



Bachelorarbeit

Variational Autoencoder zur Optimierung von Videos

Erstprüfer: Prof. Dr. ...

Zweitprüfer: Prof. Dr. ...

Jana Mustermann

Matrikelnummer: 123456789

Musterstraße. 1

44801 Bochum

Tel.: 0234/123456

E-Mail: jana.mustermann@hs-bochum.de

Studiengang:

Wirtschaftsinformatik

7. Fachsemester, Sommersemester 2013

Abgabedatum 11. Dezember 2018

Inhaltsverzeichnis

Abkürzungsverzeichnis	III
Abbildungsverzeichnis	IV
1 Einleitung	5
2 Maschinelles lernen	6
2.1 Modelle des Überwachten Lernens	7
2.2 Algorithmen des Überwachten Lernens.....	8
2.2.1 Neuronale Netze	8
2.2.2 Konvolutionale Netzwerke	12
2.3 Lernen	14
2.3.1 Performancemessungen	14
2.3.2 Maximum-Likelihood Estimation	17
2.3.3 Lernen	17
3 Inferenz	22
3.1 Bayes Theorem	22
3.2 Inferenzmethoden	24
3.3 Expectation-Maximization-Algorithmus.....	25
3.3.1 K-Means als EM-Algorithmus	25
3.3.2 EM-Algorithmus für ein mehrdimensionales Verteilungsmodell.....	26
3.3.3 Herleitung der Variational Inferenz	28
3.4 Variational Inferenz	30
4 Statistisches Maschinelles Lernen	33
4.1 Mannigfaltigkeit.....	34
4.2 Hauptachsentransformation	35

4.3 Autoencoder	37
4.4 Variational Autoencoder	40
4.5 State of the Art für Generative Modelle	44
5 Praxisprojekt: Optimierung von Videos mithilfe von Variational Autoencodern	46
5.1 Der Datensatz sowie die Vorverarbeitung	46
5.2 Model.....	47
5.3 Durchführung	49
5.3.1 Die Mannigfaltigkeit der Hand.....	49
5.3.2 Die Komponenten der Repräsentation.....	51
5.4 Optimierung von Videos.....	54
5.4.1 Versuchsaufbau	54
5.4.2 Beobachtbare Probleme	56
5.4.3 Ergebnisse	58
5.5 Fazit des Experiments.....	60
6 Ausblick	61
Literaturverzeichnis	XXXVII

Abkürzungsverzeichnis

ELBO	Evidence Lower Bound
EM-Algorithmus	Expectation-Maximization Algorithmus
GAN	Generative Adversarial Network
KL-Abweichung	Kullback-Leibler-Abweichung
MLE	Maximum Likelihood Estimation
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
VAE	Variational Autoencoder
VI	Variational Inferenz
BMU	Bundesministerium für Umwelt
DEHSt	Deutsche Emissionshandelsstelle
EEX	European Energy Exchange
EG	Europäische Gemeinschaft
ETS	Emission Trading System
EU	Europäische Union
EUA	European Union Allowances
FCOM	Flight Crew Operating Manual
IATA	International Air Transport Association
ICAO	International Civil Aviation Organisation
IPCC	Intergovernmental Panel on Climate Change
LCC	Low Cost Carrier
OECD	Organisation für wirtschaftliche Zusammenarbeit und Entwicklung
TEHG	Treibhausgas-Emissionshandelsgesetz
UBA	Umweltbundesamt

Abbildungsverzeichnis

Abbildung 1: Unter- und Überanpassung anhand eines Regressionsproblems	7
Abbildung 2: Neuronales Netz mit einem einzigen Neuron	9
Abbildung 3: Der Aufbau eines tiefen neuronalen Netzes	10
Abbildung 4: Gängige Aktivierungsfunktion. Links die Sigmoid Funktion in der Mitte die Softplus Funktion und rechts die ReLU Funktion.....	11
Abbildung 5: Schematischer Aufbau eines Konvolutionalen Netzwerkes	12
Abbildung 6 Konvolutionsoperation	13
Abbildung 7: Gradientenabstiegsverfahren.....	21
Abbildung 8: K-Means Algorithmus	25
Abbildung 9: Clustering mit einer Gaußschen Mischverteilung	27
Abbildung 10: Variational Inferenz	30
Abbildung 11 KL-Abweichung und negative KL-Abweichung	32
Abbildung 12: Visualisierung der Mannigfaltigkeit.....	35
Abbildung 13: Autoencoder Struktur.....	38
Abbildung 14: Variational Autoencoder	40
Abbildung 15: Variational Autoencoder vorhergesagte Distributionen	42
Abbildung 16: Entangled Repräsentationen	43
Abbildung 17: Vergleich der Resultate zwischen MSE und GAN	45
Abbildung 18: Maskieren des Videos.....	46
Abbildung 19: Trainingsdaten Beispiele.....	47
Abbildung 20: Die Mannigfaltigkeit der Hand	50
Abbildung 21: Auffälligkeiten der latenten Plots.....	52
Abbildung 22: Latenter Plot Nr. 5	53
Abbildung 23: Performancemessungen der einzelnen Models.....	55
Abbildung 24: Unschärfe des Videos	56
Abbildung 25: Bildverunreinigung	58

1 Einleitung

// Wird später erstellt

2 Maschinelles Lernen

Hinter dem Begriff Maschinelles Lernen (engl. Machine Learning) verstecken sich Algorithmen die fähig sind von Daten zu lernen, um mit neuen gleichartigen Daten umzugehen.

Mitchell liefert eine Definition zu dem Lernprozess:

„A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.“¹

Dabei ist die Erfahrung E eine Sammlung von Datensätzen, die von der Aufgabe abhängig sind, wie beispielsweise Sensordaten, Bilder, Texte oder Ähnliches. Die Aufgabe T beschreibt was der Algorithmus tun soll wie beispielsweise Werte vorhersagen, Datensätze klassifizieren, Bilder generieren usw. Wie gut der Algorithmus diese Aufgaben ausführt wird durch die Messung von P bestimmt. P wird häufig auch als Fehlerfunktion, Verlustfunktion oder auch als Zielsetzungsfunktion bezeichnet.²

Generell wird zwischen drei verschiedenen übergeordneten Lerntypen unterschieden:

- **Überwachtes Lernen:** Dabei hat jeder Datensatz neben den eigentlichen Daten noch den gewünschten Output bzw. ein Label. Das Ziel ist es eine Funktion zu finden die den Datensatz auf den Output mappt. Der Term „überwacht“ entwickelte sich daraus, dass ein „Lehrer“ oder auch Supervisor dem Algorithmus beim Lernen hilft und ihm den richtigen Wert vorgibt. Das überwachte Lernen stellt das Herz des maschinellen Lernens da und viele Algorithmen des maschinellen Lernens versuchen, wie wir später sehen werden, die Aufgabe auf eine Problemstellung des überwachten Lernens zurückzuführen.
- **Unüberwachtes Lernen:** Beim unüberwachten Lernen wird dem Algorithmus kein entsprechendes Zielattribut vorgegeben. Es geht überwiegend darum verborgenes Wissen bzw. Informationen zu extrahieren und dem Anwender tiefere Einblicke zu gewähren, um Schlüsse über Verhalten oder Kausalitäten aufzudecken. Es ist eng verwandt mit der Mustererkennung und Data Mining. Eine Besonderheit an dieser Problemstellung ist, dass die Anzahl der verfügbaren Daten enorm ist im Vergleich zu dem überwachten Lernen, in welchem der Output vorgegeben wird. Unüberwachtes Lernen wird auch als „Information Retrieval“-Prozess bezeichnet.
- **Bestärkendes Lernen:** Eine weitere Lernform ist das bestärkende Lernen. Dabei interagiert der Algorithmus mit der Umgebung und erhält Inputreize aufgrund von vorherigen

¹ Mitchell (1997), S. 2.

² Vgl. Goodfellow (2016), S. 82.

Handlungen in einer Umgebung. Ziel ist es, basierend auf dem Muster von Inputreizen entsprechende Aktionen bzw. Reaktionen auszuführen mit dem Ziel, seine Belohnungen (engl. rewards) zu maximieren. Der Algorithmus lernt mit fortschreitender Simulation, wie gut oder wie schlecht eine Aktion war und passt seine Aktionen dementsprechend an.³

Das Lernen im eigentlichen Sinne geschieht meistens dadurch, dass der Algorithmus durch eine Reihe von Parametern θ sein Verhalten gegenüber den Inputmustern bestimmt wird. Durch gezielte Beeinflussung dieser Parameter erlernt der Algorithmus das gewünschte Verhalten. Zusammengefasst lässt sich also sagen, dass wir Erfahrungen bzw. Daten brauchen sowie ein Modell bzw. einen Algorithmus, der diese Daten verarbeitet, und eine Performance-Funktion im Bezug auf die Aufgabe.

2.1 Modelle des Überwachten Lernens

Ziel des überwachten lernen ist es ein Modell zu finden das in der Lage ist, auf neue, unbekannte Daten richtig zu reagieren.⁴ Damit ein Modell diese Anforderung erfüllt, führen wir die Begriffe Unteranpassung, Überanpassung und die Fähigkeit zu generalisieren ein, um Modelle nach diesen Punkten zu bewerten. Als Beispiel nehmen wir uns ein Regressionsproblem, bei dem wir versuchen, eine Reihe von Datenpunkten mit einer Funktion zu approximieren, sodass die Punkte auf oder möglichst nahe an dem Graphen der Funktion liegen. Das Modell erhält eine Eingabe x und generiert dazu den geschätzten Wert für das Zielattribut y .

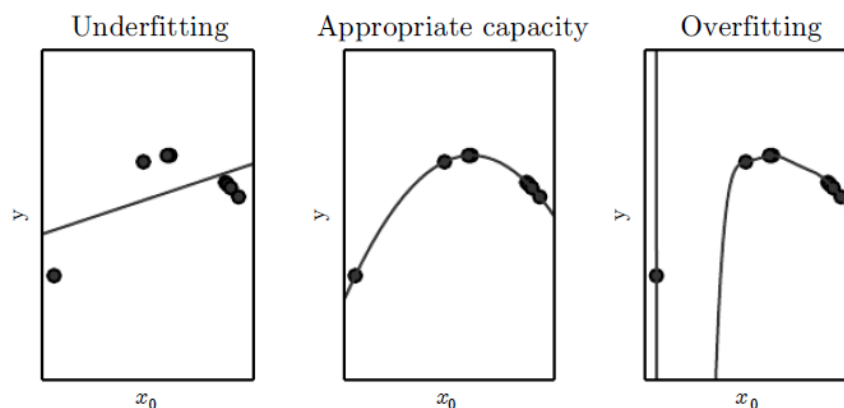


Abbildung 1: Unter- und Überanpassung anhand eines Regressionsproblems

Quelle: Goodfellow (2016), S. 113.

Es ist sehr deutlich erkennbar, dass der Graph auf der linken Seite die ursprünglichen Daten suboptimal abbildet. Das liegt daran, dass das Modell versucht, eine nicht lineare Funktion mit

³ Vgl. Kriesel (2007), S. 53.

⁴ Vgl. Goodfellow (2016), S. 110.

einer linearen Funktion zu approximieren. Allerdings liegt der Fehler nicht an dem Modell, sondern an dem Anwender, der den Daten einen linearen Verlauf unterstellt und das Modell falsch designt hat. Der Graph auf der rechten Seite zeigt eine Überanpassung des Graphen. Dabei wird versucht, ein Polynom höheren Grades auf ein Polynom kleineren Grades abzubilden. Wenn wir die Funktion nutzen würden um einen neuen Wert abzubilden, würden wir untypische Werte erhalten, die zwar den Trainingsdaten entsprechen, allerdings nicht den neuen Daten. Das Bild in der Mitte zeigt ein Modell, dass eine Funktion gelernt hat, welche die Daten am ehesten repräsentiert. Eine Aufgabe ist es, einen guten Trade-off zwischen einer Unter- und Überanpassung zu finden, um die Funktion möglichst exakt zu approximieren. Generell lässt sich sagen, dass die Komplexität der Aufgabe ungefähr der Komplexität des Modells entsprechen soll.⁵ Um die Fähigkeit des Generalisierens zu testen werden in der Praxis häufig gezielt Datensätze entfernt bzw. dem Model vorenthalten, um das Modell auf genau diese Eigenschaften hin zu überprüfen. Diese Überprüfung kann beispielsweise durch die Performancemessung stattfinden, sofern die Labels bekannt sind, um einen Fehlerwert zu berechnen und diese miteinander zu vergleichen.

2.2 Algorithmen des überwachten Lernens

Es existiert eine Vielzahl an Modellen, um die Aufgabenstellung des überwachten Lernens zu lösen. Je nach Menge der Daten, der Problemstellung sowie der Verfügbarkeit von Rechenressourcen hängt die Wahl des entsprechenden Algorithmus davon ab. Wir werden in dieser Arbeit zwei gängige Vertreter betrachten, die für eine Vielzahl an Problemstellungen einsetzbar sind, nämlich künstliche neuronale Netze, die sich insbesondere durch ihre vielfältige Einsatzmöglichkeiten auszeichnen, und konvolutionale Netze, die meist in der Bilderkennung eingesetzt werden.

2.2.1 Neuronale Netze

In dem vergangenen Jahrzehnt haben sich künstliche neuronale Netze als effiziente Möglichkeit herausgestellt, um nicht lineare Funktionen zu approximieren.⁶ Wie der Name bereits vermuten lässt, war die ursprüngliche Idee dahinter, das zentrale Nervensystem auf einer mathematischen Ebene abzubilden. Zu den Pionieren auf diesem Themenfeld gehören Warren McCulloch und

⁵ Vgl. Goodfellow (2016), S. 112.

⁶ Vgl. Goodfellow (2016), S. 224.

Walter Pitts, die in ihrer Arbeit im Jahr 1943 erste Ideen für ein vorwärts gerichtetes neuronales Modell vorstellen.⁷

Die Idee hinter neuronalen Netzen ist, dass ein Neuron durch eine Reihe von Inputs aktiviert wird bzw. Informationen erhält, diese dann durch eine mathematische Vorschrift miteinander kombiniert und anschließend den berechneten Wert weitergibt. Wir werden uns in diesem Kapitel mit den wesentlichen Elementen von neuronalen Netzen vertraut machen, die in einer Problemstellung des überwachten Lernens angewendet werden.

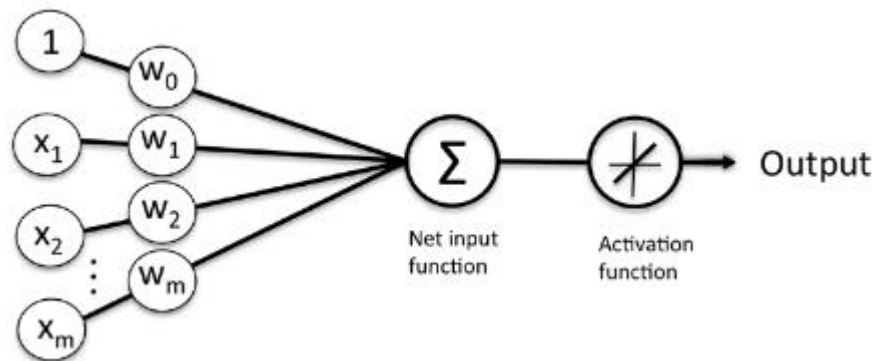


Abbildung 2: Neuronales Netz mit einem einzigen Neuron

Quelle: in Anlehnung an Raschka (2017), S. 42.

Die Abbildung zeigt den schematischen Aufbau eines neuronalen Netzes mit einer Inputschicht und einem Ausgabeneuron bzw. einer Ausgabeschicht mit einem einzigen Neuron. Auf der linken Seite befinden sich die Inputreize in Form eines Vektors $x \in \mathbb{R}^n$, die mit einem Gewichtsvektor $w \in \mathbb{R}^n$ multipliziert werden und so den vorläufigen Output des Neurons darstellen. Es wird noch ein Offset-Bias-Parameter w_0 hinzuaddiert (im nachfolgenden mit b abgekürzt), da der vorhergesagte Graph sonst nur durch den Ursprung laufen würde.⁸ Beispielsweise könnte die Funktion $f(x) = 1$ nicht abgebildet werden ohne den Bias-Parameter. Abschließend wird noch eine sogenannte Aktivierungsfunktion angewandt, um nicht-lineare Funktionen abzubilden zu können. Die Gewichte lassen sich wie folgt interpretieren: Ein negatives Gewicht ist ein für das Neuron hemmender Reiz, das die Aktivierung des Neurons mindern soll. Ein positives Gewicht hingegen hat einen erregenden Einfluss und soll das Neuron in einen aktiven Status zu versetzen. Wenn das Gewicht nahe Null ist übt es keine oder nur eine geringe Wirkung aus, es hat also weder eine animierende noch eine hemmende Wirkung.⁹

Diesem Modell ist es allerdings nur beschränkt möglich, eine beliebig komplexe Funktion zu erlernen, deswegen wird in dem Kontext von neuronalen Netzen häufig der Begriff „Deep

⁷ McCulloch (1943), S. 113-133.

