein Pixel eines Bildes.

Folgende **technische Kriterien** wurden zur Auswahl der xAI-Methoden herangezogen:

- Eignung der xAI-Methode Bilder zu verarbeiten. Bilder können in Form von Pixelwerten übergeben werden und müssen nicht aufwendig vor- oder aufbereitet werden.
- Aussagekraft der xAI-Methode ist gegeben (z.B. werden relevante Pixel markiert).
- Vertretbarer Implementationsaufwand der xAI-Methode (z.B. kein aufwendiges setzen von Umgebungsvariablen zur RAM-Freigabe o.Ä.).
- Vertretbarer Ressourcenaufwand der xAI-Methode (z.B. keine Notwendigkeit von GPUs oder TPUs).
- Das Ergebnis der xAI-Methode erlaubt die Berechnung von Metriken, um die xAI-Methoden zu vergleichen.

Anhand dieser Kriterien wurden folgende xAI-Methoden ausgewählt, um die Entscheidungen von D in einem GAN nachvollziehen zu können:

- SHapley Additive exPlanations (SHAP) (Lundberg und Lee (2017))
- Locally Interpretable Model-agnostic Explanation (LIME) (Ribeiro u. a. (2016))
- Integrated Gradients (Sundararajan u. a. (2017))
- Gradient-weighted Class Activation Mapping (GradCAM) (Selvaraju u. a. (2020))

3.2 Proof of Concept (PoC)

Um die Fragestellung nach der Eignung der ausgewählten vier xAI-Methoden zu beantworten, bedarf es eines Vergleichs der xAI-Methoden. Das PoC soll zeigen, ob und wie gut die xAI-Methoden geeignet sind, die Klassifikation von D in einem GAN nachzuvollziehen. Dabei beschränkt sich die vorliegende Arbeit auf GANs die Bilder generieren. Mit Hilfe des PoC soll visuell dargestellt werden, welche Parameter die Entscheidung von D beeinflussen. Außerdem soll ein quantitativer Vergleich der Güte der Ergebnisse der xAI-Methoden mit Hilfe von Metriken möglich sein. Auf Basis dieser Metriken sind weitere statistische Analysen der Ergebnisgüte möglich.

Da D nur während des Trainings des GAN (vgl. 2.4.4) aktiv ist, können nur dann die Ausgaben von D betrachtet werden. Daher eignet sich das PoC zusätzlich dazu, den Trainingsprozess eines GAN besser zu verstehen. Das Training von GANs ist schwer nachzuvollziehen und basiert teilweise auf unbewiesenen Annahmen und Heuristiken (Chakraborty u. a. (2024)). Instabiles und nicht konvergierendes Training sind nur einige der auftretenden Probleme (Arjovsky u. a. (2017), Mescheder u. a. (2018)). Eine Möglichkeit zum visuellen Verständnis ist daher wünschenswert, um so das Training von GANs stabiler zu machen.

3.2.1 Anforderung und Ziele

Ziel des PoC ist es, unter Zuhilfenahme von xAI-Methoden die Entscheidungen von D nachvollziehen zu können. Es soll ermittelt werden, welche dieser Methoden besonders gut für diese Aufgabe geeignet sind. Dabei sollen die in 2.3 definierten Kategorien Explainable-, Interpretable- und Responsible AI berücksichtigt werden. Es soll überprüft werden, welcher dieser Kategorien die jeweilige xAI-Methode genügt.
Weiterhin soll es möglich sein, den Verlauf des GAN-Trainings zu verstehen. Insbesondere die Entwicklung der Qualität der Bilder lässt sich so über mehrere Trainingsschleifen beobachten (vgl. 2.4.4). Auch könnte es möglich sein *Mode Collapse* (Rashid (2020), S.95) zu vermeiden, indem das Training zu einem visuell bestimmten Zeitpunkt beendet wird. Visuell in dem Sinne, dass ein Nutzer die Ergebnisse auf ausreichende Qualität prüft. Das Training würde dann manuell beendet werden und nicht erst, wenn die Bedingungen zum Beenden des Trainings erfüllt sind (vgl. 2.4.4). Bisher lässt sich Mode Collapse nur mit speziellen GAN-Architekturen wie den *Wasserstein-GANs* vollständig beheben (Arjovsky u. a. (2017)).

Um dieses Ziel zu erreichen und um einen fairen und aussagekräftigen Vergleich zu ermöglichen, werden folgende Anforderungen an das PoC formuliert:

1. Visuelle Darstellung des Ergebnisses.
2. Verfügbarkeit von Metriken für weitere Analysen.
3. Einheitliche Datenquelle für das Training des GAN.
4. Einheitliche Datenquelle für die xAI-Methoden resultierend aus dem gegebenen GAN.

5. Einheitliche Laufzeitumgebungen für alle Komponenten des PoC.

Die Anforderungen 3-5 erscheinen essenziell, um einen aussagekräftigen Vergleich der xAI-Methoden zu ermöglichen.

3.2.2 Architektur

Um die in 3.2.1 genannten Ziele zu erreichen, bedarf es einer einheitlichen Datenbasis mit der die xAI-Methoden arbeiten können. Das PoC hat daher eine zentrale Datenquelle in Form eines cGAN (vgl. 2.5), welches künstliche Bilder handgeschriebener Ziffern generiert.
Die generierten Instanzen werden zentral mit notwendigen Zusatzinformationen gespeichert. Von dort können die xAI-Methoden die generierten Instanzen laden und diese dann verarbeiten. Am Ende der Verarbeitung steht eine visuelle Aufbereitung der Ergebnisse, sowie eine Bestimmung der Ergebnisgüte mit geeigneten Metriken. Für die Auswertung steht ein Excel-Export bereit.

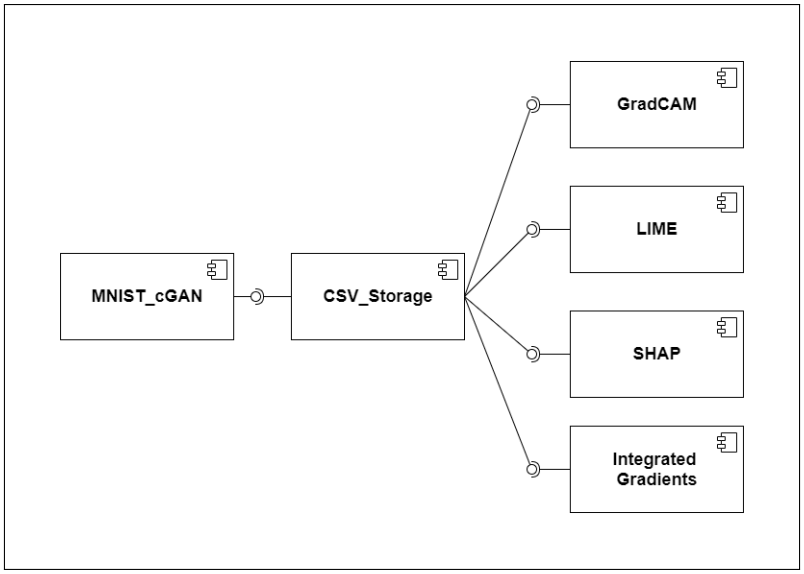

Abbildung 3.1: Komponentendarstellung des PoC (eigene Abbildung).

3.2.3 Datenbasis

Als Trainingsdatensatz für das cGAN dient der MNIST-Datensatz mit Bildern handgeschriebener Ziffern. Die Bilder sind monochrom und haben eine Größe von 28 x 28 Pixeln. Wenn die xAI-Methoden über eigene Klassifizierer verfügen, verwenden sie ebenfalls diesen Trainingsdatensatz.
Die generierten Bilder des cGAN sind ebenfalls monochrom und haben dasselbe Format wie die Trainingsbilder. Sie werden den xAI-Methoden als CSV-Datei zur Verfügung gestellt. Ein Bild ist eine Zeile in der CSV-Datei. Ein Zeile ist wie folgt aufgebaut:

- Erster Eintrag: Die durch D vorgenommene Klassifikation.
- Zweiter Eintrag: Das Label der Zahl welche das cGAN generieren sollte.
- 3 - 786 Eintrag: Die Pixelwerte des von G generierten Bildes.

Die Bilder werden nach der Ausgabe von D ($D(x, y)$) entsprechend Tabelle 3.1 kategorisiert und in CSV-Dateien gespeichert. So können Analysen unter Berücksichtigung der Ausgabe von D vorgenommen werden. Mit dem Datensatz 'Timeline' kann der Verlauf des Trainings analysiert werden.
Für das Zwischenspeichern der CSV-Dateien wurde Google Drive verwendet.

Name des Datensatzes	Zuordnungskriterium	Länge
`c_gan_numbers_low` ('generiert')	$0.0 < D(x, y) \leq 0.33$	55313
`c_gan_numbers_mid` ('unsicher')	$0.33 < D(x, y) \leq 0.66$	12462
`c_gan_numbers_up` ('echt')	$0.66 < D(x, y) \leq 1.0$	4225
`c_gan_numbers_timeline`	jedes 1000. Bild	720

Tabelle 3.1: Kategorisierte Datensätze, welche die xAI-Komponenten verwenden.

3.2.4 Allgemeine Implementationsdetails

Das PoC wurde mit Python in Colab-Notebooks umgesetzt. Colab bietet extern gehostete Jupyter-Notebooks mit einer einheitlichen Laufzeitumgebung für die einzelnen Komponenten des PoC. Außerdem bietet Colab einen umfangreichen Ressourcenpool. Das Training und die Verarbeitung ist so lokal unabhängig von der Leistungsfähigkeit der vorhandenen Hardware.
Zur Erstellung des PoC wurden folgende Bibliotheken benutzt:

- PyTorch (Erstellung neuronaler Netze)
- Keras (Nutzung externer Klassifizierer)
- TensorFlow (Verarbeitung und Transformation von Bildern)
- Pandas (Allokation von Rohdaten)
- NumPy (Verarbeitung von Arrays)
- Matplotlib (Erstellung von Diagrammen)

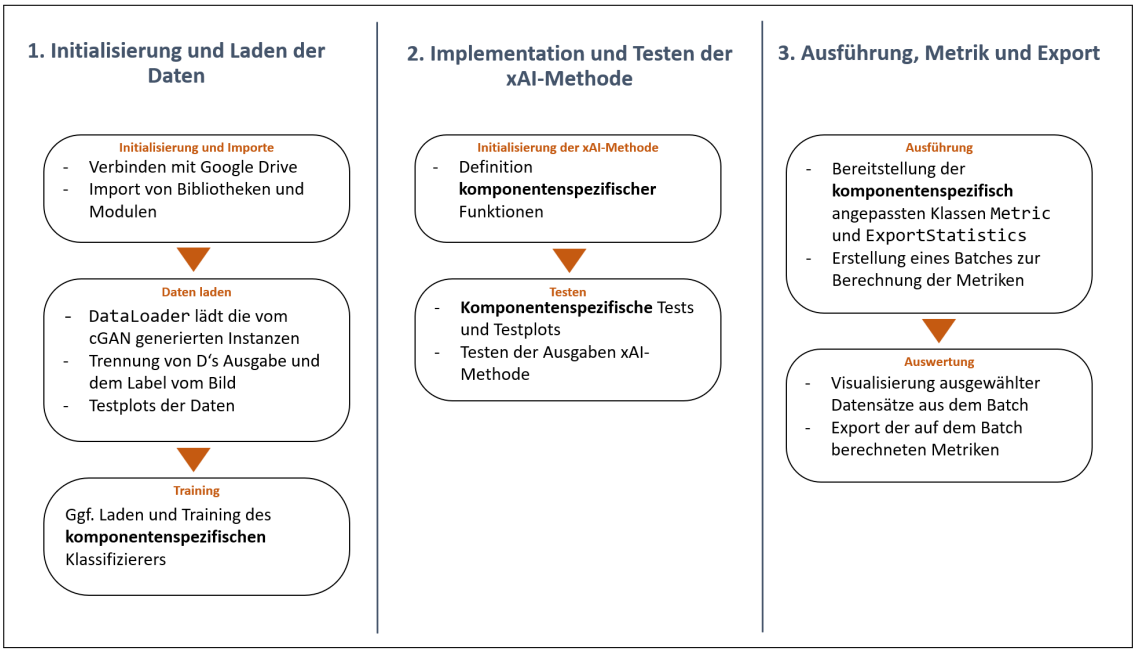

Abbildung 3.2: Standardabläufe die innerhalb aller xAI-Komponenten ausgeführt werden (eigene Abbildung).

Jede der vier Komponenten, die eine der xAI-Methoden realisiert, ist nach einem wiederkehrenden Schema aufgebaut. Abbildung 3.2 zeigt die drei wesentlichen Schritte die alle xAI-Komponenten abarbeiten, sowie den sequentiellen Ablauf innerhalb dieser Schritte. In jeder der vier xAI-Komponenten gibt es der xAI-Methode entsprechend unterschiedliche Implementationsdetails. Diese werden in den jeweiligen Abschnitten beschrieben. Dies betrifft alle Schritte, die in Abbildung 3.2 als **komponentenspezifisch** beschrieben sind.

3.2.5 LIME

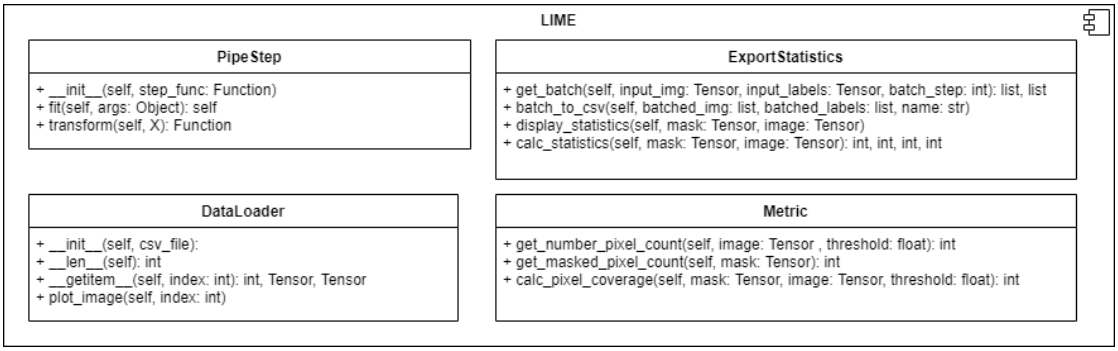

Abbildung 3.3: Klassen und Methoden von LIME (eigene Abbildung).

Abbildung 3.3 zeigt die Klassen und Funktionen die **LIME** besitzt. Die Komponente basiert auf dem Programmcode von Ribeiro u. a. (2016) und wurde entsprechend der Anforderungen des PoC angepasst.
Die visuelle Darstellung der Ergebnisse wurde so geändert, dass sie zusammen mit den Metriken angezeigt wird. Außerdem musste der Programmcode so angepasst werden, dass dieser mit den Bildern des cGAN aus der CSV-Storage arbeiten konnte. Als Datenquelle für den Klassifizierer von LIME wird der MNIST-Datensatz separat geladen.
Als zusätzliche Bibliotheken werden *Scikit-image* und *Sklearn* verwendet. LIME arbeitet mit RGB-Bildern. Mit Scikit-image werden monochrome Bilder in RGB-Bilder umgewandelt. Sklearn wird für die Pipeline zum Training des Klassifizierers verwendet. Folgende komponentenspezifische Schritte werden abgearbeitet, um die Entscheidungen von D mit LIME nachvollziehen zu können:

1. Der Datensatz zum Trainieren des Klassifizierers wird geladen. Mit der Klasse `PipeStep` wird eine Pipeline zur Verarbeitung der Trainingsbilder erstellt. Die Bilder werden zuerst in RGB-Bilder umgewandelt (Pixelwerte $0-1$). Anschließend werden die Bilder in eine eindimensionale Liste umgewandelt. Im letzten Schritt wird der Klassifizierer trainiert. LIME verwendet hier einen *Random Forest Classifier*. Tabelle 3.2 zeigt die Hyperparameter des verwendeten Random Forest Classifiers.

2. LIME wird importiert. Es können testweise Bilder des cGAN übergeben werden. Die Ergebnisse werden visualisiert. Es sind Visualisierungen für ein gewähltes Bild einzeln oder im Vergleich mit allen Labeln $(0-9)$ möglich. Alle Pixel die LIME für

eine Klassifzierung eines bestimmten Labels als relevant identifiziert hat, werden mit einer Maske über dem Bild kenntlich gemacht.

3. Die Klassen `Metric` und `ExportStatistics` verwenden zum Erkennen der Zahl im Bild einen Schwellenwert von 0.01 (`coverage_threshold`). Die für eine Confusion Matrix benötigten Parameter werden auf dieser Basis berechnet.

Die für die Metriken berechneten Parameter werden nach dem Export zur Bestimmung der Güte (Explainability Power) von LIME verwendet (vgl. 4.2).

Art	Variablenname	Wert
Anzahl Bäume	`n_estimators`	100
Maximale Tiefe	`max_depth`	unbegrenzt
Wahl des Splitattributs	`criterion`	Gini

Tabelle 3.2: Hyperparameter des Random Forest Classifiers.

3.2.6 SHAP

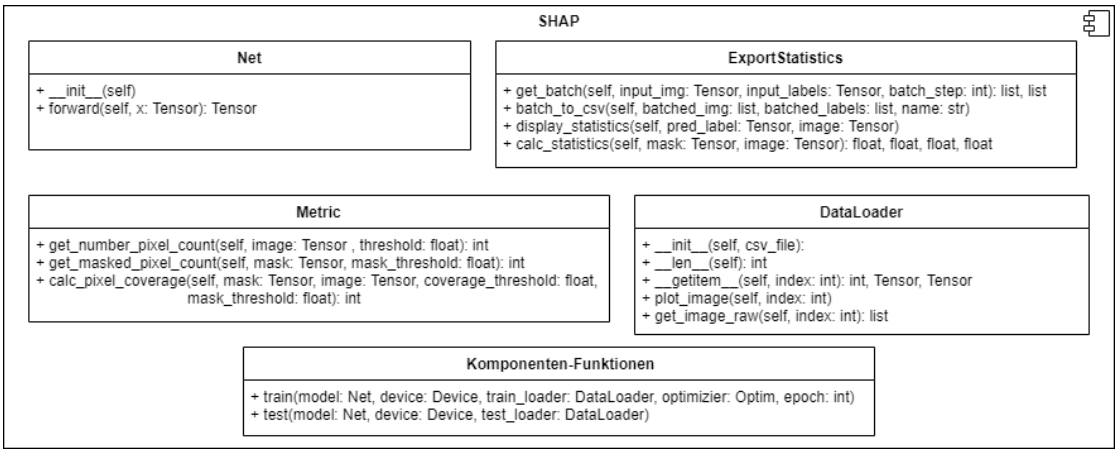

Abbildung 3.4: Klassen und Methoden von SHAP (eigene Abbildung).

