



BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Beuth Hochschule für Technik

Seminararbeit

Einführung in die wissenschaftliche Projektarbeit

Thema: Generative Modelle im maschinellen Lernen

Autor: Frank Trojanowski <info@franktrojanowski.com>
Matrikelnummer: 881001

Version vom: 1. Juli 2020

Betreuer: Dr. Hussein Hussein

Kurzfassung

In vorliegender Seminararbeit soll ein grober Überblick über die Funktionsweise und Anwendung von generativen Modellen im maschinellen Lernen gegeben werden. Im ersten Teil der Arbeit werden Grundbegriffe erklärt und im Zusammenhang mit Deep Learning eingeordnet, um im zweiten Teil zwei Beispiele der gängigsten generativen Modellarchitekturen und deren Arbeitsweise zu skizzieren. Danach sollen die Anwendungsmöglichkeiten der beschriebenen Systeme in verschiedenen Medienformaten umschrieben werden und schlussendlich ein Ausblick auf die zukünftige Entwicklung gegeben werden.

Abstract

This seminar paper is intended to provide a rough overview of the functionality and application of generative models in machine learning. In the first part of the work, basic terms are explained and classified in connection with deep learning. In the second part, two examples of the most common generative model architectures are described to outline how they work. Afterwards, the possible uses of the generative systems previously described are examined in various media formats and finally an outlook on future developments in the subject is given.

Inhaltsverzeichnis

Abbildungsverzeichnis	ii
Abkürzungsverzeichnis	iii
1 Einführung in Machine Learning	1
1.1 Deep Learning	1
1.2 Überwachtes und unüberwachtes Lernen	1
1.3 Representation Learning	2
2 Generative Modelle	3
2.1 Vergleich diskriminativer und generativer Modelle	3
2.2 Variational Autoencoders (VAE)	4
2.2.1 Autoencoder	4
2.2.2 VAE	6
2.3 Generative Adversarial Networks (GANs)	8
2.3.1 StyleGAN	9
2.3.2 CycleGAN	12
3 Anwendungsmöglichkeiten generativer Modelle	13
3.1 Bild	13
3.2 Audio	14
3.3 Text	14
4 Zusammenfassung	15
5 Ausblick	16
Literaturverzeichnis	17

Abbildungsverzeichnis

2.1	schematische Darstellung eines Autoencoders	5
2.2	Funktionsweise eines Autoencoders	5
2.3	schematische Darstellung eines VAE	6
2.4	Unterschied im Encoding - Autoencoder vs. VAE	7
2.5	Verteilung der Cluster mit und ohne KL-Divergenz	7
2.6	Verteilung im Encoding eines VAE	8
2.7	Funktionsweise eines GAN	9
2.8	Beispielbilder StyleGAN	10
2.9	PG-GAN Schema	10
2.10	Schema eines StyleGAN	11
2.11	Beispiele von Stiltransfer mit CycleGAN	12

Abkürzungsverzeichnis

AdaIN	Adaptive Instance Normalization
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DCGAN	Deep Convolutional Generative Adversarial Networks
GAN	Generative Adversarial Network
IEEE	Institute of Electrical and Electronics Engineers
KI	Künstliche Intelligenz
KL-Divergenz ..	Kullback-Leibler-Divergenz
KNN	Künstliches Neurales Netz
LSTM	Long Short-Term Memory
ML	Machine Learning
NIPS	Neural Information Processing Systems (Konferenz)
NN	Neural Network
PG-GAN	Progressive Growing Generative Adversarial Network
RNN	Recurrent Neural Network
SRGAN	Super Resolution Generative Adversarial Network
VAE	Variational Autoencoder

1 Einführung in Machine Learning

Machine Learning (ML) ist ein interdisziplinäres Teilgebiet der künstlichen Intelligenz, welches die Generierung von »Wissen« aus »Erfahrung« bezieht. Mithilfe von ML werden IT-Systeme in die Lage versetzt, auf Basis vorhandener Datenbestände und Algorithmen Muster und Gesetzmäßigkeiten zu erkennen und ein komplexes Modell der Daten zu entwickeln. Das Modell, und damit die automatisch erworbene Wissensrepräsentation, kann dann verallgemeinert für neue Problemlösungen oder für die Analyse von bisher unbekannten Daten verwendet werden. Dazu verwendet ML zahlreiche Methoden, wobei in vorliegender Seminararbeit nur jene erläutert werden, welche für das allgemeine Verständnis generativer Modelle nötig sind.

1.1 Deep Learning

Deep Learning umfasst eine Reihe von Algorithmen, deren Architektur auf mehreren nacheinander angeordneten Verarbeitungsschichten basiert und deren Aufgabe es ist, aus unstrukturierten Daten übergeordnete Repräsentationen zu lernen. Deep-Learning-Modelle können auf strukturierte Daten angewendet werden, aber ihre eigentliche Stärke, insbesondere im Hinblick auf die generative Modellierung, ergibt sich aus ihrer Fähigkeit, unstrukturierte Daten verarbeiten zu können. In den meisten Fällen liegt es in der Natur der Aufgabe, unstrukturierte Daten wie neue Bilder oder echte Textzeilen zu erzeugen. Daher verwundert es wenig, dass vor allem die Entwicklungen im Deep Learning einen bedeutenden Einfluss auf die generative Modellierung hatten[Fos19]. Bei der Mehrheit aller Deep-Learning-Modelle handelt es sich um künstliche neuronale Netze (KNNs, kurz »neuronale Netze«) mit mehreren hintereinanderliegenden verborgenen Schichten. Aus diesem Grund ist Deep Learning inzwischen fast zum Synonym für tiefe neuronale Netzwerke geworden. Es ist jedoch wichtig darauf hinzuweisen, dass jedes System, das mehrere Schichten verwendet, um übergeordnete Darstellungen der Eingangsdaten zu lernen, auch eine Form des Deep Learning ist (z. B. Deep-Belief-Netzwerke und Deep-Boltzmann-Maschinen). Um Annahmen nicht bereits im Voraus treffen zu müssen, wird ein Modell benötigt, das die relevanten Strukturen aus den Daten selbst ableiten kann. Hierbei zeichnet sich Deep Learning aus. Die Tatsache, dass Deep Learning seine eigenen Merkmale in einem niederdimensionalen Raum finden kann, bedeutet eine Form des Representation Learnings[GBC16].