⁸ Vgl. Goodfellow (2016), S. 109.

⁹ Vgl. Rey (2011), S. 15.

Learning“ verwendet. Es handelt sich dabei um die Aneinanderreihung von neuronalen Netzschichten, um komplexe geschachtelte Funktionen zu erlernen.

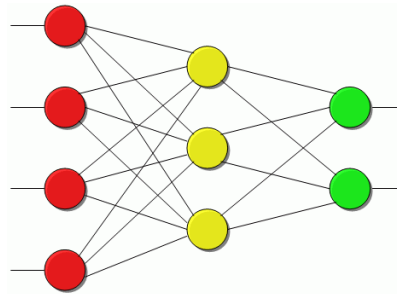


Abbildung 3: Der Aufbau eines tiefen neuronalen Netzes

Quelle: Rey (2011), S. 15.

Die Abbildung zeigt ein tiefes neuronales Netz mit einer Inputschicht (rot), einer versteckten Schicht (gelb) und einer Ausgabeschicht (grün). Die mathematische Vorschrift zu Berechnung der Ausgabe sieht wie folgt aus:

$$f(x) = f^2(f^1(x)) \text{ mit } f^i(x) = \text{activation}(W_i x + b_i)$$

Jede Schicht bekommt eine Gewichtsmatrix W und einen Bias-Parameter b zugeordnet, mit denen sich die Eingaben einer höheren Schicht berechnen lassen. Diese Prozedur wird auch als ein vorwärts gerichtetes Propagieren bezeichnet.¹⁰

Das Designen von versteckten Schichten wie beispielsweise die Auswahl von Aktivitätsfunktionen sowie die Festlegung der Anzahl der Neuronen pro Schicht wird im Zusammenhang mit der Struktur, mit der die einzelnen Schichten in Verbindung stehen, auch als Modellarchitektur bezeichnet.¹¹ Die Auswahl wird vor dem Trainingsprozess festgelegt und kann in der Regel nicht nachträglich beeinflusst werden.

Ein weiterer wichtiger Faktor ist die Wahl der Aktivierungsfunktionen. Üblicherweise werden in dem gesamten Netzwerk die gleichen Aktivierungsfunktionen verwendet, außer der Ausgabeschicht, in der diese nach den Bedürfnissen angepasst werden (wie beispielsweise der Wertebereich der Ausgabe). Dabei ist es in den meisten Fällen so, dass zwingend Aktivierungsfunktionen benötigt werden, um die Mächtigkeit des Netzwerkes zu erhöhen. Präziser ausgedrückt werden Aktivierungsfunktionen benötigt, um nicht-lineare Funktionen approximieren zu können. Um deren Relevanz zu zeigen modifizieren wir die Funktion für die vorwärts gerichtete Propagation. In ihr wird auf die Aktivierungsfunktion verzichtet sowie auf den Bias-Parameter, um die Gleichung übersichtlicher zu gestalten.

$$f(x) = f^2(f^1(x)) \text{ mit } f^i(x) = W_i x$$

Anders formuliert entsteht:

¹⁰ Vgl. Rey (2011), S. 52.

¹¹ Vgl. Goodfellow (2016), S. 197.

$$f(x) = W_2 W_1 x = W^* x$$

Das tiefe neuronale Netz hat also nicht die Fähigkeit, komplexere Funktion zu erlernen, da sie mit einer einzigen Gewichtsmatrix W^* ausgetauscht werden kann. Es werden demzufolge zwingend Aktivierungsfunktionen benötigt, um nicht-lineare Funktionen zu erlernen.¹²

Prinzipiell erfüllt jede Funktion diese Eigenschaft, die nicht ausschließlich linear ist. Allerdings haben sich in der Literatur sowie in der Praxis einige Aktivitätsfunktionen etabliert, wobei in diesem Bereich noch aktiv Forschung betrieben wird. Die Auswahl kann nach verschiedenen Kriterien erfolgen, wie beispielsweise der Begrenzung des Wertebereichs, die biologische Plausibilität oder aber auch die Differenzierbarkeit der Funktion.¹³ Eine weitere interessante Eigenschaft ist, wie sich die Ableitung der Funktion verhält, also ob der Gradient auf bestimmten Teilbereichen der Funktion sehr klein ist oder sehr groß ist.¹⁴ Sollte der Gradient klein sein, macht diese Eigenschaft das Lernen für den Algorithmus schwerer und es wird mehr Zeit benötigt, um ähnliche Ergebnisse zu erzielen wie bei anderen Aktivierungsfunktionen. Nachfolgend sind einige gängige Aktivierungsfunktionen aufgelistet.

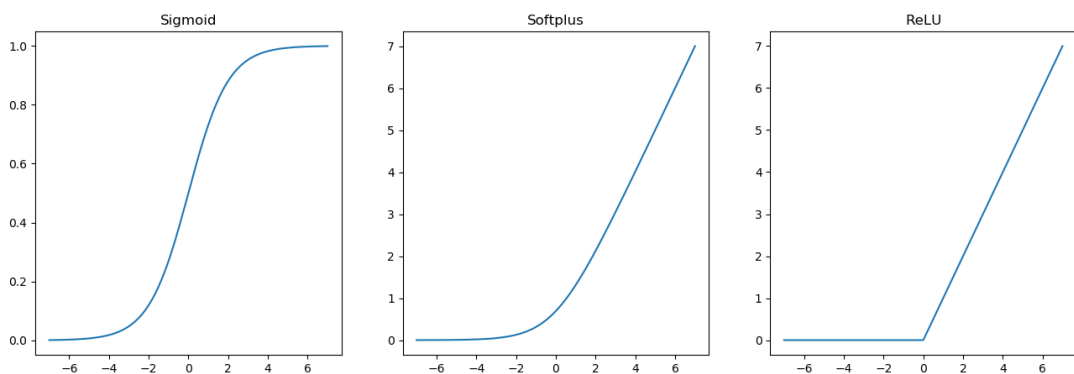


Abbildung 4: Gängige Aktivierungsfunktion

Quelle: Eigene Darstellung

- Sigmoid: Bei der Sigmoid-Funktion liegt der Wertebereich zwischen 0 und 1. Daher wird diese Funktion häufig bei Klassifizierungsproblemen eingesetzt um zu bestimmen, mit welcher Sicherheit der Algorithmus eine Klassenzugehörigkeit vorhersagt. Eine weitere nützliche Eigenschaft ist die leichte Differenzierbarkeit $\sigma'(z) = \sigma(z)(1 - \sigma(z))$, da die Werte beim Maschinellen Lernen oftmals bereits berechnet wurden und sie somit wiederverwendet werden können. Ein Nachteil dieser Funktion ist, dass der Gradient bei $z \ll 0$ und $z \gg 0$ sehr klein ist und das Lernen somit erschwert wird.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

¹² Vgl. Goodfellow (2016), S. 172.

¹³ Vgl. Rey (2011), S. 22.

¹⁴ Vgl. Goodfellow (2016), S. 179.

- Softplus: Die Softplus-Funktion hat einen Wertebereich 0 bis ∞ . Sie kommt zur Vorhersage von positiven Werten zum Einsatz.

$$\zeta(x) = \ln(1 + e^x)$$

- Rectified Linear Unit (ReLU): Die ReLU-Funktion verhält sich ähnlich wie die Softplus-Funktion, allerdings ist der Übergang deutlich härter. Die ReLU-Funktion ist streng genommen nicht differenzierbar bei $x = 0$, jedoch wird sie in der Praxis häufig verwendet, da sie einfach zu implementieren ist und die Steigung konstant bleibt.¹⁵

$$\text{ReLU}(x) = \max(0, x)$$

2.2.2 Konvolutionale Netzwerke

Konvolutionale Netze werden überwiegend bei der Analyse von Bilddaten verwendet. Der Unterschied zu klassischen neuronalen Netzen ist, dass nicht länger ein Vektor an Daten analysiert wird, sondern eine Matrix an Daten, das heißt die Bilddaten stehen in einer Korrelation zueinander. Bildpunkte, die näher beieinander liegen, stehen in einer engeren Verbindung als weit entfernte Bildpunkte.¹⁶ Vergleichbar wie bei einem tiefen neuronalen Netz ist das Netzwerk auch in verschiedene Schichten aufgeteilt. Jede dieser Schichten besteht typischerweise aus folgenden Elementen:

- einer Konvolutionsoperation
- einer Aktivierungsfunktion
- und einer Pooling-Operation

Durch das Verketteten dieser Schichten ist es möglich, genau wie bei neuronalen Netzen, komplexere Funktionen zu erlernen die leistungsfähiger als alleinstehende sind. Die letzten Schichten eines konvolutionalen Netzes sind üblicherweise neuronale Netze dessen Input die Ausgabe der letzten Konvolutionsschicht darstellt, der als Vektor neu arrangiert worden ist.

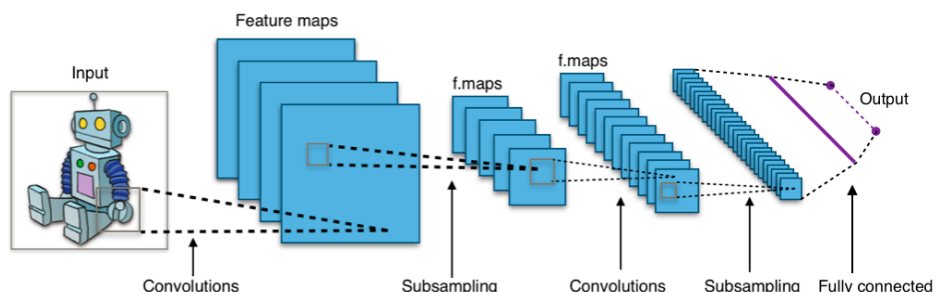


Abbildung 5: Schematischer Aufbau eines Konvolutionalen Netzwerkes

Quelle: Aphex34 (https://commons.wikimedia.org/wiki/File:Typical_cnn.png), <https://creativecommons.org/licenses/by-sa/4.0/legalcode>

¹⁵ Vgl. Goodfellow (2016), S. 192.

¹⁶ Vgl. Bishop (2006), S. 267.

Bei der Konvolutionsoperation wird ein Filter auf einen Teilbereich einer Matrix, bzw. einem Array an Matrizen, angewandt, um einen neuen Wert zu berechnen. Durch das schrittweise Verschieben des Filters, dem sogenannten Stride, wird die ganze Matrix durchlaufen und eine neue Matrix generiert, die komprimierte Informationen des Bildes enthält. Der Filter ist üblicherweise quadratischer Natur und umfasst n^2 Gewichte und einen Bias-Parameter. In der Vergangenheit wurden viele dieser Filter per Hand entworfen, um bestimmte Effekte bei der digitalen Bildverarbeitung zu erzielen, wie einen Glättungseffekt oder Kantenerfassung-Template, um beispielsweise abrupte Änderungen in dem Bild aufzuspüren. Ohne allerdings Filter von Hand zu entwerfen, benutzen wir das Maschinelle Lernen um Filter zu entdecken, die genau für die Aufgabenstellung relevante Informationen extrahieren. Die nachfolgende Abbildung zeigt die Konvolutionsoperation mit einem 3x3 Filter. Die Anzahl der Parameter umfasst somit 9. Hinzukommt noch ein Bias Parameter und somit erhöht sich die Anzahl der Parameter auf 10. Die mathematische Vorschrift um den Wert der neuen Matrix an der Stelle i, j zu berechnen lautet:

$$S(i, j) = (I * K)(i, j) = \sum_n \sum_m I(i + m, j + n) * K(m, n)$$

$S(i, j)$ entspricht den berechneten Wert der generierten Matrix an Stelle $S_{i,j}$.

I entspricht der Input Matrix bzw. der Eingabe der Schicht.

K entspricht dem Filter, der in der englischen Literatur auch als Kernel bezeichnet wird.

i, j bezeichnen die jeweilige Reihe und die jeweilige Spalte, an der die Operation durchgeführt werden soll. Die zuvor festgelegte Schrittweise gibt an, welcher Wert auf i und j hinzuaddiert wird und an welcher Stelle die darauffolgende Operation durchgeführt werden soll.

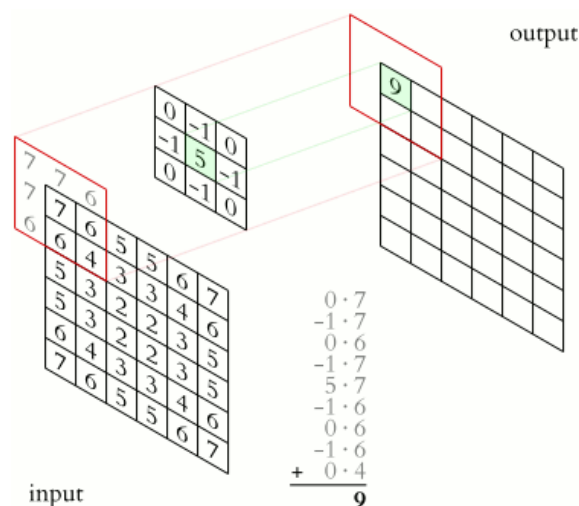


Abbildung 6: Visualisierung der Konvolutionsoperation

Quelle: Michael Plotke (https://commons.wikimedia.org/wiki/File:3D_Convolution_Animation.gif), „3D Convolution Animation“, <https://creativecommons.org/licenses/by-sa/3.0/legalcode>

Würde in der Praxis nur ein Filter pro Inputmatrix verwendet werden, würde es zu einer starken Unteranpassung kommen. Neun Parameter, bzw. 10 Parameter, sind nicht ausreichend genug,

um genügend Informationen aus der Matrix zu extrahieren, um eine brauchbare Analyse durchführen zu können. Deshalb werden mehrere Filter gleichzeitig angewendet und die resultierende Matrix wird ein Array an Matrizen mit der Länge entsprechend der Anzahl der Filter. Die erste Matrix, das ursprüngliche Bild, ist häufig auch ein Array an Matrizen da die einzelnen Kanäle für den roten, grünen und blauen Farbton separat gespeichert werden. Im Falle eines Graubildes handelt es sich allerdings um eine einfache Matrix.

Nachdem jede Konvolutionsoperation in der entsprechenden Schicht vollzogen wurde wird auf jeden berechneten Wert noch ein Bias-Parameter hinzuaddiert und eine Aktivierungsfunktion auf die berechneten Werte angewendet.

Eine weitere Operation ist die Pooling-Operation. In ihr wird auf den zuvor berechneten Schritt ein weiterer Filter angewendet, der die Werte gegenüber kleinen Änderungen weniger anfällig machen soll.¹⁷ Die Pooling-Operation erreicht dieses Verhalten dadurch, dass der Filter je nach Wahl den Mittelwert oder den Maximalwert zurückgibt.

2.3 Das Lernen der Modelle

Bisher haben wir zwei gängige Algorithmen für das maschinelle Lernen betrachtet. Allerdings sind wir noch nicht auf den charakteristischen Lernprozess eingegangen, der es erlaubt, Wissen in den Parametern abzuspeichern. Wir werden uns in diesem Kapitel erst mit der Performance-messung vertraut machen und anschließend das Konzept der Maximum-Likelihood-Estimation kennenlernen. Abschließend werden wir diese dann in dem Kontext des maschinellen Lernens anwenden.

2.3.1 Performancemessungen

In vielen Bereichen des maschinellen Lernens wird den Daten ein bestimmtes statistisches Modell unterstellt, das diese Daten generiert, den sogenannten Daten generierenden Prozess P_{Data} .¹⁸ Die zentrale Überlegung dabei ist, dass wir ein Model entwerfen wollen P_{Model} das sich gleich bzw. ähnlich verhält wie P_{Data} . Ziel der Performancemessung ist es, die Abweichung der Verteilungen, also die Menge an Unsicherheiten, zu quantifizieren. Anders formuliert messen wir die Distanz zwischen der Vorhersage des Datenpunkts und dem tatsächlichen Wert, dem Soll-Wert. In der Literatur finden sich viele Metriken, um die Distanz zwischen zwei Punkten zu bestimmen. Eine dieser Metriken ist beispielsweise die L2-Norm, die auch als „Squared“-Norm

¹⁷ Vgl. Goodfellow (2016), S. 357.

¹⁸ Vgl. Goodfellow (2016), S. 111.

bezeichnet wird. Sie entspricht am ehesten unserem natürlichen Verständnis einer Distanz. Es ist die euklidische Norm, die zusätzlich potenziert wird:

$$L_2(x, y) = \left[\sqrt{\sum_i (x_i - y_i)^2} \right]^2 = \sum_i (x_i - y_i)^2$$

Sämtliche Elemente des Differenzvektors zwischen x und y werden potenziert und aufsummiert. In der Literatur wird diese Metrik oft ohne Herleitung eingeführt und dem Leser eine intuitive Erklärung gegeben, also dass kleine Abweichungen durch das Potenzieren noch kleiner werden und das große Fehler deutlich ausschlaggebender sind. Allerdings kann die L2-Norm auch mithilfe obiger Überlegungen über zwei verschiedene Verteilungen, und einer speziellen Metrik hergeleitet werden. Die Herleitung ist deshalb relevant, da sich mit ihr auch andere Metriken herleiten lassen.

