

Machine Learning: Generative Adversarial Networks

Seminararbeit

Ausgewählte Themen der Informatik

des Studienganges Angewandte Informatik an der
Dualen Hochschule Baden-Württemberg Mosbach

von

Mirco Heck

16. Dezember 2023

Bearbeitungszeitraum Studienjahr 2023/24

Matrikelnummern , Kurs 7306389, MOS-TINF21A

Inhaltsverzeichnis

1 Einleitung	1
2 Grundlagen	2
2.1 Künstliche Neuronale Netze	2
2.1.1 Prinzip	2
2.1.2 Merkmale	3
2.2 Deep Learning	4
2.3 Generative Modelle	6
3 Generative Adversarial Networks	7
3.1 Konzept	7
3.2 Architektur	8
3.2.1 Allgemein	8
3.2.2 Variationen	10
3.3 Training	12
4 Anwendungen von GANs	13
4.1 Bildsynthese	13
4.2 Super-Resolution	15
4.3 Style Transfer	16
5 Herausforderungen	17
5.1 Failure to Converge	17
5.2 Mode Collapse	18
5.3 Ethische Aspekte	19
6 Schlussfolgerungen und Ausblick	20

Abkürzungsverzeichnis

KNN Künstliches Neuronales Netz

GAN Generative Adversarial Network

CNN Convolutional Neural Network

DCGAN Deep Convolutional Generative Adversarial Network

RAN Recurrent Adversarial Network

Abbildungsverzeichnis

1.1	Beispiel für ein Bild, das von einem Generative Adversarial Network (GAN) generiert wurde. Ausgangssatz: „Ein roter Panda, der ein Einrad fährt“	1
2.1	Schematische Darstellung eines Neuronalen Netzes	3
2.2	Beispiel für neuronales Netz mit einer „hidden layer“	4
3.1	Funktionsprinzip eines GAN	9
4.1	Demonstration von Artbreeder und Modifikation des Originals(links) durch Erhöhung des „Hundefaktors“	14
4.2	KI-genertiertes Gesicht	14
4.3	Demonstration Super-Resolution-GAN: Originalbild(links), verkleinertes Original(mitte) und per KI restaurierte Version(rechts, mit Wasserzeichen)	15
4.4	Style Transfer GAN in der Praxis, Motiv: Todd Howard, Bethesda Softworks . .	16

1 Einleitung

Generative Adversarial Networks (GANs) haben in den letzten Jahren erhebliche Aufmerksamkeit in der Forschungsgemeinschaft auf sich gezogen. Sie stellen eine neue Methode dar, generative Modelle zu trainieren und haben eine Vielzahl von Anwendungen in Bereichen wie Bildsynthese, Super-Resolution und Style Transfer.

Das Ziel dieser Arbeit ist es, einen umfassenden Überblick über GANs zu geben, ihre Funktionsweise zu erklären und einige der Herausforderungen zu diskutieren, die bei ihrer Implementierung und ihrem Training auftreten.



Abbildung 1.1: Beispiel für ein Bild, das von einem GAN generiert wurde.

Ausgangssatz: „Ein roter Panda, der ein Einrad fährt“

Das Paper ist wie folgt strukturiert: Nach dieser Einleitung werden in Kapitel 2 die Grundlagen von neuronalen Netzen, Deep Learning und generativen Modellen erläutert. Kapitel 3 ist den GANs gewidmet, wobei ihr Konzept, ihre Architektur und ihr Training im Detail besprochen werden. Kapitel 4 behandelt verschiedene Anwendungen von GANs, während Kapitel 5 einige der Herausforderungen und Lösungsansätze bei der Arbeit mit GANs diskutiert. Schließlich werden in Kapitel 6 Schlussfolgerungen gezogen und ein Ausblick auf zukünftige Forschungsrichtungen gegeben.

2 Grundlagen

2.1 Künstliche Neuronale Netze

Bevor wir uns GANs genauer anschauen können, ist es wichtig die Grundlagen von künstlichen neuronalen Netzen zu verstehen, auf welchen die Technologie fußt.

2.1.1 Prinzip

Künstliche Neuronale Netze (KNNs) bilden einen wichtigen Zweig der Künstlichen Intelligenz und formen die Basis für Deep Learning-Technologien, welche unter anderem auch GANs umfassen. Sie werden bereits heutzutage erfolgreich in verschiedenen Prozessen, wie Mustererkennung, Kategorisierung und Prognose von Daten oder Optimierung von Abläufen eingesetzt. Ihre Arbeitsweise liegt darin, eine Menge von Eingaben in sogenannte Eingabevektoren zu kodieren und durch das neuronale Netz daraus eine Menge an Ausgabevektoren zu generieren. Diese Ausgabevektoren können wiederum in ein Ergebnis beliebigen Formats (z.B. Binärdaten, Text, Audio oder Grafiken) kodiert werden. Die Struktur von KNNs sind von der Funktionsweise des menschlichen Gehirns inspiriert und bestehen aus einer Reihe von miteinander verbundenen Knoten, die als Neuronen bezeichnet werden und als simple Prozessoren fungieren. Diese Prozessoren können nur einfache Operationen ausführen, sind aber in der Lage, komplexe Aufgaben zu erledigen, wenn sie in großer Anzahl miteinander verbunden sind. Die Verbindungen zwischen den Neuronen werden als Kanten bezeichnet und haben ein Gewicht, das die Stärke der Verbindung zwischen den Neuronen angibt. Die Menge der Kanten und deren Gewichtungen definieren dabei die Transformation, welche die Eingangsvektoren beim Durchlauf des KNNs erfahren. Die Neuronen sind in Schichten angeordnet, wobei jede Schicht eine Reihe von Neuronen enthält, die eine bestimmte Funktion ausführen. Die erste Schicht wird als Eingabeschicht bezeichnet, die letzte als Ausgabeschicht und alle dazwischen liegenden Schichten werden als versteckte Schichten bezeichnet.[1]

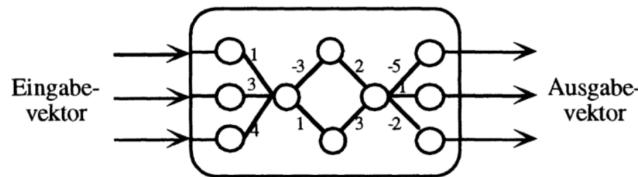


Abbildung 2.1: Schematische Darstellung eines Neuronalen Netzes

2.1.2 Merkmale

KNNs zeichnen sich durch folgende Merkmale aus:

- KNNs sind in der Lage, aus Beispielen zu lernen, ohne explizit programmiert zu werden. Dieser Vorgang wird als Lernen oder Training bezeichnet und ist der wichtigste Aspekt von KNNs. Das Lernen erfolgt durch Anpassung der Gewichtungen der Kanten zwischen den Neuronen, um die gewünschte Ausgabe zu erzeugen und wird wiederholt, bis die KI bei bestimmten Eingaben das erwünschte Resultat erzeugt, oder zumindest annähert. Näheres dazu in Kapitel 2.
- KNNs sind sehr robust und fehlertolerant, da sie in der Lage sind, auch bei fehlerhaften oder unvollständigen Daten zu arbeiten. Dies ist ein großer Vorteil gegenüber herkömmlichen Algorithmen, die bei verrauschten Daten, wie sie beispielsweise in Sensormesswerten vorkommen, häufig nicht mehr zuverlässig funktionieren.
- KNNs können mit einer großen Menge an Daten umgehen und sind in der Lage, Muster in diesen Daten zu erkennen. Dadurch können Lernstrategien oder Entscheidungen auf Basis von Erfahrungen getroffen werden, was ideal für die Inter- und Extrapolation von Daten ist.
- KNNs bieten in ihrer Anwendungsphase eine durchaus gute Performanz, die sich mit der Größe des Netzes und der Anzahl der Trainingszyklen verbessert. Die Trainingsphase hingegen ist sehr rechenintensiv und kann je nach Größe des Netzes und der Anzahl der Trainingszyklen mehrere Stunden oder sogar Tage dauern. Zudem benötigt eine erfolgreiche Trainingsphase eine große Menge an qualitativen Trainingsdaten, die in der Regel manuell ausgewählt werden und vorsichtig administriert werden müssen.

2.2 Deep Learning

Wie bereits im vorherigen Kapitel erwähnt, bieten künstliche neuronale Netze die Möglichkeit, komplexe Zusammenhänge in Daten zu erkennen. Um diese Fähigkeit zu erlangen, müssen die Netze jedoch erst gebaut werden. Dies geschieht durch die Bildung neuer Schichten und die Anpassung der Gewichte der einzelnen Neuronen. Zu den frühen Zeiten der künstlichen Intelligenz, wurden neuronale Netze von Menschen wie Algorithmen aufgebaut, was nicht nur die Komplexität der Netze begrenzte, sondern voraussetzte, dass die Entwickler den Lösungsweg mathematisch beschreiben konnten. Besonders bei Themen, wie der Erkennung von Sprache und Gesichtern, die vom Gehirn intuitiv erledigt werden, erwies sich dies als sehr schwer. Deshalb wurde der Prozess des *Deep Learnings* entwickelt. Hier soll das neuronale Netz nicht manuell gebaut werden müssen, sondern baut sich selbst, ähnlich wie es das menschliche Gehirn tut. Um dies zu erreichen, muss das Netz trainiert werden. Hierfür wird eine Sammlung von Ein- und Ausgangswertpaaren an die KI gefüttert, welche anschließend die Gewichte so anpasst, dass die Ausgabe des Netzes möglichst nahe an der gewünschten Ausgabe liegt. Dieser Vorgang wird als *Backpropagation* bezeichnet. Die Differenz zwischen der gewünschten und der tatsächlichen Ausgabe wird berechnet und auf die Gewichte der einzelnen Neuronen zurückgeführt. Die Gewichte werden dann so angepasst, dass die Differenz zwischen gewünschter und tatsächlicher Ausgabe minimiert wird. Dieser Vorgang wird so lange wiederholt, bis die Differenz zwischen gewünschter und tatsächlicher Ausgabe minimal ist. Durch diesen Prozess entstehen eine oder mehrere sogenannte *hidden layers*, abstrakte Schichten, welche die Ein- und Ausgangsschichten miteinander verbinden und die Daten in immer abstraktere Formen umwandeln. Die Anzahl der hidden layers und die Anzahl der Neuronen in diesen Schichten sind frei wählbar. Je mehr hidden layers und Neuronen vorhanden sind, desto komplexere Zusammenhänge können erkannt werden. Dieser Prozess wird in Abbildung 4.4 dargestellt.[2]

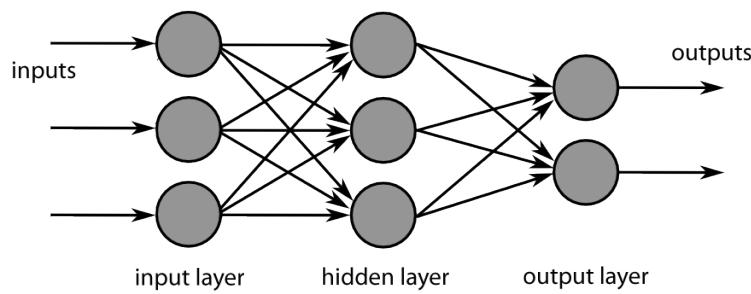


Abbildung 2.2: Beispiel für neuronales Netz mit einer „hidden layer“
 Quelle: <http://commons.wikimedia.org>

Deep Learning ist eine Unterkategorie des Machine Learnings und wird auch als *hierarchisches Lernen* bezeichnet, da die Daten in immer abstraktere Formen umgewandelt werden. Dabei ist der Oberbegriff Machine Learning als solches nicht nur auf neuronale Netze beschränkt, sondern kann auch auf andere Technologien angewendet werden, wie zum Beispiel auf Entscheidungsbäume, Bayes-Netze oder Support Vector Machines. Von Deep Learning wird allerdings nur gesprochen, wenn neuronale Netze verwendet werden. Der Name „Deep Learning“ stammt dabei von der Tiefe der verborgenen *hidden layers*.