Abbildung 3.4 zeigt, welche Klassen und Funktionen **SHAP** besitzt. Die Komponente basiert auf dem Programmcode und Framework von Lundberg (2018). Als Klassifizierer dient ein *PyTorch Deep Explainer*, der mit dem MNIST-Datensatz trainiert wird. Dieser Code wurde in das PoC eingearbeitet und entsprechend dessen Anforderungen angepasst. Für die visuelle Darstellung der Ergebnisse wurden folgende Änderungen vorgenommen:

Art	Variablenname	Wert
Anzahl Trainingsepochen	`num_epochs`	2
Angeforderte Recheneinheit	`device`	CPU
Convolutional Layers	`conv_layers`	2x `Conv2D`, 2x `MaxPool2D`, 1x `Dropout`
Aktivierung Conv. Layers	-	2x `ReLU`
Fully Connected Layers	`fc_layers`	2x `Linear`, 1x `Dropout`
Aktivierung F. C. Layers	-	1x `ReLU`, 1x `Softmax`
Optimierer	`optimizer`	Stochastischer Gradientenabstieg
Lernrate	`lr`	0.01
Verlustbestimmung	`loss`	Negativer Log Likelihood Verlust

Tabelle 3.3: Hyperparameter des PyTorch Deep Explainers. Alle Schichten und Aktivierungen sind PyTorch-Komponenten.

SHAP erklärt sowohl positive als auch negative Beiträge von Pixeln zur Gesamtklassifikation. Damit Vergleiche möglich bleiben, werden für die Metriken nur positive Beiträge berücksichtigt. Zur Umsetzung von SHAP wird zusätzlich die Bibliothek *Torchvision* verwendet, um Formate von Bildtensoren zu transformieren. Damit SHAP die Bilder des cGAN verarbeiten kann, werden diese in RGB-Bilder (Pixelwerte $0 - 1$) umgewandelt und die Achsen der Tensoren den Anforderungen von SHAP entsprechend vertauscht. Folgende komponentenspezifische Sequenzen werden abgearbeitet, um die Ausgaben von D mit SHAP nachzuvollziehen:

1. SHAP wird lokal installiert und anschließend importiert. Es ist **zwingend die Installation der Version 0.44.1 erforderlich**, da zum Zeitpunkt der Abgabe dieser Arbeit die aktuellste Version von SHAP Ergebnisse fehlerhaft visualisiert.

2. Die Klasse `Net` für den Pytorch Deep Explainer wird initalisiert. Mit den Funktionen `train()` und `test()` wird das Modell trainiert und verifiziert. Standardmäßig sind die in Tabelle 3.3 aufgeführten Hyperparamter ausgewählt. Diese können bei Bedarf angepasst werden.

3. Die Ergebnisse von SHAP können einzeln oder im Vergleich mit allen Labeln visualisiert werden. Dafür werden ein oder mehrere ausgewählte Bilder so transformiert, dass SHAP sie analysieren und anzeigen kann. Alle Pixel die SHAP für eine Klassifizierung eines bestimmten Labels als relevant identifiziert hat, werden mit einer Maske über dem Bild kenntlich gemacht.

4. `Metric` und `ExportStatistics` verwenden zum Erkennen der Zahl in einem Bild einen Schwellenwert von 0.01 (`coverage_threshold`) und zum Erkennen der positiven Beiträge der Maske einen Schwellenwert von 0.001 (`mask_threshold`). Dieser wurde heuristisch ermittelt.

Die für die Metriken berechneten Parameter werden nach dem Export zur Bestimmung der Güte (Explainability Power) von SHAP verwendet (vgl. 4.3).

3.2.7 Integrated Gradients (IG)

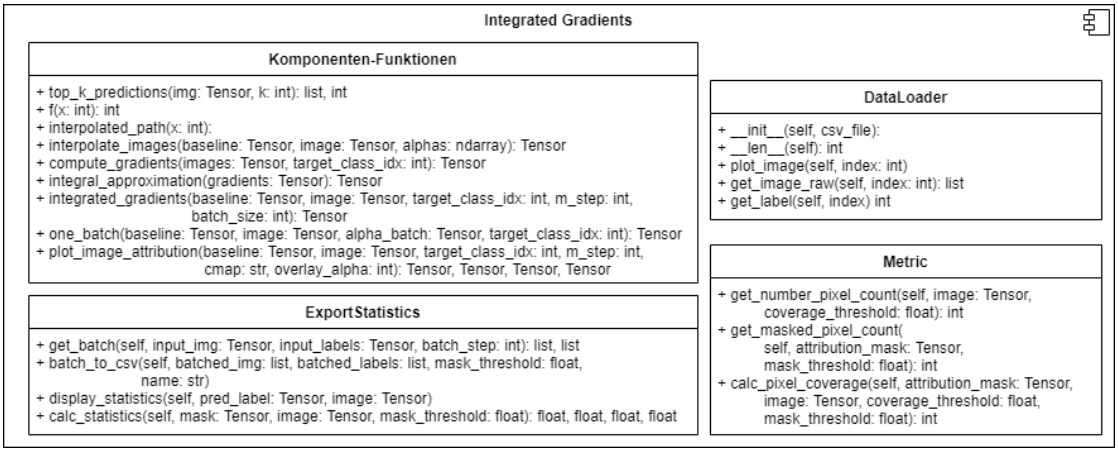

Abbildung 3.5: Klassen und Methoden von Integrated Gradients (eigene Abbildung).

Abbildung 3.5 zeigt die Funktionen und Klassen, die die Komponente **Integrated Gradients** verwendet. Im Folgenden wird die Abkürzung IG verwendet. Die Komponente basiert auf dem Programmcode von Daoust (2024) und Samoilescu (2024). Der Klassifizierer basiert wie bei SHAP auf einem neuronalen Netz. Für Aufbau und Training des Klassifizierers wird die Bibliothek Keras verwendet. Der Klassifizierer wird nach dem Training lokal gespeichert und kann so ohne erneutes Training wiederverwendet werden. Der Programmcode von Daoust (2024) musste angepasst werden, um mit dem Klassifizierer von Samoilescu (2024) kompatibel zu sein. Der ursprüngliche Klassifizierer diente der Objekterkennung und konnte keine Zahlen erkennen. Weitere Änderungen mussten an den Funktionen zur Berechnung der IG vorgenommen werden, damit diese mit den korrekten Labeln (Ziffern $0-9$) arbeiten und nicht mit beliebigen Objektlabeln (Revar (2020)). Damit IG die Bilder des cGAN verarbeiten kann, mussten diese in RGB-Bilder (Pixelwerte $0-9$) transformiert werden. Die Klassen `Metric` und `ExportStatistics`

wurden dahingehend angepasst, dass sie einen Schwellenwert für die Maske verwenden, ab dem ein Pixel als maskiert gilt (`mask_threshold`). Ebenso gibt es einen Schwellenwert zur Erkennung der eigentlichen Zahl (`coverage_threshhold`).
In der Komponente werden folgende komponentenspezifische Schritte abgearbeitet, um die Entscheidungen von D mittels IG nachvollziehen zu können:

1. Tensorflow wird in **Version 2.25.1** installiert um Kompatibilitätsprobleme zu vermeiden. Die Trainingsdaten für den Klassifizierer werden geladen. Wahlweise kann der Klassifizierer neu trainiert werden oder ein trainiertes Modell von einem vorangegangenen Durchlauf verwendet werden. Es werden sechs Trainingsepochen durchgeführt. Dieser Wert ist anpassbar (Details siehe Tabelle 3.4).

2. Die Gradienten können testweise für ein Bild berechnet und visualisiert werden. Dafür werden die in Abbildung 3.5 beschriebenen *Komponenten-Funktionen* in der abgebildeten Reihenfolge ausgeführt. Die Funktion
   `def plot_img_attributions()` stellt am Ende die Maske bereit.

3. `Metric` und `ExportStatistics` verwenden zum Berechnen der Parameter für die Confusion Matrix einen `coverage_threshold` von 0.001 und einen `mask_threshold` von 1.0e-07. Diese Werte wurden heuristisch als optimal ermittelt.

Die für die Metriken berechneten Parameter werden nach dem Export zur Bestimmung der Güte (Explainability Power) von IG verwendet (vgl. 4.4).

Art	Variablenname	Wert
Anzahl Trainingsepochen	`epochs`	6
Layers	`x`	1x `InputLayer` 2x `Conv2D`, 2x `MaxPooling2D`, 3x `Dropout`, 1x `Flatten` , 1x `Dense`
Aktivierung	-	Softmax
Optimierer	`optimizer`	Adam
Lernrate	–	0.001 (Standard für Keras-Adam)
Verlustbestimmung	`loss`	Kategorische Kreuzentropie

Tabelle 3.4: Hyperparameter des Keras MNIST Classifiers. Alle Schichten und Aktivierungen sind Keras-Komponenten.

3.2.8 GradCAM

Abbildung 3.6 zeigt den Aufbau der Komponente **GradCAM**. Diese Komponente basiert

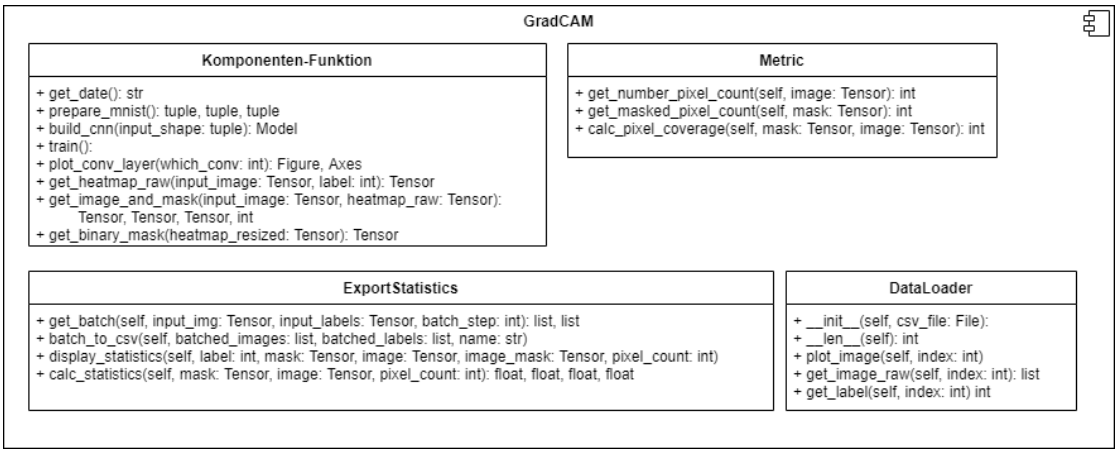

Abbildung 3.6: Klassen und Methoden von GradCAM (eigene Abbildung).

auf Programmcode von Kang (2019) und Gildenblat (2019). Sie benötigt zwei neuronale Netze. Das eine klassifiziert die vorgelegten Bilder, das andere erstellt auf Basis der Convolutional Layers des Klassifizierers die Feature Maps (Selvaraju u. a. (2020)) mit den dazugehörigen Aktivierungen (vgl. 4.5). Diese benötigt GradCAM, um die für die Klassifizierung relevanten Bildbereiche zu identifizieren. Die neuronalen Netze bestehen aus Keras-Komponenten. Als zusätzliche Bibliotheken werden *cv2* und *time* verwendet. Erstere dient der Umwandlung in RGB-Bilder, letztere zum Persistieren trainierter Modelle.
GradCAM arbeitet mit RGB-Bildern (Pixelwerte $0 - 255$, ganzzahlig), daher werden die monochromen Bilder des cGAN an geeigneter Stelle transformiert.
Eine Besonderheit bei GradCAM ist die Gestalt der Maske. GradCAM berücksichtigt nur positive Beiträge zur Klassifikation, negative werden heraus gefiltert (ReLU, vgl. 4.4). Für die Klassifikation unwichtige Bereiche werden blau hinterlegt. Je ausschlaggebender ein Bereich für die Klassifikation ist, desto grüner bzw. roter wird er hinterlegt (entsprechend dem RGB-Farbverlauf von Blau über Grün nach Rot). Alle Pixel die nicht dem Basisfarbton entsprechen (± 40), werden als relevant betrachtet. Die resultierende binäre Maske dient der besseren Vergleichbarkeit der Methoden, da nicht alle xAI-Methoden Abstufungen für die Relevanz der Pixel bereitstellen. Die Klassen `Metric` und `ExportStatistics` verwenden intern diesen Basiswert und benötigen daher ebenfalls keine Schwellenwerte für die Erkennung der Maske und der Zahl im Bild (vgl. Abbildung

3.6).
Folgende komponentenspezifische Schritte werden abgearbeitet, um die Ausgaben von D mit GradCAM nachvollziehen zu können:

1. Die Funktionen zum Training des Klassifizierers werden definiert (vgl. Tabelle 3.5). Die Klasse `HParam` wird definiert und hält alle Parameter zum Trainieren des Klassifizierers (Größe des Trainingsbatches, Anzahl der Klassenlabel, Anzahl der Trainingsepochen).

2. Erzeugung eines Modells (vgl. Tabelle 3.6), welches die Ausgaben der beiden Convolutional Layers des Klassifizierers auswertet und die Aktivierung für relevante Merkmale bestimmt, die zu einer bestimmten Klassifizierung geführt haben (Selvaraju u. a. (2020)).

3. Die Aktivierungen der Convolutional Layers können testweise visualisiert werden, ebenso eine Testausgabe von GradCAM für ein ausgewähltes Bild.

4. Definition der Funktion zur Bestimmung einer initialen Heatmap `get_heatmap_raw()`, der Funktion zur Erstellung der Ausgabe von GradCAM `get_image_and_mask()` und einer Funktion zur Erstellung der oben beschriebenen binären Maske `get_binary_mask()`.

5. Batchweise Auswertung von GradCAM unter Berücksichtigung der oben beschriebenen, komponentenspezifischen Besonderheiten.

Die für die Metriken berechneten Parameter werden nach dem Export zur Bestimmung der Güte (Explainability Power) von GradCAM verwendet (vgl. 4.5).

Art	Variablenname	Wert
Anzahl Trainingsepochen	`epochs`	1
Layers	`x, h, y`	1x `InputLayer` 2x `Conv2D`, 1x `MaxPool2D`, 2x `Dropout`, 1x `Flatten` , 2x `Dense`
Aktivierung	`activation`	ReLU, ReLU, ReLU, Softmax
Optimierer	`optimizer`	Adadelta
Lernrate	`learn_rate`	0.001
Verlustbestimmung	`loss`	Kategorische Kreuzentropie

Tabelle 3.5: Hyperparameter des Netzes, das die Klassifizierung vornimmt.

Art	Variablenname	Wert
Anzahl Trainingsepochen	`epochs`	1
Layers	`x, h`	1x `InputLayer` 2x `Conv2D`
Aktivierung	`activation`	ReLU, ReLU

Tabelle 3.6: Hyperparameter des Netzes, welches die Aktivierungen zu den Feature Maps erstellt. Das Modell ist dasselbe wie für den Klassifizierer, daher sind nur die geänderten Parameter in der Tabelle aufgeführt.

3.2.9 MNIST_cGAN

Abbildung 3.7 zeigt, welche Klassen und Funktionen das **MNIST_cGAN** (vgl. 2.5) besitzt. Das cGAN basiert auf Programmcode von Rashid (2024) und wurde so modifiziert,

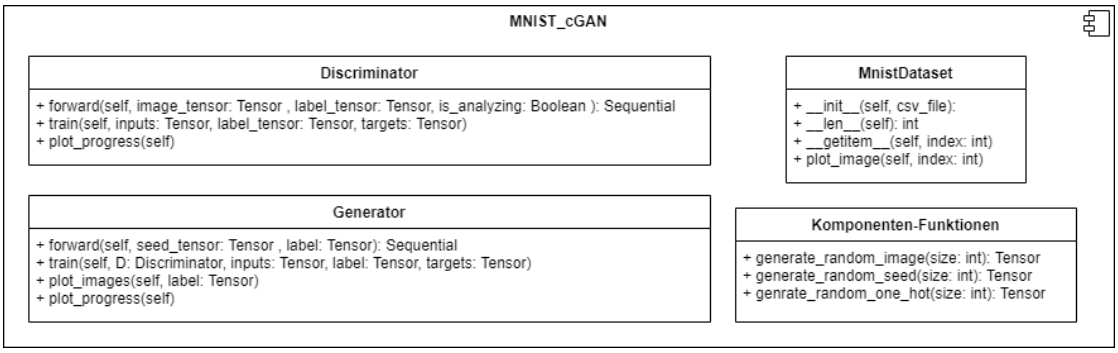

Abbildung 3.7: Klassen und Methoden des MNIST_cGAN (eigene Abbildung).

dass in der Trainingsschleife die Ausgabe von D zur von G erzeugten Instanz mit dem Label und der Instanz gespeichert werden. Dafür kann in der `forward()`-Methode des Diskriminators der Parameter `is_analyzing` gesetzt werden (vgl. Abb. 3.7). D und G besitzen dieselben Hyperparameter (vgl. Tabelle 3.7). Die Anzahl der Neuronen in der Eingabe und Ausgabeschicht ist aber unterschiedlich. D hat 794 Eingabeneuronen und 1 Ausgabeneuron. G hat 110 Eingabeneuronen und 784 Ausgabeneuronen. In der mittleren Schicht werden alle Ausgaben vor dem Weiterleiten normalisiert. Folgende Sequenzen werden abgearbeitet, um die Instanzen zusammen mit dem Label und der Ausgabe von D bereitzustellen:

1. `MnistDataset` lädt den Trainingsdatensatz. Das Label wird separiert. Ein Tensor für das Ziellabel wird erstellt. Ein weiterer Tensor enthält die von $0-255$ auf $0-1$ normalisierten Pixelwerte.

2. Training des cGAN (vgl. 2.4.4 und 2.5). Erzeugte Instanzen, Label und die Ausgabe von D werden zwischengespeichert.

3. Je nach Ausgabe von D (vgl. 3.2.3), werden die zwischengespeicherten Datensätze in eine CSV-Datei exportiert.

Auf dieser Datenbasis können die xAI-Methoden verglichen werden.