Um die Ähnlichkeit zweier statistischer Verteilungen zu bestimmen, kann die Kullback-Leibler-Abweichung (KL-Abweichung, engl. „KL-Divergence“) verwendet werden. Sie misst die Ähnlichkeit bzw. die Distanz zweier Distributionen:

$$D_{KL}(P||Q) = \mathbb{E}_{x \sim P} \left[\log \frac{P(x)}{Q(x)} \right] = \sum_{x \in X} P(x) * \log \frac{P(x)}{Q(x)}$$

Eigenschaften der KL-Abweichung sind, dass sie keine negativen Werte annehmen kann und dass die Distanz nicht symmetrisch ist, sprich: $D_{KL}(P||Q) \neq D_{KL}(Q||P)$. Die KL-Abweichung ist gleich Null, wenn die beiden Distributionen gleich sind.

Wenn wir für P bzw. Q die beiden Distributionen einsetzen, erhalten wir Folgendes:

$$D_{KL}(P_{Data}||P_{Model}) = \mathbb{E}_{x \sim P_{Data}} \left[\log \frac{P_{Data}(x)}{P_{Model}(x)} \right]$$

Um nun die Distanz zwischen den Distributionen zu bestimmen und mit der Annahme das P_{Data} unveränderlich ist, also eine Konstante darstellt, lässt sich Folgendes herleiten:

$$\begin{aligned} D_{KL}(P_{Data}||P_{Model}) &= \mathbb{E}_{x \sim P_{Data}} [\log P_{Data}(x) - \log P_{Model}(x)] \\ &= \mathbb{E}_{x \sim P_{Data}} [\log P_{Data}(x)] - \mathbb{E}_{x \sim P_{Data}} [\log P_{Model}(x)] \\ D_{KL}(P_{Data}||P_{Model}) &\propto -\mathbb{E}_{x \sim P_{Data}} [\log P_{Model}(x)] \end{aligned}$$

Der Term auf der rechten Seite wird auch als Cross-Entropy bezeichnet und durch dessen Minimierung nähert sich P_{Model} der Distribution P_{Data} an. Mit anderen Worten verhält sich die Cross-Entropy proportional zu KL-Abweichung zwischen P_{Data} und P_{Model} .

Würde man beispielsweise P_{Model} als eine Normalverteilung betrachten, deren Wert für μ wir in Abhängigkeit von dem Wert für x vorhersagen wollen, würde Folgendes gelten:¹⁹

¹⁹ Vgl. Goodfellow (2016), S. 132.

$$-\mathbb{E}_{x \sim P_{Data}}[\log P_{Model}(x)] = -\mathbb{E}_{x \sim P_{Data}}[\log P_{Model}(y|\mu, \sigma = 1)] \text{ mit } \mu = f(x; \theta)$$

,y' entspricht dabei dem richtigen Wert, der in Relation zu x steht. Je genauer unsere Vorhersage von μ mit gegebenem x ist, desto geringer ist die Cross-Entropy zwischen den Distributionen und unsere Abweichung von P_{Data} . An dieser Stelle lässt sich auch die Überanpassung neu betrachten, da den Wahrscheinlichkeiten von P_{Data} keine Unsicherheiten unterstellt werden (da diese nicht bekannt sind). Dadurch wird P_{Model} versuchen die Menge an Unsicherheit zu reduzieren, obwohl P_{Data} auch Unsicherheiten besitzt durch beispielsweise nicht erfasste weitere Faktoren. Durch einige Umformungen und der Annahme, dass σ fix bzw. eine Konstante ist kann die L2-Norm hergeleitet werden:

$$Squared Error = |y - \mu|^2 \text{ mit } \mu = f(x; \theta)$$

Um den Error für den vollständigen Datenbestand zu messen wird der Durchschnitt der Werte berechnet, da wir an dem Erwartungswert interessiert sind. Der Faktor $\frac{1}{m}$ ergibt sich durch die zusätzliche Annahme, dass es sich um eine unabhängige und eine identisch verteilte Distribution handelt (engl. independent and identical distributed – i.i.d.):

$$MSE = -\mathbb{E}_{x \sim P_{Data}}[Squared Error] = -\frac{1}{m} \sum_{i=1}^m |y_i - y'_i|^2 \text{ mit } y'_i = f(x_i; \theta)$$

Analog lässt sich eine Error Funktion auch für eine bernoullische Verteilung herleiten:

$$Binary Cross Entropy = y \log(\phi) + (1 - y) \log(1 - \phi) \text{ mit } \phi = f(x; \theta)$$

Die „Binary Cross Entropy“- Metrik wird bei der Vorhersage von Klassenzugehörigkeiten verwendet allerdings ist eine intuitive Erklärung im Gegensatz zu der L2-Norm deutlich schwerer, doch durch die formale Herleitung sind wir in der Lage, auch für diese Problemstellung eine Metrik aufzustellen.

In der Literatur wird häufig die Error-Funktion mit \mathcal{L} abgekürzt für eine Funktion, die den Abstand misst zwischen dem geschätzten Wert $f(x; \theta)$ und dem tatsächlichen Wert y. Zusammengefasst besteht die Hauptaufgabe der Performancemessung darin, zwischen zwei verschiedenen Algorithmen den besten auswählen zu können. Die Aussagekraft der verschiedenen Metriken ist aber mehr oder weniger abstrakt und gibt nur bedingt Auskunft über die Performance des Algorithmus. Ein Algorithmus könnte trotz eines Fehlerwerts richtige Ergebnisse liefern. Beispielsweise wenn wir einen Klassifizierungsalgorithmus für Spam Mails erstellen wollen würde eine 51% Chance genügen um eine „Spam-Flag“ für eine E-Mail zu setzen. Allerdings würden die Performancemessung trotzdem einen Fehlerwert enthalten, da wir eigentlich für diese E-Mail eine hundertprozentige Genauigkeit anstreben. Deswegen ist es in der Praxis häufig sinnvoll alternative Metriken zu erstellen, die eine Aussagekraft besitzen, die für Menschen nachvollziehbar

sind. Die Marketing-Abteilung wäre beispielsweise an dem Verhältnis interessiert, das eine E-Mail eine falsch oder richtig zugeordnet wird.

2.3.2 Maximum-Likelihood Estimation

Bei der Maximum Likelihood Estimation geht es im Prinzip darum ein Set von Parametern zu finden sodass die Wahrscheinlichkeit bzw. die likelihood der Daten maximiert wird. Es wird ein Wert für θ_{ML} geschätzt und anschließend die likelihood für diesen Wert berechnet. Der Wert für θ der die likelihood maximiert ist die gesuchte Konfiguration für Parameter.

Beispielsweise wenn $\mathbb{X} = \{x_1, \dots, x_m\}$ für eine Reihe von m Datenpunkten steht sieht die Formel, um die Likelihood zu berechnen, wie folgt aus:

$$\theta_{ML} = \arg_{\theta} \max p_{Model}(\mathbb{X}; \theta) = \arg_{\theta} \max \prod_{i=1}^m p_{Model}(x_i; \theta)$$

Da es bei der Berechnung zu einem numerischen Underflow kommen kann wird anstelle der likelihood die log-likelihood maximiert. Das ist möglich, da der Wert von θ_{ML} bei beiden Funktionen identisch bleibt.²⁰

$$\theta_{ML} = \arg_{\theta} \max \sum_{i=1}^m \log(p_{Model}(x_i; \theta))$$

In der Praxis ist es oftmals nicht möglich sämtliche Werte für θ durchzuprobieren. Daher wird oftmals ein iteratives Verfahren angewendet bei dem sich die Schätzung von θ schrittweise dem Wert von θ_{ML} annähern soll.

2.3.3 Lernen

Da wir uns in den vorherigen Abschnitten mit der Performance des Algorithmus beschäftigt haben, werden wir nun dieses Wissen nutzen um den Algorithmus dementsprechend zu optimieren. Das Verhalten des Algorithmus wird bestimmt durch eine Reihe von veränderbaren Parametern θ . Die geeigneten Werte für θ ergeben sich aus der Optimierung der Performance Funktion um die log-likelihood zu maximieren. Eine Möglichkeit um diese schrittweise Optimierung durchzuführen ist das Gradientenabstiegsverfahren.²¹ Dazu wird die erste Ableitung berechnet und die Parameter θ so verändert, dass die Error Funktion reduziert wird. In diesem Fall sind die Inputs \mathbb{X} und die gewünschten Outputs \mathbb{Y} fix und die Funktion $f(x; \theta)$ wird zu einer Funktion von $f(\theta)$. Der neue Wert für θ ergibt sich durch die folgende Formel:

$$\theta \leftarrow \theta + \epsilon * \Delta_{\theta} \mathcal{L}(\mathbb{X}, \mathbb{Y}, \theta)$$

²⁰ Vgl. Goodfellow (2016), S. 146.

²¹ Vgl. Goodfellow (2016), S. 98.

ϵ entspricht dem so genannten Lernparameter. Er gibt an wie die Steigung den Parameter θ beeinflussen soll. Er wird meist vorab unveränderlich festgelegt. Die Steigung ist dabei von der Performancefunktion abhängig bzw. dem Deltawert zwischen dem vorhergesagten und dem tatsächlichen Wert mit gegebenen Parametern. Eine Möglichkeit um die Gradienten von dem neuronalen Netz von den versteckten Schichten zu berechnen ist der sogenannte Backpropagation-Algorithmus, der das Lernen von tiefen neuronalen Netzen erst möglich macht.

Der Backpropagation-Algorithmus arbeitet dabei vereinfacht ausgedrückt wie ein umgekehrtes vorwärts gerichtetes neuronales Netzwerk. Das heißt, es wird von einem Output zu einem Input propagiert und die Eingabe ist der Deltawert zwischen den observierten und der gewünschten Ausgabe. Dabei ist Backpropagation nur eine Möglichkeit um die Gradienten zu bestimmen. Die eigentliche Anpassung der Gewichte wird beispielsweise durch das Gradientenabstiegsverfahren vollzogen. Die Berechnung für die Gradienten ist dabei rekursiv, bzw. es wird der Deltawert der Outputunit berechnet und anschließend schrittweise für die einzelnen versteckten Schichten.

Der Backpropagation-Algorithmus lässt sich in fünf verschiedene Schritte einteilen die nötig sind um eine Iteration des Lernprozesses auszuführen.²² Diese Schritte werden so lange wiederholt, bis der Algorithmus konvergiert oder eine maximale Anzahl von Schritten erreicht wurde.

Die einzelnen Schritte des Backpropagation-Algorithmus sind wie folgt:

1. Zunächst wird eine vorwärts gerichtete Propagation ausgeführt um die Aktivierungen der einzelnen Schichten zu berechnen und um die Vorhersage des Wertes für y zu erhalten. In der Praxis werden diese Werte zwischengespeichert um die Werte nicht erneut berechnen zu müssen. Das beschleunigt den Rechenaufwand da die gleichen Operationen nicht erneut ausgeführt werden müssen. Es ist in diesem Fall sinnvoll Speicherplatz gegen Rechenzeit einzutauschen.
2. Es werden die Deltawerte für die letzten Neuronen in der letzten Aktivierungsschicht berechnet. Der Wert für $netinput_i$ ist dabei der berechnete Wert für das Neuron i bevor eine beliebige Aktivierungsfunktion ϕ angewendet wurde.

$$\delta_i = \phi'(netinput_i) * (y_i - y'_i)$$

3. Berechne rekursiv die Deltawerte für die einzelnen Neuronen in den versteckten Schichten. Dabei iteriert L sämtliche Neuronen zu dem das Neuron i in Beziehung steht und die Informationen des Neurons i ausgehen. W ist ein Ausschnitt der Gewichtsmatrix, der die Beziehung zwischen den Neuronen i und dem Neuron l beschreibt.

²² Vgl. Mitchell (1997), S. 98

$$\delta_i = \phi'(\text{netinput}_i) * \sum_L (\delta_l * w_{li})$$

Der Term $\sum_L (\delta_l * w_{li})$ lässt sich auch als Skalarprodukt zwischen den Deltawerten der Neuronen von der nachfolgenden Schicht und der Gewichtsmatrix auffassen. Visuell entspricht es einem Propagationsschritt allerdings in der entgegengesetzten Richtung.

4. Berechne die Veränderungen der Gewichte zwischen den Einheiten i und j:

$$\Delta W_{ij} = \delta_i * a_j$$

a_j entspricht dabei dem Aktivitätslevel des Neurons j.

5. Führe einen Schritt des Gradientenabstiegsverfahren aus und berechne die neuen Elemente der Gewichtsmatrix W:

$$W_{ij} \leftarrow W_{ij} + \epsilon * \Delta W_{ij}$$

Die Herleitung des Backpropagation-Algorithmus kann dabei über die Kettenregel der Differentialrechnung erfolgen.²³ Zur Erinnerung wir berechnen eine vorläufige Ausgabe mit der Formel $z_i = \sum_l W_{il} * a_l$ und wenden anschließend eine beliebige Aktivierungsfunktion ϕ an und erhalten einen Wert, der für die neue Schicht weitergereicht wird $a_l = \phi(z_l)$. Als Performancefunktion und zur Bestimmung des Fehlerterms benutzen wir die „Squared Error“-Funktion (E) für ein einziges Trainingsbeispiel.

Wir suchen die Ableitung für die jeweiligen Gewichte W_{il} . Formal ausgedrückt:

$$\frac{\delta E}{\delta W_{il}} = \frac{\delta E}{\delta z_i} * \frac{\delta z_i}{\delta W_{il}} = \delta_i * a_l$$

Der Wert für $\frac{\delta E}{\delta z_i} = \delta_i$ ist zu diesem Zeitpunkt unbekannt und stellt eine gesuchte Größe dar. Die Ableitung für $\frac{\delta z_i}{\delta W_{il}} = a_l$ lässt sich anhand obiger Formel ablesen. Wie bereits oben beschrieben ist der Wert für a_l der zwischengespeicherte Wert, der bei der vorwärts gerichteten Propagation berechnet wurde.

Für $\delta_i = \frac{\delta E}{\delta z_i}$ existieren zwei verschiedene Fälle. Entweder handelt es sich um ein Ausgabeneuron oder ein Neuron innerhalb einer verdeckten Schicht. Im ersten Fall handelt es sich um ein Ausgabeneuron und die Vorhersage lässt sich anhand dieses Neurons ablesen. Wir gehen vereinfacht davon aus, dass es sich um eine lineare Aktivierung handelt.

Es wird der Deltawert δ_k wie folgt berechnet:

$$\delta_k = \frac{\delta E}{\delta z_k} = \frac{\delta E}{\delta y'} * \frac{\delta y'}{\delta z_k} = \frac{\delta E}{\delta z_k} = -2 * (y - y') * \frac{\delta y'}{\delta z_k} = -2 * (y - y')$$

²³ Vgl. Mitchell (1997), S. 101 ff.

Im zweiten Fall handelt es sich um ein Neuron in einer versteckten Schicht und wir können auf einen Deltawert δ_j der nachfolgenden Schichten zugreifen. Wie bereits oben beschrieben berechnen wir bei Backpropagation in entgegengesetzter Richtung. Es ist also zwingend erforderlich δ_j zu berechnen bevor wir δ_i berechnen können.

Die Deltawerte der verdeckten Schichten lassen sich wie folgt berechnen:

$$\delta_i = \frac{\delta E}{\delta z_i} = \sum_j \frac{\delta E}{\delta z_j} * \frac{\delta z_j}{\delta z_i} = \sum_j \delta_j * \frac{\delta z_j}{\delta z_i}$$

$$\frac{\delta z_j}{\delta z_i} = \frac{\delta z_j}{\delta a_i} * \frac{\delta a_i}{\delta z_i} = W_{ji} * \phi'(z_i)$$

Werden die einzelnen Formeln ineinander eingesetzt entstehen die Formeln der jeweiligen Schritte wie weiter oben beschrieben.

Prinzipiell lässt sich der Backpropagation Algorithmus auf eine Vielzahl an Modellen anwenden um die Gradienten zu bestimmen. Es bietet sich dabei an einen sogenannten Berechnungsgraphen zu definieren in dem die einzelnen Mathematischen Operationen als Knotenpunkte aufgefasst werden der die Beziehungen zwischen anderen Elementen mithilfe von Kanten beschreibt.²⁴ Aufgrund dieser Flexibilität für verschiedene Problemstellungen und der Einsatzmöglichkeiten für verschiedene Modelle bieten entsprechende Frameworks für das Maschinelle Lernen, wie beispielsweise TensorFlow, die Möglichkeit an einen Graphen zu definieren der automatisch diese Formeln berechnet.