1.2 Überwachtes und unüberwachtes Lernen

Wenn ein Modell anhand gekennzeichneter Trainingsdaten (*labeled data*) lernen soll, handelt es sich um überwachtes Lernen. Die Methode richtet sich also nach einer im Vorhinein festgelegten Ausgabe, deren Ergebnisse bekannt sind. Die Ergebnisse des Lernprozesses können mit den bekannten, richtigen Ergebnissen verglichen, also über-

wacht, werden. Die Überwachung bezieht sich dabei nur auf die sogenannten Trainingsdaten, mit denen ein KNN für die Lösung einer Aufgabe trainiert wird. Im produktiven Einsatz des Modells wird grundsätzlich nicht überwacht. Zu den überwachten Lernverfahren zählen alle Verfahren zur Klassifikation oder Regression, beispielsweise mit Algorithmen wie k-nearest-Neighbour, Random Forest, Support Vector Machines oder auch Verfahren der Dimensionsreduktion wie die lineare Diskriminanzanalyse. Der Nachteil von überwachtem Lernen besteht jedoch in einem oft sehr hohen manuellen Aufwand bei der Aufbereitung der Trainingsdaten. Eine besondere Form des überwachten Lernens ist die des bestärkenden Lernens. Bestärkendes Lernen kommt dann zum Einsatz, wenn ein Endergebnis noch nicht bestimmbar ist, jedoch der Trend hin zum Erfolg oder Misserfolg erkennbar wird. In der Trainingsphase werden beim bestärkenden Lernen die korrekten Ergebnisse also nicht zur Verfügung gestellt, jedoch wird jedes Ergebnis bewertet, ob dieses (wahrscheinlich) in die richtige oder falsche Richtung geht.[Ras18]

Im Gegensatz zum überwachten Lernen werden beim unüberwachten Lernen weder gekennzeichnete, noch klassifizierte Trainingsdaten verwendet. Ziel dieses Ansatzes ist es, aus den Daten unbekannte Muster zu erkennen und Regeln aus diesen abzuleiten. Das ML-System nutzt Algorithmen, die die Struktur der Eingabedaten erkunden und daraus sinnvolle Informationen in den Ausgabedaten abbilden. Die Vorteile des unüberwachten Lernens bestehen in der teilweise vollautomatisierten Erstellung von Modellen. Dabei können diese eine sehr gute Prognose über neue Daten hervorbringen oder gar neue Inhalte erstellen (generative Modelle). Das Modell lernt mit jedem neuen Datensatz dazu und verfeinert gleichzeitig seine Berechnungen und Klassifizierungen. Ein manueller Eingriff ist dadurch nicht mehr notwendig. KNNs basieren auf diesem selbstlernenden Verfahren.

1.3 Representation Learning

Die Grundidee beim Lernen von Repräsentationen besteht darin, nicht den hochdimensionalen Stichprobenraum direkt zu modellieren, sondern jede Beobachtung im Trainingsdatensatz mittels eines niederdimensionalen latenten Raums zu erfassen. Das Modell muss dann eine Mapping-Funktion lernen, die einen Punkt im latenten Raum nehmen und ihn auf einen Punkt in dem ursprünglichen Raum abbilden kann. Mit anderen Worten: Jeder Punkt im latenten Raum ist die Darstellung einer hochdimensionalen Repräsentation der Eingabedaten, beispielsweise eines Bilds.

2 Generative Modelle

Ein generatives Modell beschreibt, wie ein Datensatz im Rahmen eines Wahrscheinlichkeitsmodells erzeugt wird. Die aus diesem Modell gewonnenen Stichproben ermöglichen es, neue Daten zu generieren.

Gegeben sei ein Datensatz mit Bildern von bestimmten Objekten. Es soll ein Modell entwickelt werden, das ein neues Bild von einem dieser Objekte erzeugen kann, das nie existiert hat, aber trotzdem realistisch aussieht. Dies ist möglich, weil das Modell die allgemeinen Prinzipien gelernt hat, die das Aussehen dieses Objektes bestimmen. Genau dies ist die Form von Aufgabenstellung, die sich mithilfe eines generativen Modells im Machine Learning lösen lässt [Fos19, vgl. S. 54]. Zuerst benötigt man einen Datensatz, der aus zahlreichen Beispielen der zu erzeugenden Entität besteht. Man spricht in diesem Zusammenhang vom Trainingsdatensatz, bei dem jeder der Eingabedatenpunkte als Beobachtung bezeichnet wird. Jede Beobachtung besteht aus vielen Merkmalen – bei einer Bilderzeugungsaufgabe sind die Merkmale üblicherweise die jeweiligen Werte der einzelnen Pixel. Das Ziel ist es, ein Modell zu erstellen, das neue Merkmale erzeugen kann, die so aussehen, als wären sie nach den gleichen Regeln wie die Originaldaten erstellt worden. Rein konzeptionell ist dies für die Bilderzeugung eine unglaublich schwierige Aufgabe – bedenkt man, wie vielfältig die einzelnen Pixelwerte zugeordnet werden können und wie vergleichsweise gering dagegen die Anzahl der Anordnungen ist, die ein Bild der Entität erzeugen, die reproduziert werden soll.

Ein generatives Modell sollte zudem probabilistisch und nicht deterministisch sein. Wenn das Modell nur eine fest vorgegebene Berechnung umfasst, wie z. B. die Berechnung des Durchschnittswerts jedes Pixels im Datensatz, ist es nicht generativ, da das Modell jedes Mal die gleiche Ausgabe erzeugt. Folglich muss das Modell ein stochastisches (zufälliges) Element beinhalten, das die von ihm erzeugten Ausgaben beeinflusst. Es gibt also eine unbekannte Wahrscheinlichkeitsverteilung, die erklärt, warum einige Bilder wahrscheinlich im Trainingsdatensatz zu finden sind und andere Bilder hingegen nicht. Es muss ein Modell entwickelt werden, das diese Verteilung so genau wie möglich nachahmt, um daraus neue, einzigartige Beobachtungen zu generieren, die so aussehen, als würden sie dem ursprünglichen Trainingsdatensatz entstammen. In diesem Kapitel sollen zwei Beispiele der meistgenutzten generativen Modelle erläutert werden.

2.1 Vergleich diskriminativer und generativer Modelle

Um zu verstehen, was generative Modelle leisten sollen und warum sie wichtig sind, ist es sinnvoll, sie mit ihrem Gegenstück, den diskriminativen Modellen, zu vergleichen. Die meisten Fragestellungen im ML sind diskriminativer Natur.

Ein wesentlicher Unterschied besteht darin, dass bei diskriminativen Modellen jede Beobachtung in den Trainingsdaten mit einem Label versehen ist. Im Fall einer binären

Klassifikationsaufgabe wären beispielsweise Gemälde von Picasso mit einer 1 und alle anderen mit einer 0 gekennzeichnet. Das Modell lernt dann, beide Gruppen zu unterscheiden, und gibt die Wahrscheinlichkeit aus, dass eine neue Beobachtung das Label 1 trägt – gleichbedeutend damit, dass sie von Picasso gemalt wurde.[Fos19] Deshalb ist diskriminative Modellierung synonym mit überwachtem Lernen beziehungsweise dem Erlernen einer Funktion, die eine Eingabe mithilfe eines mit Labels versehenen Datensatzes auf eine Ausgabe abbildet. Für generative Modelle bedarf es für gewöhnlich nur ungelabelter Datensätze (als Form des unüberwachten Lernens). Sie kann jedoch auch auf einen mit Labels versehenen Datensatz angewendet werden, um zu lernen, wie man Beobachtungen und Merkmale aus den einzelnen Kategorien erzeugt.

2.2 Variational Autoencoders (VAE)

Im Vergleich zu den diskriminativen Aufgaben von KNN als Regressoren und Klassifikatoren haben VAEs starke generative Modelle, die mittlerweile diverse Anwendungsmöglichkeiten haben, von der Erzeugung menschlicher Gesichter bis zur Produktion reinsynthetischer Musik. Dabei möchte man aber meist bereits vorhandene Daten nicht auf zufällige Weise verändern, sondern die erzeugten Ausgaben in eine spezifische Richtung steuern. VAEs sind in dieser Hinsicht allen anderen momentan bekannten Methoden überlegen. Um die Funktionsweise von VAEs zu verstehen, ist es hilfreich, sich erst einmal einen Standard-Autoencoder anzuschauen:

2.2.1 Autoencoder

Ein Autoencoder-Netzwerk ist eigentlich ein in Encoder und Decoder geteiltes KNN. Der Encoder nimmt Eingabedaten und konvertiert diese in eine kleinere, dichtere Repräsentation. Das Encoding beschreibt die komprimierten Daten im latenten Raum. Der Decoder kann die ursprünglichen Daten fast originalgetreu aus dieser Repräsentation wiederherstellen [GBC16, S.499ff].