Der Prozess des Deep Learnings ist sehr rechenintensiv und benötigt eine große Menge an Trainingsdaten, die in der Regel manuell ausgewählt werden und vorsichtig administriert werden müssen. Zudem ist die Trainingsphase sehr langwierig und kann je nach Größe des Netzes und der Anzahl der Trainingszyklen mehrere Stunden oder sogar Tage dauern. Die Anwendungsphase hingegen ist sehr performant und kann in Echtzeit erfolgen. Der eigentliche Prozess des Deep Learnings ist dabei nicht neu, sondern wurde bereits in den 1980er Jahren entwickelt. Damals scheiterte die Technologie jedoch an der mangelnden Rechenleistung der Computer und der geringen Menge an Trainingsdaten. Erst durch die Entwicklung von leistungsstarken Grafikkarten und die Verfügbarkeit großer Datenmengen, wie sie zum Beispiel im Internet zu finden sind, wurde Deep Learning zu einer praktikablen Technologie.

Durch die Fähigkeit aus großen Datensätzen komplexe Zusammenhänge zu erkennen, ist Deep Learning gut für die Replikation abstrakter Prozesse geeignet, für welche sich das Entwickeln eines klassischen Algorithmus als schwierig herausstellt. Besonders wenn die Daten sehr komplex sind und große Datensätze für das Training verfügbar sind. Exzellente Beispiele für den Einsatz von Deep Learning wären zum Beispiel die Erkennung von Sprache und Gesichtern. Solche Technologien sind dabei schon weitläufig in Anwendergeräten im Einsatz, wie beispielsweise in Form von virtuellen Assistenten, die die Sprache des Anwenders verstehen können oder Kamera-Apps, die die Motive und Gesichter auf Fotos erkennen und anhand dessen klassifizieren. Aber auch in der Medizin und der Biologie wird Deep Learning eingesetzt, um zum Beispiel Krebszellen zu erkennen oder die Struktur von Proteinen zu analysieren. Auch in der Robotik wird Deep Learning eingesetzt, um Roboter zu entwickeln, die sich selbstständig bewegen können.

2.3 Generative Modelle

Generative Modelle sind Modelle, welche durch unüberwachtes Lernen erzeugt werden, einer Unterkategorie des Machine Learnings, in welcher sich auch GANs einordnen lassen. Im Gegensatz zu anderen Deep Learning Modellen, wie zum Beispiel Convolutional Neural Networks (CNNs), welche die Daten klassisch als Input-Output-Paar oder auch Daten-Label-Paar zum Training bereitgestellt bekommen, werden bei generativen Modellen keine Labels benötigt, um die Daten zu klassifizieren. Stattdessen werden die Daten selbst analysiert und daraus ein Modell erzeugt, welches die Daten möglichst gut abbildet. Dieses Modell kann dann verwendet werden, um neue Daten zu generieren, welche den Trainingsdaten möglichst ähnlich sind.[3]

Diese Art des Machine Learnings wird in vielen Bereichen eingesetzt, wie zum Beispiel in der Bildverarbeitung, der Spracherkennung, der Sprachsynthese, der Textverarbeitung oder der Musiksynthese. Sie werden genutzt, um Daten zu generieren, die kaum von natürlichen Daten zu unterscheiden sind.

Generative Modelle können in zwei Kategorien unterteilt werden: *explicit generative models* und *implicit generative models*. Bei *explicit generative models* wird die Wahrscheinlichkeitsverteilung der Daten explizit modelliert. Dies geschieht in der Regel durch die Verwendung von Bayes'schen Netzen. Bei *implicit generative models* wird die Wahrscheinlichkeitsverteilung der Daten nicht explizit modelliert, sondern durch ein Modell approximiert. Dies wird in der Regel mit neuronalen Netzen realisiert, welche die Wahrscheinlichkeitsverteilung der Daten approximieren. Bei GANs handelt es sich um ein *implicit generative model*, welches mit neuronalen Netzen arbeitet. In Kapitel 3 wird dabei die Architektur von GANs genauer erläutert.

3 Generative Adversarial Networks

3.1 Konzept

Generative Adversarial Networks (GANs) basieren auf der zuvor genannten Technologie des unüberwachten Lernens und bedienen somit ähnliche Einsatzzwecke, nämlich der Generation von Daten, die vom menschlichen Gehirn nicht von reellen Daten unterscheiden lassen. Im Falle der GANs geschieht dies mit der Hilfe von zwei unabhängigen, impliziten generativen Modellen: dem Generator und dem Diskriminator. Diese beiden Modelle werden als „adversarial“ (gegensätzlich) bezeichnet, da sie im Wettbewerb zueinander stehen und einander trainieren.

Diese Art der künstlichen Intelligenz hat sich vor allem bezüglich der Bildverarbeitung als sehr erfolgreich erwiesen. Dies wird nicht nur in der reinen Generation von Bildern eingesetzt, sondern erlaubt ebenfalls andere bildbezogenen Methoden, wie die Super-Resolution, welche es erlaubt Bilder mit niedrigem Detailgrad neue Details zu erschaffen, die im Original nicht vorhanden sind. Eine weitere Option ist der sogenannte Style Transfer, welcher die Möglichkeit bietet den Stil eines Bildes, z.B. realistisch, abstrakt, Zeichentrick, etc. auf ein anderes Bild übertragen zu können. Besonders die Super-Resolution, auch AI-based Upscaling genannt, wird bereits heut zutage weitläufig eingesetzt, um beispielsweise Musikvideos, welche vor langer Zeit auf analoge Formate, wie Magnetbändern aufgezeichnet wurde, zu restaurieren und den analogen, 576i-Standard Definition-Look zu entfernen. Dadurch können mit genug Feinarbeit alte Videos so bearbeitet werden, dass sie nicht nur eine moderne Auflösung wie UHD besitzen, sondern auch wesentlich schärfer und klarer aussehen.

Um besser nachvollziehen zu können, wie die zwei konkurrierenden Modelle in solch komplexen Prozessen resultieren können, lohnt es sich hier ein genauer Blick auf die Architektur der GANs zu werfen.