Das Lernen für neuronale Netze erweist sich in der Praxis als Problematisch da das Gradientenabstiegsverfahren ein lokales und kein globales Minimum anstrebt. Ein Forschungsgebiet des maschinellen Lernens beschäftigt sich mit dem entwickeln von Verfahren, die diese lokalen Minima überspringt und hoffentlich ein besseres lokales Minimum anstreben. Häufig eingesetzte Verfahren in der Praxis sind beispielsweise Lernparameter die sich mit der Zeit verringern oder auch Verfahren die das Momentum des Gradienten nutzen sowie Verfahren die eine statistische Verteilungen der Aktivierungen der Neuronen betrachten.

Graphisch kann das folgendermaßen veranschaulicht werden:

²⁴ Vgl. Goodfellow (2016), S. 205.

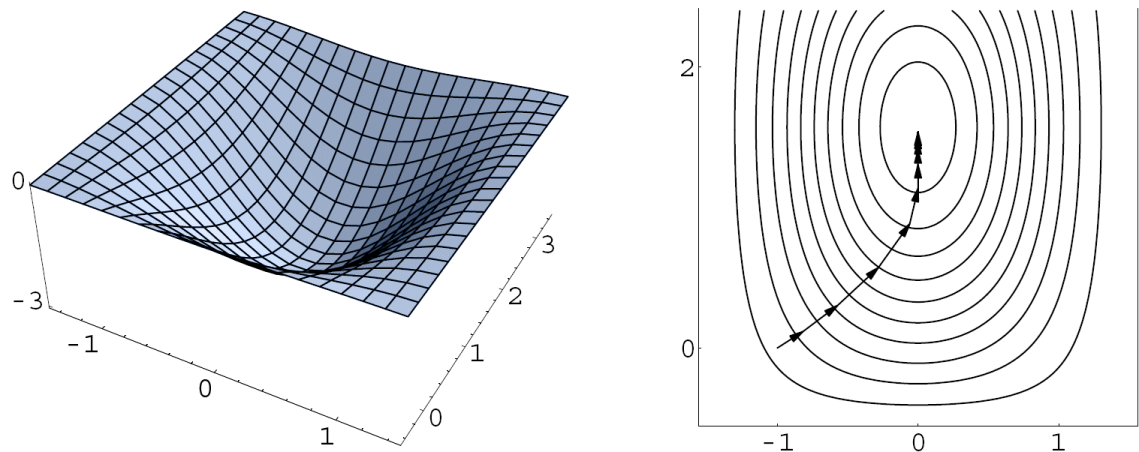


Abbildung 7: Gradientenabstiegsverfahren

Quelle: Kriesel (2007), S. 62.

Die Oberfläche zeigt die Bereiche, an denen die Performancefunktion groß bzw. klein ist. Der schwarze Strich zeigt die einzelnen Stationen des Gradientenabstiegsverfahrens an bzw. die Verschiebung der Position durch die Anpassung von Parameter θ . In dieser Abbildung wird der Algorithmus lediglich von zwei Parametern charakterisiert um die Vorgehensweise des Gradientenabstiegsverfahrens zu veranschaulichen. In der Praxis bestehen die Algorithmen aus deutlich mehr Parametern und die Performancefunktion bietet eine Vielzahl an lokalen Optima. In der Regel handelt es sich um eine nicht konvexe Optimierungsproblem. Die Gründe dafür im Fall von Neuronalen Netzen sind vielfältig.²⁵ Beispielsweise durch eine sogenannte Gewichts-Symmetrie, da die einzelnen Gewichte der Neuronen innerhalb einer Schicht ausgetauscht werden können, da die Reihenfolge der Neuronen keine Rolle spielt. Ein weiteres Optimierungsproblem sind Sattelpunkte. Mathematisch lässt es sich zeigen, dass das Verhältnis zwischen Sattelpunkten und lokalen Optima in einem hochdimensionalen Raum exponentiell wächst. Es ist wahrscheinlicher an einen Sattelpunkt zu geraten als einem lokalen Optimum. Prinzipiell sind Sattelpunkt schlecht da die Gradienten immer kleiner werden welches den Trainingsprozess verlangsamt allerdings scheint das Gradientenabstiegsverfahren, empirisch betrachtet, die Fähigkeit zu besitzen diese Sattelpunkte zu überwinden.

Die Oberfläche der Performancefunktion analysieren ist ein Forschungsgebiet im Bereich des maschinellen Lernens indem noch aktiv Forschung betrieben wird. Ein besseres Verständnis dieser erlaubt das bessere Designen der Architektur, dem entwickeln neuer Aktivierungsfunktionen und der Erstellung von Verfahren die schneller konvergieren u.v.m.

²⁵ Vgl. Goodfellow (2016), S. 284 ff.

3 Inferenz

Unter Inferenz (lat. „Inferre“) wird das Schlussfolgern von Sachverhalten verstanden. Im Wesentlichen geht es dabei zu überprüfen welche Attribute gegeben sind und welche Informationen sich aus diesen Attributen herleiten lassen. Beispielsweise wenn wir wissen, dass jeder lebende Mensch atmet und wenn wir wissen das Sokrates lebt sowie ein Mensch ist wäre eine Schlussfolgerung das Sokrates atmen würde. Diese spezielle Form der Schlussfolgerung wird auch Implikation genannt. Die Mathematik ist aufgebaut auf dem Schlussfolgern von Grundannahmen bzw. Axiomen um neues Wissen zu generieren das zum weiteren Schlussfolgern verwendet wird. Im Bereich der Mustererkennung und des unüberwachten Lernens wird der Inferenzbegriff auf eine ähnliche Weise verwendet. Es geht darum, Datensätze zu analysieren um Informationen zu finden die für die weitere Verwendung nützlich sind. Beispielsweise ist der Kunde Kreditwürdig? In wie viele Zielgruppen lassen sich unsere Kunden einteilen? Zu welcher Zielgruppe gehört der neue Kunde?

Diese erste Art von Fragestellung lässt sich durch das observieren von vergangenen Ereignissen beantworten. Es lässt die Wahrscheinlichkeit ermitteln, mit der die Aussage zutrifft bzw. nicht zutrifft. Die zweite Art von Fragestellung befasst sich mit dem erfassen einer verdeckten Struktur. Gibt es Gemeinsamkeiten zwischen unseren Kunden und welche wären das? Die dritte Fragestellung ist ähnlich unserer Problemstellung des vergangenen Kapitels. Allerdings ist diese Zielgruppe unbekannt bzw. mit der Zugehörigkeit des Kunden zur Gruppe verändert sich die Gruppe. Jedoch kann jede dieser Fragestellungen als Inferenzproblem aufgefasst werden.

In diesem Kapitel werden wir uns überwiegend mit Variational Inferenz beschäftigen, die auf diese Art von Problemstellungen angewendet werden kann. Wir werden uns zunächst mit dem Bayes Theorem vertraut machen mit der Inferenz durchgeführt werden kann sowie dem Problem mit dieser Methode. Anschließend werden wir im Detail den Expectation-Maximization-Algorithmus der das Fundament für Variational Inferenz bildet mit der wir dieses Kapitel abschließen.

3.1 Bayes Theorem

Das Bayes Theorem spielt eine zentrale Rolle im Bereich der Mustererkennung. Es wird auch häufig als „umgekehrte Wahrscheinlichkeit“ bezeichnet. In dem vergangenen Kapitel konnten wir beispielsweise die bedingte Wahrscheinlichkeit von y mit einem gegebenen x Berechnen. Das Bayes Theorem dreht nun die Fragestellung um und fragt: „Was ist die bedingte Wahrscheinlichkeit

von x mit einem gegebenen y ". Die Bayes Formel lässt sich durch die Umkehrung der Produktregel ($P(A, B) = P(A|B)P(B)$) einfach zeigen:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

In der Literatur haben die einzelnen Formelelemente bestimmte Bezeichnungen. Der Term $P(A|B)$ wird beispielsweise diesem Zusammenhang als Posterior bezeichnet. $P(B|A)$ ist die likelihood die mit dem Prior $P(A)$ multipliziert wird und anschließend durch die evidence $P(B)$ geteilt wird. Die evidence fungiert als Normalisierungskonstante und ist die Summe über sämtliche Konfigurationen von A . Sie dient dazu valide Wahrscheinlichkeitswerte zu erhalten.

Die evidence zu berechnen kann sich in der Praxis als zu komplex erweisen. Beispielsweise wenn wir erfassen möchten ob es sich bei dem Bild B um eine Katze A handelt, wäre die gegensätzliche Fragestellung ob es sich bei der Katze A um Bild B handelt. Diese Fragestellung ist deutlich kniffliger da A prinzipiell auf viele Bilder zeigt, die diese Eigenschaft erfüllen. Die Dimensionsunterschiede sind entscheidend da die evidence über sämtliche Konfigurationen berechnet wird und oftmals ist die Vertauschung der Fragestellungen ein guter erster Schritt zur Lösung des Problems ist, je nachdem ob das eine oder andere einfacher zu berechnen ist.

Eine Möglichkeit um eine Intuition über das Bayes Theorem erhalten ist durch das studieren eines Beispiels²⁶. Angenommen wir wären von unserem Arzt auf eine gefährliche Krankheit getestet und dieser Test wäre positiv ausgefallen. Der Arzt gibt den Hinweis das die Genauigkeit des Tests bei 99% liegt. Wie besorgt wären wir? Anhand von Statistiken erfahren wir das ca. 1 von 1 000 Menschen betroffen sind. Eingesetzt in die Formel ergibt sich:

$$\begin{aligned} P(Krank|Test_p) &= \frac{P(Test_p|Krank)P(Krank)}{P(Test_p|-Krank)P(-Krank) + P(Test_p|Krank)P(Krank)} \\ &= \frac{0.99 * 0.001}{0.99 * 0.001 + 0.01 * 0.999} \approx 9.01\% \end{aligned}$$

Laut Bayes Theorem besteht eine 9.01% Chance das wir eine ernsthafte Krankheit haben. Im Verhältnis zu 99% eine deutliche Besserung. Der Schlüssel dieser Verzerrung liegt bei der zusätzlichen Information zu der Auftretungswahrscheinlichkeit. Beispielsweise haben wir bei obigen Zahlen circa 10 Falschklassifizierungen. Die Chance, dass wir die Falschklassifizierung sind oder der Test wahrhaftig positiv ist drückt das Bayes Theorem in diesem Beispiel aus.

²⁶ Vgl. MacKay (2005), S. 25.

3.2 Inferenzmethoden

In dem vergangenen Beispiel konnten wir eine Inferenz durchführen, indem wir durch sämtliche verschiedene Fälle iteriert haben. In diesem Fall wird von einer vollständigen Enumeration gesprochen. Allerdings existieren noch andere Methoden²⁷ um Inferenz durchzuführen. In der Literatur haben sich folgende Verfahren etabliert:

- Komplette Enumeration
- Laplace Methoden
- Monte Carlo Methoden
- Variational Inferenz

Wir werden uns nur mit der kompletten Enumeration sowie der Variational Inferenz befassen. Es sollte allerdings erwähnt werden, dass es auch andere Methoden existieren die entsprechende Vor- bzw. Nachteile haben um das Inferenz Problem zu lösen.

Die komplette Enumeration bezieht sich darauf, dass wir sämtliche Teile der Gleichung des Bayes Theorem haben, oder diese Berechnung durchführen können. Der Vorteil ist das wir exakte Ergebnisse erhalten allerdings durch einen oftmals sehr hohen Rechenaufwand. Dieser Rechenaufwand ist teilweise so hoch das er in einer nicht annehmbaren Zeit durchführbar ist. Der Grund ist häufig die evidence, da sie über sämtliche Konfigurationen berechnet werden muss. Das heißt, dass alle Faktoren ausgegrenzt werden müssen um die evidence exakt berechnen zu können. In dem Fall von diskreten Variablen bedeutet dies eine Reihe von Summen um sämtliche verschiedene Features miteinander zu kombinieren.

$$P(B) = \sum_A P(B, A) = \sum_{A_1} \sum_{A_2} \dots \sum_{A_m} P(B, A_1, \dots A_m)$$

Die Rechendauer wächst exponentiell, da für jede weitere Dimension für A die „for-Schleife“ erneut durchlaufen werden muss. Eine Folgerung ist, dass sich die evidence nicht mehr berechnen lässt und sich keine exakte Inferenz mehr durchführen lässt. Dieses Problem wird auch als der Fluch der Dimensionen bezeichnet, der sich auch auf Anzahl der Daten bezieht die nötig sind um eine möglichst genaue Inferenz durchzuführen.²⁸

Eine Möglichkeit um dieses Problem zu lösen sind sogenannte approximierte Methoden anzuwenden. Variational Inferenz gehört zu dieser Rubrik und wir werden diese anhand des Expectation-Maximization-Algorithmus herleiten.

²⁷ Vgl. MacKay (2005), S. 294.

²⁸ Vgl. Bishop (2006), S. 33 ff.

3.3 Expectation-Maximization-Algorithmus

Der Expectation-Maximization-Algorithmus (EM-Algorithmus) ist ein Algorithmus, der sehr häufig im Bereich der Mustererkennung und im Data-Mining verwendet wird. Er gilt als Grundlage für variationale Bayesian Inferenz (VI). Der EM Algorithmus lässt sich vereinfacht ausgedrückt als Zyklus auffassen in dem zwei Parameter gesucht sind. Durch das abwechselnde fixieren und anpassen der Parameter konvergiert der Algorithmus zu einem lokalen Optimum. Wir werden zwei verschiedene Einsatzmöglichkeiten betrachten. Zunächst der k-means Algorithmus und anschließend den EM-Algorithmus für eine Gaußsche Mischverteilung.

3.3.1 K-Means als EM-Algorithmus

In der Literatur wird der Algorithmus gerne im Zusammenhang mit dem k-Means Algorithmus eingeführt, bei dem Schrittweise die Clusterzugehörigkeiten ermittelt werden und die Mittelpunkte verschoben werden.²⁹ Dieser Prozess wiederholt sich solange bis keine neuen Clusterzuweisungen mehr durchgeführt werden. Diese zwei Schritte können als einen Erwartungsschritt und einen Optimierungsschritt aufgefasst werden.

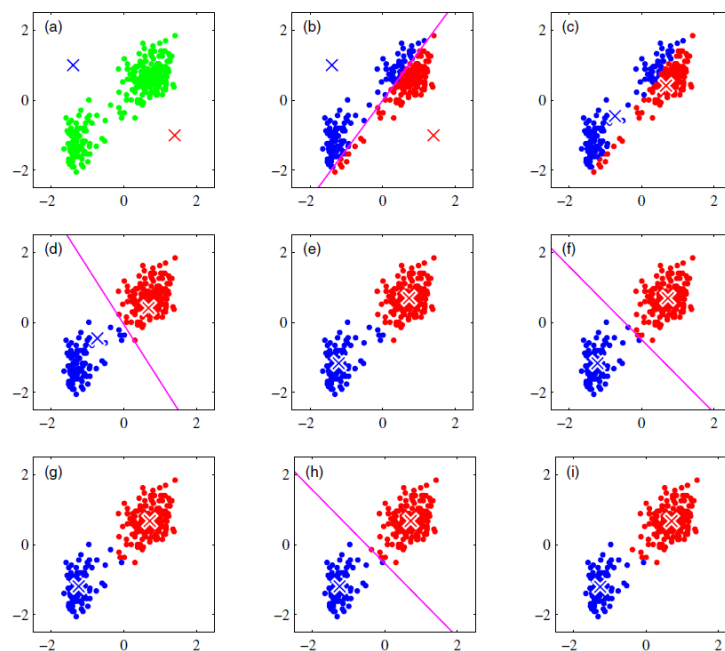


Abbildung 8: K-Means Algorithmus

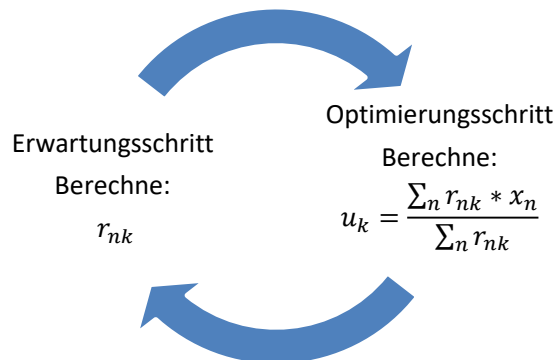
Quelle: Bishop (2006), S. 426.

Die Abbildung zeigt den Verlauf der Clusterbildung und jedes Bild beschreibt dabei einen Zyklus in dem EM Algorithmus Schritt. Formal haben wir eine Performancefunktion, die den Abstand der N Datensätze zu dem zugeordneten Cluster misst und eine Anzahl an K Clustern deren

²⁹ Vgl. Bishop (2006), S. 424 ff.