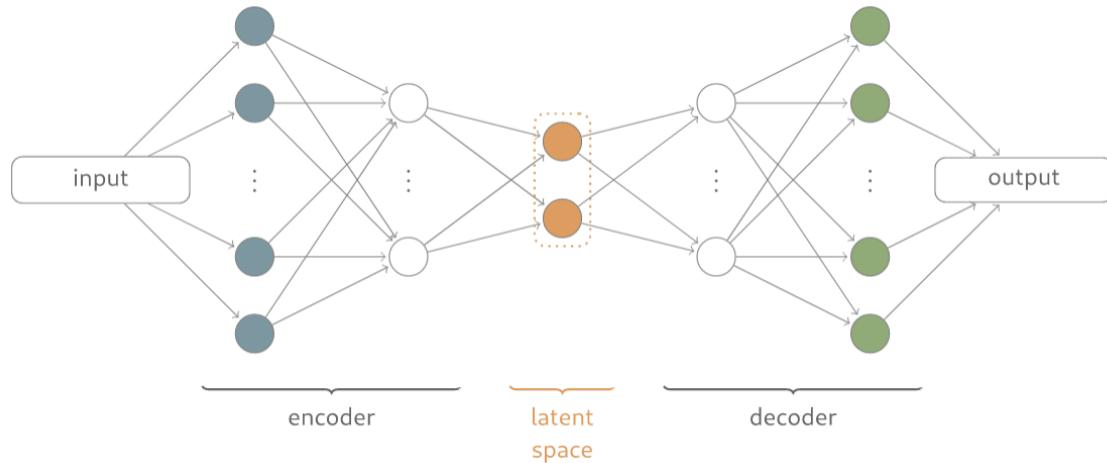


Abbildung 2.1: schematische Darstellung eines Autoencoders [SKGD18]

Dies ist als eine Art Datenkompression zu sehen, da der Encoder spezifische Encodings generiert, die für die spätere Generierung der - dem orginalen Input sehr ähnlichen - Ausgabedaten nötig ist. Die Aufgabe des Encoders ist also, die Eingabedaten zu verarbeiten und sie einem Punkt im latenten Raum zuzuordnen. Als Verlustfunktion wird normalerweise entweder »mean-squared error« oder »cross-entropy« benutzt, um das Autoencoder-Netzwerk für Ausgaben zu bestrafen, die nicht den Eingaben entsprechen. Da das Encoding bzw. der latente Raum sehr viel weniger Speicherplatz bedarf als die Originaleingabe, muss das Netzwerk Informationen verwerfen. Der Encoder lernt so viel wie nötig und so wenig wie möglich an relevanten Informationen im Encoding zu behalten, damit das Decodernetzwerk ein möglichst exaktes Abbild der Eingabedaten rekonstruieren kann [Fos19, S.64].

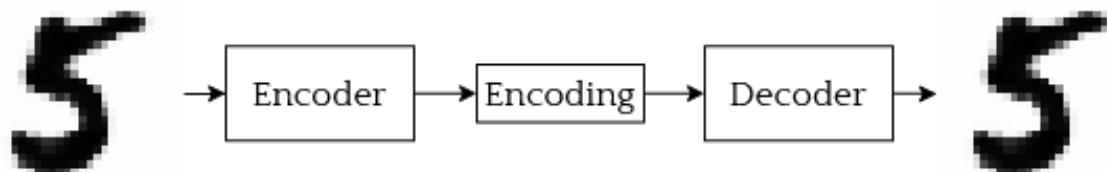


Abbildung 2.2: Funktionsweise eines Autoencoders am Beispielbild der Zahl »5« aus dem MNIST-Set

Standard-Autoencoder können also kompakte Datenrepräsentationen lernen und ihre Eingabedaten sehr gut rekonstruieren. Sie können auch sehr gut zur Bildreinigung verrauschter Bilder verwendet werden, da der Encoder lernt, dass die Position des Rauschens innerhalb des latenten Raums nicht relevant ist. Das Problem mit Standard-Autoencodern ist im Hinblick auf generative Modelle, dass sie keine Interpolation im latenten Raum zulassen [SKGD18]. Man möchte ja mit generativen Modellen nicht die Originaldaten wiederherstellen, sondern dem latenten Raum zufällige Stichproben entnehmen bzw. Variationen der Eingabedaten aus dem latenten Raum generieren. Wenn

diese Stichproben in Raumdiskontinuitäten liegen, werden inkorrekte Ausgabedaten erzeugt, weil der Decoder schlichtweg keine Erfahrung mit Datenpunkten aus dieser Region des latenten Raumes hat.

2.2.2 VAE

Variational Autoencoder haben hingegen einen *kontinuierlichen* latenten Raum, welcher eine Interpolation und zufällige Stichproben erlaubt.

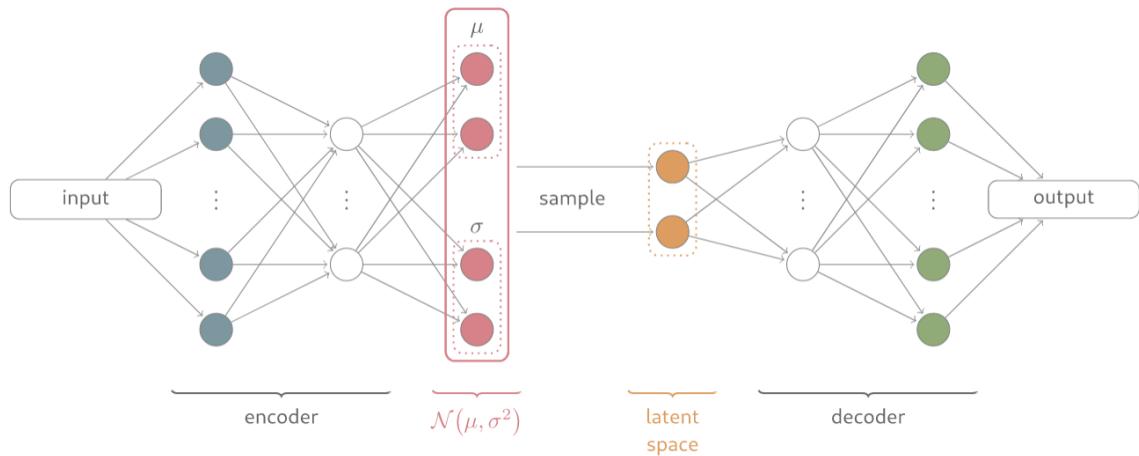


Abbildung 2.3: schematische Darstellung eines VAE [SKGD18]

Dies wird erreicht, indem der Encoder statt *einem* Vektor der Größe n , *zwei* Vektoren der Größe $n - \mu$ und σ - codiert. Dabei ist μ der Mittelwert der Verteilung und σ der Logarithmus der Varianz jeder Dimension [Sha18]. In einem Standard-Autoencoder wird jedes Bild direkt auf einen Punkt im latenten Raum abgebildet. In einem VAE hingegen wird jedes Bild auf eine multivariate Normalverteilung um einen Punkt im latenten Raum abgebildet. Das bedeutet, dass der Encoder jede Eingabe zu einem Mittelwert- und einem Varianzvektor abbilden muss und sich nicht um die Kovarianz zwischen den Dimensionen zu sorgen braucht. Darüber hinaus ist σ der Logarithmus der Varianz, da dieser eine beliebige reelle Zahl im Bereich $(-\infty, \infty)$ annehmen kann, was dem natürlichen Ausgabebereich eines neuronalen Netzes entspricht, wohingegen die Varianz nur positive Werte annehmen kann [Fos19, S.81].