Art	Variablenname	Wert
Anzahl Trainingsepochen	`epochs`	12
Layers	–	3x `Linear`
Aktivierung	–	Leaky-ReLU, Sigmoid
Optimierer	`optimiser`	Adam
Lernrate	`lr`	0.0001
Verlustbestimmung	`loss_function`	Binäre Kreuzentropie

Tabelle 3.7: Hyperparameter von Diskriminator und Generator des cGAN. Es werden PyTorch-Komponenten verwendet.

4 Ergebnisse

Dieses Kapitel legt die Ergebnisse des erarbeiteten Proof of Concept dar. Dies geschieht auf Basis der aktuellen Implementation des PoC, dem aktuellen Stand der verwendeten Bibliotheken sowie der in Kapitel 3 beschriebenen Klassifizierer, deren Hyperparametern und der ausgewählten Stichproben aus dem Training (vgl. Tabelle 3.1).
Zu jeder der vier identifizierten xAI-Methoden werden zunächst die theoretischen Ergebnisse erklärt. Es wird die Frage beantwortet, wie die xAI-Methode in der Lage ist, für die Klassifizierung relevante Instanzelemente zu identifizieren. Anschließend werden die praktischen Ergebnisse beschrieben. Die visuelle Erklärung der xAI-Methode und die Auswertung der Metriken wird erklärt. Zudem wird erläutert warum die jeweilige xAI-Methode die in 3.1 festgelegten fachlichen und technischen Anforderungen erfüllt.
In 4.6 wird beschrieben, wie die in 3.2.1 gestellten Anforderungen an das PoC umgesetzt werden.
Der Programmcode findet sich in folgendem GitHub-Repository
https://github.com/wuschee/ba_code.

4.1 Allgemeines

Im Rahmen der Nachvollziehbarkeit von KI ist es essentiell, dass Nutzende die Ergebnisse der xAI-Methode adäquat präsentiert bekommen. Um die Beiträge von Pixeln für eine Klassifikation darzustellen bieten sich Salienzkarten an (Gohel u. a. (2021)). Salienzkarten heben die Bereiche des Bildes hervor, die von der xAI-Methode als wesentlich für die Klassifikation bestimmt werden (Molnar (2022)).
Salienzkarten sind besonders geeignet, wenn eine oder mehrere Klassen je Bild zu identifizieren sind (Gohel u. a. (2021)), was bei den verwendeten Bildern von Ziffern der Fall ist. Daher erfolgt die visuelle Darstellung mit Salienzkarten.
Die Ergebnisse der Metriken werden in einer *Confusion Matrix* zusammengefasst, in der folgende Parameter aufgeführt sind:

- TP (True Positive) - Quote der Pixel, die zur Ziffer gehören und von der xAI-Methode korrekt als solche erkannt werden.
- FN (False Negative) - Quote der Pixel, die zur Ziffer gehören und nicht von der xAI-Methode als solche erkannt werden.
- FP (False Positive) - Quote der Pixel, die nicht zur Ziffer gehören, aber von der xAI-Methode als zur Ziffer zugehörig erkannt werden.
- TN (True Negative) - Quote der Pixel, die nicht zur Ziffer gehören und von der xAI-Methode als solche erkannt werden.

Die Metriken werden in jeder Komponente mit der Methode `calc_statistics()` berechnet. Die dazugehörigen Formeln und Variablen finden sich im Programmcode. Die Quoten von TP und TN sollen idealerweise gegen eins gehen, die von FN und FP gegen null. Daher sind Erstere in den Matrizen grün, Letztere rot hinterlegt.
Es wird sich primär auf die korrekte Erkennung der Ziffern (TP) fokussiert, daher wird TP für jede xAI-Methode mit D in Relation gesetzt. So wird ein Zusammenhang zwischen der Diskriminatorentscheidung und der Güte der Erkennung der Ziffer hergestellt. Die durchschnittliche Ausgabe von D beträgt über das gesamte Training hinweg 0,205.

4.2 LIME

LIME wird mit einer importierten Implementation von Ribeiro u. a. (2016) umgesetzt. Um die für eine bestimmte Klassifikation relevanten Pixel zu bestimmen, arbeitet LIME nach folgendem Prinzip:
Für jedes Bild wird ein eigenes Surrogatmodell trainiert, um die Klassifikation des Bildes nachvollziehbar zu machen. Dieses Surrogatmodell muss dafür 'interpretierbar' sein (Ribeiro u. a. (2016)). Das am besten interpretierbare Modell $g \in G$ wird mit Formel 4.1 bestimmt.

$$\xi(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{4.1}$$

G ist die Menge möglicher Surrogatmodelle, g ist das Modell, welches die Ausgaben des Originalmodells f am besten in einer lokalen Umgebung π_x für einen Datenpunkt x approximiert. L berechnet die Güte der Approximation zwischen f und g in der Umgebung π_x. Die Summe von L und der Komplexiät Ω von g wird dann minimiert. Da g interpretierbar ist (Molnar u. a. (2020)), kann für jedes Element einer Instanz überprüft werden,

wie sich eine Änderung eines Elements auf die Klassifikation der Instanz auswirkt. Ist g eine lineare Regressionen, dann gibt der Koeffizient der Elementvariablen an, wie stark eine Änderung um eins die Klassifikation beeinflusst (Fahrmeir (2016)). Im Falle von Bildern, wird dies über das aktivieren und deaktivieren von Pixelgruppen realisiert, sogenannter 'Super-Pixel' (Ribeiro u. a. (2016), Molnar u. a. (2020)). Auf deren Basis wird bestimmt, welche Pixel relevant für die Klassifikation sind. Die Klassifiktion des Originalmodells wird so durch die Interpretation des lokalen Modells nachvollziehbar gemacht (Molnar u. a. (2020)).
Aus der beschriebenen Funktionsweise von LIME lässt sich ableiten, dass LIME lokale, aber keine globale Nachvollziehbarkeit bietet. Da LIME beliebige Modelle erklären kann (Ribeiro u. a. (2016)) können auch Bilder und deren Klassifikation nachvollziehbar gemacht werden. LIME ist ebenfalls in der Lage, Beiträge einzelner Pixel für die Klassifikation festzustellen (Ribeiro u. a. (2016)). Damit erfüllt LIME die in Abschnitt 3.1 festgelegten **fachlichen Kriterien**.

Die verwendete Implementation ist ohne aufwendiges Vorbereiten der Bilder nutzbar und durch das Hervorheben relevanter Pixel können die Ergebnisse von LIME visualisiert werden. Zudem benötigt LIME keine besonderen Ressourcen (GPU oder TPU) und ist verhältnismäßig leicht umzusetzten (Molnar (2022)). Da die Maske separat verfügbar ist, können Metriken auf Basis hervorgehobener Pixel bestimmt werden. Damit sind die in Abschnitt 3.1 festgelegten **technischen Kriterien** erfüllt.

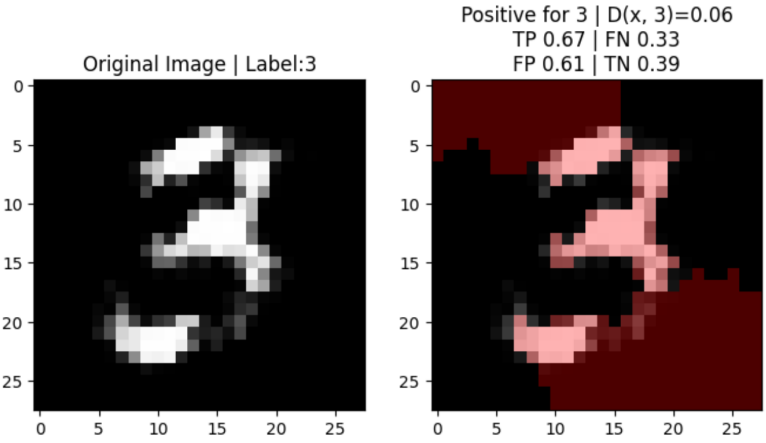

Abbildung 4.1: Die Maske markiert alle Pixel rot, welche für die Klassifizierung als 'Drei' ausschlaggebend sind (eigene Abbildung).

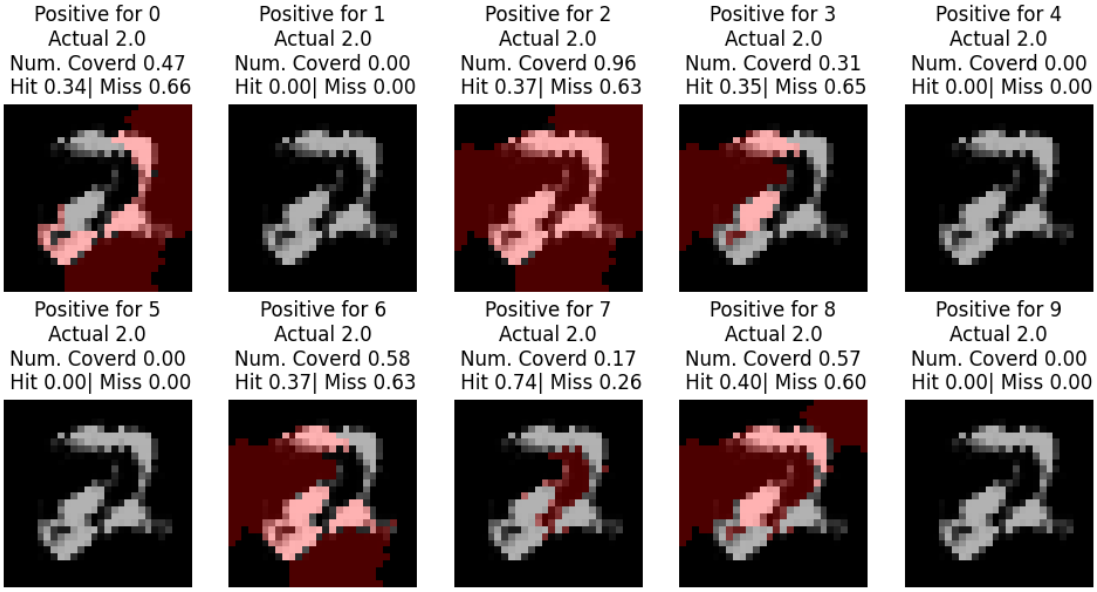

Abbildung 4.2: Visualisierung die hervorhebt, welche Pixel für das ausgewählte Bild unter Berücksichtigung aller Label relevant sind (eigene Abbildung).

Visuelle und statistische Ergebnisse

Im Falle von LIME ist keine Salienzkarte wie in 4.1 beschrieben verfügbar. Stattdessen wird eine einfarbigen Maske über ein ausgewähltes Bild gelegt. Es kann daher nicht abgeleitet werden, wie wichtig einzelne Pixel relativ zu anderen für die Klassifikation sind. Für die Darstellung wie in Abbildung 4.1 muss ein konkretes Bild per Index aus einem Datensatz ausgewählt werden. Neben der visuellen Darstellung werden auch die Metriken und die Ausgabe von D ($D(x, y)$) für das gewählte Bild angezeigt.

LIME bietet zusätzlich einen Vergleich des gewählten Bildes mit allen möglichen Labeln. Die Metriken sind in diesen Plots ebenfalls verfügbar (vgl. Abbildung 4.2).

Es ist nicht möglich und nicht zielführend, tausende Instanzen parallel zu visualisieren. Daher können für jeden der Datensätze (vgl. Tabelle 3.1) , über eine hinreichend große Anzahl an Bildern, die Metriken berechnet werden. So kann bestimmt werden, wie zuverlässig LIME die Ziffern in den Bildern identifiziert. Es ergibt sich folgende Auswertung:

Abbildung 4.3 zeigt die Ergebnisse über den gesamten Trainingsverlauf. Über das gesamte Training hinweg, ist LIME in der Lage 81% der Pixel, die zur Ziffer gehören korrekt zu erkennen (TP). 18% der Pixel, die zur Ziffer gehören werden nicht erkannt (FN).

55% der Pixel, die nicht zur Ziffer gehören, werden trotzdem von LIME als zur Ziffer zugehörig erkannt (FP). 44% der Pixel, die nicht zur Ziffer gehören werden korrekt als nicht zu dieser zugehörig erkannt (TN). LIME erreicht einen hohen Wert bezüglich der Erkennung von Pixeln, die zur Ziffer gehören (TP), allerdings zu dem Preis, dass viele nicht zur Ziffer gehörige Pixel ebenfalls als solche erkannt werden (FP). Diese Werte sind über den gesamten Trainingsverlauf und alle zehn möglichen Klassenlabel gemittelt.

Nun werden die Erkennungsraten der Datensätze, die nach der Ausgabe von D sortiert werden gemäß Tabelle 3.1 betrachtet:

Trainingsverlauf		Von LIME bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,81 TP	FN 0,18
	Pixel gehört nicht zur Ziffer	0,55 FP	TN 0,44

Abbildung 4.3: **LIME**: Confusion Matrix für die Erkennung der Bildpixel im Trainingsverlauf (eigene Abbildung).

'echt'		Von LIME bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,84 TP	FN 0,16
	Pixel gehört nicht zur Ziffer	0,57 FP	TN 0,43

Abbildung 4.4: **LIME**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert größer als 0.66 klassifiziert wurden (eigene Abbildung).

Es zeigt sich, dass der Datensatz mit den Bildern deren Ziffern von D als 'generiert' klassifiziert werden, die niedrigste Erkennungsrate (TP) von 82% hat (vgl. Abbildung 4.6). Dieser Wert liegt 2 bzw. 3 Prozentpunkte unter den Datensätzen deren Ziffern als 'echt' klassifiziert wurden (vgl. Abbildung 4.4) bzw. die D als 'unsicher' klassifiziert hat (vgl. Abbildung 4.5).

'unsicher'		Von LIME bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,85 TP	FN 0,15
	Pixel gehört nicht zur Ziffer	0,58 FP	TN 0,42

Abbildung 4.5: **LIME**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert zwischen 0.66 und 0.33 klassifiziert wurden (eigene Abbildung).

'generiert'		Von LIME bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,82 TP	FN 0,18
	Pixel gehört nicht zur Ziffer	0,57 FP	TN 0,43

Abbildung 4.6: **LIME**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert kleiner als 0.33 klassifiziert wurden (eigene Abbildung).

4.3 SHAP

SHAP wird mit der Implementation von Lundberg und Lee (2017) umgesetzt. SHAP wird dazu importiert und ermittelt für die Klassifikation relevante Elemente einer Instanz. Für Bilder macht sich SHAP das von LIME bekannte Konzept der 'Super-Pixel' zu nutze (Molnar (2022)). Der Beitrag ϕ_i (Shapley-Value (Shapley (1953))) der i-ten Gruppe von Pixeln zur Klassifikation wird mit Formel 4.2 bestimmt:

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!(M-|z'|-1)!}{M!}[f_x(z') - f_x(z' \backslash i)] \tag{4.2}$$

Für ein Bild x und das klassifizierende Modell f ergibt sich ϕ_i aus der Summe über die Potenzmenge x' der Menge aller 'Super-Pixel' in die ein Bild unterteilt wird. z' ist eine beliebige Teilmenge von x', die Koalition genannt wird. Eine Koalition enthält aktivierte und nicht aktivierte 'Super-Pixel', codiert mit 0 oder 1 für nicht aktiv bzw. aktiv. Mit der Differenz aus $[f_x(z') - f_x(z' \backslash i)]$ wird der Einfluss des i-ten 'Super-Pixels' innerhalb einer Koalition z' bestimmt. Dieser Wert wird mit dem davor stehenden Bruch gewichtet.

Die Gewichtung erfolgt mit der Gesamtzahl existierender 'Super-Pixel' M im Bild und der Anzahl $|z'|$ der betrachteten Teilmenge von 'Super-Pixel' in z' (Lundberg und Lee (2017), Molnar (2022)). So wird sichergestellt, dass die Bedeutung eines 'Super-Pixels' anders gewichtet wird, je nach dem ob eine große oder kleine Menge an 'Super-Pixeln' beeinflusst wird. Dementsprechend ist die Wichtigkeit dieses 'Super-Pixels' innerhalb der Koalition größer oder kleiner.

Mit diesen Eigenschaften bietet SHAP sowohl lokale Erklärbarkeit als auch globale. Von Letzterer wird kein Gebrauch gemacht. SHAP ist inhaltlich in der Lage Bilder zu verarbeiten, da SHAP modellagnostisch ist (Lundberg und Lee (2017)). SHAP kann die Beiträge einzelner Pixel bzw. 'Super-Pixel' zur Klassifikation des Bildes bestimmen. Dazu wird analog zu LIME das Konzept der 'Super-Pixel' verwendet (Molnar (2022)). Es wird für alle möglichen Kombinationen von Super-Pixeln überprüft, wie diese zur Klassifikation des Bildes beitragen. Damit erfüllt SHAP alle in Abschnitt 3.1 bestimmten **fachlichen Kriterien**.

Die verwendete Implementation ist mit leichten Anpassungen an den Dimensionen der RGB-Bilder nutzbar. Zur Nutzung von SHAP sind keine besonderen Hardware-Ressourcen notwendig, obwohl die Berechnung der Shapley-Werte sehr rechenaufwendig ist (Lundberg und Lee (2017)). Eigene Experimente (Nutzung von GPU bzw. TPU) haben gezeigt, dass die Verwendung von GPUs die Rechenzeit für große Batches von Bildern leicht verringert. SHAP ist ausreichend aussagekräftig, da relevante Pixel mit einer Maske hervorgehoben werden. Die Maske ist separat verfügbar, so können Metriken auf Pixelbasis bestimmt werden. Damit werden auch die in 3.1 geforderten **technischen Kriterien** von SHAP erfüllt.