Mittelwert μ_k den Cluster Ursprung darstellt. Die Zugehörigkeit sowie die Clustermittelpunkte sind vorab unbekannt und stellen die gesuchten Werte des Algorithmus dar. Formal lässt sich das folgendermaßen ausdrücken. Dabei stellt r_{nk} die Zugehörigkeit von dem Datensatz n zu dem Cluster k dar.

$$r_{nk} = \begin{cases} 1 & \text{wenn } k = \arg \min_j |x_n - \mu_j|^2 \\ 0 & \text{andernfalls} \end{cases}$$



Der Erwartungsschritt ist die Berechnung der Clusterzugehörigkeit und der Optimierungsschritt ist die Verschiebung der Clustermittelpunkte bzw. die Berechnung des Durchschnitts (Mean) der Datensätze. Es ist besonders hervorzuheben, dass der EM Algorithmus und VI Methoden von der Initialisierung der Startpunkte abhängen. Beispielsweise könnte bei einer schlechten Initialisierung ein Cluster statistische Ausreißer darstellen wohingegen die anderen Cluster die gesamten üblichen Werte abdecken. Dadurch, dass die Initialisierung Einfluss auf das Resultat nimmt bedeutet dies das nicht ein globales, sondern ein lokales Optimum gefunden wurde. In der Praxis ist es üblich mehrere Durchläufe mit unterschiedlichen Initialisierungen durchzuführen. Die Ergebnisse werden mithilfe einer Performance Funktion, wie die Summe der Distanzen zu den jeweiligen Clustern, miteinander verglichen und das beste Clustering ausgewählt.

3.3.2 EM-Algorithmus für ein mehrdimensionales Verteilungsmodell

Das gleiche Prinzip lässt sich auch für eine Gaußsche Mischverteilung anwenden bei der die einzelnen Verteilungen verantwortlich für die Generierung des jeweiligen Datenpunkts sind.³⁰ Dabei wird anstelle einer harten Clusterzuweisung eine weiche Clusterzuweisung gefunden, die eine Wahrscheinlichkeit angibt, mit dem der Datenpunkt zu dieser Verteilung gehört bzw. generiert wurde. Im Gegensatz zu k-Means sind die einzelnen r_{nk} Wahrscheinlichkeiten.

$$r_{nk} = P(k|x_n) = \frac{P(x_n, k)}{P(x_n)} = \frac{P(k)N(x_n|\mu_k, \Sigma_k)}{\sum_{i=1}^K P(k_i)N(x_n|\mu_i, \Sigma_i)} \text{ mit } P(k) = \frac{N_k}{N} = \frac{\sum_{n=1}^N r_{nk}}{N}$$

³⁰ Vgl. Bishop (2006), S. 439 f.

In jeder Iteration des Zyklus werden μ und Σ für jeden Cluster erneut berechnet und die Verteilungen verschieben sich entsprechend. Eine wesentliche Änderung zu dem k-Means Algorithmus ist, dass die Kovarianzmatrix Σ verwendet wird, mit der es ermöglicht wird, komplexere Clusterstrukturen zu finden. Ein Problem des k-Means Algorithmus ist, dass die Distanzen zwischen den einzelnen Attributen mit dem gleichen Gewicht in die Berechnung eingehen. Die Gaußsche Mischverteilung in Verbindung mit dem EM Algorithmus löst elegant dieses Problem indem die Wahrscheinlichkeiten, dass dieser Datenpunkt zu dem Cluster gehört in die Berechnung einkalkuliert wird. Die neuen Werte für μ und Σ werden wie folgt berechnet:

$$\mu'_k = \frac{\sum_n^N r_{nk} * x_n}{\sum_n^N r_{nk}}$$

$$\Sigma'_{kjk} = \frac{1}{N_k} \sum_{i=1}^N r_{nk} * (x_j^{(i)} - \mu_{kj})(x_k^{(i)} - \mu_{kk})$$

μ_k ist der Mittelwert von dem Cluster k und μ_{k_i} ist das i'te Element von dem Vektor μ_k . Analog ist $x_j^{(i)}$ von dem i'te Datensatz das Attribut j.

Die folgende Abbildung zeigt ein beispielhaftes Clustering mit zwei Gaußschen Mischverteilungen.

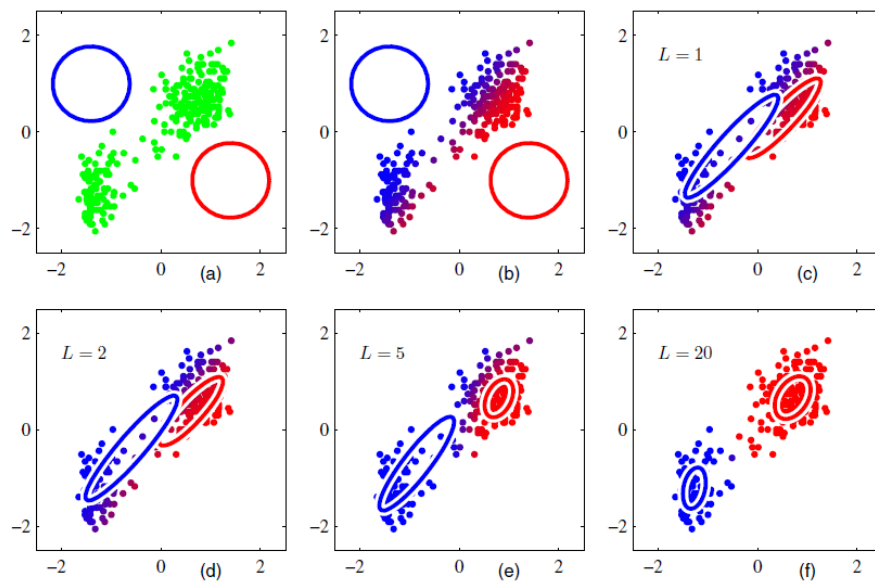


Abbildung 9: Clustering mit einer Gaußschen Mischverteilung

Quelle: Bishop (2006), S. 437.

3.3.3 Herleitung der Variational Inferenz

Wie zum Anfang beschrieben stellt der Expectation-Maximization Algorithmus das Herz von Variational Inferenz Methoden dar. Wir wollen uns daher noch das generelle Konzept betrachten sowie die Herleitung.³¹

Für die Herleitung führen wir eine Funktion Dichtefunktion $q(z)$ ein, die sich dem wahren optimalen Posterior $P(z|x)$ annähern soll. Diese Dichtefunktion stammt aus einer Familie an Dichtefunktionen D , die sich als der Suchraum herausstellen und wird vorab festgelegt. Wie wir sehen werden bildet $q(z)$ eine untere Schranke für den wahren Posterior $P(z|x)$. Um die Abweichung zwischen den beiden Distributionen zu ermitteln benutzen wir die Kullback-Leibler-Abweichung (KL). Für sie gilt $KL \geq 0$ und die KL-Abweichung ist dann und nur dann gleich null, wenn die beiden Distributionen gleich sind.

$$q^*(z) = \arg_{q(z) \in D} \min KL(q(z) || P(z|x))$$

Per Definition der KL-Abweichung ergibt sich:

$$KL(q(z) || P(z|x)) = \mathbb{E} \left[\log \left(\frac{q(z)}{P(z|x)} \right) \right] = \mathbb{E}[\log(q(z))] - \mathbb{E}[\log P(z|x)]$$

Durch Umformung der konditionalen Wahrscheinlichkeit und der Tatsache das $\mathbb{E}[\log(P(x))] = \log P(x)$ ist, da sie unabhängig von dem Erwartungswert von z ist ergibt sich:

$$\begin{aligned} KL(q(z) || P(z|x)) &= \mathbb{E}[\log(q(z))] - \mathbb{E} \left[\log \left(\frac{P(z, x)}{P(x)} \right) \right] \\ &= \mathbb{E}[\log(q(z))] - \mathbb{E}[\log(P(z, x))] + \log P(x) \end{aligned}$$

Es ergibt sich folgende Gleichung:

$$\begin{aligned} KL(q(z) || P(z|x)) &= \mathbb{E}[\log(q(z))] - \mathbb{E}[\log(P(z, x))] + \log P(x) \\ \log P(x) &= \mathbb{E}[\log(P(z, x))] - \mathbb{E}[\log(q(z))] + KL(q(z) || P(z|x)) \end{aligned}$$

Unser Ziel ist es den Term $\log P(x) = \log(\sum_z P(x, z))$ zu maximieren. Dadurch, dass es sich um eine Summe innerhalb eines Logarithmus handelt kann die Maximum Likelihood Estimation nicht exakt durchgeführt und wir haben keinen Zugriff auf eine geschlossene Lösung mehr.

Wir können allerdings die beiden anderen Terme benutzen um $\log P(x)$ trotzdem zu optimieren. Sie bilden den sogenannten „Evidence Lower Bound“ (ELBO) und die KL-Abweichung zwischen unserer Zielfunktion $q(z)$ und $P(z|x)$. Da die KL-Abweichung nicht negativ sein kann und $\log P(x)$ eine Konstante darstellt gilt $ELBO \leq \log P(x)$. Wie gut unsere Approximation ist lässt sich also anhand der Formel ablesen:

$$\log P(x) = ELBO - KL$$

³¹ Vgl. Blei (2018), S. 6 f.

Durch Maximierung des ELBO wird die KL-Abweichung immer geringer und nähert sich immer weiter dem Wert von $\log P(x)$ an. Der ELBO dient in dem Sinne als Proxy für unsere eigene Optimierungsfunktion. Durch weitere Umformungen entsteht:

$$\begin{aligned} ELBO &= \mathbb{E}[\log(P(z, x))] - \mathbb{E}[\log(q(z))] = \mathbb{E}\left[\log\left(\frac{P(z, x)}{q(z)}\right)\right] \\ &= \mathbb{E}[\log(P(z))] + \mathbb{E}[\log(P(x|z))] - \mathbb{E}[\log(q(z))] \\ &= \mathbb{E}[\log(P(x|z))] - KL(q(z) || P(z)) \end{aligned}$$

Es ergibt sich die log-likelihood von $P(x|z)$ und die KL-Abweichung von unserem gewählten Prior und dem tatsächlichen Prior. Daran lässt sich schon ein Problem festmachen, nämlich dass sich unser Suchraum nur auf eine Dichtefunktionsfamilie beschränkt. Eine Folge dessen ist, dass wir in der Praxis meist nie exakte Ergebnisse erhalten werden. Die KL-Abweichung bestraft in dem Sinne unsere ursprüngliche Annahme, mit der wir die Daten abbilden wollen, wenn sich diese Daten nicht, oder nicht gut, mit der Annahme abbilden lassen. Es ist in dem Sinne eine Beschränkung, dass wir verletzen können allerdings nur zu einem Preis.

Die Parallele zu dem EM Algorithmus ist, dass wir zwei Parameter haben. Einen neuen gesuchten Parameter θ^{new} und einen alten Parameter θ^{old} die unsere Dichtefunktion beschreibt.³² Wenn wir für $q(z) = P(z|x, \theta^{old})$ einsetzen bzw. unserem aktuell besten Model können wir auf folgende Gleichung kommen:

$$\begin{aligned} ELBO &= \mathbb{E}\left[\log\left(\frac{P(z, x)}{q(z)}\right)\right] = \sum_z q(z) * \log \frac{P(z, x)}{q(z)} = \sum_z P(z|x, \theta^{old}) * \log \frac{P(z, x)}{P(z|x, \theta^{old})} \\ &= \sum_z P(z|x, \theta^{old}) * \log P(z, x) - \sum_z P(z|x, \theta^{old}) * \log P(z|x, \theta^{old}) \\ &= \sum_z P(z|x, \theta^{old}) * \log P(z, x) + Const \end{aligned}$$

Wenn wir für die gemeinsame Wahrscheinlichkeit einen neuen Parameter einführen θ können wir eine neue Funktion erstellen die sich wie folgt ergibt:

$$Q(\theta, \theta^{old}) = \sum_z P(z|x, \theta^{old}) * \log P(z, x|\theta)$$

Für diese neue Funktion können wir die Maximum Likelihood Estimation anwenden und den neuen Parameter θ^{new} bestimmen der im nächsten Zyklus den Wert für θ^{old} darstellt.

$$\theta^{new} = \arg_{\theta} \max Q(\theta, \theta^{old})$$

Die Optimierung erfolgt dabei über $\log P(z, x|\theta)$. Es lässt sich zeigen, dass wenn dieser Term nach dem entsprechenden Parameter abgeleitet wird und die Nullstelle berechnet wird die Formeln von dem vergangenen Kapitel entstehen. Im Falle von einer Normalverteilung handelt es

³² Vgl. Bishop (2006), S. 452.

sich um eine Parabel, die nach unten geöffnet ist. Es existiert somit eine Extremstelle, die das Optimum darstellt. Die neue Parameterkonfiguration für θ stellt in dem neuen Zyklus den Wert für θ^{old} da. Die Funktion $Q(\theta, \theta^{old})$ lässt sich also so interpretieren, dass die Funktion die jeweils letzten Elemente der Reihe $\theta_1, \theta_2, \dots, \theta_n$ enthält.

Grafisch kann der EM Algorithmus folgendermaßen aufgefasst werden:

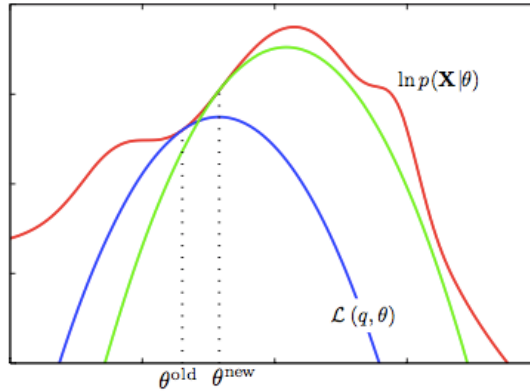


Abbildung 10: Variational Inferenz

Quelle: Bishop (2006), S. 453.

3.4 Variational Inferenz

Bisher haben wir uns mit dem Expectation-Maximization-Algorithmus vertraut gemacht sowie deren Herleitung. Wir haben für unsere Funktion mit dem Posterior gleichgesetzt $q(z) = P(z|x, \theta^{old})$ um das generelle Framework der Variational Inferenz kennenzulernen. Es ist allerdings nicht zwingend notwendig diesen Ansatz zu folgen um das Inferenz Problem zu lösen. Wir wollen $q(z)$ so wählen, dass sich die gemeinsame Wahrscheinlichkeit $\log P(z, x)$ möglichst einfach berechnen lässt allerdings sollte die Funktion flexibel genug sein um eben genau zu approximieren. Häufig wird $q(z)$ dabei aus einer Familie von Dichtefunktionen ausgewählt deren Elemente disjunkt sind bzw. die folgendermaßen beschränkt ist:

$$q(z) = \prod_i q_i(z_i)$$

Mit anderen Worten die Elemente sollen unabhängig voneinander sein. Jede latente Variable wird von einer eigenen Distribution abgedeckt, die untereinander keine Abhängigkeiten haben. Dieser Ansatz wird auch als „Mean field“-Ansatz bezeichnet und gehört zu den einfacheren Varianten von Variational Inferenz. Wissenschaftler haben auch andere Dichtefunktionsfamilien untersucht die beispielsweise Abhängigkeiten untereinander aufweisen. Dieser Ansatz wird auch als strukturierte variational Inferenz bezeichnet.³³ Dieser Ansatz würde Allerdings an dieser

³³ Vgl. Blei (2018), S. 8.