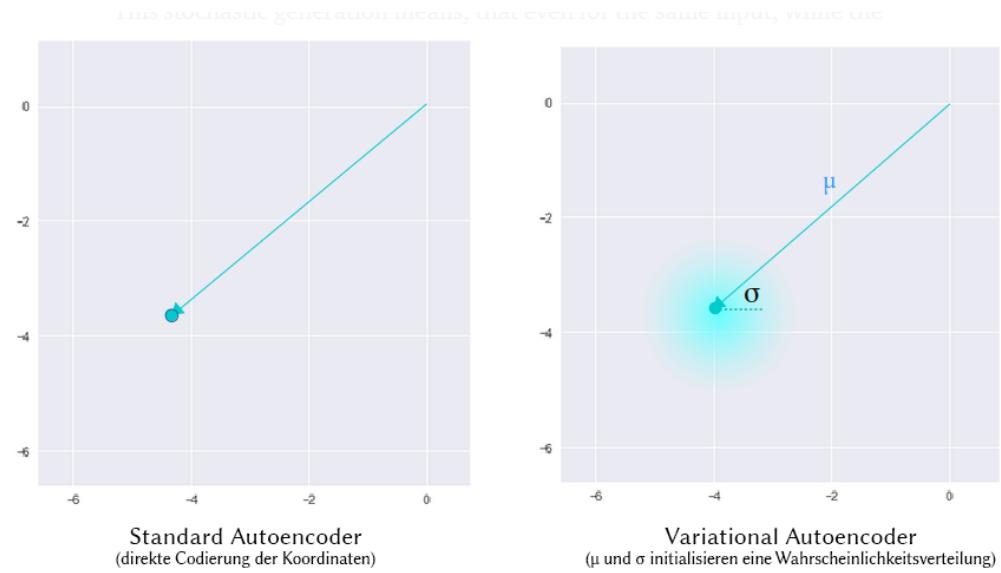


Abbildung 2.4: Unterschied im Encoding - Autoencoder vs. VAE [Sha18]

Das Modell hat damit einen gewissen Grad an lokaler Variation im Encoding, jedoch keine überlappenden Cluster an variierenden Stichprobenencodings. Im Idealfall möchte man nicht nur im lokalen Stichprobenraum Variationen entnehmen, wo sich die Stichproben sehr ähnlich sind, sondern zwischen verschiedenen Klassen von Stichproben kontinuierlich interpolieren [Doe16]. Alle erkannten Merkmalcluster sollen also so nah wie möglich aneinander im latenten Raum angeordnet sein. Dies wird mit der Einrechnung der Kullback-Leibler-Divergenz in die Verlustfunktion erreicht [Sha18]. Die KL-Divergenz ist eine Methode, um zu messen, wie sehr sich eine Wahrscheinlichkeitsverteilung von einer anderen unterscheidet. Im Fall eines VAE soll gemessen werden, wie sehr sich die Normalverteilung mit den Parametern μ und σ von der Standardnormalverteilung (Glockenkurve) unterscheidet. Das Netzwerk wird damit für die Codierung von Beobachtungen zu μ und σ bestraft, die deutlich unterschiedlich von den Parametern einer Standardnormalverteilung sind [Fos19, S.102].

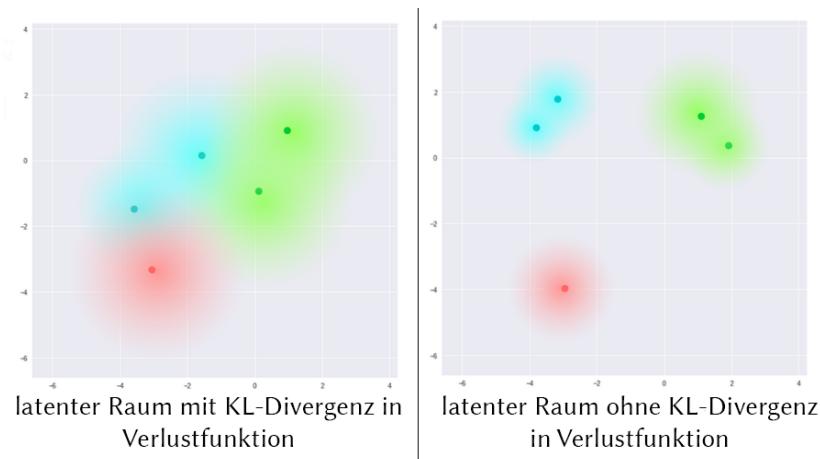


Abbildung 2.5: Verteilung der Cluster mit und ohne KL-Divergenz [Sha18]

Mit einer optimierten Verlustfunktion und der KL-Divergenz können Verteilungen wie in Abbildung 2.6 erreicht werden.

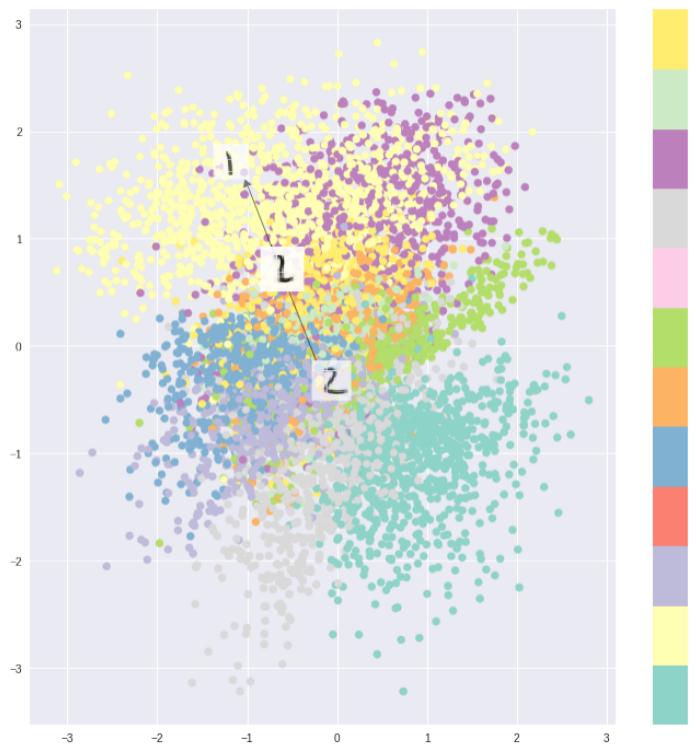


Abbildung 2.6: Verteilung im Encoding eines VAE am Beispiel des MNIST-Sets [Sha18]