Visuelle und statistische Ergebnisse

SHAP hebt die für die Klassifikation relevanten Pixel mit detaillierten Salienzkarten hervor. Dafür wird eine mehrfahrbige Maske über das klassifizierte Bild gelegt. Für die Klassifikation ausschlaggebende Pixel werden rot hinterlegt. Pixel, die einen negativen Einfluss auf die Klassifikation haben, werden blau hinterlegt. Letztere werden von den Metriken nicht berücksichtigt, da nicht alle xAI-Methoden negative Einflüsse bestimmen können. Je wichtiger ein Pixel ist, desto intensiver ist es eingefärbt. Dies ist möglich, da die Intensität aus der Summe der Shapley-Values je Pixel über alle Farbkanäle resultiert (Lundberg und Lee (2017)). Für die Darstellung in Abbildung 4.7 muss ein konkretes

Bild per Index aus einem Datensatz (vgl. Tabelle 3.1) gewählt werden. Die Metriken werden unter dem Bild dargestellt, da es nicht möglich ist sie in den SHAP-spezifischen Plot zu integrieren. Abbildung 4.8 zeigt einen zusätzlichen Plot, der für ein Bild einen

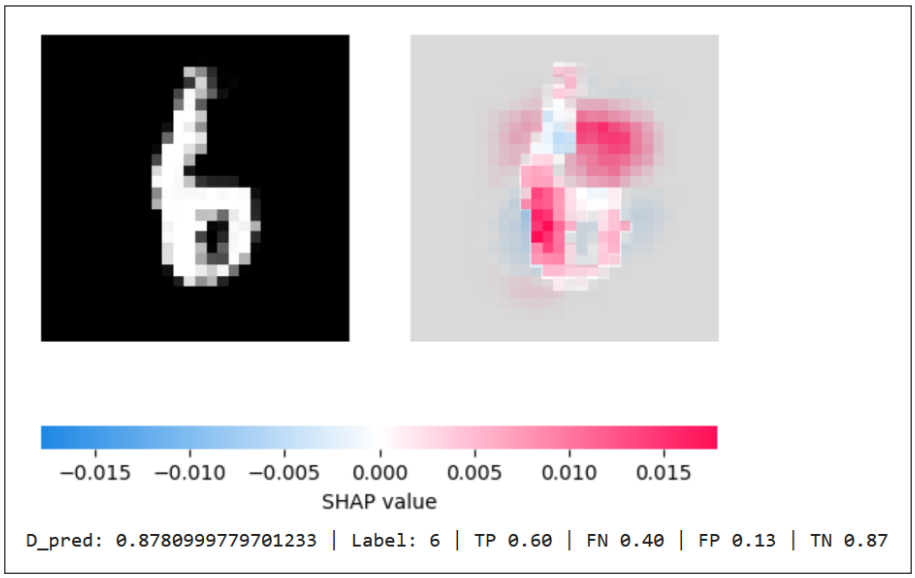

Abbildung 4.7: Visualisierung mit Salienzkarte (rechts) die hervorhebt, welche Pixel für die Klassifizierung als 'Sechs' ausschlaggebend sind (eigene Abbildung).

Vergleich mit allen möglichen zehn Labeln visualisiert. Zur Bestimmung der Güte werden

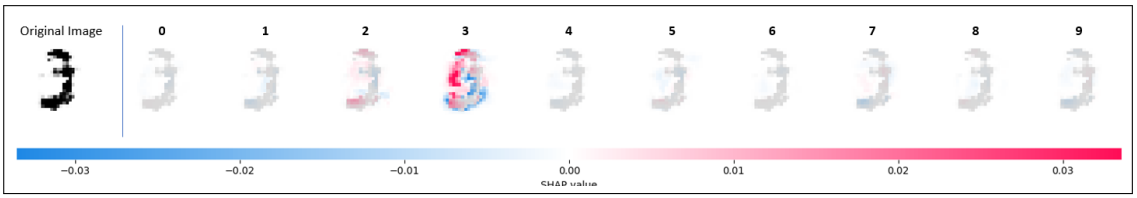

Abbildung 4.8: Visualisierung mit Salienzkarte die hervorhebt, welche Pixel für die Klassifizierung als 'Drei' ausschlaggebend sind unter Berücksichtigung aller Label (eigene Abbildung).

die Metriken auf einer hinreichend großen Datenmenge für jeden der vier exportierten Datensätze berechnet (vgl. Tabelle 3.1). Folgende Ergebnisse sind für diese vier Datensätze bestimmt worden:
Über das gesamte Training hinweg betrachtet, hat SHAP 54% der zur Ziffer gehörigen Pixel erkannt (TP). 46% der Pixel, die zu einer Ziffer gehören, werden nicht als solche erkannt (FN). 11% der Pixel, die nicht Teil der Ziffer auf einem Bild sind, werden fälschlicherweise als zur Ziffer zugehörig erkannt (FP). 89% der Pixel, die nicht zu einer Ziffer

gehören, werden korrekt als solche erkannt (TN). Etwas mehr als die Hälfte aller zur Ziffer gehörigen Pixel werden korrekt identifiziert. Es werden nur sehr wenige Pixel, die nicht zur Ziffer gehören, fälschlicherweise als zur Ziffer zugehörig erkannt.
Die folgenden Abbildungen zeigen die Erkennungsraten für die einzelnen Datensätze, die wie in Tabelle 3.1 sortiert werden: Die Bilder welche D als 'generiert' klassifiziert

Trainingsverlauf		Von SHAP bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,54 TP	FN 0,46
	Pixel gehört nicht zur Ziffer	0,11 FP	TN 0,89

Abbildung 4.9: **SHAP**: Confusion Matrix für die Erkennung der Bildpixel im Trainingsverlauf (eigene Abbildung).

'echt'		Von SHAP bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,57 TP	FN 0,43
	Pixel gehört nicht zur Ziffer	0,13 FP	TN 0,87

Abbildung 4.10: **SHAP**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert größer als 0.66 klassifiziert wurden (eigene Abbildung).

hat, haben eine TP-Quote von 54%. Die TN-Quote liegt einen Prozentpunkt über der TN-Quote der anderen Datensätze (vgl. Abbildung 4.12). Für den Datensatz bei dem D 'unsicher' ist, liegt die TP-Quote zwei Prozentpunkte über der der 'generierten' Bilder (vgl. Abbildung 4.11). Drei Prozentpunkte beträgt die Differenz der TP-Quote für den Vergleich zwischen den als 'echt' klassifizierten Bildern (vgl. Abbildung 4.10) und den als 'generiert' klassifizierten Bildern.

'unsicher'		Von SHAP bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,56 TP	FN 0,44
	Pixel gehört nicht zur Ziffer	0,13 FP	TN 0,87

Abbildung 4.11: **SHAP**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert zwischen 0.66 und 0.33 klassifiziert wurden (eigene Abbildung).

'generiert'		Von SHAP bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,54 TP	FN 0,46
	Pixel gehört nicht zur Ziffer	0,12 FP	TN 0,88

Abbildung 4.12: **SHAP**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert kleiner als 0.33 klassifiziert wurden (eigene Abbildung).

4.4 Integrated Gradients (IG)

Integrated Gradients, im Folgenden mit IG abgekürzt, kann nicht importiert oder lokal installiert werden. Alle relevanten Funktionen zu Nutzung von IG finden sich daher in der IG-Komponente (vgl. 3.2.7).
IG ist eine Methode, die die Beiträge einzelner Elemente einer Instanz zu einem Gesamtergebnis bestimmen kann. Im Falle eines Bildes x bedeutet dies, dass für jedes Pixel (x_i) ein numerischer Wert $(IntegratedGrads_i^{approx}(x))$ bestimmt wird, der angibt wie wichtig ein Pixel relativ zu allen anderen Pixeln für die Klassifikation des Bildes ist. Diese Werte werden mit Formel 4.3 berechnet:

$$IntegratedGrads_i^{approx}(x) = (x_i - x_i') \times \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m} \times (x - x'))}{\partial x_i} \times \frac{1}{m} \tag{4.3}$$

Für die Bestimmung der IGs bedarf es eines Referenzbilds welches möglichst informationsarm ist (Sundararajan u. a. (2017)). Dafür wird ein vollständig schwarzes Bild gewählt dessen Pixelwerte alle null sind (vgl. das rot markierte Bild in Abbildung 4.13). Für ein

Pixel x_i wird die Differenz zum vollständig schwarzen Pixel bestimmt. Dann wird die Intensität des Pixels in k Schritten erhöht (vgl. Abbildung 4.13). Ist der Gradient für dieses Pixel bei einer Intensität $\frac{k}{m}$ hoch, bedeutet eine kleine Änderung des Pixelwerts eine große Änderung der Ausgabe, also der Klassifikation. Verändert sich der Gradient nicht mit zunehmender Intensität des Pixels, dann ist dieser nicht wichtig für die Klassifikation. Sind die Gradienten für kleine k hoch und sättigen sich mit zunehmenden Werten für k, dann ist dieses Pixel relevant für die Klassifizierung. Um dem Pixel einen

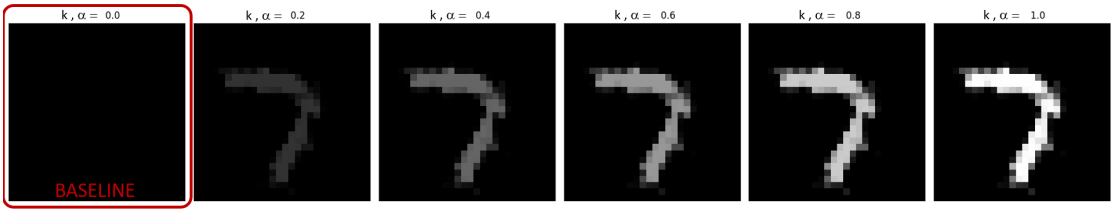

Abbildung 4.13: Interpolation der Pixel in k bzw. α Schritten (eigene Abbildung).

numerischen Wert für dessen Wichtigkeit zuzuordnen, wird die Summe aller Gradienten für alle m Schritte gebildet und mit $\frac{1}{m}$ gemittelt. Dieser Wert wird mit dem Abstand des Pixelwertes x_i zum Pixelwert des Referenzbilds x' gewichtet. So wird sicher gestellt, dass Pixel die einen Pixelwert mit großer Differenz zum 'informationslosen' Schwarz haben stärker gewichtet werden. Auf Basis dieses Werts für jedes Pixel wird eine Salienzkarte erstellt (vgl. Abbildung 4.14).
Mit diesen Eigenschaften bietet IG lokale Nachvollziehbarkeit, da einzelne Bilder betrachtet werden können. Es gibt aber keine Erklärungen über das ganze Modell. IG ist nicht vollständig modellagnostisch, allerdings können IGs für jede Form neuronaler Netze berechnet werden (Sundararajan u. a. (2017)). IG wurde mit der Intention, entwickelt die Bedeutung einzelner Elemente einer Instanz zu deren Klassifikation zu bestimmen. So ist IG dazu geeignet, die Wichtigkeit einzelner Pixel für eine Klassifizierung zu bestimmen (Sundararajan u. a. (2017), Molnar (2022)). Damit erfüllt IG die in 3.1 festgelegten **fachlichen Kriterien**.

Die Bilder des cGAN können mit Anpassungen an den Dimensionen der Bild-Tensoren von der verwendeten IG-Implementation benutzt werden. Durch die Erstellung einer Salienzkarte, welche die Wichtigkeit der Pixel farblich hervorhebt, ist IG ausreichend aussagekräftig. Da die Berechnung von IG sehr aufwendig ist, wird für die Implementation die in Formel 4.3 beschriebene Approximation zur Berechnung verwendet (Sundararajan u. a. (2017)). So sind keine besonderen Hardwareressourcen oder Konfigurationen von Umgebungsvariablen zur Erhöhung der Leistungsfähigkeit des ausführenden Systems nö-

tig. Da es für IG keine Bibliothek gibt ist die Implementation vergleichsweise aufwendig. Die Maske auf Basis der IG-Werte für jedes Pixel erlaubt das Berechnen von Metriken. Damit werden die in 3.1 festgelegten **technischen Kriterien** erfüllt.

Visuelle und statistische Ergebnisse

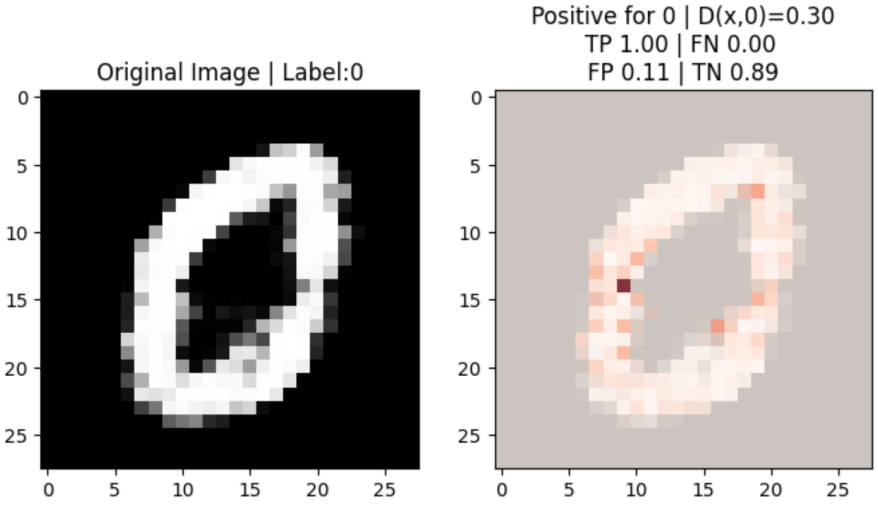

Abbildung 4.14: Die Salienzkarte (rechts) visualisiert die Wichtigkeit der Pixel für die Klassifizierung als 'Null'. Je roter ein Pixel hervorgehoben ist, desto relevanter ist es für die Klassifikation (eigene Abbildung).

IG hebt mit einer Maske, die über das Bild gelegt wird, alle für die Klassifikation relevanten Pixel hervor. Je wichtiger ein Pixel ist, desto roter ist es eingefärbt.
Ein zu visualisierendes Bild muss per Index aus einem Datensatz gewählt werden. Dann wird es, wie in Abbildung 4.14, zusammen mit den berechneten Metriken für dieses Bild angezeigt. Es gibt keine vergleichende Visualisierung mit allen möglichen Labeln wie bei SHAP und LIME.

Um die Güte von IG zu bestimmen werden für alle vier Datensätze (vgl. Tabelle 3.1) die Metriken bestimmt. Folgende Ergebnisse wurden für die vier Datensätze bestimmt:
Über den gesamten Verlauf des Trainings hat IG 100% der zu einer Ziffer gehörigen Pixel erkannt (TP). Dementsprechend ist die FN-Quote 0%. 10% der Pixel, die nicht zur Ziffer gehören werden trotzdem als solche erkannt (FP). 90% der Pixel, die nicht zur Ziffer gehören werden korrekt als solche erkannt (TN). Es werden über das Training

hinweg alle Pixel, die zur Ziffer gehören korrekt erkannt (vgl. Abbildung 4.15). Das ist außergewöhnlich und wird in Kapitel 5.1 kritisch analysiert.

Folgende Abbildungen zeigen die Erkennungsraten für die in Tabelle 3.1 beschriebenen Datensätze: Unabhängig davon, ob D die Bilder als 'echt' (vgl. Abbildung 4.16), 'unsi-

Trainingsverlauf		Von IG bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	1,00 TP	FN 0,00
	Pixel gehört nicht zur Ziffer	0,10 FP	TN 0,90

Abbildung 4.15: **IG**: Confusion Matrix für die Erkennung der Bildpixel im Trainingsverlauf (eigene Abbildung).

'echt'		Von IG bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	1,00 TP	FN 0,00
	Pixel gehört nicht zur Ziffer	0,09 FP	TN 0,91

Abbildung 4.16: **IG**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert größer als 0.66 klassifiziert wurden (eigene Abbildung).

cher' (vgl. Abbildung 4.17) oder 'generiert' (vgl. Abbildung 4.18) klassifiziert hat, liegt die TP-Quote bei 100%. Die TN-Quote liegt bei den als 'echt' klassifizierten Instanzen einen Prozentpunkt unter denen der beiden anderen Datensätze.

'unsicher'		Von IG bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	1,00 TP	FN 0,00
	Pixel gehört nicht zur Ziffer	0,09 FP	TN 0,91

Abbildung 4.17: **IG**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert zwischen 0.66 und 0.33 klassifiziert wurden (eigene Abbildung).

'generiert'		Von IG bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	1,00 TP	FN 0,00
	Pixel gehört nicht zur Ziffer	0,10 FP	TN 0,90

Abbildung 4.18: **IG**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert kleiner als 0.33 klassifiziert wurden (eigene Abbildung).

4.5 GradCAM

GradCAM wird mit einer Implementation von Gildenblat (2019) umgesetzt. Die dafür notwendigen Funktionen finden sich in der Komponente, da GradCAM nicht lokal installiert und importiert werden kann. GradCAM ermöglicht die Visualisierung der Ergebnisse von *Convolutional Neural Networks*, im Folgenden mit CNN abgekürzt (Selvaraju u. a. (2020), Molnar (2022)). Um diese Ergebnisse, hier eine Klassifikation der Zahlen, zu erklären werden relevante Bildbereiche folgendermaßen bestimmt:
Die Grundidee ist, mit der letzten Convolutional Layer des klassifizierenden CNN, eine Salienzkarte zu erstellen. Die Neuronen dieser Schicht identifizieren die für die Klassifikation entscheidenden Bildbereiche (Selvaraju u. a. (2020)). GradCAM arbeitet im Gegensatz zu LIME und SHAP rein gradientenbasiert. Die Gradienten der letzten Convolutional Layer werden genutzt, um zu bewerten wie wichtig ein Neuron und der damit verknüpfte Bildbereich für die Klassifikation ist. Die Convolutional Layer erzeugt eine Karte für jeden als relevant identifizierte Bildbereich k. Für jeden Bildbereich k wird in einer eigenen Matrix A^k vermerkt, wie relevant ein Pixel für den Bildbereich ist.

Mit Formel 4.4 wird zunächst die Wichtigkeit der einzelnen Neuronen α_k^c für ein Klassenlabel c unter der Berücksichtigung des k-ten identifizierte Bildbereichs bestimmt. Die Salienzkarte $L^c_{Grad-CAM} \in \mathbb{R}^{u \times v}$ wird dann unter Verwendung von α_k^c berechnet. Die Salienzkarte hat die Breite u und die Höhe v. Die Maße entsprechen dem zu klassifizierenden Bild, das Klassenlabel ist c.

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \qquad L^c_{Grad-CAM} = ReLU\left(\sum_k \alpha_k^c A^k\right) \tag{4.4}$$

α_k^c wird für jeden Bildbereich k berechnet. Dazu wird der Gradient von $\frac{\partial y^c}{\partial A_{ij}^k}$ pixelweise bestimmt, also wie stark eine Änderung der Eingabe in den Convolutional Layer sich auf den Wert y^c auswirkt, wie 'sicher' sich das Modell ist, ein Bild einem Klassenlabel c korrekt zugeordnet zu haben.
Zur Berechnung der Salienzkarte $L^c_{Grad-CAM}$ wird jede A^k für jedes Pixel mit dem zugehörigen α gewichtet. So kann für ein Klassenlabel die Wichtigkeit eines Bildbereichs k für jedes Pixel bestimmt werden. Durch die Anwendung einer $ReLU$-Aktivierungsfunktion werden alle negativen Beiträge zu einer Klassifikation entfernt. Je höher der Wert in der entstandenen Matrix L ist, desto wichtiger ist dieses Pixel für die Klassifikation c.