3.2 Architektur

3.2.1 Allgemein

Die Basis eines GANs bilden wie bereits erwähnt zwei Modelle, der Generator und der Diskriminator. Diese beiden Modelle agieren als Gegenspieler und trainieren einander, um zu einem besseren Gesamtresultat zu führen. Somit gehören beide Komponenten stets zueinander und sind jeweils beide für die korrekte Funktion der Architektur unverzichtbar. Die Daten, mit welchen die Modelle arbeiten, können in Form von verschiedenen Formaten definiert sein, wie Bildern, Texten oder sogar Musik, in den nachfolgenden Kapiteln wird allerdings ausschließlich der visuelle Aspekt von GAN betrachtet. Dabei unterscheiden sich die Arbeitsweisen der beiden Modelle grundlegend:

Generator: Der Generator erzeugt neue Daten, die echten Daten möglichst ähnlich sehen sollen. Es ist das Modell der beiden, welches nach dem Abschließen des Trainings für die tatsächliche Generation von neuen Daten verwendet wird. Der Generator wird mit zufälligem Rauschen als Eingabe gestartet, welches als Latent Space bezeichnet wird. Dieses Rauschen in Form eines Vektors wird als Eingabe in den Generator eingespeist, welcher durch das Feedback des Diskriminators im Laufe der Zeit lernt, Daten zu erzeugen, die von einem echten Datensatz nicht zu unterscheiden sind. Dieser Prozess wird auch als *Backpropagation* bezeichnet und erlaubt das unüberwachte Lernen der Netze.

Diskriminator: Der Diskriminator hat die Aufgabe, zwischen echten Daten und den vom Generator erzeugten Daten zu unterscheiden. Er wird mit einer Mischung aus echten und generierten Daten trainiert und wird nach Abschluss des Trainings für die eigentliche Generation von Bildern nicht mehr benötigt, ist jedoch dennoch für das Training des GANs unverzichtbar. Der Diskriminator lernt, die beiden Arten von Daten auseinanderzuhalten, indem er die Wahrscheinlichkeit berechnet, mit welcher das aktuelle Objekt künstlich erzeugt wird, woraufhin eine binäre Klassifikation durchgeführt wird, welche eine binäre Entscheidung festlegt und die Korrektheit der Entscheidung auswertet. Dies ist möglich, da der Diskriminator im Gegensatz zum Generator Zugriff auf die Realdaten hat und geschieht ebenfalls durch Backpropagation.

Der Prozess des Trainings wird in einem nachfolgenden Kapitel noch näher behandelt.

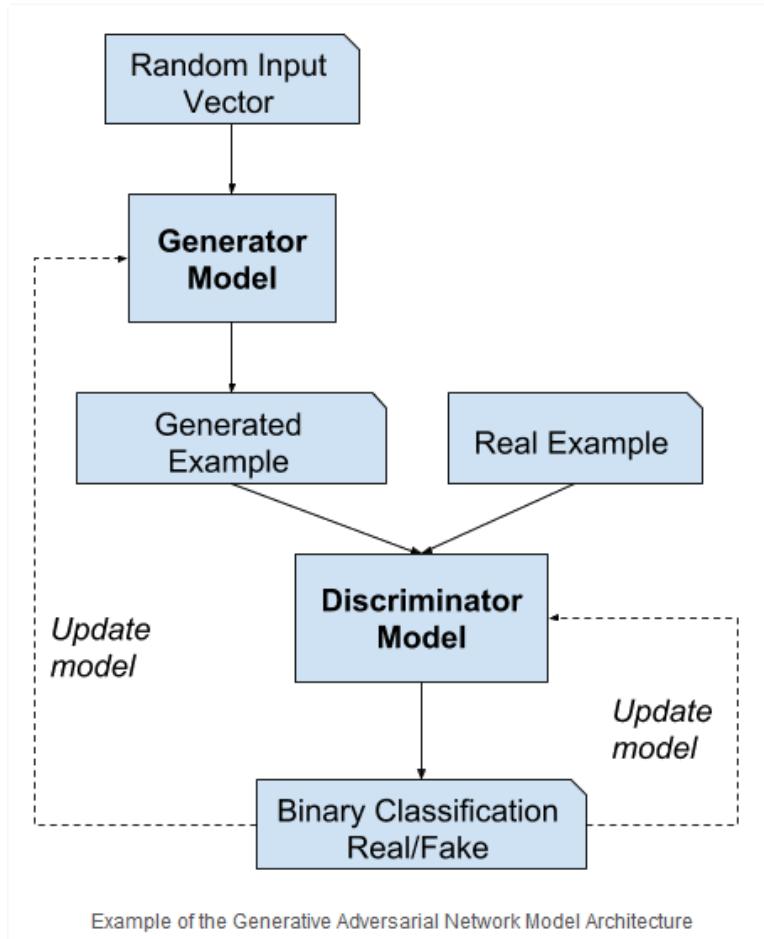


Abbildung 3.1: Funktionsprinzip eines GAN

Die Abbildung zeigt die grundlegende Architektur eines GAN, wie es auch schon in den Grundlagen zuvor erklärt wurde. Der Generator erhält als Eingabe einen zufälligen Vektor, der als Latent Space bezeichnet wird. Dieser Vektor wird als Eingabe in den Generator eingespeist und durchläuft eine Reihe von Schichten, die jeweils eine Reihe von Neuronen enthalten. Die letzte Schicht des Generators, die Ausgabeschicht, gibt einen Vektor aus, der die generierten Daten enthält. Der Diskriminatator erhält schließlich als Eingabe entweder diesen oder einen aus echten Daten stammenden Vektor als Eingangswert. Dieser Vektor durchläuft anschließend einen ähnlichen Prozess, wie der Latent Space in Form des Diskriminators. Die letzte Schicht des Diskriminators gibt schließlich einen Vektor aus, der die Wahrscheinlichkeit angibt, ob die Eingabe aus echten oder generierten Daten besteht. Diese Wahrscheinlichkeit bietet in Form der *Backpropagation* die Grundlage für das Training beider Modelle, allerdings auf zwei unterschiedliche Arten und Weisen. Sollte der Diskriminatator nämlich falsch liegen und beispielsweise ein künstlich generiertes Bild als real anerkennen, so gilt dies für den Generator als Erfolg, für den Diskriminatator allerdings als Fehlschlag. Dieses Verhältnis der beiden Modelle kann durch

eine Funktion, die sogenannte *Minimax-Verlustfunktion* 3.1 beschrieben werden.

$$L_{GAN} = \frac{1}{N_1} \sum_{i=1}^{N_1} \ln D(\mathbf{x}_i) + \frac{1}{N_2} \sum_{j=1}^{N_2} \ln (1 - D(G(\mathbf{z}_j))) \quad (3.1)$$

Der Diskriminatator versucht dabei den oberen Term zu maximieren, indem korrekt zwischen echten und generierten Daten unterschieden wird. Der Generator hat allerdings die Fähigkeit den hinteren Teil des Terms zu beeinflussen, welchen er zu minimieren versucht. Diese Balance zwischen den beiden Modellen und ihren gegenseitigen Einflüssen, wird durch diesen Term gezeigt, welcher seinen Namen *Minimax* exakt aufgrund dieser Eigenschaft besitzt.

