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:	22. Juni 2020
Betreuer:	Dr. Hussein Hussein

Kurzfassung

Hier steht der Text, welcher den Inhalte der Seminararbeit zusammenfasst...

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DCGAN	Deep Convolutional Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Networks
GAN	Generative Adversarial Network
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
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« bezweckt. 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-Believe-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. Deshalb ist diskriminative Modellierung synonym mit überwachtem Lernen beziehungsweise dem Erlernen einer Funktion, die eine Eingabe mithilfe eines mit Labeln 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 Labeln 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].

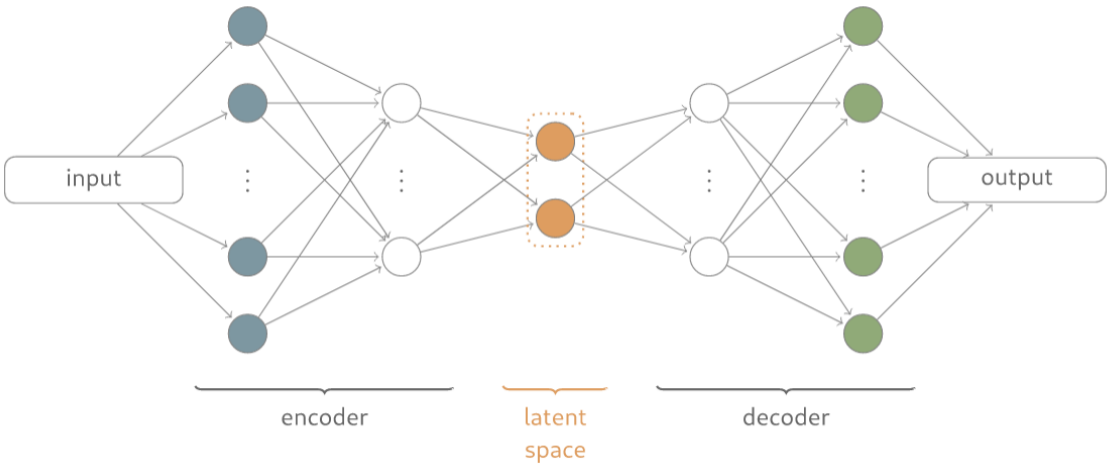

Abbildung 2.1: schematische Darstellung eines Autoencoders [SKGD18]

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].

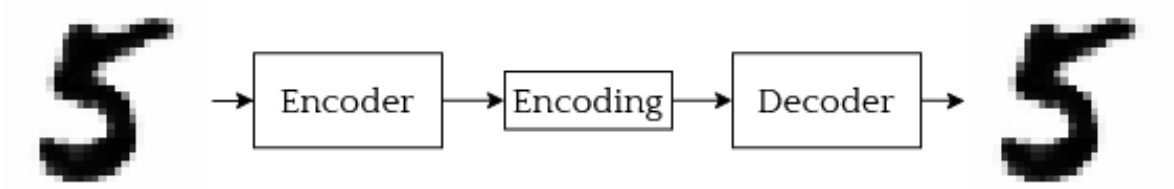

Abbildung 2.2: Funktionsweise eines Autoencoders am Beispielbild der Zahl »5« aus dem MNIST-Set

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.

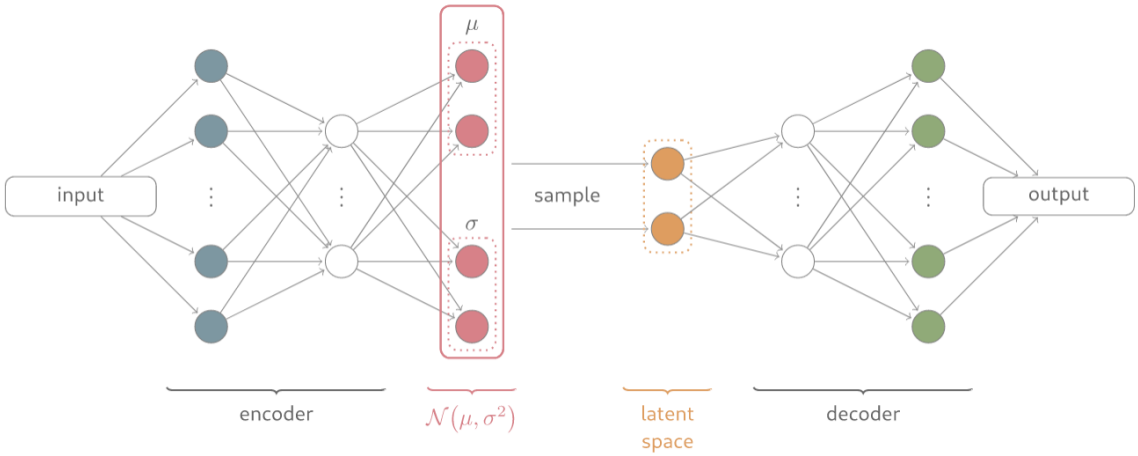

Abbildung 2.3: schematische Darstellung eines VAE [SKGD18]

Dies wird erreicht, indem der Encoder statt *einem* Vektor der Größe n, *zwei* Vektoren der Größe n - μ 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].