2.3 Generative Adversarial Networks (GANs)

Die GAN-Architektur für ein auf Deep Learning basierendes, generatives Modell wurde erstmals 2014 von Ian Goodfellow et al. auf der NIPS-Konferenz vorgestellt. Sie besteht aus zwei Submodellen:

- Generator (G) - Modell, das neue Beobachtungen als Eingabedaten generiert
- Diskriminatör (D) - Modell, das klassifiziert, ob die zugeführten Daten vom Trainingsset oder vom Generator stammen

Beide Modelle werden gleichzeitig trainiert. Der Generator versucht, Zufallsrauschen in Beobachtungen umzuwandeln, die so aussehen, als würden sie dem Trainingsdatensatz entstammen, und der Diskriminatör versucht zu erkennen, ob eine Beobachtung aus dem ursprünglichen Datensatz stammt oder eine der Fälschungen des Generators ist [Bro19]. Am Anfang gibt der Generator sehr verrauschte Bilder aus und der Diskriminatör prognostiziert zufällig. Der entscheidende Punkt von GANs ist, wie das Training der beiden Netzwerke abgewechselt wird, sodass, sobald der Generator geschickter im Täuschen des Diskriminators wird, dieser sich anpassen muss, um weiterhin die gefälschten Bilder zu erkennen. Das wiederum treibt den Generator dazu an, neue Wege zu finden, um den Diskriminatör zu täuschen [Fos19, S.99].

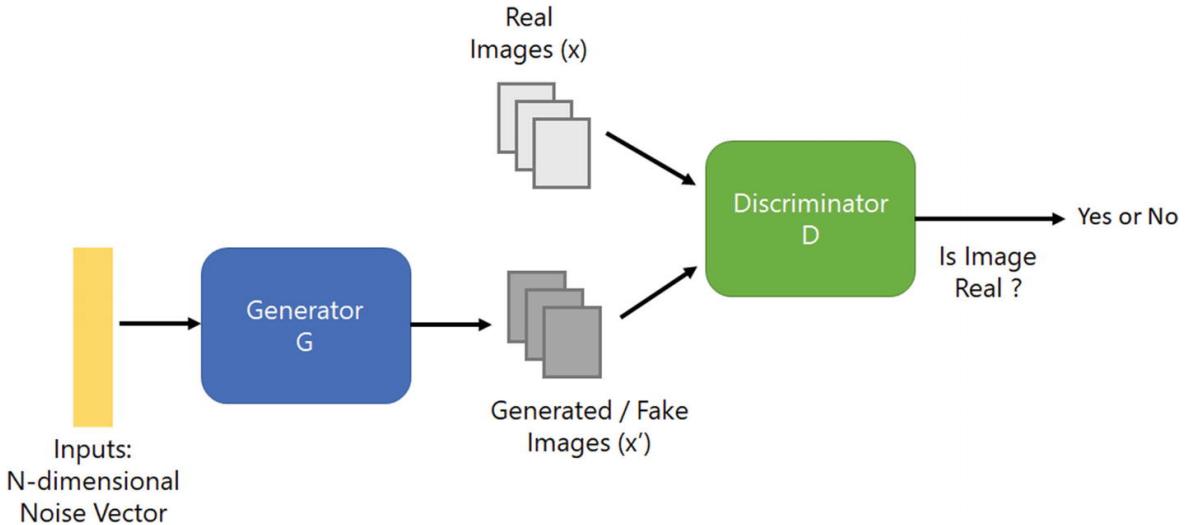


Abbildung 2.7: Funktionsweise eines GAN [SDT18]

Man kann sich das Training eines GAN als Nullsummenspiel vorstellen: Das Generatornetzwerk sei dabei ein Geldfälscher, der versucht, Falschgeld zu produzieren. Das Diskriminatorennetzwerk sei die Polizei, die versucht, echtes Geld von gefälschtem Geld zu unterscheiden. Um das Spiel zu gewinnen, muss der Geldfälscher lernen, Geldscheine zu erzeugen, welche von echtem Geld ununterscheidbar sind[Goo16]. In diesem Fall bedeutet Nullsummenspiel, dass das jeweilige Netzwerk per Fehlerrückführung in jedem Trainingsschritt bestraft wird und dessen Modellparameter variiert werden. So entsteht nach sehr vielen Trainingsschritten ein generatives Modell, welches täuschend echte Daten generieren kann. Die GAN-Architektur wird vor allem bei der Generierung von Bilddaten verwendet.

Im Original-GAN-Forschungspaper von Ian Goodfellow et al. wurden dichte Schichten anstelle von Konvolutionsschichten für das Training verwendet. Seitdem hat sich jedoch gezeigt, dass Konvolutionsschichten dem Diskriminator eine bessere Vorher sagekraft verleihen. Diese Art von GANs wird in der Literatur auch DCGAN (Deep Convolutional Generative Adversarial Network) genannt. Heutzutage enthalten fast alle GAN-Architekturen Konvolutionsschichten, sodass das »DC« bereits enthalten ist, wenn über GANs geredet wird [Bro19]. Seit der ersten Vorstellung 2014 wurden bis heute mehr als 500 verschiedene Erweiterungen von GAN-Architekturen in wissenschaftlichen Papers vorgestellt. Diese haben verschiedene spezielle Eigenschaften, die zu einer verbesserten Lösung der jeweiligen Aufgabenstellung führt. Zwei der berühmtesten Erweiterungen von GANs sollen in dieser Seminararbeit erläutert werden.

2.3.1 StyleGAN

In dem IEEE-Konferenzpaper »A Style-Based Generator Architecture for Generative Adversarial Networks« wurde das Konzept des StyleGAN 2019 von Nvidia-Mitarbeitern

vorgestellt. Die Bilder von menschlichen Gesichtern in Abbildung 2.8 wurden mit diesem generativen Modell erzeugt. StyleGAN besteht im Wesentlichen aus zwei Erweiterungen eines GAN: Die Erzeugung hochauflösender Bilder durch PG-GAN (Progressively Growing GAN) und Einbindung von Stiltransfer durch AdaIN.



Abbildung 2.8: Beispielbilder StyleGAN [KLA19]

PG-GAN »Progressives Wachstum« ist eine Methode zum Generieren von hochauflösenden Bildern. Ein PG-GAN fängt mit einer niedrigauflösten Repräsentation eines Bildes an und fügt im Trainingsprozess allmählich hochauflösendere Generatoren und Diskriminatoren hinzu. Im Schema in Abbildung 2.9 wird mit einer Auflösung von 4×4 begonnen und diese schrittweise auf 1024×1024 erhöht.