3.2.2 Variationen

Die Grundlagen der Funktionsweise eines GANs sind nun bekannt, wobei einige Parameter, wie die Art der eingesetzten Modelle generell gehalten werden und von der verwendeten GAN-Art abhängen. Außerhalb dessen existieren viele weitere Variationen der Architektur, die sich in der Regel auf bestimmte Anwendungsfälle spezialisieren. Die Grundlagen der Funktionsweise bleiben allerdings bei allen Variationen gleich.

Conditional GAN Ein bekanntes Beispiel für eine Erweiterung der GAN-Architektur ist die Verwendung von Datenlabels. Diese Datenlabel werden in der Regel von Menschenhand erstellt und geben an, ob ein Bild beispielsweise eine Katze oder einen Hund zeigt. Diese Datenlabel können nun in den Trainingsprozess mit einbezogen werden, indem der Diskriminatator nicht nur zwischen echten und generierten Daten unterscheidet, sondern auch zwischen den verschiedenen Klassen, die durch die Datenlabel definiert werden. Dieser Prozess wird als *Conditional GAN* bezeichnet und besitzt in der Praxis meist einen größeren Nutzen für die Bildgeneration als ein einfaches GAN. Dessen Generator wird nämlich lediglich durch den randomisierten Eingang gesteuert, welcher es nicht zulässt Einfluss auf die Ausgabe zu nehmen. Die Ausgabe ist also zufällig. Bei einer Conditional GAN hingegen, kann dem Generator zusätzlich bestimmte Eingangsbedingungen mitgegeben werden, die beispielsweise den Stil oder das Motiv des resultierenden Bildes beeinflussen können.

Deep Convolutional GAN Eine weitere Möglichkeit der Verbesserung der Architektur sind die sogenannten Deep Convolutional Generative Adversarial Networks (DCGANs). Diese Variation der klassischen GANs nutzt spezielle neuronale Netze als Generatoren und Diskriminatoren, die CNNs. Diese besitzen auf der Eingangsseite sogenannte *Convolutional Layer*, was auf Deutsch etwa „faltende Schicht“ bedeutet. Diese Schicht erlaubt es dem Netzwerk Verhältnisse zwischen benachbarten Pixeln zu erkennen, wodurch sich CNNs hervorragend zur Erkennung von Mustern eignet. Dies vereinfacht die Konvertierung von Bilddaten in einem zweidimensionalen Raster zu den Modell-internen Datenstrukturen. Dadurch eignen sich DCGANs besonders gut für visuelle Einsatzzwecke, wobei klassische GANs auch für andere Gebiete eignen.

Recurrent Adversarial Networks Die Architektur der Recurrent Adversarial Networks (RANs) ist eine weitere Variation der GANs, die sich besonders für die Verarbeitung von Texten und Audiodaten eignet und bietet einen Gegenpol zu den DCGANs, welche auf Bilddaten spezialisiert sind. Diese Variation nutzt die sogenannten *Recurrent Neural Networks*, welche sich besonders gut für die Verarbeitung von Sequenzen und Echtzeit-Daten eignen. Sie bestehen aus einem *Convolutional Encoder* und einem *Recurrent Decoder*. Der Encoder wandelt die Eingangsdaten in einen Vektor um, der anschließend vom Decoder in die gewünschte Form zurückgewandelt wird. Diese Architektur eignet sich besonders gut für die Verarbeitung von Texten und Audiodaten, da diese in der Regel in Form von Sequenzen vorliegen.

Dies sind nur einige der vielen Variationen, die sich in der Regel auf bestimmte Anwendungsfälle spezialisieren. Die Grundlagen der Funktionsweise bleiben allerdings bei allen Variationen gleich. Weitere Beispiele für GAN-Variationen, auf die in diesem Paper nicht näher eingegangen wird, sind:

- Laplacian GAN
- Wasserstein GAN
- Energy-based GAN
- CycleGAN
- InfoGAN
- ...

3.3 Training

Das Training von GANs basiert auf unüberwachtem Lernen und erfolgt in einem iterativen Prozess, der in drei Schritten durchgeführt wird:

1. Der Generator erzeugt eine Reihe von Daten auf Basis des Latent Spaces.
2. Der Diskriminator erhält zufällig das vom Generator erzeugte Bild oder ein echtes Bild und überprüft die Echtheit.
3. Das Ergebnis des Diskriminators wird ausgewertet und beide Netze werden durch das Resultat beeinflusst.

Dabei werden beiden Modelle abwechselnd trainiert, damit die einzelnen Modelle sich besser auf das Ergebnis des jeweiligen Gegenspielers reagieren kann und sich das Ziel nicht konstant verschiebt. Der Generator wird trainiert, um den Diskriminator zu täuschen, indem er Daten erzeugt, die von echten Daten nicht zu unterscheiden sind. Der Diskriminator wird trainiert, um den Generator zu täuschen, indem er die vom Generator erzeugten Daten nicht von echten Daten unterscheiden kann. Die beiden Modelle trainieren sich schließlich gegenseitig, bis der Diskriminator circa die Hälfte der Daten nicht korrekt zuordnen kann. Die Gründe hierfür liegen in der Konvergenz des Systems und werden in Kapitel 5.1 näher besprochen. Zu diesem Zeitpunkt gilt das Netz als fertig trainiert[4].

Trotz seiner simplen Architektur ist das Training eines GANs ein komplexer Prozess, der viel Zeit und Rechenleistung benötigt. Zusätzlich benötigt der Prozess ein großes Maß an Daten. Die rechenintensive Natur wird allerdings im Laufe der Zeit immer weniger zum Problem werden, aufgrund Moore's Law. Dieses besagt, „dass sich die Anzahl an Komponenten in einem einzigen integrierten Schaltkreis bei minimalen Kosten jedes Jahr verdoppelt.“[5] Dieser Trend wird sich zwar in den nächsten Jahren verlangsamen, dennoch ist davon auszugehen, dass sich die Leistung von Rechnern weiterhin steigern wird. Für das Problem der großen benötigten Datensätze bietet das Internet eine Lösung. Dieses ist nämlich eine nahezu unerschöpfliche Quelle an Daten. Diese Daten können dabei jedoch nicht einfach so übernommen werden, denn das Trainieren einer KI erfordert Ausgangsdaten, die vorsichtig ausgewählt und überprüft werden müssen. Dies erzeugt einen erheblichen Mehraufwand und ist ein großer einschränkender Faktor für die Entwicklungen von künstlichen Intelligenzen und somit auch GANs.