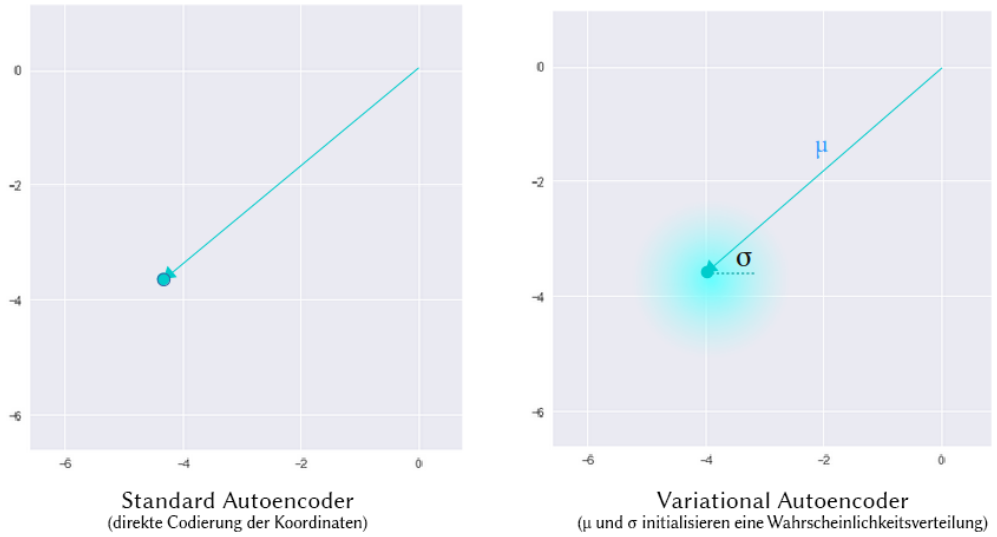

Abbildung 2.4: Unterschied im Encoding - Autoencoder vs. VAE [Sha18]

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. 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].

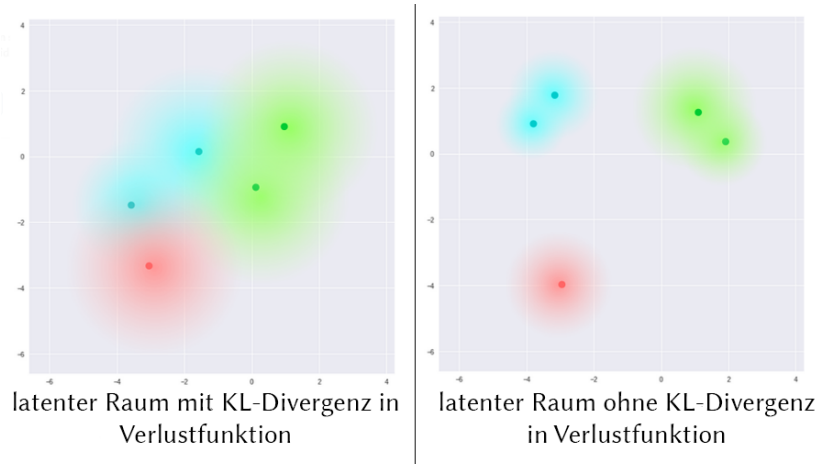

Abbildung 2.5: Verteilung der Cluster mit und ohne KL-Divergenz [Sha18]

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

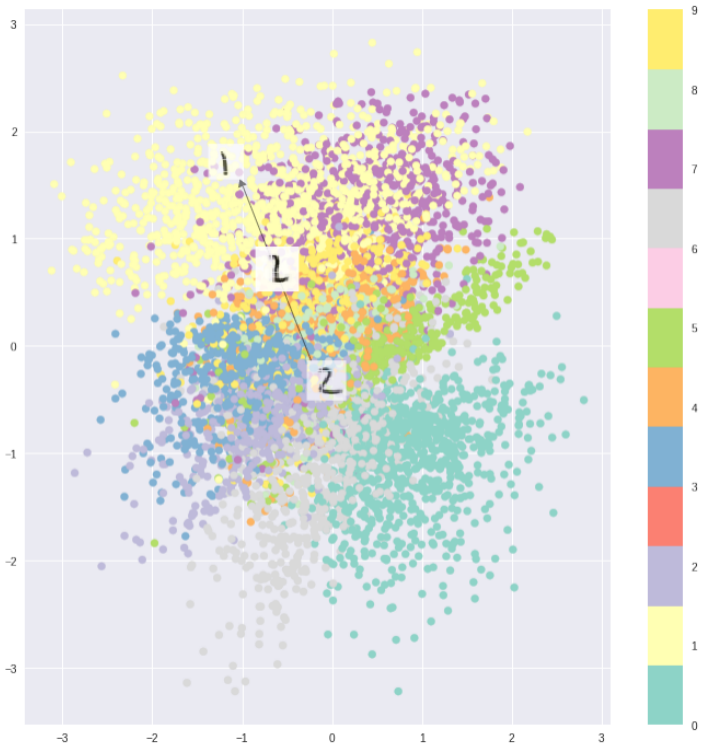

Abbildung 2.6: Verteilung im Encoding eines VAE am Beispiel des MNIST-Sets [Sha18]