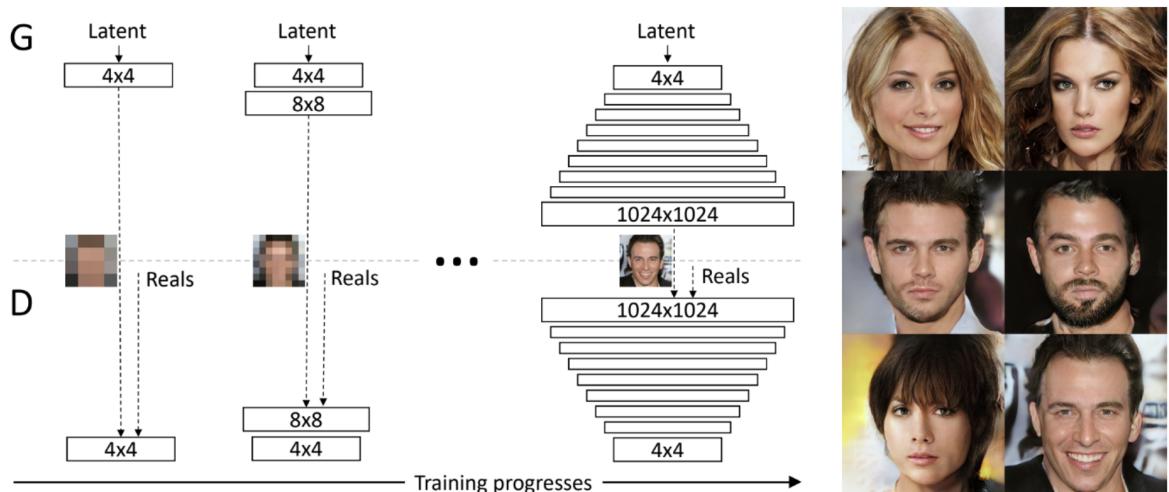


Abbildung 2.9: PG-GAN Schema [KLA19]

AdaIN »Adaptive Image Normalization« (kurz AdaIN) ist ein Stiltransferalgorithmus und wurde 2017 von Xun Huang et al. vorgestellt. Im StyleGAN wird er als Normalisierungsmodul verwendet und in jeder Auflösungsebene des Generators im PG-GAN angewendet. AdaIN entspricht einer Art neuronaler Netzwerkschicht, die den Mittelwert und die Varianz der Ausgaben x_i der einzelnen Feature-Maps aus einer jeweils vorgegebenen Schicht im Synthesenetzwerk mit einem Referenzstil-Bias $y_{b,i}$ und einer Skalierung $y_{s,i}$ anpasst. Die Stilparameter werden berechnet, indem zuerst ein latenter Vektor z durch ein Mapping-Netzwerk f geleitet wird, um einen Zwischenvektor w zu erzeugen. Dieser wird dann durch eine dicht verbundene Schicht umgewandelt, um die Vektoren $y_{b,i}$ und $y_{s,i}$ mit jeweils einer Länge n (der Anzahl der von der Konvolutionsschicht im Synthesenetzwerk ausgegebenen Kanäle) zu erzeugen. Dabei geht es vor allem darum, den Auswahlprozess des Stils für das Bild (das Mapping-Netzwerk) von dem eigentlichen Erzeugungsvorgang eines Bilds mit einem bestimmten Stil (das Synthesenetzwerk) zu trennen. Die adaptiven Instanz-Normalisierungsschichten stellen sicher, dass die Stilvektoren, die in jede Schicht eingespeist werden, nur Merkmale auf dieser Schicht beeinflussen, indem sie verhindern, dass Informationen bezüglich des Stils zwischen den Layern durchsickern [Fos19, S.277f].

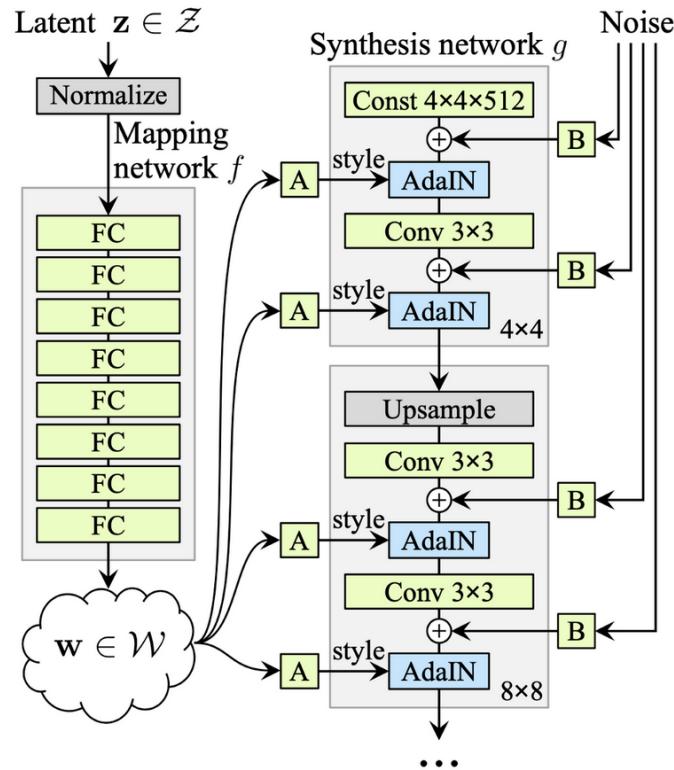


Abbildung 2.10: Schema eines StyleGAN [KLA19]

2.3.2 CycleGAN

Das IEEE-Konferenzpaper von 2017 [ZPIE17] stellte einen bedeutenden Fortschritt im Bereich des Stiltransfers dar. Es zeigt, wie ein Modell trainiert werden kann, das den Stil von einer Reihe von Referenzbildern auf ein anderes Bild überträgt, ohne einen Trainingssatz gepaarter Beispiele zu benötigen. Dies stellt eine Form des unüberwachten Lernens dar. Ein CycleGAN generiert Bilder nicht aus Zufallsrauschen, sondern benutzt bereits vorhandene Bilder um eine neue Version der Bilder herzustellen. Diese sogenannte »Bild-zu-Bild Translation« erfolgt auf Basis des Bildinhalts auf der einen Seite, und des Bildstils auf der anderen. Die Trainingsdaten liegen also ungepaart aus zwei verschiedenen Domänen vor und müssen nicht über 1-zu-1-Zuordnungen verfügen. Das Netzwerk besteht aus zwei kombinierten GANs. Es nimmt ein Bild aus der Domäne X als Input für den ersten Generator $G_{X \rightarrow Y}$, dessen Aufgabe die Transformation eines Bildes der Domäne X in die Zieldomäne Y ist. Dieses neu generierte Bild wird dann Generator $G_{Y \rightarrow X}$ zugeführt, welcher das Bild in ein Bild der Domäne D_X zurückkonvertiert. Beide Generatoren haben korrespondierende Diskriminatoren, welche versuchen, die generierten Bilder von den Bildern ihrer jeweiligen Domäne zu unterscheiden [ZPIE17]. Im genannten Paper wurde das Modell für 200 Epochen trainiert, um die Bildbeispiele in Abbildung 2.11 zu erhalten.