4 Anwendungen von GANs

Wie in den vorherigen Kapiteln gezeigt, sind GANs und seine Vielzahl an Varianten, in der Lage, eine Vielzahl von unterschiedlichen Aufgaben zu erfüllen, wobei die Arbeit mit visuellen und auditiven Inhalten aufgrund ihrer abstrakten Natur besonders hervorsticht. Im Folgenden werden die Möglichkeiten der Technologie in Form von drei konkreten visuellen Einsatzzwecken näher untersucht.

4.1 Bildsynthese

Einer der prominentesten und eindrucksvollsten Einsatzzwecke ist die Bildsynthese. Hierbei wird ein GAN trainiert, um Bilder zu generieren, die von einem menschlichen Auge nicht von echten Bildern unterschieden werden können. Häufig wird eine Conditional GAN verwendet, damit der Anwender die Eigenschaften des generierten Bildes genau bestimmen kann. Einige Produkte und Technologien, welche von GANs für die Bildsynthese Gebrauch machen, sind bereits heute über das Internet verfügbar, zum Beispiel Artbreeder.

Artbreeder, ehemals GANbreeder, ist eine Webseite, welche es dem Anwender ermöglicht, Bilder zu generieren, indem er die Eigenschaften von mehreren Bildern kombiniert. Die Webseite verwendet hierfür BigGAN, ein großes Modell, welches auf einem Datensatz von 150 Gigabyte an gelabelten Daten trainiert wurde. Dabei erlaubt Artbreeder nicht nur die Kombination bestimmter Motive, sondern erlaubt auch das freie Modifizieren bestimmter Faktoren. Bild 4.1 zeigt beispielsweise das stilisierte Bild einer Frau, welches durch Zugabe des Hundeparameters modifiziert wurde. So lassen sich abstrakte Konzepte realisieren, wie das Darstellen des Gegenteils eines Objekts.



Abbildung 4.1: Demonstration von Artbreeder und Modifikation des Originals(links) durch Erhöhung des „Hundefaktors“

Quelle: <https://www.artbreeder.com>

Woran Artbreeder häufig scheitert, ist die Darstellung realistischer menschlicher Gesichter, welche häufig unnatürlich und seltsam wirken können.



Abbildung 4.2: KI-genertiertes Gesicht

Quelle:

<https://thispersondoesnotexist.com/>

Hier kommt StyleGAN ins Spiel. StyleGAN ist ein GAN, welches von Nvidia entwickelt wurde und sich auf die Generierung von Gesichtern spezialisiert hat. StyleGAN ist in der Lage, hochrealistische Bilder menschlicher Gesichter zu generieren, welche mit dem bloßen Auge kaum von echten menschlichen Gesichtern unterscheiden werden können. Dies gilt besonders für StyleGAN 2, eine überarbeitete Version, die häufig auftretenden Artefakte reduziert und die generelle Bildqualität steigert. Ein bekannter und für jeden zugänglicher Beispiel für StyleGAN 2 ist „This Person does not exist“(siehe Bild 4.2), eine Webseite die bei Aufruf mithilfe dieses Netzes eine fotorealistisches Bild eines menschlichen Gesichts generiert.

4.2 Super-Resolution

Ein GAN, welches trainiert wurde, um das Prinzip der Super-Resolution anzuwenden, generiert statt neuen Bildinhalten, eine verbesserte Version des Originals mit erweiterter visuellen Qualität. Diese Verbesserung kann in unterschiedlichen Formen auftreten, beispielsweise durch eine höhere Auflösung, eine verbesserte Farbgebung, entfernen von Rauschen und Schmutz, sowie das Schärfen der Bilder. Im unteren Beispiel 4.3 ist eine Demonstration einer Super-Resolution-GAN zu sehen, welche ein herunterskaliertes Bild auf die ursprüngliche Auflösung hochskaliert. Dabei ist zu beobachten, dass die KI die hochfrequenten Details, wie die zerfransten Kanten nicht replizieren kann und das Resultat sehr geglättet wird. Dies liegt daran, dass diese Information durch das Herunterskalieren irreparabel verloren gegangen ist und die KI lediglich raten kann, wie das Bild ursprünglich aussah.

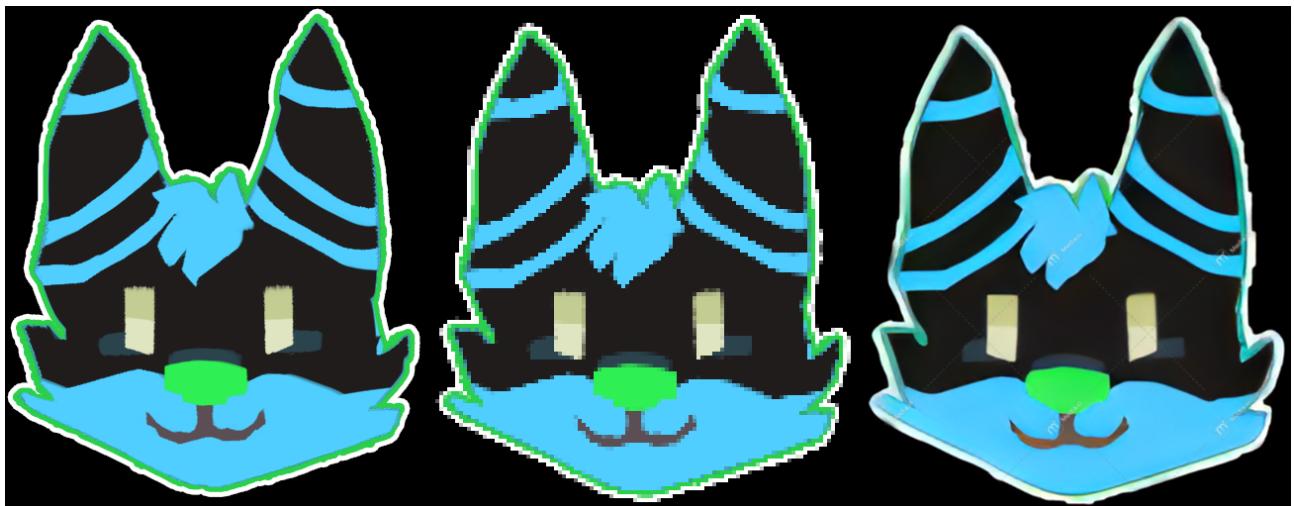


Abbildung 4.3: Demonstration Super-Resolution-GAN: Originalbild(links), verkleinertes Original(mitte) und per KI restaurierte Version(rechts, mit Wasserzeichen)
Quelle: <https://imgupscaler.media.io>