Stelle zu weit führen. Das Model kann Aufgrund einer zu flexiblen Distribution nicht an einer Überanpassung leiden. Die Approximation kann dadurch nur noch genauer werden.³⁴

Wie wir gesehen haben geht es bei VI darum den ELBO zu maximieren. Wenn wir in die Gleichung $q(z) = \prod_i q_i(z_i)$ erhalten wir folgendes:

$$\begin{aligned} ELBO(q) &= \mathbb{E}[\log P(z, x)] - \mathbb{E}[\log q(z)] = \mathbb{E}[\log P(z, x) - \log q(z)] \\ &= \sum_z \prod_i q_i \left[\log P(z, x) - \log \left(\prod_i q_i \right) \right] = \sum_z \prod_i q_i \left[\log P(z, x) - \sum_i \log q_i \right] \end{aligned}$$

Ein großer Vorteil, wenn die einzelnen Distributionen unabhängig sind, ist das die Gleichungen nach genau dieser Distribution umgestellt werden können. Die folgenden Gleichungen isolieren eine einzelne Distribution q_j die stellvertretend für eine Komponente steht. Nach jeweils einer dieser Komponente wird zyklisch optimiert. Bei der Notation steht $\sum_{z_{i \neq j}}$ für eine Reihe über Summen bzw. $\sum_{z_1} \dots \sum_{z_m}$ jedoch ohne \sum_{z_j} . Die Anzahl der latenten Variablen bestimmt die Anzahl der Sigmas und die Anzahl der Distributionen.

$$\begin{aligned} &= \sum_{z_j} q_j \left[\sum_{z_{i \neq j}} \log P(z, x) \prod_{i \neq j} q_i \right] - \sum_z \prod_i q_i \sum_i \log q_i \\ &= \sum_{z_j} q_j \left[\sum_{z_{i \neq j}} \log P(z, x) \prod_{i \neq j} q_i \right] - \sum_z q_j \prod_{i \neq j} q_i [\log q_j + \sum_{z_{i \neq j}} \log q_i] \\ &= \sum_{z_j} q_j \left[\sum_{z_{i \neq j}} \log P(z, x) \prod_{i \neq j} q_i \right] - \sum_{z_j} q_j \log q_j \underbrace{\sum_{z_{i \neq j}} \prod_{i \neq j} q_i}_{\sum_x P(x) = 1} - \underbrace{\sum_{z_j} q_j}_{\sum_x P(x) = 1} \underbrace{\sum_{z_{i \neq j}} \prod_{i \neq j} q_i \sum_{z_{i \neq j}} \log q_i}_{const} \\ &= \sum_{z_j} q_j \left[\sum_{z_{i \neq j}} \log P(z, x) \prod_{i \neq j} q_i \right] - \sum_{z_j} q_j \log q_j - const \end{aligned}$$

Die Gleichung kann als KL-Abweichung zwischen $P^\sim(x, z) = \mathbb{E}_{i \neq j}[\log P(z, x)]$ und q_j interpretiert werden. Die Maximierung des ELBO ist gleichzusetzen mit der Minimierung der negativen KL-Abweichung und es ergibt sich folgendes:

$$\log q_j^*(z_j) = \mathbb{E}_{i \neq j}[\log P(z, x)] + const$$

Jeder Term der unabhängig von der Zufallsvariable z_j ist gilt als Konstante und ist unabhängig von der Optimierung. Bei einem Optimierungsschritt wird eine Distribution angepasst und die anderen werden fixiert bzw. eingefroren. Das ist vergleichbar mit dem EM Algorithmus allerdings werden die Distributionen einzeln und nicht zusammen optimiert. Dadurch ist prinzipiell die VI Methode schneller als der EM Algorithmus. Das erlaubt es VI Methoden auch auf große

³⁴ Vgl. Bishop (2006), S. 464.

Datenbestände anzuwenden. Ein Anwendungsgebiet ist beispielsweise Topic Modeling, bei dem Dokumente, wie z.B. Nachrichtenfeeds, analysiert und geclustert werden.

Ein Nachteil ist, dass VI Methoden über die KL-Abweichung optimiert werden. Ein Charakteristikum dieser Methode ist, dass sie asymmetrisch ist und wir verschiedene Ergebnisse erhalten, wenn wir die Terme vertauschen. Eine weitere charakteristische Eigenart ist, dass die KL-Abweichung, z.B. von q und p , versucht die Wahrscheinlichkeitsmasse von q dorthin zu verlagern wo die Wahrscheinlichkeit von p hoch ist. Dieses Verhalten kann auch anhand der Formel inspiziert werden.

$$D_{KL}(q, p) = \sum_x q(x) \log \frac{q(x)}{p(x)}$$

Die vier Fälle in denen jeweils q bzw. p groß bzw. klein ist:

q	p	$D_{KL}(q, p)$
Klein	Klein	Klein
Klein	Groß	Klein
Groß	Klein	Groß
Groß	Groß	Klein

Grafisch kann dieses Verhalten folgendermaßen interpretiert werden. Die Approximation q (roter Kreis) versucht Bereiche zu vermeiden in denen p klein ist. Auf der rechten Seite ist der Sachverhalt umgekehrt.³⁵ Beide Szenarien sind nur bedingt optimal, da Bereiche ein- bzw. ausgegrenzt werden.

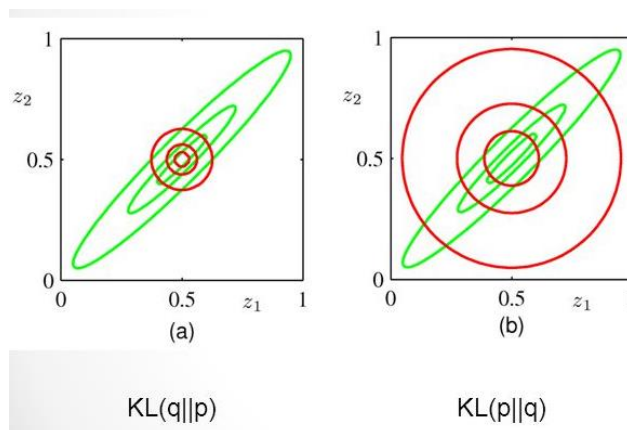


Abbildung 11 KL-Abweichung und negative KL-Abweichung

Quelle: https://images.slideplayer.com/15/4703655/slides/slide_21.jpg (oder bishop)

Eine Folge des „Mean field“-Ansatzes ist das keine Korrelation zwischen den Zufallsvariablen abgebildet werden können. Die Abbildung auf der rechten Seite zeigt das Ergebnis einer negativen KL-Abweichung bzw. wenn die Terme vertauscht sind.

³⁵ Vgl. Bishop (2006), S. 468.

Ein weiterer Nachteil ist, dass die Initialisierung entscheidend ist und die VI Methode nur einzelne Modale erfassen kann. Je nachdem welches Modal erfasst wurde unterscheiden sich die Resultate. Der Algorithmus konvergiert, allerdings zu einem lokalen Optimum.

Es sollte noch erwähnt werden, dass es andere Methoden gibt um das Inferenz Problem zu lösen. Beispielsweise bietet die Monte Carlo Markov Chain Methode (MCMC) eine Alternative. MCMC Methoden sind exakter aber sind auch langsamer im Vergleich mit VI Methoden. Die VI Methoden lassen sich also einfacher auf große Datenbestände anzuwenden.³⁶

4 Statistisches Maschinelles Lernen

In den vergangenen Kapiteln haben wir uns mit den Grundzügen des Maschinellen Lernens vertraut gemacht und die wesentlichen Grundkenntnisse kennengelernt die nötig sind um eine Inferenz basierend auf dem EM-Algorithmus durchzuführen. In diesem Kapitel werden wir diese beiden Themen miteinander kombinieren um eine spezielle Sorte von neuronalen Netzen zu erstellen die das Inferenzproblem mithilfe des maschinellen Lernens löst. Die Kombination von statistischen Elementen und Elementen des maschinellen Lernens umfasst anspruchsvollere Modelle des maschinellen Lernens die für komplexere unüberwachte Problemstellungen eingesetzt werden können.

Wir werden uns mit dem Konzept der Mannigfaltigkeit und der Hauptachsentransformation vertraut machen sowie auf sogenannte Autoencoder generell eingehen und auf die spezielle Variante davon die Variational Autoencoder den Gegenstand des Projektes in dem nachfolgenden Kapitel ist.

³⁶ Vgl. Salimans (2014), S. 1.

4.1 Mannigfaltigkeit

Ein wesentliches Konzept im Bereich maschinelles lernen und Mustererkennung ist das der Mannigfaltigkeit.³⁷ Unter einer Mannigfaltigkeit wird eine zusammenhängende Region an Datenpunkten verstanden. Die Formen dieser Regionen sind oftmals unbekannt stellen aber eine zentrale Rolle im maschinellen lernen da, da sie die wesentlichen Informationen über das Objekt enthält. Beispielsweise wenn wir ein Spracherkennungssystem erstellt hätten, dass auf ein Signalwort entsprechend reagiert. Dieses Signalwort richtig zu erfassen und auszuwerten erweist sich als äußerst komplexe Aufgabe. Das System darf nicht von der Stimme, Tonlage, Lautstärke, dem Geschlecht, der Aussprache, Hintergrundgeräusche oder ähnlichem in der Form beeinflusst werden das es nicht funktioniert. Das System ist gezwungen sich auf die wesentlichen Merkmale des Wortes zu konzentrieren. Das geschieht dadurch, dass das System die Informationen filtert und irrelevante Informationen verwirft. Das Konzept der Mannigfaltigkeit ist insofern wichtig da es dem Rahmen angibt in wie weit das ungefilterte Wort auf das gefilterte Wort zeigt. Diese Relation machen beispielsweise andere Aufgaben komplex wie das generieren von Audio oder Bildern, da beispielsweise das zu generierende Wort auf unzählige viele verschiedene valide Formen des Wortes zeigt die aus unterschiedlichen Sequenzen der Sprache bestehen. Die Mannigfaltigkeit zu lernen stellt also eine wesentliche Aufgabe im Bereich des maschinellen Lernens da sie das Konzept des Generalisierens am ehesten widerspiegelt.

Ein konvolutionales Netz, dass beispielsweise handgeschriebene Zahlen in einem Bild erkennen soll, transformiert das Bild in eine niedrigere Dimension und soll sämtliche Variationen des Bildes, das die gleiche Zahl zeigt, nahe beieinander Clustern. Prinzipiell wird der Algorithmus eine Struktur bei den Zahlen finden und ähnliche Zahlen eher beieinander anordnen als Zahlen, die absolut unähnlich sind. Beispielsweise ist der Übergang von der Zahl Eins zu der Sieben nahezu fließend. Es wäre also zu erwarten, dass diese Cluster eher beieinander liegen als der Verbund von den Clustern Sechs, Fünf, Acht, Neun und Null. Die Zahlen Zwei und Drei erhalten wahrscheinlich eine eigene Region, da sie zu stark von den anderen abweichen. Die Regionen müssen also nicht zwingend ineinander übergehen jedoch müssen sie ineinander übergehen, wenn sie dasselbe Objekt abbilden. Also anstatt des Wertes der Zahl zu erfassen könnte ein Algorithmus erfassen ob es sich in dem Bild um eine Zahl oder ein nichtzuordnungsbares Rauschen handelt. Also das Bild auf einen eindimensionalen Wert zu reduzieren der die Wahrscheinlichkeit angibt ob es sich um eine Zahl handelt oder nicht. Das gewünschte Resultat wäre eine zusammenhängende Region, die angibt ob das Bild eine valide Zahl darstellt. Die folgende Abbildung zeigt eine

³⁷ Vgl. Goodfellow (2016), S. 161 ff.

Kollektion an Vektorrepräsentationen von Bildern des Projekts, auf das wir später weiter eingehen. Es zeigt sehr deutlich den keilförmigen Verlauf und die ersten Cluster Versuche in der Lernphase.

Die Datenpunkte richtig anzuordnen, also das die Nachbarn eines Punktes eine ähnliche semantische Bedeutung haben, und die Entfernungen richtig zu bestimmen wird innerhalb des Lernprozesses eine zentrale Rolle einnehmen.

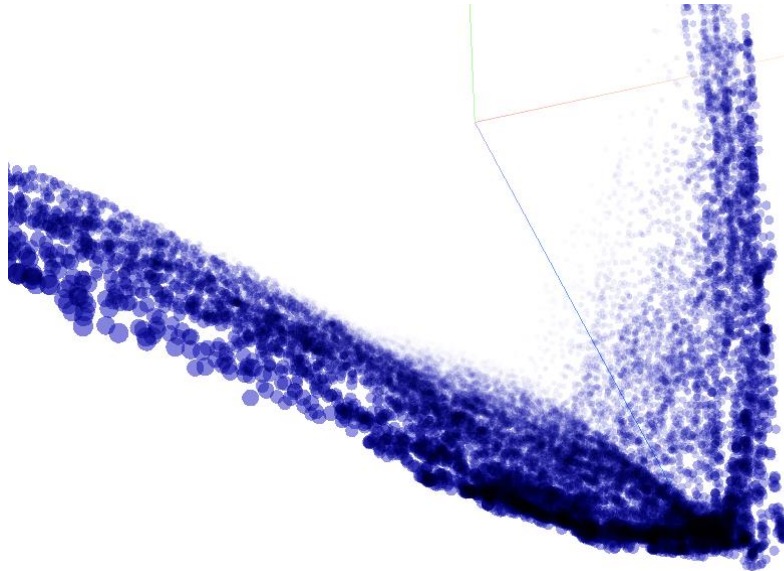


Abbildung 12: Visualisierung der Mannigfaltigkeit
Quelle: Eigene Darstellung

4.2 Hauptachsentransformation

Das Konzept der Mannigfaltigkeit machen sich viele Algorithmen des unüberwachten Lernens zu eigen. Ein effizienter und leistungsstarker Algorithmus, der sich durch seine Einfachheit und seiner vielfältigen Einsatzmöglichkeiten auszeichnet ist die Hauptachsentransformation. Formal ausgedrückt sucht der Algorithmus eine lineare Transformation der Daten in eine niedrigere Dimension transformiert mit dem Ziel die Varianz zu maximieren. Dabei unterstellt der Algorithmus den Daten, dass es einfachere Repräsentation in einem Raum in einer niedrigeren Dimension existiert. Beispielsweise könnte die Abbildung in dem vergangenen Abschnitt auch als Ebene interpretiert werden. Sofern wir einen Algorithmus haben, der überprüft ob wir den kritischen Punkt des Keils erreicht haben kennen wir die Vorschrift um die Datenpunkte nahezu richtig anzuordnen. Wir haben dadurch die Möglichkeit dreidimensionale Daten als zweidimensionale Daten abzubilden ohne wichtige Informationen zu verlieren. Einsatzmöglichkeiten sind beispielsweise die Datenvisualisierung, den Rechenaufwand zu reduzieren indem einfachere

Modelle erstellt werden können, Speicherplatz zu reduzieren oder auch um redundante, korrelierende Informationen zu entfernen um dem Algorithmus ein stabileres Training zu ermöglichen. Die Hauptachsentransformation bedient sich dabei hauptsächlich einer Eigenwertdekomposition um die gewünschte Reduzierung der Dimension durchzuführen.

Die Schritte um die Hauptachsentransformation durchzuführen sind:³⁸

1. Standardisierung der Daten $x \in \mathbb{R}^d$ für die Menge $\mathbb{X} = \{x_1, \dots, x_n\}$
2. Erstellung der Kovarianzmatrix $\Sigma \in \mathbb{R}^{d \times d}$
3. Zerlegung der Kovarianzmatrix in Eigenvektoren und Eigenwerte
4. Auswahl von k Eigenvektoren um die größte Varianz in den Daten abzubilden
5. Konstruieren der Projektionsmatrix $W \in \mathbb{R}^{d \times k}$
6. Transformation der Eingabemenge \mathbb{X} mit der Projektionsmatrix W in den Unterraum mit der Dimension k

Der erste Schritt ist die Standardisierung der Daten. Dabei werden die Daten skaliert und oftmals auf einem bestimmten Intervall abgebildet. Dadurch werden Eigenschaften wie beispielsweise die Varianz aussagekräftiger und können mit den Varianzen von anderen Attributen besser verglichen werden.

Der nächste Schritt ist die Erstellung der Kovarianzmatrix. Die Kovarianz gibt Aufschluss über die Relation der Attribute untereinander. Beispielsweise wenn wir die Aktivität eines Computers überwachen würden könnten wir eine Relation zwischen den Attributen CPU Auslastung und dem Schreiben bzw. dem Lesen einer Festplatte entdecken können. Die Relation ist nicht eindeutig da das eine nicht zwingend das andere impliziert. Allerdings würde man statistisch diese Relation erfassen können.

Die Kovarianzmatrix Σ ist eine $d \times d$ Matrix in der die einzelnen Elemente wie folgt berechnet werden:

$$\Sigma = \begin{bmatrix} \sigma_{11}^2 & \cdots & \sigma_{1d}^2 \\ \vdots & \ddots & \vdots \\ \sigma_{d1}^2 & \cdots & \sigma_{dd}^2 \end{bmatrix} \text{ mit } \sigma_{jk}^2 = \frac{1}{n} \sum_{i=1}^n (x_j^{(i)} - \mu_j)(x_k^{(i)} - \mu_k)$$

Die Elemente der Diagonalen beschreiben die Varianz des jeweiligen Attributs. Außerdem ist die Matrix symmetrisch durch das Kommutativgesetz und es gilt: $\sigma_{jk}^2 = \sigma_{kj}^2$.