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
- Diskriminator (D) - Modell, das klassifiziert, ob die zugeführten Daten vom Trainingset 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 Diskriminator 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 Diskriminator 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 Diskriminator zu täuschen [Fos19, S.99].

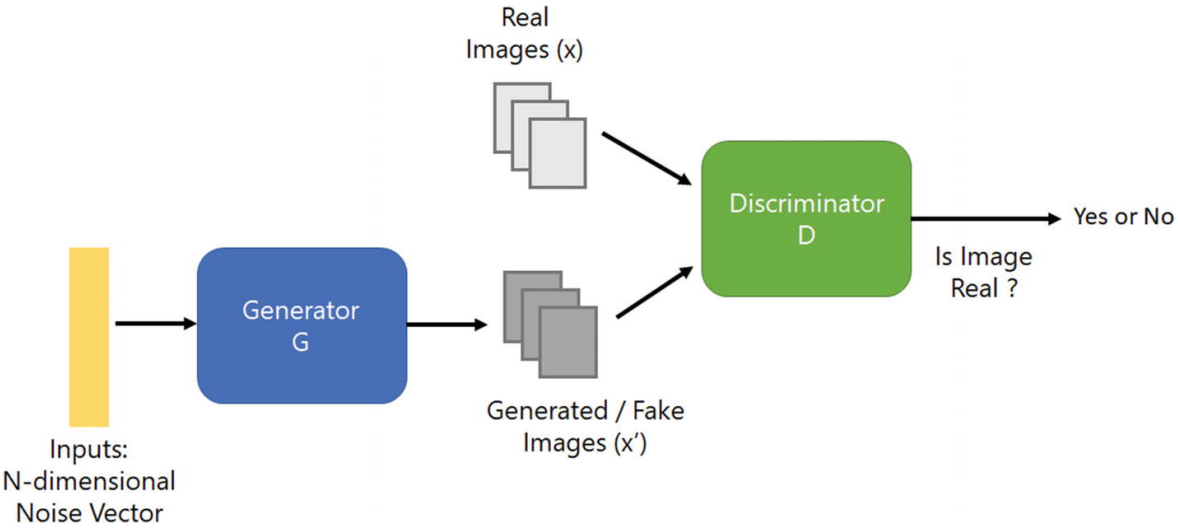

Abbildung 2.7: Funktionsweise eines GAN [SDT18]

Man kann sich das Training eines GAN als Nullsummenspiel vorstellen: "Das Generatornetzwerk sei dabei ein Geldfälscher, der versucht, Falschgeld zu produzieren. Das Diskriminatornetzwerk 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 wurden dichte Schichten anstelle der Konvolutionsschichten für das Training verwendet. Seitdem hat sich jedoch gezeigt, dass Konvolutionsschichten dem Diskriminator eine bessere Vorhersagekraft verleihen. Diese Art von GANs wird in der Literatur auch DCGAN (Deep Convolutional Generative Adversarial Network) genannt, aber 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 dann spezielle Eigenschaften, die zu einer verbesserten Lösung der jeweiligen Aufgabenstellung führt.

3 Anwendungsmöglichkeiten generativer Modelle

Im Gegensatz zu diskriminativen Modellen bieten generative Modelle kaum Lösungansätze 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. Deshalb liegt das größte Potenzial der Entwicklung in der Kreativbranche.

Bild - Stiltransfer (aiportraits.com) - Entrauschen (Nvidia denoiser) - SRGAN - SyleGAN2 (thispersondoesnotexist.com)

Ton -Stiltranfer Text

4 Zusammenfassung

Text des Fazits... Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

[GBC16] [Doe16] [Goo16] [Roc19] [Ras18] [Sha18] [Fos19] [KW13] [SKGD18]

5 Ausblick

Text des Ausblicks... Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

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

[KW13] Kingma, Diederik ; Welling, Max: Auto-encoding variational bayes. In: *arXiv preprint arXiv:1312.6114* (2013). https://arxiv.org/abs/1312.6114

[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

[Roc19] Rocca, Joseph: Understanding Variational Autoencoders (VAEs) - Towards Data Science. In: *Medium* (2019). https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73. – abgerufen am 08.06.2020

[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 Autoencoders with Variational Autoencoders. In: *Proceedings of the Workshop on Visualization for AI Explainability 2018 (VISxAI)*, 2018