Auch bei Videos kann das Prinzip angewendet werden, wobei einer Verbesserung intra-frame stattfinden kann, heißt die einzelnen Standbilder können visuell optimiert werden, doch Bewegtbilder bieten ebenfalls die Möglichkeit Optimierungen inter-Frame durchzuführen. Genauer gesagt bedeutet dies, dass die oben genannten Verbesserungen, wie höhere Auflösung und Optimierung des Bildinhalts nicht nur auf den zweidimensionalen Raum, sondern ebenfalls auf den dreidimensionalen Raum angewendet werden kann, wobei die dritte Achse die Zeit darstellt. So kann beispielsweise die Bildwiederholungsrate erhöht werden, indem zwischen Bildern interpoliert wird, was die Bewegungen weicher und flüssiger aussehen lässt.

4.3 Style Transfer

Die letzte Technik, auf die wir einen Blick werfen werden, ist der Style Transfer. Diese Technik erlaubt es, den Stil eines Bildes auf ein anderes Bild zu übertragen. Dieser Stil kann dabei sehr unterschiedlich sein, von realistisch bis abstrakt, von Zeichentrick bis Fotografie. Das Prinzip des Style Transfers basiert auf der Trennung von Inhalt und Stil. Der Inhalt eines Bildes ist dabei der eigentliche Bildinhalt, also die Objekte, die auf dem Bild zu sehen sind. Der Stil hingegen ist die Art und Weise, wie das Bild dargestellt wird, also die Farbgebung, die Pinselstriche, die Textur, etc. Das Modell ist schließlich in der Lage den Bildinhalt in eine andere Form zu transformieren. Im Beispiel wurde das Tool ToonMe verwendet, um überzeugende Variationen der ursprünglichen Fotografie zu erzeugen.



Abbildung 4.4: Style Transfer GAN in der Praxis, Motiv: Todd Howard, Bethesda Softworks
Quelle: <https://toonme.com> & <https://commons.wikimedia.org/>

5 Herausforderungen

5.1 Failure to Converge

Auch wenn es sich bei GAN um eine Technologie handelt, die sehr beeindruckende Resultate erzielen kann, so bietet sich aufgrund der komplexen Natur der Technologie auch viel Raum für Fehler und Fragilität. Einer dieser Fehler ist das sogenannte *Failure to Converge*, was auf Deutsch so viel wie „Fehlschlagen der Konvergenz“ bedeutet. Dieser Fehler tritt auf, wenn die beiden Modelle nicht mehr in der Lage sind, einander zu trainieren und somit keine Konvergenz stattfindet. Dies liegt vor allem an den Wechselwirkungen zwischen Generator und Diskriminatator.

Die treibende Kraft hinter dem System stammen ist die Rivalitätsbeziehung zwischen den beiden Modellen. Sollte allerdings eins der beiden Modelle im Vergleich zu gut werden, wird dadurch das jeweils andere Modell geschädigt. Problematisch wird das vor allem, wenn der Generator zu gut wird. Wird angenommen, dass der Generator nur noch perfekte Resultate produziert, hat der Diskriminatator keine Möglichkeit mehr ein künstliches und ein reales Bild zu unterscheiden. In diesem Fall hat der Diskriminatator keine Basis mehr für die Berechnung der Wahrscheinlichkeit und muss praktisch gesehen zufällig entscheiden, schafft also statistisch nur noch eine Erfolgsquote von 50 %. Da die Entwicklung des Generators allerdings auf inhaltsvollem Feedback des Diskriminators beruht, kann dieses sich nicht mehr weiterentwickeln. Im schlimmsten Fall wirkt das Feedback des Diskriminators schädlich auf den Generator und verringert seine Erfolgsquote wieder. Für ein GAN ist Konvergenz also oft nur ein transitiver Zustand, statt ein stabiler.

Mit der Zeit wurden einige Methoden eingeführt, welche die Wahrscheinlichkeit des *Failure to Converge* verringern. Diese greifen häufig in die Funktionsweise des Diskriminators ein und fügen beispielsweise Rauschen in die Eingangsdaten des Diskriminators hinzu. Dies soll dabei

helfen dem Diskriminator trotz hoher Qualität der generierten Bilder das Unterscheiden der Daten zu erleichtern, wodurch durch vorsichtige Konfiguration der „Münzwurf“ verhindert werden kann.[4]

5.2 Mode Collapse

Ein weiteres Problem, welches bei der Verwendung von GANs auftreten kann, ist das sogenannte *Mode Collapse*, was auf Deutsch so viel wie „Modus-Kollaps“ bedeutet. Der Modus ist ein statistischer Lagewert und bezeichnet den Wert aus einer Sammlung von Werten, der am häufigsten vorkommt. Der Begriff *Mode Collapse* bezeichnet dabei den Kollaps des Generators zu seinem eigenen Modus, sprich der Generator generiert nur noch eine bestimmte Art von Bildern, die sich in der Regel nur minimal voneinander unterscheiden. Da der Generator versucht das Resultat zu produzieren, welches den Diskriminator am besten zufriedenstellt, ist ein einseitiger Ausgang nichts Ungewöhnliches. Allerdings ist die Aufgabe des Diskriminators ein Ergebnis, das zu häufig vorkommt abzulehnen und zu verhindern. Dadurch muss der Generator sein Repertoire an Ausgaben erweitern, um den Diskriminator weiterhin überlisten zu können. Sollte die nächste Diskriminator-Generation allerdings in einem lokalen Minimum feststecken, kann es passieren, dass diese nicht lernt den einseitigen Ausgang abzulehnen. Dadurch wird der Generator nicht mehr dazu gezwungen sein Repertoire zu erweitern, sondern fixiert sich auf einen speziellen Diskriminator und lernt diesen zu überlisten. Der Diskriminator schafft es dann auch nicht mehr das Problem zu überwinden und stagniert. In diesem Fall ist der Generator nicht mehr in der Lage, neue Bilder zu generieren, sondern produziert nur noch ähnliche Bilder: Das Modell ist kollabiert.