Mit diesen Eigenschaften ermöglicht GradCAM lokale Erklärbarkeit, da jedes Bild einzeln betrachtet werden kann. Es können keine gemittelten Werte über das gesamte Modell bestimmt werden (Molnar (2022)). GradCAM ist insbesondere dazu geeignet Klassifikationen von Bildern nachzuvollziehen, da es explizit mit den Gradienten der Convolutional Layer arbeitet (Selvaraju u. a. (2020)). Außerdem ist GradCAM in der Lage, die Bedeutung einzelner Pixel zu der Klassifikation zu bestimmten. Diese werden mit einer Salienzkarte visualisiert. Damit erfüllt GradCAM alle in Abschnitt 3.1 gestellten **fachlichen Kriterien**.

Die Implementation kann die generierten Bilder verarbeiten, wenn die Dimensionen der Bild-Tensoren den Spezifikationen der GradCAM-Implementation entsprechend angepasst werden. Durch das Hervorheben relevanter Pixel können die Ergebnisse von GradCAM visualisiert werden. Es werden keine besonderen Hardwareressourcen benötigt. Es werden keine individuellen Einstellungen an der Laufzeitumgebung vorgenommen. Da die Salienzkarte separat verfügbar ist, kann diese zum Bestimmen von Metriken herangezogen werden. Damit sind die in Abschnitt 3.1 festgelegten **technischen Kriterien** erfüllt.

Visuelle und statistische Ergebnisse

GradCAM hebt die für die Klassifikation relevanten Bildbereiche mit einer Salienzkarte hervor. Je wichtiger ein Pixel für die Klassifikation ist, desto roter wird dieses im RGB-Farbverlauf von Blau über Grün nach Rot eingefärbt. Die in Abbildung 4.19 gezeigte Darstellung kann für einzelne Bilder des ausgewählten Datensatzes (vgl. Tabelle 3.1) angezeigt werden. Die Auswahl des Bildes erfolgt über den Index des Datensatzes. Die von GradCAM verarbeiteten Bilder haben eine höhere Auflösung von 100x100 Pixeln statt 28x28 Pixel, da bei niedrigerer Auflösung der Farbverlauf der Maske nicht gut zu erkennen ist. Die Metriken werden für das gewählte Bild im Plot angezeigt. Es gibt

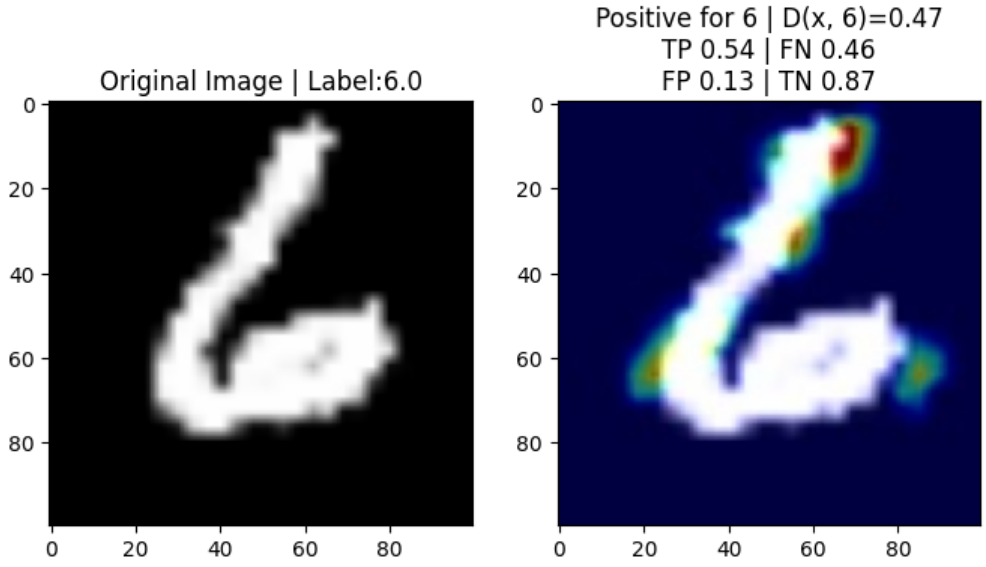

Abbildung 4.19: Visualisierung, welche Pixel für die Klassifizierung als 'Sechs' relevant sind (eigene Abbildung).

keinen Plot, der den Vergleich einer Zahl mit allen Labeln ermöglicht, wie bei SHAP und LIME.

Zur Bestimmung der Güte werden die Metriken auf hinreichend große Datensätze angewendet (vgl. Tabelle 3.1).
Für den Datensatz, der Bilder über den gesamten Trainingsverlauf enthält, sind folgende Ergebnisse bestimmt worden:
Abbildung 4.20 zeigt, dass über das gesamte Training hinweg GradCAM 57% der zur Ziffer gehörenden Pixel korrekt erkennt (TP). 42% der Pixel, die zur Ziffer gehören wer-

den nicht erkannt (FN). 21% der Pixel die nicht zur Ziffer gehören werden trotzdem als dieser zugehörig eingeordnet (FP). 78% der Pixel, die nicht zur Ziffer gehören werden korrekt als solche erkannt (TN). Dieser Wert ist über das gesamte Training und alle zehn möglichen Klassenlabel gemittelt.

Folgende Abbildungen zeigen die Erkennungsraten für die einzelnen Datensätze die gemäß Tabelle 3.1 sortiert werden: Abbildung 4.21 zeigt, dass als 'echt' klassifizierte Bilder

Trainingsverlauf		Von GradCAM bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,57 TP	FN 0,43
	Pixel gehört nicht zur Ziffer	0,21 FP	TN 0,79

Abbildung 4.20: **GradCAM**: Confusion Matrix für die Erkennung der Bildpixel im Trainingsverlauf (eigene Abbildung).

'echt'		Von GradCAM bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,59 TP	FN 0,41
	Pixel gehört nicht zur Ziffer	0,22 FP	TN 0,78

Abbildung 4.21: **GradCAM**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert größer als 0.66 klassifiziert wurden (eigene Abbildung).

eine TP-Quote von 59% haben. Die TN-Quote ist die niedrigste aller vier Datensätze. Die TP-Quote für die als 'unsicher' klassifizierten Datensätze liegt einen Prozentpunkt unter der als 'echt' klassifizierten (vgl. 4.22). Für die als 'generiert' klassifizierten Bilder ist die TP-Quote am niedrigsten. Die TN-Quote ist mit 79% durschnittlich und höher als die der als 'echt' klassifizierten Bilder (vgl. Abbildung 4.23).

'unsicher'		Von GradCAM bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,58 TP	FN 0,42
	Pixel gehört nicht zur Ziffer	FP 0,20	TN 0,80

Abbildung 4.22: **GradCAM**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert zwischen 0.66 und 0.33 klassifiziert wurden (eigene Abbildung).

'generiert'		Von GradCAM bestimmte Zugehörigkeit	
		Pixel als zur Ziffer zugehörig erkannt	Pixel nicht als zur Ziffer zugehörig erkannt
Tatsächliche Zugehörigkeit	Pixel gehört zur Ziffer	0,56 TP	FN 0,44
	Pixel gehört nicht zur Ziffer	FP 0,21	TN 0,79

Abbildung 4.23: **GradCAM**: Confusion Matrix für die Erkennung der Bildpixel von Ziffern, die von D mit einem Wert kleiner als 0.33 klassifiziert wurden (eigene Abbildung).

4.6 Umsetzung der Anforderung

Für die Umsetzung des PoC wurden in Abschnitt 3.2.1 Anforderungen formuliert. Diese Anforderungen werden wie folgt umgesetzt:
Die Ergebnisse der xAI-Methoden zur Nachvollziehbarkeit einer Klassifikation sind für alle Implementationen der xAI-Methoden gegeben (siehe oben). Diese Anforderung ist wichtig, damit die Ergebnisse möglichst niederschwellig verstanden werden können. Und so das Potential besteht, nicht nur den Ansprüchen der Explainable AI (vgl. 2.3.1) zu genügen sondern auch Ansprüchen der Interpretable AI (vgl. 2.3.2).
Da alle vier xAI-Methoden separat eine Maske erzeugen, können die Metriken berechnet werden, die die Güte der jeweiligen xAI-Methode bestimmt. Dies ist wichtig, um die Methoden qualifiziert miteinander vergleichen zu können. Die berechneten vier Parameter bilden eine Confusion Matrix, so lässt sich genau nachvollziehen wie gut die jeweilige xAI-Methode die relevanten Pixel erkennt.
Um einen fairen Vergleich zu ermöglichen, muss die Datenbasis einheitlich sein. Alle vier xAI-Methoden verwenden dieselben vier Datensätze, somit ist die Datenbasis für alle

xAI-Methoden gleich und eine Basis für den Vergleich gegeben (vgl. Tabelle 3.1).
Für das Training der Klassifizierer in den Komponenten der xAI-Methoden wird der MNIST-Datensatz verwendet. Damit erfolgt das Training der Klassifizierer wie gefordert auf denselben Trainingsdaten.
Alle xAI-Methoden werden mit Jupyter-Notebooks in Colab umgesetzt. Es besteht daher kein Risiko, dass unterschiedliche Laufzeitumgebungen die Ergebnisse der xAI-Methoden verfälschen.
Mit der Erfüllung aller fünf Anforderungen (vgl. 3.2.1) kann angenommen werden, dass die Bedingungen für einen möglichst fairen Vergleich der xAI-Methoden gegeben sind.

5 Diskussion

In diesem Kapitel werden die Ergebnisse aus Kapitel 4 diskutiert. In 5.1 werden die xAI-Methoden verglichen, indem ihre Vor- und Nachteile gegeneinander abgewogen werden. Anschließend werden in 5.2 die Methodik und die verwendeten Technologien bewertet. In 5.3 werden die Grenzen der verwendeten Methodik und der damit erzielten Ergebnisse betrachtet. Abschließend wird in 5.4 auf Möglichkeiten eingegangen, die Fragestellung der vorliegenden Arbeit zu erweitern.

5.1 Vergleich der xAI-Methoden

Die verwendeten xAI-Methoden werden in diesem Abschnitt hinsichtlich der Güte der bereitgestellten Erklärungen betrachtet. In 5.1.1 wird die Anwendbarkeit des PoC und damit der xAI-Methoden untersucht.

In 5.1.2 die Aussagekraft der xAI-Methoden. Außerdem wird diskutiert, ob und wie gut die xAI-Methoden geeignet sind einen Bezug zur Ausgabe von D zu erstellen und diese nachvollziehbar zu machen. Auf Basis dieser Diskussion wird die in 1.1 formulierte Forschungsfrage beantwortet.

5.1.1 Anwendbarkeit von xAI-Methoden und PoC

Mit einer einheitlichen Datenbasis erlaubt das PoC Vergleiche verschiedener xAI-Methoden. Die Ansätze der xAI-Methoden unterscheiden sich teilweise stark. Daher braucht jede der Komponenten, die eine xAI-Methode implementiert einen Klassifizierer (vgl. 3.2.2). Je nach xAI-Methode müssen diese Klassifizierer unterschiedlichen Anforderungen genügen (z.B. Dimension der Eingabetensoren).

IG und GradCAM sind gradientenbasierte Verfahren. IG kann als Klassifizierer ein beliebiges neuronales Netz verwenden (Sundararajan u. a. (2017)). GradCAM benötigt als

klassifizierendes neuronales Netz zwingend ein CNN (Selvaraju u. a. (2020)). SHAP und LIME basieren auf dem Exkludieren von Bildinformationen ('Super-Pixel'). SHAP und LIME sind modellagnostisch (Molnar (2022)), daher unterliegt die Wahl des Klassifizierers keinen Einschränkungen.
Auf technischer Ebene lassen sich die vier Komponenten flexibel verwenden, wobei die gradientenbasierten xAI-Methoden keine beliebigen Klassifizierer zulassen. Das PoC erlaubt es beliebige cGANs zu analysieren. Der genaue Aufbau von G und D ist für die Nachvollziehbarkeit der Entscheidungen nicht relevant, da das PoC auf Basis der Ein- und Ausgaben von D arbeitet (vgl. 3.2.2). D muss aber modifiziert werden, um die Eingaben gemäß Tabelle 3.1 zu sortieren und abzuspeichern. Da explizit auch die Label verwendet werden, die das cGAN generieren soll, können keine beliebigen GANs analysiert werden. Diesen fehlt die Zusatzinformation y (vgl. 2.5). Das zu generierende Label (y) ist aber essenziell, um in der Auswertung einen Rückschluss auf die zu generierende Zahl zu ziehen. Zudem können die Ergebnisse nach Labeln gruppiert werden. So werden verschiedene Analysen ermöglicht, um die Ausgaben von D besser nachvollziehen zu können (vgl. 5.1.2).

5.1.2 Aussagekraft der xAI-Methoden

Dieser Abschnitt diskutiert die Eignung der vier gewählten xAI-Methoden, um die in 1.1 gestellte Forschungsfrage *'Welche xAI-Methoden sind geeignet, um die Entscheidungen dieses Teilmodells (Diskriminator) nachzuvollziehen?'* zu beantworten.

Bezüglich der Aussagekraft der xAI-Methoden werden zunächst die visuellen Ergebnisse betrachtet. Anschließend wird ein Vergleich auf Basis der Ergebnisse der berechneten Metriken (TP, FN, FP, TN) vorgenommen (vgl. 4.1). Die visuellen und statistischen Ergebnisse werden bezüglich der in 2.3 beschriebenen Ansätze zur Nachvollziehbarkeit von KI betrachtet.

Alle vier Methoden visualisieren ihre Ergebnisse für ein ausgewähltes Bild, indem eine Maske über das Bild gelegt wird, die aussagt, wie wichtig jedes einzelne Pixel für die Klassifikation ist. Diese Masken sind nach Möglichkeit Salienzkarten, welche besonders geeignet sind, die Klassifikation von Bildern nachvollziehbar zu machen (Simonyan u. a. (2014), Kapishnikov u. a. (2019)). Die Wichtigkeit der Pixel wird durch die Sättigung der Maske (SHAP, IG) oder den Farbverlauf der Maske (GradCAM) ausgedrückt. Wie diese Wichtigkeit auf Basis des Inaktivierens von Pixel-Koalitionen (SHAP) oder Gradienten

(IG, GradCAM) bestimmt wird, ist bestenfalls für Data Scientists verständlich (Yang und Berdine (2023)). Salienzkarten sind für Data Scientists leicht zu verstehen, können aber auch sonstigen Nutzenden ein grundsätzliches Verständnis vermitteln, welche Pixel relevant sind. Das 'Warum' wird dabei nicht direkt erklärt (Yang und Berdine (2023)). So bieten Salienzkarten eine intuitive Form visueller Nachvollziehbarkeit.

Um den Bezug zur Ausgabe von D zu erstellen, wird diese zusammen mit der Salienzkarte angezeigt ($D(x, y)$). So kann visuell überprüft werden, ob eine Ausgabe von D mit einem Bild einhergeht, das der zu generierenden Ziffer in einer Qualität entspricht, die der Ausgabe von D entspricht. Falls das Bild nicht eindeutig oder von minderer Qualität ist, kann anhand des Labels festgestellt werden, welche Ziffer generiert werden sollte. Ebenfalls werden die Metriken (TP, FN, FP, TN) angezeigt. So kann für ein Bild bestimmt werden, wie gut die Erkennungsrate der xAI-Methode ist. Dies ist besonders dann relevant, wenn die Pixel nur sehr geringe Beiträge haben und die Maske sie entsprechend schwach hervorhebt.
Es ist zu beachten, dass Ausgaben von D nahe eins, nicht zwangsläufig bedeuten, dass die jeweilige Ziffer den menschlichen Ansprüchen an eine bestimmte Ziffer genügt (vgl. Abbildung 5.1). Beide Achten wurde von D mit einem Wert > 0.66 ('echt') klassifiziert. Die linke Acht hat D als 'echter' als die rechte bewertet. Ein menschlicher Beobachter würde aber die rechte Acht eher als solche erkennen. Das in Abbildung 5.1 gezeigte Ver-

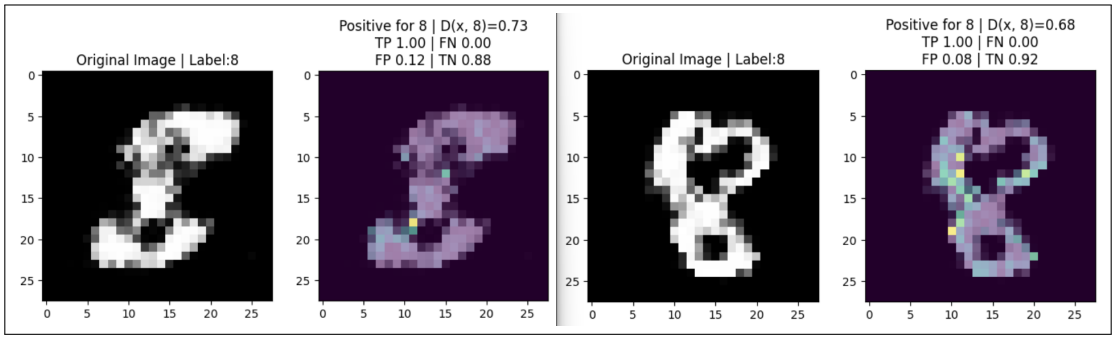

Abbildung 5.1: Vergleich zweier Achten die D mit 'echt' bewertet hat (eigene Abbildung).

halten lässt sich durch das zyklische Training des cGANs bzw. GANs im Allgemeinen erklären (vgl. 2.4.4). D soll Bilder als 'generiert' oder 'echt' klassifizieren. Diese Bilder sind in frühen Trainingsepochen noch nicht oder nicht so gut als Ziffern zu erkennen. Ursächlich dafür ist, dass G ebenfalls nur relativ zum Trainingsfortschritt Ergebnisse liefert. Je nach Trainingsfortschritt genügen diese Bilder allerdings nicht den Ansprüchen eines menschlichen Betrachters an eine handgeschriebene Ziffer. So klassifiziert D Ziffern als

'echt', da diese zwar relativ besser sind als zuvor von G vorgelegte Bilder, aber absolut gesehen nur Rauschen mit minimaler Struktur darstellen. Die linke Acht in Abbildung 5.1 wurde zeitlich vor der rechten generiert und damit mutmaßlich auch in einer früheren Trainingsepoche. So mussten D von G noch nicht so gute Ergebnisse vorgelegt werden, damit D diese als möglichst 'echt' klassifiziert. Im weiteren Trainingsverlauf hat D dann eine deutlich gelungenere Acht, als weniger 'echt' klassifiziert. So lässt sich die Ausgabe von D im Trainingsverlauf beobachten.