Wenn die Kovarianzmatrix berechnet ist können die Eigenwerte sowie die Eigenvektoren ermittelt werden. Dabei wird folgende Gleichung nach λ umgestellt:

$$\Sigma v = \lambda v$$

³⁸ Vgl. Raschka (2017), S. 137 – 144.

$$\det \left(\begin{bmatrix} \sigma_{11}^2 - \lambda & \cdots & \sigma_{1d}^2 \\ \vdots & \ddots & \vdots \\ \sigma_{d1}^2 & \cdots & \sigma_{dd}^2 - \lambda \end{bmatrix} \right) = 0$$

Die einzelnen $\lambda_1, \dots, \lambda_d$ die die Gleichung erfüllen sind die Eigenwerte. Werden die Eigenwerte wieder in die Gleichung eingesetzt lassen sich die einzelnen Eigenvektoren berechnen.

$$\begin{bmatrix} \sigma_{11}^2 - \lambda_z & \cdots & \sigma_{1d}^2 \\ \vdots & \ddots & \vdots \\ \sigma_{d1}^2 & \cdots & \sigma_{dd}^2 - \lambda_z \end{bmatrix} * v_z = 0$$

Anschließend werden die Vektoren ausgewählt, die am ehesten die Daten widerspiegeln und somit die höchste Varianz aufweisen. Die größten Eigenwerte erfüllen diese Eigenschaft. Formal lässt sich sagen, dass die Summe der Eigenwerte die maximale Varianz abbilden und jeder Eigenwert einen Beitrag an Informationen liefert. Dieser Beitrag an Informationen ist der Eigenwert geteilt durch die Summe der Eigenwerte:

$$\frac{\lambda_j}{\sum_{j=1}^d \lambda_j}$$

Die Projektionsmatrix $W = [v_1, \dots, v_k]$ wird durch das horizontale aneinanderreihen der ausgewählten Eigenvektoren erstellt. Mit dieser Projektionsmatrix wird der Datensatz x in auf eine neue reduzierte Repräsentation transformiert:

$$x' = Wx \text{ wobei } x' \text{ den neuen transformierten Datensatz darstellt}$$

Bei der Hauptachsentransformation handelt es sich allerdings um eine lineare Transformation. Sie ist also nur bedingt bei nicht linearen Verläufen einsetzbar. Ein weiteres Problem sind statistische Ausreißer. Bei der Standardisierung werden diese nicht weiter berücksichtigt und so können falsche Skalierungen angewendet werden, die das Ursprüngliche Bild zu weit verzerren. Die Idee dahinter zur Transformation von aussagekräftigen Features lässt sich auch in einem Ansatz des maschinellen Lernens darstellen in Form von Autoencodern.

4.3 Autoencoder

Eine spezielle Variante von neuronalen Netzen sind Autoencoder. Diese können in einer Problemstellung des ungestützten Lernens eingesetzt werden, um wichtige relevante Merkmale in den Daten zu entschlüsseln. Vereinfacht ausgedrückt komprimieren diese Netze die Daten in eine Repräsentation und rekonstruieren wieder die Eingabe.³⁹ Autoencoder haben historisch betrachtet einen besonderen Stellenwert und sind maßgeblich an dem heutigen Erfolg von neuronalen Netzen beteiligt. Restricted-Boltzman-Machines sowie Deep Belief Networks waren

³⁹ Vgl. Goodfellow (2016), S. 502.

frühere Forschungsthemen und haben die Entwicklung von neuronalen Netzen in den Vordergrund gerückt. In der Literatur werden Autoencoder auch als „latent variable models“ bezeichnet, da sie eine nicht Observierbare Repräsentation beherbergen, die es gilt zu entschlüsseln. Der Autoencoder besteht dabei im Wesentlichen aus zwei Komponenten einer Kodierungsfunktion und einer Dekodierungsfunktion. Der Kodierungsfunktion ist verantwortlich relevante Informationen zu finden die nützlich sind um es dem Dekodierer ermöglichen die ursprüngliche Eingabe wiederherzustellen.

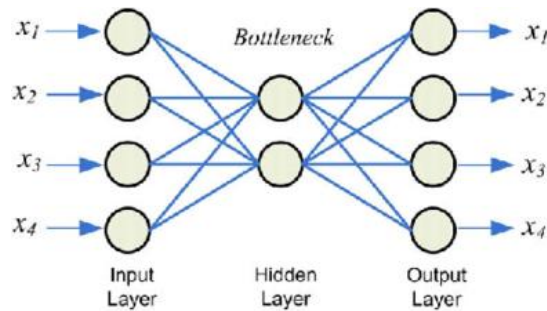


Abbildung 13: Autoencoder Struktur

Quelle: <https://www.doc.ic.ac.uk/~js4416/163/website/autoencoders/>

$$g(f(x)) = x \text{ und die Performancefunktion } \mathcal{L}(x, g(f(x)))$$

Dabei ist f eine Kodierungsfunktion und g eine Dekodierungsfunktion.

Der Vektor, der die codierte Form des Inputs enthält, wird in dem Zusammenhang mit Autoencodern auch als embedding bezeichnet. Dadurch, dass es sich um einen Vektor handelt, können sämtliche mathematische Operationen durchgeführt werden wie die Messung der Distanz zwischen zwei embeddings durch die Berechnung der Länge des Differenzvektors.

Die Struktur der Autoencoder kann wie auch bei neuronalen willkürlich gewählt werden. Es können Deep Autoencoder mit mehreren versteckten Schichten erstellt werden, die komplexe, willkürliche, nicht lineare Funktionen mit Aktivierungsfunktionen erlernen können oder es können auch oberflächliche lineare Funktionen erlernt werden deren Ergebnisse vergleichbar sind mit der Hauptachsentransformation.⁴⁰ Je nach Aufgabenkomplexität, Anzahl der Datensätze sowie Rechenkapazitäten kann ein geeignetes Modell gewählt werden.

Das charakteristische Merkmal eines Autoencoders ist vermutlich die Sanduhrförmige Struktur, der in der Mitte die codierte Repräsentation enthält. Diese Mitte wird auch als Flaschenhals (engl. Bottleneck) bezeichnet. Die Wahl der Dimension dieser Repräsentation kann sich in der Praxis als äußerst schwierig erweisen. Beispielsweise wenn die Dimension gleich oder größer der Dimension der Eingabeschicht ist, wird der das neuronale Netz dazu neigen die Werte einfach zu kopieren und überflüssige Neuronen nicht weiter zu verwenden. Das Ergebnis ist nicht

⁴⁰ Vgl. Goodfellow (2016), S. 503.

zufriedenstellend, da wir nur eine bzw. zwei Identitätsfunktion erlernen. Autoencoder deren Dimension der Repräsentation größer oder gleich der Eingabeschicht sind werden auch übervollendet (engl. overcomplete) bezeichnet. Ist die Dimension kleiner werden sie als untervollendet (engl. undercomplete) bezeichnet. Untervollendete Autoencoder haben in der Regel einen bedeutsameren Stellenwert als übervollendete. Allerdings muss ein Equilibrium zwischen der Repräsentations- und der Machtkapazität des Dekodierers bestehen. Es wäre beispielsweise möglich, sämtliche Eingaben des Kodierers als Index auf einer einzelnen Dimension abzuspeichern. Ein zu mächtiger Dekodierer könnte trotz dieses Fehlverhalten aus diesem Index die passende Eingabe wiederherstellen. Allerdings lernt der Autoencoder keine vernünftige Struktur und kann somit nur als Hashfunktion verwendet werden, der Autoencoder leidet an einer Überanpassung und würde bei neuen Daten komplett versagen.⁴¹

In der Praxis haben sich einige Varianten von Autoencodern hervor getan die zusätzlich zu dem lernen einer kodierten Repräsentation einige Einschränkungen (engl. Constraints) besitzen.

Eine dieser Formen sind Sparse-Activation-Autoencoder. Diese spezielle Form sucht eine Repräsentation, die nur spärlich aktiviert ist bzw. viele Elemente Null sind. Diese Form folgt dem biologischen Paradigma, das die Neuronen nicht bei jeder Eingabe ein wenig aktiviert werden, sondern nur bei speziellen Eingabemustern stark angesprochen werden.⁴² Das Ziel dahinter ist das die einzelnen Neuronen eindeutige Charakteristika, die sich von den anderen unterscheiden.

Eine weitere Form von Autoencoder sind Rauschfilternde Autoencoder.⁴³ Dabei wird die Eingabe des neuronalen Netzes mit einem zusätzlichen Noise-Vektor verunreinigt und der Autoencoder muss das Rauschen beseitigen, um den ursprünglichen nicht verunreinigten Wert wiederherzustellen. Diese Form kann beispielsweise dazu verwendet werden, um fehlerhafte Bilder wiederherzustellen oder auch zum Lernen von captcha Bildern bzw. die captcha Verunreinigung zu entfernen. Präziser ausgedrückt lernen die Autoencoder ein Vektorfeld der an jeder Stelle einen geeigneten Vektor finden soll, um die verunreinigte Eingabe auf die Ursprüngliche Mannigfaltigkeit abzubilden.

$$\mathcal{L}(x, g(f(x + \epsilon)))$$

⁴¹ Vgl. Goodfellow et al (2016), S. 504.

⁴² Vgl. Goodfellow et al (2016), S. 227.

⁴³ Vgl. Goodfellow et al (2016), S. 507.

4.4 Variational Autoencoder

Eine besondere Variante sind Variational Autoencoder.⁴⁴ Dabei werden die deterministischen Autoencoder in ein probabilistisches Setting überführt. Der Kodierer wird eine Wahrscheinlichkeitsverteilung von $P_{Kodierer}(z|x)$ und der Dekodierer wird zu einer Wahrscheinlichkeitsverteilung von $P_{Dekodierer}(x|z)$.

Wir können von X auf Z schließen und umgekehrt. Der Autoencoder soll diese Repräsentation finden, um einen unvollständigen Datensatz X zu komplettieren $P_{Model}(x, z)$. Die Grundannahme ist, dass es in dem Datensatz X versteckte, latente (fehlende) Variablen gibt, die mit dem X Datensatz korrelieren.

Diese Problemstellung kann als Inferenzproblem aufgefasst werden und mithilfe der Variational Inferenz Methode gelöst werden. Diese Form von Autoencodern wird auch als Variational Autoencoder (VAE) bezeichnet.

Variational Autoencoder lernen anstatt einer deterministischen Repräsentation eine bzw. mehrere Distributionen, aus denen eine Repräsentation gezogen wird. Der Kodierer wird zu einem neuronalen Netzwerk das, im Falle einer Normalverteilung, die Werte für den Mittelwert μ und die Varianz σ (oder auch die Kovarianzmatrix Σ) vorhersagt. Die folgende Abbildung zeigt den schematischen Aufbau eines VAE:

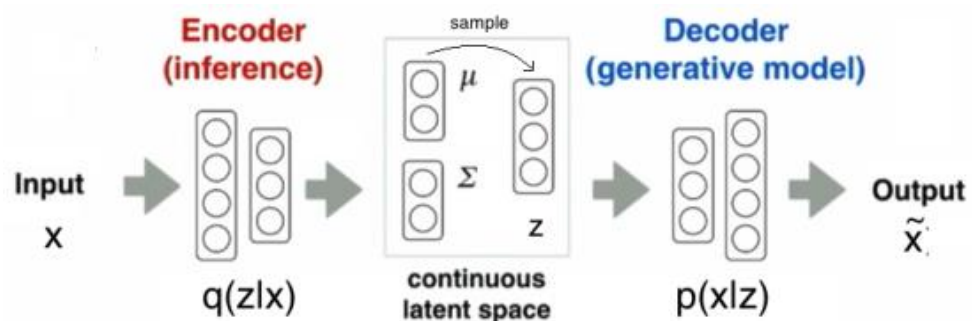


Abbildung 14: Variational Autoencoder

Quelle: <https://calculatedcontent.com/2017/09/06/free-energies-and-variational-inference/>

Der Dekodierer zieht aus dieser vorhergesagten Distribution eine Repräsentation und versucht anhand dieser wieder die ursprüngliche Eingabe herzustellen. Wie wir sehen werden kann diese Vorgehensweise auch als verrauschte Eingabe interpretiert werden und sie gilt als natürlicher Regulator der das generalisieren Autoencoders sicherstellt. Wie in dem Abschnitt über Variational Inferenz beschrieben verändert sich die Performancefunktion dementsprechend und besteht aus den beiden Termen der ELBO und der KL-Abweichung. Der VAE muss einen Balanceakt lernen zwischen diesen beiden Größen, um das ursprüngliche Ziel des Autoencoders zu erfüllen.

⁴⁴ Vgl. Kingma (2013), S. 1-4.

$$ELBO = \mathbb{E}[\log(P(x|z))] - KL(q(z)||P(z))$$

Die beiden Größen können als Rekonstruktionsfehler und als Abweichung von unserer Modelannahme interpretiert werden. Das neuronale Netzwerk, das die Distribution q vorhersagt, könnte ohne die KL-Abweichung dazu neigen keine oder nur geringe Varianz vorherzusagen. Dadurch würde das Model eher deterministisch werden und die Zufallskomponente komplett aushebeln. Wenn wir beispielsweise eine Normalverteilung vorhersagen wollen können wir als Prior $P(z)$ eine Normalverteilung annehmen mit den Werten $\mu = 0$ und $\sigma = 1$. Der VAE wird bestraft, wenn er sich von dieser Verteilung zu weit entfernt. Die Aufgabe ist also eine Repräsentation auf einem beschränkten Raum zu lernen und die Trainingsbeispiele in diesem Raum anzuordnen. Dadurch, dass Distributionen vorhergesagt werden entstehen Bereiche, in denen sich die Wahrscheinlichkeitsmasse überlappt. Das ist in dem Sinne ein Problem, da der VAE oftmals nicht unterscheiden kann zwischen zwei verschiedenen Eingaben und somit nicht exakt die Eingabe wiederherstellen kann. Der VAE kann das Problem auf prinzipiell auf drei verschiedene Arten lösen. Die erste Lösung wäre es, μ so anzupassen, dass sich möglichst wenig Wahrscheinlichkeitsmasse überlappt. Mit anderen Worten verschiebe die Distribution nach links bzw. nach rechts und verringere die gemeinsame Wahrscheinlichkeit. Allerdings verstößt dieses Verhalten gegen die weiche Beschränkung der KL-Abweichung. Das Gleiche gilt auch für den zweiten Ansatz der Reduzierung der Varianz, bzw. die Zufallskomponente zu entfernen, um möglichst deterministische Repräsentation zu erhalten, welches ebenfalls gegen die KL-Abweichung verstößt. Die letzte Möglichkeit ist es ähnliche Objekte nahe beieinander anzuordnen. Diese Objekte unterscheiden sich beispielsweise in der x-y-Position, der Größe, der Rotation und ähnlichem. Es muss dazu allerdings noch erwähnt werden, dass die latenten Variablen keine expliziten physikalischen Interpretationen aufweisen müssen.⁴⁵ Es werden nur Features entdeckt, die für die Dekodierung am hilfreichsten sind. Der Dekodierer muss diese selbstgewählten Variablen nur richtig interpretieren. Dieser Ansatz verstößt gegen keinerlei Beschränkungen und sorgt dafür, dass (hoffentlich) plausible Repräsentationen gelernt werden. Die nachfolgende Abbildung zeigt die Problemstellung samt den ersten beiden Lösungsansätze anhand von zwei Trainingsbeispielen x_1 und x_2 .

⁴⁵ Vgl. Bishop (2006), S. 366.

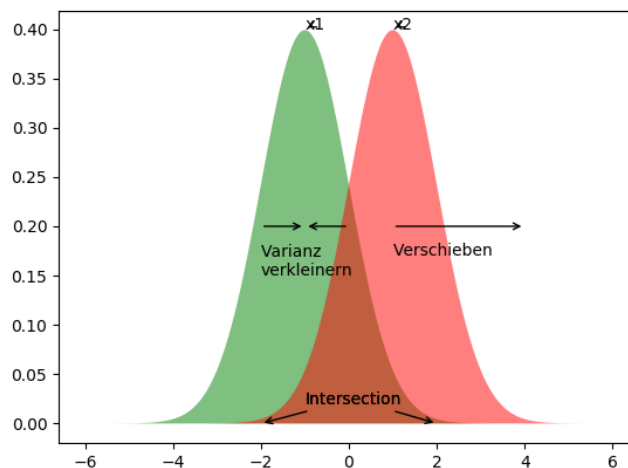


Abbildung 15: Variational Autoencoder vorhergesagte Distributionen

Quelle: Eigene Darstellung

Der VAE gilt als eine verlässlichere Form des Autoencoders das ein stabileres Training ermöglicht. Da der VAE eine Distribution lernt wird er in der Literatur auch als generatives Model bezeichnet. Die Idee eines generativen Models ist es, vereinfacht ausgedrückt, aus einer Distribution ein Sample ziehen zu können und anhand dieses Samples Inhalte generieren zu können wie beispielsweise Texte, Bilder, Töne und ähnlichem.