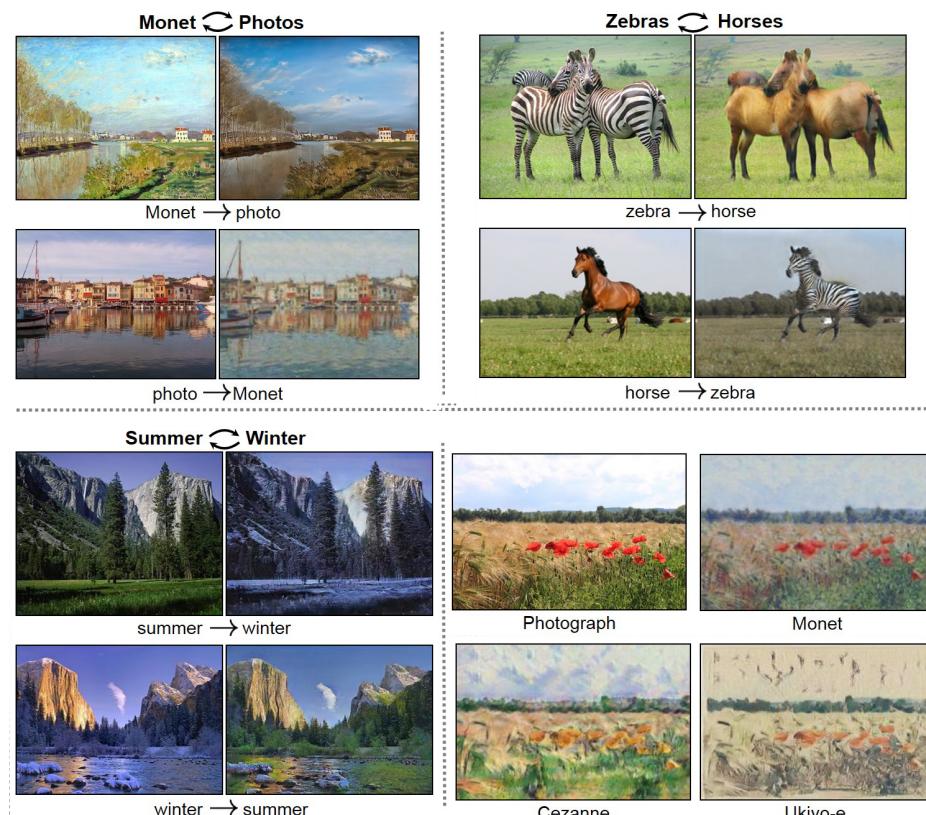


Abbildung 2.11: Beispiele von Stiltransfer mit CycleGAN [ZPIE17]

3 Anwendungsmöglichkeiten generativer Modelle

Im Gegensatz zu diskriminativen Modellen bieten generative Modelle kaum Lösungsansätze für die Automatisierung von Validierungs- und Überwachungsprozessen in der Industrie. Generative Modelle eignen sich hingegen vor allem für die Erstellung von Inhalten wie Bilder, Texte oder Musik [Fos19, S.7]. Deshalb liegt das größte Potenzial der Entwicklung in der Kreativbranche. Schon heute zeigt sich der Trend zur prozeduralen Arbeitsweise in professionellen Softwareprodukten für die Erstellung von Inhalten, wie Substance Designer, Cinema4D und Houdini. Dort werden Texturen, Shader und ganze 3D-Modelle durchgehend parametrisiert erstellt und sind zur Laufzeit variabel. Wenn generative Deep-Learning-Modelle auf diese Parameter angewendet werden, lassen sich per Knopfdruck einfach Variationen und Abstraktionen erstellen, welche sich qualitativ nicht von den von Menschen gestalteten Entwürfen unterscheiden [Pri18]. Der Trend zu »user generated content« begünstigt dazu den Datenwachstum und bereichert somit die Datenbasis zur Erforschung neuer generativer Methoden.

3.1 Bild

Bei der Erläuterung des CycleGAN wurde bereits erwähnt, dass der Stiltransfer ein bedeutender Punkt in der Forschung ist, wenn es um generative Modelle geht [SKLO18]. Zahlreiche mobile Applikationen setzen generatives Deep-Learning ein, um Filter zu entwickeln, die aus den Benutzeraufnahmen - teilweise in Echtzeit - kunstvolle Effekte durch Stiltransfer erzeugen [HB17]. In diesen Apps werden auch Kombinationen aus diskriminativen Modellen und generativen Modellen benutzt, um Gesichter einerseits zu erkennen und andererseits beispielsweise altern zu lassen oder das Geschlecht zu ändern. Sogenannte »Deepfakes« bedienen sich dieser Architekturen, um Gesichter von Personen in Videos auszutauschen. Dieselben Methoden werden in der Filmproduktion eingesetzt, um fehlerhaft gedrehte Szenen nicht erneut drehen zu müssen. Auch wenn Webseiten wie thispersondoesnotexist.com eher als Machbarkeitsstudien für realistische Gesichtsfotos zu sehen sind, ist das Wirkpotenzial realistisch wirkender Bilder von Objekten für die Medienwirtschaft enorm.

Beim Rendern von physikalisch korrekten Lichtsituationen entsteht unerwünschtes Bildrauschen, welches nur vermieden werden kann, wenn die Samplerate des Renderers erhöht wird. Damit werden aber auch die Renderzeiten verlängert, was im Endeffekt immer eine Kostenabwägung ist. Mit Technologien wie Intels *Open Image Denoise* oder *Optix AI-Accelerated Denoiser* von Nvidia, welche auf dem Prinzip eines Autoencoders beruhen, kann man mittlerweile sehr einfach selbst stark verrauschte Bilder in ein akzeptables Endprodukt verwandeln und dadurch Kosten sparen [Pri18]. SRGANs (Super Resolution Generative Adversarial Network) sind eine weitere Klasse von GANs und können niedrigauflöste Bilder in hochauflösende Bilder umwandeln. Dadurch

können stark vergrößerte Bildausschnitte oder qualitativ minderwertiges Bildmaterial rekonstruiert werden und neue Erkenntnisse in Bildgebungsverfahrensforschung geben [LTH⁺16].

3.2 Audio

Die meisten gezeigten Beispiele waren bisher von der Verarbeitung von Bilddaten geprägt. Neben der Erzeugung optisch ansprechender und aussagekräftiger Bilder finden generative Modelle jedoch auch praktische Anwendung im Bereich der Erzeugung von Musik, insbesondere bei Videospielen und Filmen. MuseNet von OpenAI ist in der Lage, unbegrenzt Musik in einem bestimmten Stil zu erzeugen und Genres zu neuen Kompositionen zu mischen. Mit der Magenta API von Google wurden leistungsstarke Werkzeuge geschaffen, die die Arbeitsweise von Künstlern erweitern, indem Kompositionen anhand von Parametern erstellt und verändert werden können [REM⁺19]. Die Aufgabenstellungen generativer Modelle können im Audiobereich in zwei wesentliche Felder aufgeteilt werden. Zum einen werden MIDI-Daten erzeugt, welche dann virtuelle Instrumente kontrollieren, um wohlklingende Kompositionen zu erhalten. Zum anderen besteht die Aufgabe in der Erzeugung digitaler Rohdaten, welche aus Zufallsrauschen synthetisiert werden, ähnlich dem StyleGAN. Darüber hinaus kommen Stiltransfermethoden zum Einsatz, um Sprecher auszutauschen oder die Stimme eines *Text-To-Speech-Systems* den Nutzervorlieben anzupassen.

3.3 Text

Maschinengenerierte Texte sind eine besondere Herausforderung für generative Modelle, da semantische Hürden überwunden werden müssen, damit sie authentisch wirken. Ein wichtiger Anwendungsfall für automatisierte Texterstellung ist journalistischer Natur. Live-Berichterstattung, Wetterbericht, Newsticker und Spielberichte aus dem Massensport bedienen sich bereits seit längerem maschinengenerierter Texte. Auch im Content Marketing kommen generative Modelle zum Einsatz. In einfachen Textformen wie Bildunterschriften und Produkttexten können diese mit einer guten Datenbasis bereits überzeugende Texte liefern.