Zusätzlich wäre es für Data Scientists interessant, aus welcher Trainingsepoche und aus welcher Iteration innerhalb der Epoche das betrachtete Bild stammt. Damit kann beispielsweise bei visuell gut befundener Bildqualität eine Trainingsepoche bestimmt werden, in der das Training beendet werden kann. So könnte das Training zu einem sinnvollen Ende gebracht werden, falls es trotz sinnvoller Ergebnisse nicht konvergiert (Mescheder u. a. (2018)).

LIME erzeugt als einzige der vier xAI-Methoden keine Salienzkarte. Es wird nur binär klassifiziert, ob ein Pixel relevant für die Klassifikation ist oder nicht. So bietet LIME dem Nutzenden keine Möglichkeit, die Wichtigkeit eines Pixels für die Klassifikation relativ zu anderen Pixeln zu bewerten (Yang und Berdine (2023)). Im Falle von LIME wäre diese Information wichtig, insbesondere wegen der hohen FP-Quoten (vgl. Abbildung 4.1), welche im Vergleich mit dem Originalbild besonders hervortreten. Die visuelle Darstellung der Ergebnisse von LIME ist weniger aussagekräftig, da weniger wichtige Pixel nicht von relevanteren unterschieden werden können.

SHAP, IG und GradCAM ermöglichen diese relativen Bewertungen, da hier Salienzkarten verwendet werden. So kann differenziert werden, welche Pixel mehr oder weniger relevant für die Klassifikation sind (Simonyan u. a. (2014)), Yang und Berdine (2023).
SHAP markiert zusätzlich noch die Pixel, die explizit nicht zur vorgenommenen Klassifikation beitragen (Lundberg und Lee (2017)). Vor dem Hintergrund der Ausgabe von D kann überprüft werden, ob sich eine hohe Erkennungrate relevanter Pixel (rot, vgl. Abbildung 4.7) in der Salienzkarte widerspiegelt.
Diese Überprüfung kann für IG und GradCAM analog vorgenommen werden. IG markiert relevante Pixel analog zu SHAP rot (vgl. Abbildung 4.14) und GradCAM entsprechend dem in 4.5 beschriebenen Farbverlauf (vgl. Abbildung 4.19). Mit Hilfe der visuellen Darstellung kann ein Zusammenhang zwischen den Ausgaben von D und der Qualität des generierten Bildes erstellt werden. Auch können Bilder aus unterschiedlichen Trainingsepochen verglichen werden.

Bei allen visuellen Darstellungen ist zu beachten, dass diese nur für das gewählte Bild gelten. Um genereller Aussagen treffen zu können, müssen Serien von generierten Bildern betrachtet werden.
Die Güte der xAI-Methoden wird über das gesamte Training hinweg gruppiert nach Ausgaben von D betrachtet (vgl. Tabelle 3.1). Für alle vier xAI-Methoden sind die jeweiligen Metriken für die unterschiedlichen Datensätze sehr ähnlich. Die berechneten Metriken unterscheiden sich innerhalb der Datensätze nur um wenige Prozentpunkte. Allerdings muss berücksichtigt werden, dass die Datensätze den gesamten Trainingsverlauf über alle Trainingsepochen abbilden. Das gilt auch für die nach Ausgaben von D gruppierten Datensätze. So eignen sich die erhobenen Daten für einen Vergleich der Methoden untereinander.

LIME hat eine hohe TP-Quote von etwa 80% über alle Datensätze hinweg, allerdings auf Kosten einer hohen FP-Quote von etwa 55% (vgl. 4.2). Abbildung 5.2 zeigt an einigen

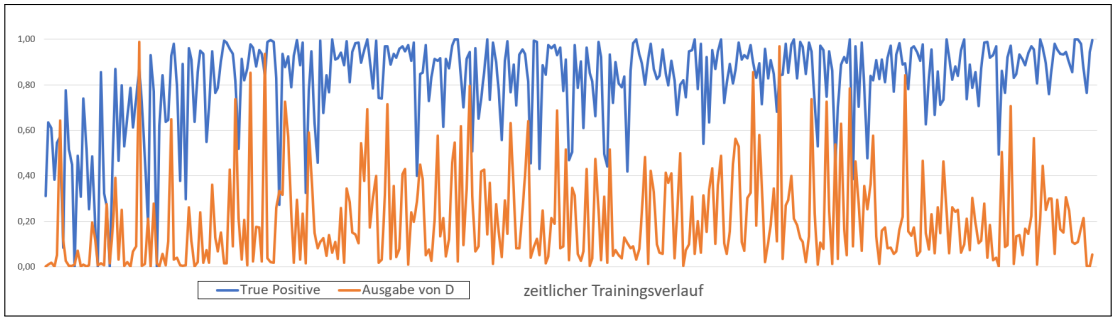

Abbildung 5.2: **LIME**: Entwicklung von TP und den Ausgaben von D im Trainingsverlauf (eigene Abbildung).

Stellen, dass Ausgaben von D nahe eins mit niedrigeren TP-Quoten einher gehen. Dies ist über den gesamten Trainingsverlauf zu beobachten und lässt den Schluss zu, dass LIME bei Ziffern, die als 'echt' klassifiziert werden, viele zur Ziffer gehörige Pixel nicht erkennt. Dieses Verhalten nimmt zum Trainingsende hin ab, also dann, wenn die Qualität der generierten Bilder besser wird (vgl. Abbildung 5.2). Für detailliertere Betrachtungen wäre eine Korrelationsanalyse der Ausgaben von D und den berechneten Metriken zielführend.

SHAP erkennt mit einer TP-Quote von etwa 50% jedes zweite Pixel korrekt, das zu einer Ziffer gehört. Allerdings hat SHAP mit etwa 89% eine auffallend hohe TN-Quote (vgl. 4.3). Es werden also im Gegensatz zu LIME kaum Pixel, die nicht zur Ziffer gehören, fälschlicherweise dieser zugeordnet. Abbildung 5.3 zeigt zum Ende des Trainings, dass

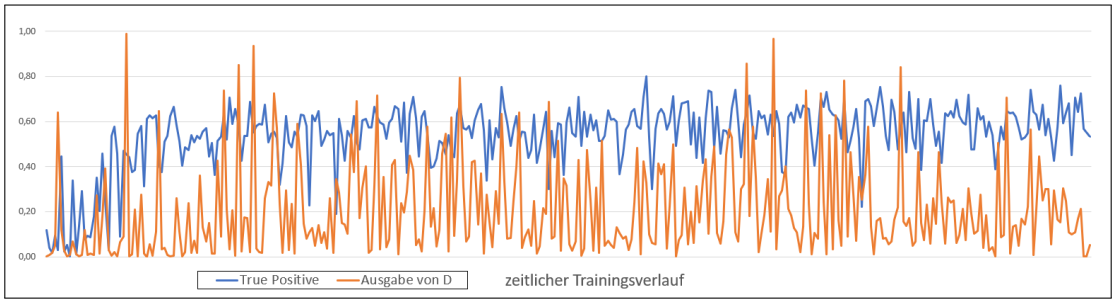

Abbildung 5.3: **SHAP**: Entwicklung von TP und den Ausgaben von D im Trainingsverlauf (eigene Abbildung).

eine höhere TP-Quote mit größeren Ausgaben von D einhergeht. Somit ist SHAP in der Lage, Ziffern hoher Qualität auch als solche zu erkennen. Allerdings müsste auch hier der Zusammenhang der Ausgabe von D und der bestimmten Metriken mit einer Korrelationsanalyse verifiziert werden.

IG hat eine außergewöhnlich hohe TP-Quote von 100%. Die FP-Quote ist mit 9% außergewöhnlich niedrig (vgl. 4.4). Dies ist unabhängig davon welche Datensätze ausgewertet werden.

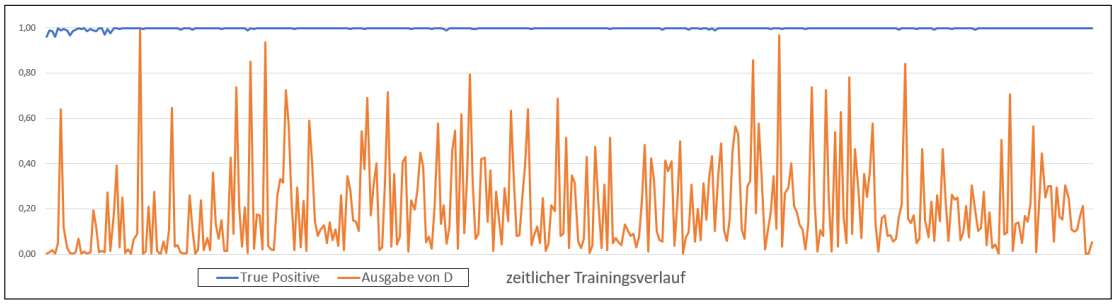

Abbildung 5.4: **IG**: Entwicklung von TP und den Ausgaben von D im Trainingsverlauf (eigene Abbildung).

Abbildung 5.4 zeigt die Entwicklung der TP-Quote und die Ausgabe von D. Die Werte sind über alle Klassenlabel gemittelt. Die TP-Quote ist von Anfang an hoch und steigt dann schnell auf den Maximalwert von eins. Es gibt eine minimale Oszillation. Es ist nicht erkennbar, ob die Ausgabe von D und die TP-Quote voneinander abhängen. Die außergewöhnlich guten Ergebnisse lassen vermuten, dass IG nicht so arbeitet wie vorgesehen. Der Klassifizierer scheint nicht zu lernen was eine Ziffer ist. Vielmehr scheint es, dass Pixel die nicht den selben Farbwert wie die Baseline (vgl. 3.2.7) haben, als die zu erkennende Ziffer markiert werden. Aufgrund der technischen Funktionsweise von IG,

ist es auch erklärbar, dass nur IG dieses Verhalten zeigt. Die weißen Ziffern sind auf schwarzem Grund abgebildet. Damit entspricht die Farbe der nicht zum Bild gehörigen Pixel, der der Baseline. Wenn die Sättigung des Bildes erhöht wird, ändern sich nur die Gradienten für die Pixel, die ohnehin zur Ziffer gehören. IG muss nicht zwischen Pixeln unterscheiden, die zur Ziffer gehören und welchen die es nicht tun, da es keine weiteren Objekte im Bild gibt die nicht Teil der Ziffer sind. Die Gradienten sättigen sich daher mit zunehmender Intensität der Pixel für alle nicht schwarzen Pixel (Sundararajan u. a. (2017)).
IG ist daher trotz guter Ergebnisse nicht geeignet, die Ausgaben von D in einem GAN nachzuvollziehen.

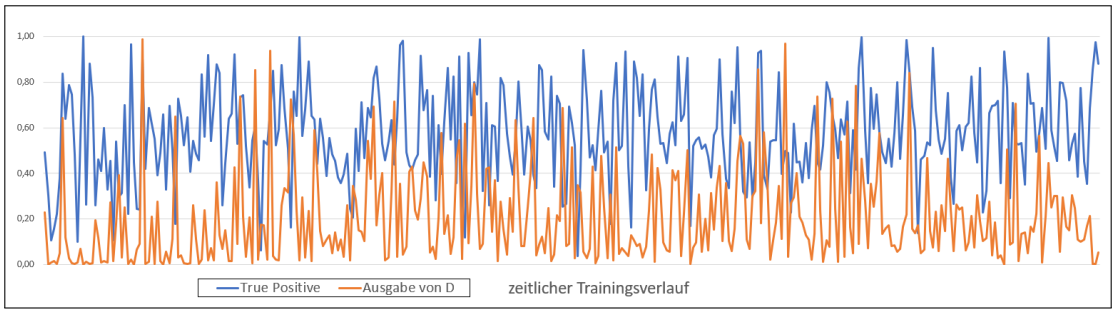

Abbildung 5.5: **GradCAM**: Entwicklung von TP und den Ausgaben von D im Trainingsverlauf (eigene Abbildung).

GradCAM erreicht eine TP-Quote von etwa 58%. Die FP-Quote ist mit etwa 20% vergleichsweise niedrig, die dazugehörig TN-Quote mit etwa 80% hoch, allerdings nicht so hoch wie bei SHAP (vgl. 4.5). Abbildung 5.5 zeigt eine starke Oszillation der TP-Quote. Wenn überhaupt ist zum Ende des Trainings ein Zusammenhang zwischen hohen Ausgaben von D und der TP-Quote zu erkennen. Für eine Nachweis bedürfte es auch hier einer Korrelationsanalyse.

Auf Basis der Metriken können, wie in 5.2.1 angemerkt, weitere Auswertungen der Ergebnisse vorgenommen werden. Je Datensatz (vgl. Tabelle 3.1) und xAI-Methode können die berechneten Metriken und die Ausgaben von D gruppiert nach Labeln betrachtet werden. Je besser der jeweilige Parameter im Vergleich zu den anderen Labeln ist, desto grüner ist dieser hervorgehoben (vgl. Abbildung 5.6). Die äquivalenten Abbildungen zu Abbildungen 5.6 für LIME, SHAP und IG finden sich in Anhang A.

Am Beispiel von GradCAM lässt sich erkennen, dass auf Basis der Metrik weiterführende Analysen vorgenommen werden können. Die Gruppierung nach Labeln ermöglicht es,

Vorkommen und Ausgabe von D nach Labeln				Confusionmatrix nach Klassenlabeln			
Label	#Absolut	Relativ	Ø-Ausgabe von D	Ø-True Positive	Ø-False Negative	Ø-False Positive	Ø-True Negative
0	31	0,0861	0,2645	0,4998	0,5002	0,2062	0,7938
1	28	0,0778	0,2392	0,6656	0,3344	0,1540	0,8460
2	37	0,1028	0,2208	0,8408	0,1592	0,3708	0,6292
3	35	0,0972	0,2039	0,5468	0,4532	0,2999	0,7001
4	40	0,1111	0,1698	0,6930	0,3070	0,2503	0,7497
5	39	0,1083	0,2313	0,4913	0,5087	0,2889	0,7111
6	31	0,0861	0,2309	0,5342	0,4658	0,1700	0,8300
7	39	0,1083	0,1374	0,4846	0,5154	0,2252	0,7748
8	39	0,1083	0,1804	0,6122	0,3878	0,0936	0,9064
9	41	0,1139	0,2065	0,3353	0,5915	0,0270	0,8998

Abbildung 5.6: Beispiel GradCAM: Metriken und Ausgaben von D gruppiert nach Labeln für den Trainingsverlauf (eigene Abbildung).

die Ausgaben von D im Zusammenhang mit den Metriken besser zu verstehen. So kann festgestellt werden, dass für die Ziffer 'Null' D im Durchschnitt die höchsten Bewertungen vornimmt. So lassen sich die Ergebnisse der Metrik auf Basis der xAI-Methoden nutzen, um den Generierungsprozess besser zu verstehen. Es kann überprüft werden, ob Zahlen häufiger als andere generiert werden und wie die Ausgaben von D je Label ausfallen. Auf Basis dieser Information wäre es möglich das Training anzupassen. Beispielsweise könnte die Zusatzinformation y, welche dem cGAN das zu generierende Label vorgibt, nicht zufällig gewählt werden. Auf Basis der Auswertung könnten unterrepräsentierte oder von D schlechter bewertete Label erkannt werden. Diese könnten dann häufiger generiert werden. So ließen sich die Qualität der Ziffern dieser Label gegebenenfalls verbessern.

Auf Basis des PoC werden verschiedene xAI-Methoden verglichen. Die vorliegenden Ergebnisse basieren auf einer einheitlichen Datenbasis und den Bewertungen, welche die xAI-Methoden vorgenommen haben. Dies erfolgt auf Basis des in Kapitel 3 beschriebenen Vorgehens, der verwendeten Komponenten und Bibliotheken, der verwendeten Laufzeitumgebung und der für die einzelnen Komponenten gesetzten Hyperparameter.
Die vorliegenden Ergebnisse zeigen, dass die gewählten xAI-Methoden grundsätzlich geeignet sind, die Ausgaben von D besser zu verstehen. Allerdings sind die xAI-Methoden unterschiedlich gut geeignet. Die Ergebnisse von SHAP zeigen über alle Datensätze hinweg gute Ergebnisse für die Metriken. Außerdem ist die Salienzkarte für einzelne Bilder detailliert und aussagekräftig. Dies liegt insbesondere an der niedrigen FP-Quote und der damit einhergehenden hohen TN-Quote. Dafür sprechen auch die Ausgaben von D und die im Trainingsverlauf beobachtete TP-Quote (vgl. Abbildung 5.2).

GradCAM liefert detaillierte Salienzkarten. Es werden häufiger Bildbereiche als wichtig markiert, die gar nicht zur eigentlichen Ziffer gehören. Da GradCAM besonders geeignet

ist, unterschiedliche Klassen in komplexeren Bildern zu identifizieren (Selvaraju u. a. (2020)), scheint die verwendete Datenbasis nicht optimal zu sein.

LIME entspricht bei den visuellen Ergebnissen nicht dem Standard der anderen xAI-Methoden, da diese im Gegensatz zu LIME Salienzkarten verwenden. Die Metriken betreffend liefert LIME dieselben Ergebnisse wie die anderen xAI-Methoden, da die Metriken nicht Gewichtungen der Pixel beachten, um eine bessere Vergleichbarkeit zu ermöglichen.

Weiterhin hat sich IG im Rahmen des PoC als nicht nutzbar herausgestellt. Die generierten Bilder sind nicht geeignet um von IG analysiert zu werden. Grundsätzlich ist IG aber in der Lage, mit komplexeren Bilder die sich stärker von der Baseline unterscheiden nützliche Ergebnisse zu liefern.

Zusammenfassend sind SHAP, GradCAM und LIME geeignet Diskriminatorentscheidungen nachvollziehbar zu machen. IG ist es im Rahmen des umgesetzten PoC nicht.