Eine Möglichkeit dieses Problem zu umgehen, ist die Verwendung von alternativen Verlustfunktionen, wie die Wasserstein-Verlustfunktion, welche dem Diskriminator hilft wiederkehrende Ergebnisse abzulehnen. Eine weitere Möglichkeit ist die Verwendung von *Unrolled GANs*, welche eine Verlustfunktion für den Generator verwenden, die nicht nur die Klassifikation des momentanen Diskriminators einbezieht, sondern auch die möglichen zukünftigen Diskriminator-Varianten. Dadurch, dass der Generator sich an mehrere Diskriminatoren anpassen muss, kann er sich also nicht auf einen speziellen Diskriminator verstießen und das Ergebnis bleibt variiert. [4]

5.3 Ethische Aspekte

Die Verwendung von GANs bietet viele Möglichkeiten, die in der Praxis sehr nützlich sein können. Allerdings ist die Technologie auch sehr komplex und bietet viel Raum für Fehler und Missbrauch. Besonders die Verwendung von GANs in Bezug auf die Generation visueller Daten war in jüngster Zeit besonders kontrovers.

Ein großes Problem war dabei die Verwendung von Deepfakes. Deepfakes sind künstlich generierte Bilder oder Videos, die durch die Verwendung von künstlicher Intelligenz erstellt wurden. Durch diese kann beispielsweise das Gesicht einer Person in ein Video einzufügen, in dem diese nicht vorkommt. Aus technischer Sicht ist dies zwar sehr eindrucksvoll, kann jedoch verwendet werden um Menschen, die den Unterschied nicht erkennen können, zu täuschen und zu betrügen. Und da die Technologie mit der Zeit sich immer mehr verbessert, wird es mit der Zeit nur noch schwieriger werden, künstlich erzeugte Medien mit krimineller Absicht und echte Videos zu unterscheiden. Eine Möglichkeit der Lösung dieses Problems ist, dass sich alle Anbieter von künstlichen Intelligenzen dazu verpflichten das Einfügen gewisser „Wasserzeichen“ sicherzustellen, sodass jederzeit zwischen künstlich erzeugten Daten und echten Daten unterschieden werden kann.

Ein weiteres Problem, das insbesondere für GAN eine Rolle spielt, ist der Datenschutz. Das Training von GANs benötigt wie bereits vorher erwähnt neben viel Rechenleistung eine riesige Sammlung an Trainingsdaten. Die Quelle für diese Trainingsdaten ist dabei heutzutage größtenteils das Internet aufgrund der riesigen Masse an qualitativen Daten. Allerdings haben die Individuen, welche selbsterstellte Bilder oder Kunst ins Internet hochladen, oft keine Ahnung, dass ihre Daten ohne Zustimmung für das Training einer KI verwendet werden, oder sie haben keine Möglichkeit explizit die Verwendung zu verbieten. Und selbst wenn die Möglichkeit besteht, ist das illegale Beziehen der Daten immer noch sehr einfach und der Datendiebstahl schwer nachvollziehbar. Aufgrund dessen ist die Auseinandersetzung mit dem Thema Datenschutz sehr wichtig, besonders in Zeiten von Social Media.

6 Schlussfolgerungen und Ausblick

Schlussendlich lässt sich sagen, dass GANs eine sehr beeindruckende Technologie sind, die in der Zukunft noch viel Potenzial haben. Die Technologie ist zwar sehr komplex und bietet viel Raum für Fehler, allerdings ist sie auch sehr vielseitig und kann in vielen Bereichen eingesetzt werden. Bereits heute werden GANs in spannenden Projekten, wie „thispersondoesnotexist.com“ eingesetzt, welche die Möglichkeiten bereits heute greifbar machen, auch wenn es sich um eine sehr junge Technologie handelt und die Qualität, sowie die Effizienz in Umgang mit Ressourcen hoffentlich in der Zukunft weiter verbessert wird. Trotz dessen wird auch GAN keine Universallösung bleiben und ist besser als Werkzeug in einem prall gefüllten Werkzeugkasten zu verstehen, der für jeden Einsatzzweck ein entsprechendes Werkzeug bereitstellt.

Ein Beispiel für ein weiteres Werkzeug, das einen ähnlichen Aufgabenbereich wie GANs abdeckt, sind Diffusionsmodelle. Diese bieten deutlich feineren Zugriff auf die Generationsparameter und arbeiten stabiler als GANs, zum Beispiel betrifft sie Probleme, wie Mode Collapse nicht. Allerdings sind diese noch rechenintensiver als GANs und benötigen viel Fine-tuning, um ansprechende Resultate zu erhalten, was bei GANs die KI selbst übernimmt. Eine perfekte Methode gibt es schließlich nicht und so werden GANs wohl auch zukünftig ein wichtiges Werkzeug im Werkzeugkasten der künstlichen Intelligenz bleiben.

Literatur

- [1] Andreas Scherer. *Neuronale Netze: Grundlagen und Anwendungen*. Vieweg, Wiesbaden, 1997. ISBN: 978-3-528-05465-6.
- [2] John D. Kelleher. *Deep Learning*. The MIT Press, Cambridge, Massachusetts, 2019. ISBN: 978-0-262-53755-1.
- [3] Kenny Choo. *Machine Learning kompakt*. Springer Spektrum, Wiesbaden, 2021. ISBN: 978-3-658-32267-0.
- [4] Google for Developers. *GAN training*. Juli 2022. URL: <https://developers.google.com/machine-learning/gan/training>.
- [5] Mitja Schmakeit. *Moore's Law*. Juni 2023. URL: <cl-informatik.uibk.ac.at/teaching/ss14/ewa/reports/ss13-MS.pdf>.