GPT-2 von OpenAI ist ein generatives Modell, welches insbesondere für Aufgaben im Rahmen der Erzeugung von sinnvollen Sätzen ausgelegt ist. Aufgrund von Bedenken, dass dieses Modell von böswilligen Dritten missbraucht werden könnte, z. B. um gefälschte Nachrichten, gefälschte Aufsätze, gefälschte Konten in den sozialen Medien oder Online-Imitationen von Personen zu erzeugen, hat OpenAI beschlossen, den Datensatz, den Quellcode und die Modellgewichte von GPT-2 nicht zu veröffentlichen [Fos19].

4 Zusammenfassung

In dieser Seminararbeit wurde ein Überblick über die Methoden und Anwendungen von generativen Modellen in Machine Learning gegeben. Es sei darauf hingewiesen, dass dieser Überblick in keiner Weise einen Anspruch der Vollständigkeit hat. Machine Learning ist ein riesiges, interdisziplinäres Fachgebiet und wird ständig erweitert. Generative Modelle und ihre Anwendungen stellen nur einen kleinen Teil dieses Fachgebietes dar und sind trotzdem nicht im Umfang dieser Seminararbeit vollständig zu erfassen. Der Leser sollte aber trotzdem einen Einblick in die Funktionsweise von generativen Deep Learning bekommen haben und eine Vorstellung davon bekommen haben, zu welchen Leistungen generative Modelle in der Lage sind. Es sollte klar geworden sein, wie durch kleine Veränderungen an einem generativen System immer bessere Ergebnisse erzielt werden können. Bei der Beschreibung der beiden Beispielmodelle wurde auf die Verwendung mathematischer Notation und die Erläuterung von tiefgehenden, wahrscheinlichkeitstheoretischen Ansätzen weitgehend verzichtet.

5 Ausblick

Durch die rasante Entwicklung im Forschungsfeld generativer Modelle wurde in den letzten Jahren die Grenze der lösbareren Aufgaben immer weiter verschoben. Es lässt sich bereits heute erkennen, welche Schlüsselrolle die Methoden der generativen Modellierung in der digitalen Medienproduktion einnehmen werden. Dies wird im Gestaltungsprozess dazu führen, dass Medienproduzenten allein aus Gründen der Kosteneffizienz auf generative Modelle zurückgreifen werden. Den Designer bzw. Künstler werden diese Systeme auch auf längere Zeit jedoch nicht ersetzen können [Pri18]. Sie werden eher den kreativen Prozess unterstützen und Ideevariationen erstellen, aus denen der Produzent dann auswählen kann. Mit Text-To-Image-Verfahren ist es bereits heute möglich aus rein deskriptivem Text Bilder von Objekten und Landschaften zu erzeugen [ZXL⁺17].

Darüber hinaus ist eine Entwicklung hin zu personalisierten Medienprodukten wahrscheinlich. So könnten beispielsweise Radiosender für einzelne Personen entstehen, die je nach Situation die Hörpräferenzen des Nutzers bedienen. Die Handlung in Romanen könnte individualisiert verlaufen und Filme mit den bevorzugten Schauspielern ausgestattet werden [Fos19, S.283]. Dadurch wird sich aber auch die rechtliche Frage der Lizenzierung von maschinengenerierten Inhalten stellen und wie Medienprodukte als geistiges Eigentum geschützt werden können. Dieser Aspekt wird sicherlich in Zukunft noch häufig zu gesellschaftlichen Auseinandersetzungen führen.

Literaturverzeichnis

- [Bro19] BROWNLEE, Jason: A Gentle Introduction to Generative Adversarial Networks (GANs). (2019). <https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/>. – abgerufen am 16.06.2020
- [Doe16] DOERSCH, Carl: Tutorial on Variational Autoencoders. In: *arXiv:1606.05908 [cs, stat]* (2016). <http://arxiv.org/abs/1606.05908>
- [Fos19] FOSTER, David: *Generative deep learning: teaching machines to paint, write, compose, and play*. O'Reilly Media, 2019
- [GBC16] GOODFELLOW, I. ; BENGIO, Y. ; COURVILLE, A.: *Deep Learning*. MIT Press, 2016
- [Goo16] GOODFELLOW, Ian: NIPS 2016 tutorial: Generative adversarial networks. In: *arXiv preprint arXiv:1701.00160* (2016). <https://arxiv.org/abs/1701.00160.pdf>
- [HB17] HUANG, Xun ; BELONGIE, Serge: Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In: *ICCV*, 2017
- [KLA19] KARRAS, Tero ; LAINE, Samuli ; AILA, Timo: A Style-Based Generator Architecture for Generative Adversarial Networks. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019
- [LTH⁺16] LEDIG, Christian ; THEIS, Lucas ; HUSZAR, Ferenc ; CABALLERO, Jose ;AITKEN, Andrew P. ; TEJANI, Alykhan ; TOTZ, Johannes ; WANG, Zehan ; SHI, Wenzhe: Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In: *CoRR* abs/1609.04802 (2016). <http://arxiv.org/abs/1609.04802>
- [Pri18] PRICE, Andrew: *The Next Leap: How A.I. will change the 3D industry*. <https://www.youtube.com/watch?v=F1gLxSLsYWQ>. Version: 2018
- [Ras18] RASTISLAV, Paluv: Grundlagen des Machine Learning – überwachtes und unüberwachtes Lernen. In: *Plus IT* (2018). <https://plus-it.de/blog/machine-learning-ueberwachtes-vs-unueberwachtes-lernen/>. – abgerufen am 02.06.2020
- [REM⁺19] ROBERTS, Adam ; ENGEL, Jesse ; MANN, Yotam ; GILICK, Jon ; KAYACIK, Claire ; NØRLY, Signe ; DINCULESU, Monica ; RADEBAUGH, Carey ; HAWTHORNE, Curtis ; ECK, Douglas: Magenta Studio: Augmenting Creativity with Deep Learning in Ableton Live. In: *Proceedings of the International Workshop on Musical Metacreation (MUME)*, 2019
- [SDT18] SALVARIS, M. ; DEAN, D. ; TOK, W.H.: *Deep Learning with Azure*. Apress, Berkeley, CA, 2018
- [Sha18] SHAFKAT, Irhum: Intuitively Understanding Variational Autoencoders. In: *Medium* (2018). <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf>. – abgerufen am 10.06.2020

- [SKGD18] SPINNER, Thilo ; KÖRNER, Jonas ; GÖRTLER, Jochen ; DEUSSEN, Oliver: Towards an Interpretable Latent Space: an Intuitive Comparison of Auto-encoders with Variational Autoencoders. In: *Proceedings of the Workshop on Visualization for AI Explainability 2018 (VISxAI)*, 2018
- [SKLO18] SANAKOYEU, Artsiom ; KOTOVENKO, Dmytro ; LANG, Sabine ; OMMER, Bjorn: A style-aware content loss for real-time hd style transfer. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018
- [ZPIE17] ZHU, Jun-Yan ; PARK, Taesung ; ISOLA, Phillip ; EFROS, Alexei A.: Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. In: *The IEEE International Conference on Computer Vision (ICCV)*, 2017
- [ZXL⁺17] ZHANG, Han ; XU, Tao ; LI, Hongsheng ; ZHANG, Shaoting ; WANG, Xiaogang ; HUANG, Xiaolei ; METAXAS, Dimitris N.: StackGAN: Text to Photo-Realistic Image Synthesis With Stacked Generative Adversarial Networks. In: *The IEEE International Conference on Computer Vision (ICCV)*, 2017