Der in der Forschungsfrage formulierte Begriff Nachvollziehbarkeit subsumiert im Kontext der vorliegenden Arbeit (vgl. 1.1) unterschiedliche Anforderungen an die Güte der Erklärungen der Ausgabe einer KI.
Die untersuchten xAI-Methoden sind nicht in der Lage, Ergebnisse zu liefern, die konkret aussagen 'Instanz x wurde von D mit y bewertet, weil...'. Das wäre zwar wünschenswert, aber ließe sich nur mit einer komplexen KI-Anwendung auf Basis der xAI-Methoden realisieren und wäre der Interpretable bzw. Responsible AI zuzuordnen (Vainio-Pekka u. a. (2023)).

Die aktuellen Ergebnisse sind nur eingeschränkt für beliebige Nutzendengruppen geeignet. Dies wäre aber gerade im Kontext von Interpretable AI wünschenswert (Graziani u. a. (2023), Miller (2023)). Mit Ausnahme intuitiv verständlicher Salienzkarten, sind die Ergebnisse nur für Personen mit Kompetenzen im Bereich Data Science verwertbar (Yang und Berdine (2023)). Zudem müssen noch weitere Analysen auf Basis der Metriken vorgenommen werden. Zwar ist Excel ein niederschwellig verwendbares Tool dafür, aber es entspricht nicht den Ansprüchen von Interpretable AI, dass Nutzende selbst Analysen vornehmen (Graziani u. a. (2023)).

Abschließend kann festgestellt werden, dass die xAI-Methoden den Ansprüchen von Explainable AI genügen, da die Ergebnisse für Data Scientists verständlich und hilfreich sind. Die Ergebnisse werden mit Surrogatmodellen erzeugt und nicht intrinsisch aus D

generiert. Daher kann den Ansprüchen der Interpretable AI nach intrinsischer Nachvollziehbarkeit nicht nachgekommen werden (Graziani u. a. (2023)). Außerdem ist es nicht möglich die Bedeutung der Ausgaben von D direkt einem Teil der Eingabe, also Pixeln, zuzuordnen (Mahya und Fürnkranz (2023)). Die Erklärungen können ebenfalls nicht an das Wissen einer bestimmten Nutzendengruppe angepasst werden (Saeed und Omlin (2023)).
Dahingehend genügen die Erklärungen noch weniger den Ansprüchen der Responsible AI (vgl. 2.3.3). Für den konkreten Anwendungsfall ist dies weniger relevant, da keine sensiblen Daten verarbeitet werden. Aber für Szenarien in denen GANs beispielsweise Trainingsdatensätze ergänzen, wäre es wichtig nachvollziehen zu können, dass diese fair, nicht diskriminierend und verzerrungsfrei sind (Chakraborty u. a. (2024)).

5.2 Konzept und Technologien

Dieser Abschnitt bewertet die für das Erstellen des PoC verwendete Konzeption und die Technologien zur technischen Umsetzung des PoC.
Dazu werden in 5.2.1 die Methodik und in 5.2.2 explizit die Anforderungen und Ziele kritisch betrachtet. Abschließend werden in 5.2.3 die verwendeten Technologien hinsichtlich ihrer Eignung bewertet.

5.2.1 Methodik

Um die Ausgaben von D nachvollziehen zu können, wurden potentiell geeignete xAI-Methoden ausgewählt. Auf Basis von Molnar (2022) konnte ein Überblick über eine Vielzahl von xAI-Methoden erstellt werden. Anhand der fachlichen und technischen Anforderungen (vgl. 3.1) wurden vier Methoden ausgewählt. Hier wäre eine systematischere Auswahl auf Basis einer strukturierten Literaturanalyse wünschenswert gewesen. Dies lag aber außerhalb des Rahmens der vorliegenden Arbeit.

Das PoC besitzt fünf Komponenten, eine davon liefert eine einheitliche Datenbasis. An diese sind die vier xAI-Komponenten angeschlossen. Die Datenübermittlung erfolgt über einen Excel-Export. Dieser sowie der Export für die Auswertung sind etwas umständlich aber zweckmäßig. So kann das PoC in cloudbasierten Laufzeitumgebungen wie Colab genutzt werden, während die generierten Daten extern gespeichert werden. Bei Bedarf

können die CSV-Dateien der Exporte mit einem geeigneten Programm betrachtet oder bearbeitet werden. Mit Excel ist die Hürde hierfür niedrig.

Die exportierten Datensätze ermöglichen eine Analyse gruppiert nach der Ausgabe von D bzw. über das gesamte Training hinweg. So kann die Ausgabe von D aus verschiedenen Perspektiven (z.B. nach Labeln oder Trainingsepochen) betrachtet werden. Weitere Exportarten sind aber nur durch Änderungen am PoC möglich. Mit einem geeigneten Programm oder Jupyter-Notebook können beliebige Analysen der exportierten CSV-Dateien vorgenommen werden.

Das PoC ist somit leicht nutzbar, allerdings auf Kosten der Flexibilität. Tiefgreifendere Anpassungen, z.B. an der Auswertung oder den ausgewählten xAI-Methoden, erfordern Änderungen im Programmcode. Das PoC ist nicht an spezifische Hardware gebunden. Verbunden mit dem CSV-Export der Auswertung sorgt dies für niedrige Hürden bei Nutzung und Auswertung.

5.2.2 Anforderungen und Ziele

Nachdem im vorherigen Abschnitt die Methodik diskutiert wurde, sollen Anforderung und Ziele des PoC noch einmal separat betrachtet werden:

Die xAI-Methoden liefern auf Basis der generierten Bilder Informationen, die erklären welche Pixel eines Bildes relevant für dessen Klassifikation sind. Diese Ergebnisse werden dann in Bezug zu der Ausgabe von D zu dem Bild gesetzt. Die Details dazu werden in 5.1 diskutiert, insbesondere wie den Kriterien der Explainable-, Interpretable- und Responsible AI genügt wird.

Das formulierte Ziel des PoC ist in 3.2.1 festgelegt, aber weit gefasst. Unter dem Oberbegriff Nachvollziehbarkeit wird nicht weiter festgelegt welchen Gütekriterien die xAI-Methoden genügen müssen. Entsprechend sind die fünf Anforderungen an das PoC (vgl. 3.2.1) nicht streng ausspezifiziert. Es war zu diesem Zeitpunkt noch nicht absehbar, wie gut die xAI-Methoden nutzbar sind und inwiefern den Anforderungen genügt werden kann.

Der Anforderung nach visueller Darstellung konnte in genügendem Maße entsprochen werden. Auch haben alle xAI-Methoden die Möglichkeit geboten eine Metrik zu bestimmen. Das war nicht von vornherein absehbar. Der Forderung nach einer einheitlichen Datenquelle konnte nachgekommen werden. Nicht vorhersehbar war, ob sich für jede

xAI-Methode ein geeigneter Klassifizierer findet. Es wurden Änderungen an den Implementationen vorgenommen, diese gefährden aber nicht die Integrität der Klassifizierer. Um eine einheitliche Laufzeitumgebung zu gewähren wird Colab genutzt. So ist es aber schwierig Daten zu persistieren. Daher musste der Weg über die CSV-Exporte gewählt werden.

Es wurden alle Anforderungen erfüllt. Jedoch musste der Programmcode stark angepasst werden und der Programmcode der einzelnen Komponenten unterscheidet sich teilweise erheblich. Dadurch verliert das PoC auf Implementationsseite an Übersichtlichkeit. Bezüglich der Nutzbarkeit wurde auf Kosten eines umständlichen Datentransfers zwischen den Komponenten ein Ansatz gewählt, der das PoC hardwareunabhänig macht.

5.2.3 Verwendete Technologien

Das PoC verwendet mit Python eine für KI-Anwendungen prädestinierte Programmiersprache. Zusätzlich werden die in 3.2.4 genannten Bibliotheken verwendet. PyTorch, Keras und Tensorflow sind Säulen moderner KI-Anwendungen, da sie Python um viele nützliche und teilweise elementare Klassen ergänzen. Mit ihnen werden neuronale Netze aufgebaut und trainiert.
Pandas, NumPy und Matplotlib sind Standardtechnologien, um Daten mit Python zu analysieren, auszuwerten und zu visualisieren. Deshalb werden alle sechs Bibliotheken im KI-Kontext häufig verwendet.

Das PoC stützt sich somit auf eine moderne, KI-geeignete Programmiersprache und die dazugehörigen Bibliotheken. Die genannten Bibliotheken werden regelmäßig aktualisiert. Die neuesten Versionen sind wegen ihrer starken Verbreitung in der Regel stabil. Bei weniger weit verbreiteten Bibliotheken, wie der verwendeten LIME/SHAP-Bibliotheken (vgl. 3.2.6) ist es möglich, dass die neuesten Versionen nicht stabil sind. Im Rahmen der vorliegenden Arbeit war dies bei SHAP der Fall. Es musste manuell eine stabile Version für das PoC ausgewählt werden (vgl. Programmcode).
Die Wahl der Technologien entspricht damit dem aktuellen Stand für KI-Anwendungen. Die weniger häufig genutzten Bibliotheken bergen aber das Risiko, dass die aktuellste Version nicht stabil ist. Mit Colab wird eine hardwareunabhängige Laufzeitumgebung genutzt. So kann das PoC dezentral und unabhängig genutzt werden. Allerdings ist ein stabiler Netzwerkzugang Voraussetzung, damit das PoC genutzt werden kann.

Abschießend können die verwendeten Technologien als zeitgemäß und zweckmäßig angesehen werden.

5.3 Einschränkungen

Die Ergebnisse und Interpretationen unterliegen einigen Einschränkungen, die aus den Rahmenbedingungen der vorliegenden Arbeit resultieren. Die Ergebnisse des PoC basieren auf dem MNIST-Datensatz. Dieser bietet zwar Vorteile, was das Training des cGAN und der xAI-Methoden angeht. Die wenig komplexen Bilder des MNIST-Datensatzes bringen aber folgende Nachteile mit sich:
Die Ziffernbilder sind wenig komplex, für jedes Bild gibt es nur exakt ein zu identifizierendes Objekt. Somit ist die Klassifikation die D vorzunehmen hat einfacher, und D wird nicht unter schwierigeren Bedingungen geprüft. Selbiges gilt für die xAI-Methoden, sie müssen einfache Objekte in einfach strukturierten Bildern erkennen. Diese Einschränkung wurde in Kauf genommen. Komplexere Bilder erfordern deutlich leistungsfähigere Hardware (GPUs) und verlängern den Trainingsprozess erheblich (Rashid (2020), S.119). Die Verwendung von GPUs und damit einhergehend CUDA, würde wiederum Änderungen am Programmcode bedeuten, da Berechnungen per Methodenaufruf auf GPUs verlagert werden müssen. Dann würde das PoC wiederum nicht in einer CPU basierten Laufzeitumgebung funktionieren (Rashid (2020). S.121).

Eine weitere Einschränkung ist, dass das PoC mit einem cGAN arbeitet. Dies erlaubt aussagekräftigere Interpretationen, aber das PoC kann so nicht genutzt werden um die Ausgaben von D eines beliebigen GANs nachvollziehbar zu machen. Hier wäre eine Implementation oder eine Auswahl des GAN-Typs eine wünschenswerte Ergänzung. Zudem muss das cGAN modifiziert werden, damit die Ein- und Ausgaben von D gespeichert werden können. Es können nicht beliebige cGANs analysiert werden. Eine weitere Einschränkung ist der inhaltliche Bezug zwischen der Ausgabe von D und der Möglichkeit diese nachzuvollziehen (vgl. 5.1). Dieser Bezug muss mit Hilfe der xAI-Methoden von Nutzenden des PoC auf Basis der Salienzkarten, der Metriken und der darauf basierenden Diagramme, erstellt werden. Somit ist das PoC nur für Data Scientists nutzbar. Eine Möglichkeit die Ausgaben von D einem beliebigen Nutzenden nachvollziehbar zu machen ist nicht möglich.

5.4 Weitere Untersuchungen

Im Hinblick auf die beschriebenen Einschränkungen des PoC wären folgende weitere Untersuchungen wünschenswert:

Das PoC könnte so umgestaltet werden, dass es universellere Schnittstellen zu beliebigen GANs zur Verfügung stellt. Zielführend wäre hier auch, wenn das PoC nicht nur auf einen konkreten Typ Bilder festgelegt ist. Hier ist ebenfalls eine universellere Schnittstelle wünschenswert, die beliebige Daten (Bild, Schrift, Ton) verarbeiten kann, welche GANs erzeugt haben.

Zum aktuellen Stand muss D modifiziert werden, um die relevanten Daten zu erhalten. Hier wäre eine unkompliziertere Lösung wünschenswert. Es könnte hilfreich sein, direkt Gradienten aus D zu verwenden und diese zur Erklärung der Ausgabe von D zu nutzen. Vergleichbar mit dem Ansatz von GradCAM, um relevante Instanzelemente zu identifizieren.

In Bezug auf die Güte der bereitgestellten Erklärungen wäre es erstrebenswert, dem Nutzenden seinem Wissen entsprechende Erklärungen zu liefern (Graziani u. a. (2023), Miller (2023)). Das würde bedeuten, die Erklärungen sind so gestaltet, dass sie den Definitionen von Interpretable AI (2.3.2) und Responsible AI (vgl. 2.3.3) genügen. Besonders für sensible und schützenswerte Daten ist dies wichtig (Graziani u. a. (2023)).

6 Zusammenfassung und Ausblick

Die vorliegende Arbeit vergleicht verschiedene xAI-Methoden ob ihrer Eignung, die Ausgaben des Diskriminators eines GAN nachvollziehbar zu machen. Dazu wurden xAI-Methoden hinsichtlich ihrer visuellen Ergebnisse bewertet. Für einen aussagekräftigeren Vergleich wurden geeignete Metriken herangezogen. Dafür wurde geprüft, ob und welche Zusammenhänge sich zwischen den Ausgaben von D und den Metriken ergeben. Anhand dieser Ergebnisse wurden die Vor- und Nachteile der xAI-Methoden gegeneinander abgewogen. Im Kontext der Nachvollziehbarkeit von Entscheidungen bzw. Ausgaben einer KI, ist es wichtig die Güte der Erklärungen zu bewerten. Dies wurde anhand der Kriterien, der im Rahmen dieser Arbeit definierten Begriffe für Explainable, Interpretable und Responsible AI bewertet.

Der Vergleich der xAI-Methoden hat gezeigt, dass diese unterschiedlich gut geeignet sind, um die Ausgaben von D nachvollziehbar zu machen. Auf Basis der bestimmten Metriken sind weitere Analysen möglich. So kann das PoC genutzt werden, die Ausgaben von D zu verstehen und das Training eines GAN zu optimieren.

Es wurde festgestellt, dass diese Art von Erkenntnissen und die gesammelten Daten hauptsächlich für Data Scientists relevant sind. Nutzende ohne Domänenkompetenz profitieren wenig von den Ergebnissen. Daher genügen die betrachteten xAI-Methoden den Kriterien der Explainable AI, aber nicht denen der Interpretable- oder der Responsible AI.

Die vorliegende Arbeit hat vier xAI-Methoden verglichen, die zu diesen Zielen beitragen können. Insbesondere wurden die xAI-Methoden nicht in eine Reihenfolge gebracht, was ihre Eignung angeht. Vielmehr wurden die verschiedenen Stärken und Schwächen aufgezeigt. Je nach dem in welchem Szenario ein GAN genutzt wird, können unterschiedliche xAI-Methoden die richtige Wahl zur Nachvollziehbarkeit sein.

Es wäre daher wünschenswert, die Erklärungen so zu gestalten, dass beliebige Nutzende sie verstehen können. Insbesondere wenn die Nutzenden von den generierten Ergebnissen

einer KI in irgendeiner Weise betroffen sind. Dazu wäre es sinnvoll die Ausgaben von D möglichst auf Basis der internen Vorgänge in D nachvollziehbar zu machen. Weiterhin wäre es zielführend, KI direkter zu erklären. Das heißt sich möglichst im Bereich intuitiver Erklärungen zu bewegen, die von beliebigen Nutzenden verstanden werden können. Zusätzlich könnten Data Scientists technische Informationen zur Verfügung gestellt werden. Diese können Data Scientists dann nutzen, um das Training und die Ausgaben der Modelle zu verbessern.
Es können Fehler aus den Modellen entfernen werden. Außerdem können verzerrte Trainingsdaten und Modellausgaben identifiziert und entfernt werden. Falls ein GAN so genutzt wird, dass beliebige Nutzende auf die Ergebnisse zugreifen können, sollten perspektivisch auch die Ausgaben von G nachvollziehbar gemacht werden. So kann ein GAN als Ganzes besser verstanden werden.

Vor dem Hintergrund des am 21.05.2024 von der EU verabschiedeten Gesetztes *Vorschlag für ein Gesetz über künstliche Intelligenz* ist es anzunehmen, dass der Druck auf KI-Anwendungen eigene Ausgaben zu erklären zukünftig steigen wird. Daher kann nur nochmal betont werden, dass neben generativen Modellen wie GANs diverse weitere Anwendungen so umgestaltet werden müssen, dass ihre Ausgaben für beliebige Nutzende nachvollziehbar sind.

Literaturverzeichnis

[Antipov u. a. 2017] Antipov, Grigory ; Baccouche, Moez ; Dugelay, Jean-Luc: *Face Aging With Conditional Generative Adversarial Networks*. Mai 2017. – URL http://arxiv.org/abs/1702.01983. – Zugriffsdatum: 2023-10-07. – arXiv:1702.01983 [cs]

[Arjovsky u. a. 2017] Arjovsky, Martin ; Chintala, Soumith ; Bottou, Léon: *Wasserstein GAN*. Dezember 2017. – URL http://arxiv.org/abs/1701.07875. – Zugriffsdatum: 2024-02-08. – arXiv:1701.07875 [cs, stat]

[Arrieta u. a. 2019] Arrieta, Alejandro B. ; Díaz-Rodríguez, Natalia ; Del Ser, Javier ; Bennetot, Adrien ; Tabik, Siham ; Barbado, Alberto ; García, Salvador ; Gil-López, Sergio ; Molina, Daniel ; Benjamins, Richard ; Chatila, Raja ; Herrera, Francisco: *Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI*. Dezember 2019. – URL http://arxiv.org/abs/1910.10045. – Zugriffsdatum: 2023-09-18. – arXiv:1910.10045 [cs]

[Chakraborty u. a. 2024] Chakraborty, Tanujit ; S, Ujjwal Reddy K. ; Naik, Shraddha M. ; Panja, Madhurima ; Manvitha, Bayapureddy: Ten years of generative adversarial nets (GANs): a survey of the state-of-the-art. In: *Machine Learning: Science and Technology* 5 (2024), Januar, Nr. 1, S. 011001. – URL https://dx.doi.org/10.1088/2632-2153/ad1f77. – Zugriffsdatum: 2024-05-09. – Publisher: IOP Publishing. – ISSN 2632-2153

[Chang u. a. 2022] Chang, Chun-Hao ; Caruana, Rich ; Goldenberg, Anna: *NODE-GAM: Neural Generalized Additive Model for Interpretable Deep Learning*. März 2022. – URL http://arxiv.org/abs/2106.01613. – Zugriffsdatum: 2023-12-03. – arXiv:2106.01613 [cs]

[Copeland 2019] Copeland, B.: *Artificial Intelligence*. 2019. – URL https://www.britannica.com/technology/artificial-intelligence

[Creswell u. a. 2018] Creswell, Antonia ; White, Tom ; Dumoulin, Vincent ; Arulkumaran, Kai ; Sengupta, Biswa ; Bharath, Anil A.: Generative Adversarial Networks: An Overview. In: *IEEE Signal Processing Magazine* 35 (2018), Januar, Nr. 1, S. 53–65. – URL http://arxiv.org/abs/1710.07035. – Zugriffsdatum: 2023-09-15. – arXiv:1710.07035 [cs]. – ISSN 1053-5888

[Daoust 2024] Daoust, Mark: *tensorflow*. https://github.com/tensorflow/docs/blob/master/site/en/tutorials/interpretability/integrated_gradients.ipynb. 2024

[EU-Parlament 2023] EU-Parlament: Europäisches Parlament News. June 18 2023. – Accessed on January 24, 2024

[Fahrmeir 2016] Fahrmeir, Ludwig: *Statistik: Der Weg zur Datenanalyse*. 6. Springer-Verlag, 2016. – 153–169 S

[Frochte 2019] Frochte, Jörg: *Maschinelles Lernen*. 2. Hanser, 2019. – ISBN 978-3-446-45996-0

[Gildenblat 2019] Gildenblat, Jacob: *Grad-CAM with PyTorch*. https://github.com/jacobgil/pytorch-grad-cam/. 2019

[Gohel u. a. 2021] Gohel, Prashant ; Singh, Priyanka ; Mohanty, Manoranjan: *Explainable AI: current status and future directions*. Juli 2021. – URL http://arxiv.org/abs/2107.07045. – Zugriffsdatum: 2024-02-14. – arXiv:2107.07045 [cs]

[Goodfellow u. a. 2014] Goodfellow, Ian J. ; Pouget-Abadie, Jean ; Mirza, Mehdi ; Xu, Bing ; Warde-Farley, David ; Ozair, Sherjil ; Courville, Aaron ; Bengio, Yoshua: *Generative Adversarial Networks*. Juni 2014. – URL http://arxiv.org/abs/1406.2661. – Zugriffsdatum: 2023-09-15. – arXiv:1406.2661 [cs, stat]

[Goodman und Flaxman 2017] Goodman, Bryce ; Flaxman, Seth: European Union regulations on algorithmic decision-making and a "right to explanation". In: *AI Magazine* 38 (2017), September, Nr. 3, S. 50–57. – URL http://arxiv.org/abs/1606.08813. – Zugriffsdatum: 2023-11-19. – arXiv:1606.08813 [cs, stat]. – ISSN 0738-4602, 2371-9621

[Graziani u. a. 2023] Graziani, Mara ; Dutkiewicz, Lidia ; Calvaresi, Davide ; Amorim, José P. ; Yordanova, Katerina ; Vered, Mor ; Nair, Rahul ; Abreu,

Pedro H. ; Blanke, Tobias ; Pulignano, Valeria ; Prior, John O. ; Lauwaert, Lode ; Reijers, Wessel ; Depeursinge, Adrien ; AndreArczyk, Vincent ; Müller, Henning: A global taxonomy of interpretable AI: unifying the terminology for the technical and social sciences. In: *Artificial Intelligence Review* 56 (2023), April, Nr. 4, S. 3473–3504. – URL https://doi.org/10.1007/s10462-022-10256-8. – Zugriffsdatum: 2023-09-18. – ISSN 1573-7462

[Kang 2019] Kang, Jaekoo: *Grad-CAM on MNIST*. https://github.com/jaekookang/mnist-grad-cam.git. 2019

[Kapishnikov u. a. 2019] Kapishnikov, Andrei ; Bolukbasi, Tolga ; Viégas, Fernanda ; Terry, Michael: *XRAI: Better Attributions Through Regions*. August 2019. – URL http://arxiv.org/abs/1906.02825. – Zugriffsdatum: 2023-09-15. – arXiv:1906.02825 [cs, stat]

[Li 2018] Li, Yuxi: *Deep Reinforcement Learning*. Oktober 2018. – URL http://arxiv.org/abs/1810.06339. – Zugriffsdatum: 2023-11-26. – arXiv:1810.06339 [cs, stat]

[Lundberg 2018] Lundberg, Scott: *PyTorch Deep Explainer MNIST example*. https://github.com/shap/shap/tree/master/docs/notebooks/deep_explainer. 2018

[Lundberg und Lee 2017] Lundberg, Scott ; Lee, Su-In: *A Unified Approach to Interpreting Model Predictions*. November 2017. – URL http://arxiv.org/abs/1705.07874. – Zugriffsdatum: 2023-09-15. – arXiv:1705.07874 [cs, stat]

[Mahya und Fürnkranz 2023] Mahya, Parisa ; Fürnkranz, Johannes: An Empirical Comparison of Interpretable Models to Post-Hoc Explanations. In: *AI* 4 (2023), Juni, Nr. 2, S. 426–436. – URL https://www.mdpi.com/2673-2688/4/2/23. – Zugriffsdatum: 2023-09-18. – ISSN 2673-2688

[McCulloch und Pitts 1943] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), Dezember, Nr. 4, S. 115–133. – URL https://doi.org/10.1007/BF02478259. – Zugriffsdatum: 2023-11-19. – ISSN 1522-9602

[Mescheder u. a. 2018] Mescheder, Lars ; Geiger, Andreas ; Nowozin, Sebastian: *Which Training Methods for GANs do actually Converge?* Juli 2018. – URL http:

//arxiv.org/abs/1801.04406. – Zugriffsdatum: 2023-12-03. – arXiv:1801.04406 [cs]

[Microsoft 2020] Microsoft: *Microsoft erklärt: Was ist künstliche Intelligenz? Definition & Funktionen von KI.* News Center Microsoft. March 4 2020. – Accessed on January 24, 2021

[Miller 2019] Miller, Tim: Explanation in artificial intelligence: Insights from the social sciences. In: *Artificial Intelligence* 267 (2019), Februar, S. 1–38. – URL https://www.sciencedirect.com/science/article/pii/S0004370218305988. – Zugriffsdatum: 2024-02-04. – ISSN 0004-3702

[Miller 2023] Miller, Tim: *Explainable AI is Dead, Long Live Explainable AI! Hypothesis-driven decision support.* März 2023. – URL http://arxiv.org/abs/2302.12389. – Zugriffsdatum: 2023-09-18. – arXiv:2302.12389 [cs]

[Mirza und Osindero 2014] Mirza, Mehdi ; Osindero, Simon: *Conditional Generative Adversarial Nets.* November 2014. – URL http://arxiv.org/abs/1411.1784. – Zugriffsdatum: 2024-02-07. – arXiv:1411.1784 [cs, stat]

[Molnar 2022] Molnar, Christoph: *Interpretable Machine Learning.* 2. URL https://christophm.github.io/interpretable-ml-book, 2022

[Molnar u. a. 2020] Molnar, Christoph ; Casalicchio, Giuseppe ; Bischl, Bernd: Interpretable Machine Learning - A Brief History, State-of-the-Art and Challenges. arXiv, 2020, S. 417–431. – URL http://arxiv.org/abs/2010.09337. – Zugriffsdatum: 2023-09-18. – arXiv:2010.09337 [cs, stat]

[Nagisetty u. a. 2022] Nagisetty, Vineel ; Graves, Laura ; Scott, Joseph ; Ganesh, Vijay: *xAI-GAN: Enhancing Generative Adversarial Networks via Explainable AI Systems.* März 2022. – URL http://arxiv.org/abs/2002.10438. – Zugriffsdatum: 2023-09-15. – arXiv:2002.10438 [cs, stat]

[Nesterov 2013] Nesterov, Y.: *Introductory Lectures on Convex Optimization: A Basic Course.* Springer Science & Business Media, Dezember 2013. – ISBN 978-1-4419-8853-9

[Norvig und Russell 2002] Norvig, Peter ; Russell, Stuart J.: *Artificial Intelligence: A Modern Approach, 4th US ed.* 2002. – URL https://aima.cs.berkeley.edu/. – Zugriffsdatum: 2023-11-19

[Paaß und Hecker 2020] Paass, Gerhard ; Hecker, Dirk: *Künstliche Intelligenz: Was steckt hinter der Technologie der Zukunft?* Wiesbaden : Springer Fachmedien, 2020. – URL http://link.springer.com/10.1007/978-3-658-30211-5. – Zugriffsdatum: 2023-11-25. – ISBN 978-3-658-30210-8 978-3-658-30211-5

[Palacio u. a. 2021] Palacio, Sebastian ; Lucieri, Adriano ; Munir, Mohsin ; Hees, Jörn ; Ahmed, Sheraz ; Dengel, Andreas: *XAI Handbook: Towards a Unified Framework for Explainable AI*. Mai 2021. – URL http://arxiv.org/abs/2105.06677. – Zugriffsdatum: 2023-09-18. – arXiv:2105.06677 [cs]

[Radford u. a. 2016] Radford, Alec ; Metz, Luke ; Chintala, Soumith: *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. Januar 2016. – URL http://arxiv.org/abs/1511.06434. – Zugriffsdatum: 2023-11-27. – arXiv:1511.06434 [cs]

[Rashid 2020] Rashid, Tariq: *GANs mit PyTorch selbst programmieren: Ein verständlicher Einstieg in Generative Adversarial Networks*. dpunkt.verlag, 2020

[Rashid 2024] Rashid, Tariq: *ImageNetLabels*. https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a. 2024

[Revar 2020] Revar, Y.: *Conditional GAN - MNIST*. https://github.com/makeyourownneuralnetwork/gan/blob/master/16_cgan_mnist.ipynb. 2020

[Ribeiro u. a. 2016] Ribeiro, Marco T. ; Singh, Sameer ; Guestrin, Carlos: *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. August 2016. – URL http://arxiv.org/abs/1602.04938. – Zugriffsdatum: 2023-09-15. – arXiv:1602.04938 [cs, stat]

[Rudin 2019] Rudin, Cynthia: *Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead*. September 2019. – URL http://arxiv.org/abs/1811.10154. – Zugriffsdatum: 2023-09-18. – arXiv:1811.10154 [cs, stat]

[Saeed und Omlin 2023] Saeed, Waddah ; Omlin, Christian: Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities. In: *Knowledge-Based Systems* 263 (2023), März, S. 110273. – URL https://www.sciencedirect.com/science/article/pii/S0950705123000230. – Zugriffsdatum: 2023-09-18

[Salimans u. a. 2016] Salimans, Tim ; Goodfellow, Ian ; Zaremba, Wojciech ; Cheung, Vicki ; Radford, Alec ; Chen, Xi: *Improved Techniques for Training GANs*. Juni 2016. – URL http://arxiv.org/abs/1606.03498. – Zugriffsdatum: 2023-09-24. – arXiv:1606.03498 [cs]

[Samoilescu 2024] Samoilescu, Robert: *Integrated gradients for MNIST*. https://github.com/SeldonIO/alibi/blob/ab0b6a600eaeb1a9952eed769bfb5997cfa083a3/doc/source/examples/integrated_gradients_mnist.ipynb. 2024

[Selvaraju u. a. 2020] 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 (2020), Februar, Nr. 2, S. 336–359. – URL http://arxiv.org/abs/1610.02391. – Zugriffsdatum: 2023-12-11. – arXiv:1610.02391 [cs]. – ISSN 0920-5691, 1573-1405

[Shapley 1953] Shapley, L. S.: *17. A Value for n-Person Games*. S. 307–318. In: Kuhn, Harold W. (Hrsg.) ; Tucker, Albert W. (Hrsg.): *Contributions to the Theory of Games (AM-28), Volume II*. Princeton : Princeton University Press, 1953. – URL https://doi.org/10.1515/9781400881970-018. – ISBN 9781400881970

[Simonyan u. a. 2014] Simonyan, Karen ; Vedaldi, Andrea ; Zisserman, Andrew: *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. April 2014. – URL http://arxiv.org/abs/1312.6034. – Zugriffsdatum: 2024-05-04

[Steinwendner und Schwaiger 2020] Steinwendner, Joachim ; Schwaiger, Roland: *Neuronale Netze programmieren mit Python: Der Einstieg in die Künstliche Intelligenz*. dpunkt.verlag, 2020. – ISBN 978-3-8362-7450-0

[Sundararajan u. a. 2017] Sundararajan, Mukund ; Taly, Ankur ; Yan, Qiqi: *Axiomatic Attribution for Deep Networks*. Juni 2017. – URL http://arxiv.org/abs/1703.01365. – Zugriffsdatum: 2023-09-15. – arXiv:1703.01365 [cs]

[Vainio-Pekka u. a. 2023] Vainio-Pekka, Heidi ; Agbese, Mamia Ori-Otse ; Jantunen, Marianna ; Vakkuri, Ville ; Mikkonen, Tommi ; Rousi, Rebekah ; Abrahamsson, Pekka: The Role of Explainable AI in the Research Field of AI Ethics. In: *ACM Transactions on Interactive Intelligent Systems* 13 (2023), Dezember, Nr. 4,

S. 26:1–26:39. – URL https://dl.acm.org/doi/10.1145/3599974. – ISSN 2160-6455

[Wittpahl 2019] Wittpahl, Volker (Hrsg.): *Künstliche Intelligenz: Technologie | Anwendung | Gesellschaft*. Berlin, Heidelberg : Springer, 2019. – URL http://link.springer.com/10.1007/978-3-662-58042-4. – Zugriffsdatum: 2023-11-25. – ISBN 978-3-662-58041-7 978-3-662-58042-4

[Yang und Berdine 2023] Yang, Shengping ; Berdine, Gilbert: Interpretable artificial intelligence (AI) – saliency maps. In: *The Southwest Respiratory and Critical Care Chronicles* 11 (2023), Juli, Nr. 48, S. 31–37. – URL https://pulmonarychronicles.com/index.php/pulmonarychronicles/article/view/1209. – Zugriffsdatum: 2024-05-14. – Number: 48

A Anhang

Vorkommen und Ausgabe von D nach Labeln				Confusionmatrix nach Klassenlabeln			
Label	#Absolut	Relativ	Ø-Ausgabe von D	Ø-True Positive	Ø-False Negative	Ø-False Positive	Ø-True Negative
0	31	0,0864	0,2645	0,9994	0,0006	0,1038	0,8962
1	28	0,0780	0,2392	0,9980	0,0020	0,0589	0,9411
2	37	0,1031	0,2208	0,9988	0,0012	0,1112	0,8888
3	34	0,0947	0,2032	0,9981	0,0019	0,1280	0,8720
4	40	0,1114	0,1698	0,9967	0,0033	0,1012	0,8988
5	39	0,1086	0,2313	0,9997	0,0003	0,1113	0,8887
6	31	0,0864	0,2309	0,9986	0,0014	0,0873	0,9127
7	39	0,1086	0,1374	0,9983	0,0017	0,0903	0,9097
8	39	0,1086	0,1804	0,9992	0,0008	0,0984	0,9016
9	41	0,1142	0,2065	0,9997	0,0003	0,0731	0,9269

Abbildung A.1: **IG**: Metriken und Ausgaben von D gruppiert nach Labeln für den Trainingsverlauf (eigene Abbildung).

Vorkommen und Ausgabe von D nach Labeln				Confusionmatrix nach Klassenlabeln			
Label	#Absolut	Relativ	Ø-Ausgabe von D	Ø-True Positive	Ø-False Negative	Ø-False Positive	Ø-True Negative
0	30	0,0847	0,2733	0,5860	0,4140	0,0749	0,9251
1	28	0,0791	0,2392	0,5956	0,4044	0,1730	0,8270
2	37	0,1045	0,2208	0,4950	0,5050	0,1094	0,8906
3	34	0,0960	0,2032	0,5064	0,4936	0,0914	0,9086
4	38	0,1073	0,1786	0,4652	0,5348	0,1050	0,8950
5	39	0,1102	0,2313	0,5125	0,4875	0,1095	0,8905
6	31	0,0876	0,2309	0,5998	0,4002	0,1230	0,8770
7	39	0,1102	0,1374	0,5730	0,4270	0,1502	0,8498
8	38	0,1073	0,1786	0,5042	0,4958	0,0623	0,9377
9	40	0,1130	0,2068	0,5818	0,4182	0,1393	0,8607

Abbildung A.2: **SHAP**: Metriken und Ausgaben von D gruppiert nach Labeln für den Trainingsverlauf (eigene Abbildung).

Vorkommen und Ausgabe von D nach Labeln					Confusionmatrix nach Klassenlabeln			
Label	#Absolut	Relativ	Ø-Ausgabe von D		Ø-True Positive	Ø-False Negative	Ø-False Positive	Ø-True Negative
0	31	0,0864	0,2645		0,8901	0,1099	0,7015	0,2985
1	28	0,0780	0,2392		0,6810	0,2476	0,2950	0,6335
2	37	0,1031	0,2208		0,7952	0,2048	0,5963	0,4037
3	34	0,0947	0,2032		0,7986	0,2014	0,6018	0,3982
4	40	0,1114	0,1698		0,7282	0,2718	0,4721	0,5279
5	39	0,1086	0,2313		0,8154	0,1846	0,6077	0,3923
6	31	0,0864	0,2309		0,9111	0,0889	0,5904	0,4096
7	39	0,1086	0,1374		0,6954	0,2534	0,4305	0,5182
8	39	0,1086	0,1804		0,9175	0,0825	0,6138	0,3862
9	41	0,1142	0,2065		0,8920	0,1080	0,5712	0,4288

Abbildung A.3: **LIME**: Metriken und Ausgaben von D gruppiert nach Labeln für den Trainingsverlauf (eigene Abbildung).

Erklärung zur selbstständigen 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