Für die Generierung von Inhalten wäre es beispielsweise von Vorteil, wenn die Attribute bzw. die Elemente des Samplevektors, keinerlei Relation aufweisen. Allerdings ist das in den meisten Fällen nicht möglich. Beispielsweise wenn wir ein generatives Model hätten, das einen Kreis auf einer Leinwand zeichnet, könnten beispielsweise die Farbe des Kreises entscheidend sein, die Größe, die x Position und die y Position. Wenn unsere Trainingsdaten allerdings daraus bestehen das wir einen Kreis in einer linearen Bewegung vor- und zurück bewegen, könnte das Model genau ein Feature lernen. Nämlich der Verlauf dieser linearen Bewegung. Da anstelle von zwei Attributen nur eins effektiv genutzt werden würde. Das Feature würde die Steigung (bzw. der Vektor bei vertikaler Bewegung) sein und es wäre nicht möglich den Kreis anders anzuordnen. In der Literatur spricht man in diesem Zusammenhang auch von entangled (verstrickte) und disentangled (entwirrte) Repräsentationen. Die folgende Abbildung zeigt diese Problemstellung an dem Beispiel von Handgeschriebenen Zahlen:

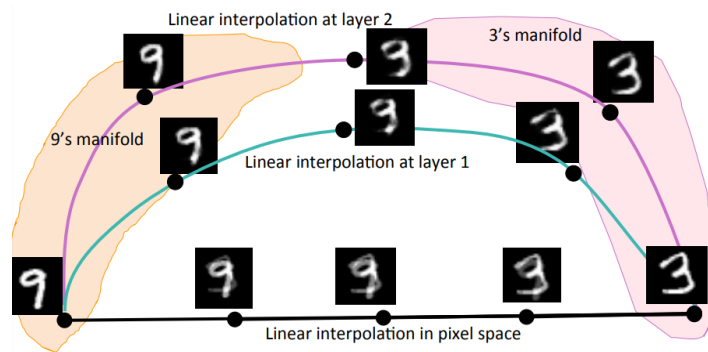


Abbildung 16: Entangled Repräsentationen

Quelle: <https://speakerd.s3.amazonaws.com/presentations/056b60e0cfee0130eacc16a85b2fb3f8/aaai2013-tutorial.pdf>

S. 199

Eine Möglichkeit, um dieses Verhalten zu forcieren sind sogenannte Beta-VAE.⁴⁶ Sie funktionieren auf nahezu die gleiche Weise wie normale VAE allerdings wird die Performancefunktion leicht angepasst und es wird eine stärkere Präferenz für die Beschränkung der KL-Abweichung genommen. Mit anderen Worten die KL-Abweichung bekommt mehr Gewicht bei der Performancefunktion als der Rekonstruktionsserror.

$$ELBO = \mathbb{E}[\log(P(x|z))] - \beta * KL(q(z)|| P(z))$$

Der Wert für β ist üblicherweise größer oder gleich 1. Das Verhalten lässt sich so interpretieren, dass der Balanceakt zugunsten der KL-Abweichung entschieden wird. Es ist für den Algorithmus erstrebenswerter größere Bereiche Wahrscheinlichkeitsmasse einzubeziehen und die Varianz ähnlich zu dem Prior zu machen. Gleichzeitig werden weniger Feature benutzt, da es von Nachteil ist von dem Prior zu weit abzuweichen. Ein Nachteil allerdings sein, dass der β -VAE multiple Distributionen benutzt, um ein Feature vorherzusagen, da mit jeder Verteilung ein Stück der Zufallskomponente entfernt wird und aus dem Zufallswerts ein Durchschnitt bzw. ein Erwartungswert wird.

Ein technisches Detail bei der Implementierung von VAE ist die Tatsache das sich die Gradienten von Distributionen nicht berechnen lassen. Dadurch das eine zufällige Sample gezogen wird ist es nicht möglich den Gradienten zu berechnen der nötig ist, um die Parameter anzupassen. Eine Lösung für dieses Problem ist der Reparameterization Trick. Der Reparameterization Trick bedient sich eines mathematischen Tricks, in dem sich die Distributionen anders darstellen lassen. Beispielsweise könnte eine Normalverteilung folgendermaßen umformuliert werden:

$$\epsilon \sim N(x; \mu, \sigma^2) \Leftrightarrow \mu + \sigma * \epsilon \text{ mit } \epsilon \sim N(0,1)$$

Diese Form lässt sich einfach herleiten, indem die Normalverteilung ausgeschrieben wird und entsprechend umgestellt wird. Der Unterschied zwischen diesen beiden Formen ist, das die

⁴⁶ Vgl. Higgins (2016), S. 2 f.

Zufallskomponente ausgelagert ist und die vorhergesagten Parameter sich so isolieren lassen. Dadurch ist es möglich die Gradienten zu berechnen um die Parameter dementsprechend anzupassen. Es ist also möglich Backpropagation anzuwenden und das Model zu trainieren trotz der Zufallskomponente.

4.5 State of the Art für Generative Modelle

VAE haben den Fokus der Wissenschaftler erneut auf Autoencoder gelenkt. Insbesondere gelten diese heute immer noch als „State of the Art“ im Bereich des Lernens einer Repräsentation.⁴⁷ Allerdings wurden bessere Modelle für generative Models entworfen, die in der Lage sind neue Inhalte zu generieren die detaillierter sind. Wie beispielsweise „Generative Adversarial Networks“ (GANs). Ein GAN besteht aus einem generativen Netzwerk und einen Diskriminator Netzwerk. Die Idee hinter GANs basiert auf einer Art Täuschungsspiel. Das generative Netzwerk versucht dabei neue Trainingsdaten zu erstellen wohingegen der Diskriminator versucht diese versuchten Täuschungen aufzudecken und zu unterscheiden ob es sich bei den aktuellen Daten um Täuschungen handelt oder aus den ursprünglichen, originalen, Datenbestand handelt. Das generative Model wird mit der Zeit immer bessere Täuschungen entwerfen und versuchen den Diskriminator gleichzeitig dazu animieren immer besser zu zwischen den beiden Distributionen zu unterscheiden. Im Vergleich zu VAE bieten GANs allerdings bessere Resultate. Beispielsweise VAE, die Bilder generieren, tendieren dazu eher verwaschen zu wirken. Die Ursachen dafür sind noch unbekannt.⁴⁸ Im Gegensatz zu VAE tendieren GANs zu einem deutlich instabileren Training da das sogenannte Nash-Equilibrium zwischen den Kräften des Diskriminators und des Generators bestehen muss. Ein zu starker Diskriminator wäre in der Lage die Täuschungen sofort zu erkennen und der Generator wäre nicht in der Lage sich zu verbessern da die Gradienten zu klein sind. Ein zu schwacher Diskriminator wäre für das Endresultat nicht gut genug. Je stärker der Diskriminator unterscheiden kann desto besser sollten die Resultate werden. Es gibt zwar einige Tricks, um das Training zu stabilisieren die sich empirisch bewährt haben allerdings ist es generell schwieriger ein GAN als ein VAE zu trainieren und es braucht deutlich mehr Zeit, um brauchbare Resultate zu erzielen. Die Resultate von GANs sind allerdings deutlich besser. Die folgende Abbildung zeigt die verschiedenen Resultate zwischen MSE und einem GAN:

⁴⁷ Vgl. Goodfellow (2016), S. 699.

⁴⁸ Vgl. Goodfellow (2016), S. 697.

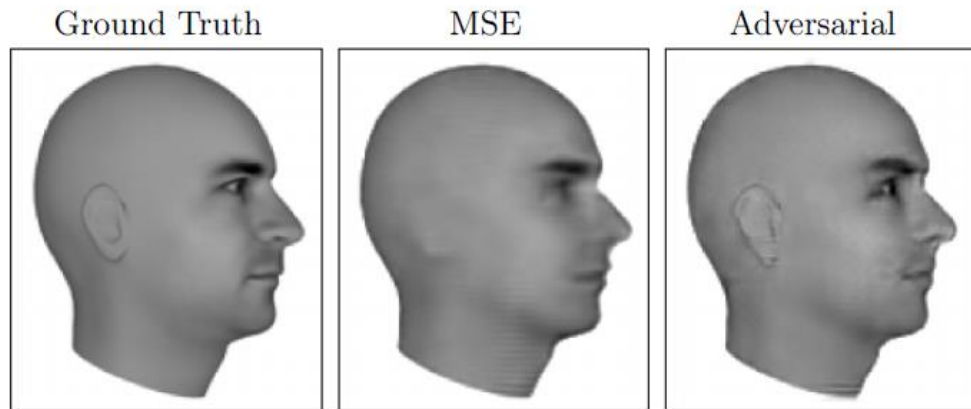


Abbildung 17: Vergleich der Resultate zwischen MSE und GAN

Quelle: DeepLearningbook 545

Trotzdem gelten VAE immer noch als mächtige Netzwerke, die für eine Vielzahl an Problemstellungen eingesetzt werden können. Insbesondere zum Lernen der Mannigfaltigkeit. Eine dieser Problemstellungen ist beispielsweise eine Domain Adaption. Die Idee hinter einer Domain Adaption ist, zwei VAE auf zwei unterschiedliche Bereiche zu trainieren. Beispielsweise könnte ein VAE die Mannigfaltigkeit von einem Bild Objekt lernen wohingegen ein weiterer VAE antrainiert werden kann der die Mannigfaltigkeit von Textdaten lernt. Werden diese beiden Netzwerke kombiniert können Netzwerke erstellt werden, die beispielsweise von einem Text auf ein Bild zeigen und umgekehrt. Es muss allerdings noch ein zusätzliches neuronales Netzwerk zwischengeschaltet werden das die eine Mannigfaltigkeit auf die andere abbildet, da die beiden Strukturen nicht zwingend identisch sind.

5 Praxisprojekt: Optimierung von Videos mithilfe von Variational Autoencodern

Diese Arbeit wird sich mit Optimierung von Videos mithilfe von variationalen Autoencodern befassen. Für diese Arbeit wurde ein kurzes simples Video aufgenommen in denen einige Operationen einer Hand durchgeführt wurden. Wir werden uns zunächst mit der Datenvorverarbeitung, also mit dem Überführen des Videos in einen geeigneten Datensatz, befassen und anschließend auf das VAE-Model eingehen, das in dieser Arbeit verwendet wurde. Anschließend werden wir uns mit den Ergebnissen sowie der Evaluation befassen, um das Projekt abzuschließen.

In dem Experiment werden wir versuchen die Bildwiederholrate (engl. Framerate) künstlich mit VAEs zu erhöhen. Dabei werden wir aus unserem Datenbestand, auf dem wir im folgenden Kapitel eingehen, sukzessiv verringern und überprüfen ob wir mit einem kleineren Datenbestand, also einer geringen Bildwiederholrate, die Videos künstlich zu erhöhen, um flüssigere Videos zu erhalten. Wir werden dabei auf die Problematiken bei dieser Methode eingehen.

5.1 Der Datensatz sowie die Vorverarbeitung

Der Datensatz, der in dieser Arbeit verwendet wird, besteht aus einem ca. 11-minütigem Video, in dem einige Basisoperationen einer Hand gezeigt werden. Das Video wurde dabei vor einem blauen Hintergrund aufgenommen der anschließend mithilfe einer Maske entfernt wurde. Insgesamt besteht der Datensatz aus 19860 Einzelbildern, die auf die Abmaßen von 64x64 Pixeln herunterskaliert wurden. Die Bilder wurden anschließend in eine Graustufe umgewandelt, um das Lernen für den Algorithmus zu vereinfachen. Da anstelle der drei Komponenten RGB nur eine kombinierte Komponente verwendet wird, die den Mittelwert der drei Komponenten darstellt. Die folgende Abbildung zeigt den Prozess des Maskierens:

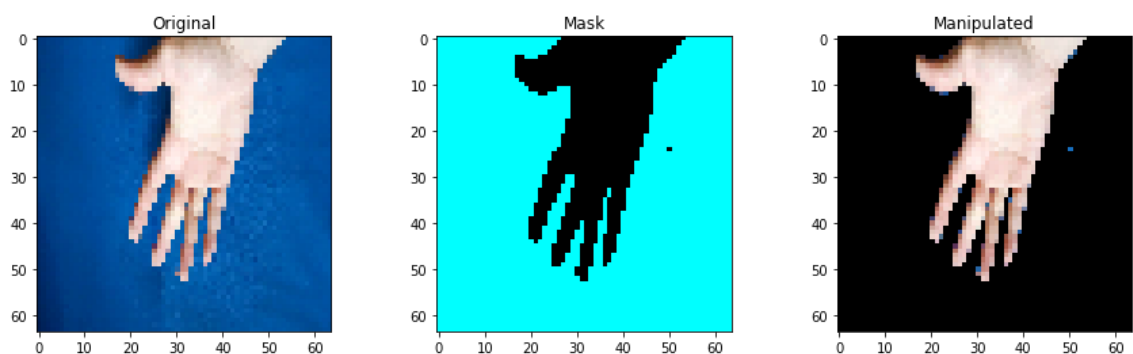


Abbildung 18: Maskieren des Videos

Quelle: Eigene Darstellung

Das Video besteht dabei aus einer zusammenhängenden Aufnahme, in der die Umformung von einem Zustand zu einem anderen eine fließende Bewegung darstellen.

Einige dieser Zustände sind beispielsweise:

- Die Bewegungen der Hand in einem Dreidimensionalen Koordinatensystem
- Das spreizen der Finger
- Das bilden einer Faust
- Das rotieren der Hand
- Die ausschließliche Bewegung des Daumens
- Das bewegen des ersten Fingerglieds. Die anderen Glieder bleiben verborgen

Da das Video eine fließende Bewegung darstellt ist es möglich, die Relation zwischen den verschiedenen Zuständen zu erfassen. Beispielsweise schließt das bilden einer Faust das spreizen der Finger aus jedoch nicht die Bewegung im Koordinatensystem. Das folgende Bild zeigt einen Auszug aus dem Datenbestand an denen Beispielhaft einige Zustände dargestellt sind.

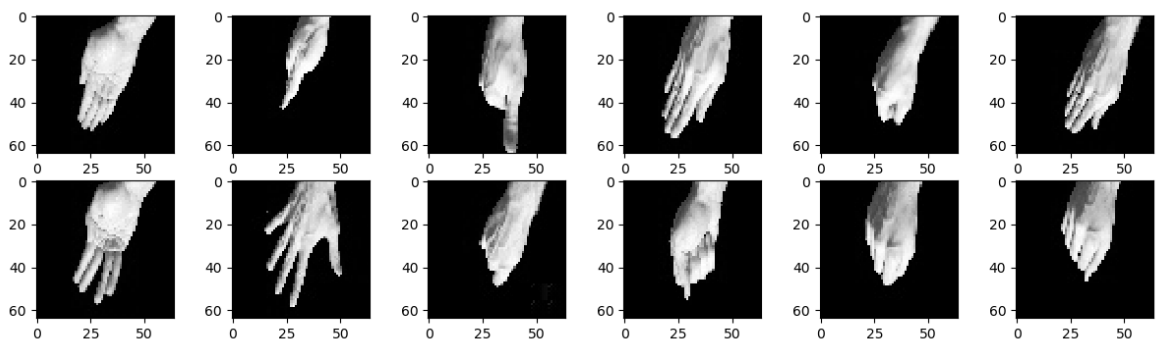


Abbildung 19: Trainingsdaten Beispiele

Quelle: Eigene Darstellung

Wie an der Abbildung erkennbar sind noch teilweise viele Details erkennbar wie beispielsweise Knochen, Adern, Lebenslinien, teilweise Schattierungen, Falten usw. Allerdings sollten diese Attribute in Relation zu den einzelnen Zuständen stehen. Da beispielsweise bei einer Faust die Knöchel deutlich sichtbarer sind als bei einer ausgestreckten Handfläche. Es ist also möglich Aufgrund der Stellung der Hand der Rotation usw. errahnen zu können wo, und wie stark, der Knöchel sichtbar ist. Dasselbe Prinzip ist auf andere Details anwendbar.

5.2 Model

Der VAE, der in dieser Arbeit verwendet wird, besteht dabei im Wesentlichen aus einem konvolutionalen Netzwerk mit folgender Architektur:

Schicht:	Enthält:
----------	----------

1	<ol style="list-style-type: none"> 1. Konvolutionale Schicht mit 32 Filtern je 5×5×1px und einer Schrittweite von 1 2. ReLU Aktivierungsfunktion 3. Averagepoolingoperation mit einer Filtergröße von 2×2px und einer Schrittweite von 2
2	<ol style="list-style-type: none"> 1. Konvolutionale Schicht mit 64 Filtern je 5×5×32px und einer Schrittweite von 1 2. ReLU Aktivierungsfunktion 3. Averagepoolingoperation mit einer Filtergröße von 2×2px und einer Schrittweite von 2
3	<ol style="list-style-type: none"> 1. Konvolutionale Schicht mit 128 Filtern je 5×5×64px und einer Schrittweite von 1 2. ReLU Aktivierungsfunktion 3. Averagepoolingoperation mit einer Filtergröße von 2×2px und einer Schrittweite von 2
4	<ol style="list-style-type: none"> 1. Neuronale Schicht mit 512 Neuronen 2. ReLU Aktivierungsfunktion
5	<ol style="list-style-type: none"> 1. Neuronale Schicht mit 25 Neuronen. Anhand dieser Schicht werden die Werte für μ und σ vorhergesagt. Unsere Repräsentation für die Eingabe x mithilfe von 25 Werten abgebildet. 2. ReLU Aktivierungsfunktion
6.1	<ol style="list-style-type: none"> 1. Vorhersage von μ mit einer neuronalen Schicht mit 25 Neuronen 2. Als Aktivierungsfunktion wurde die Identitätsfunktion gewählt
6.2	<ol style="list-style-type: none"> 1. Vorhersage von σ mit einer neuronalen Schicht mit 25 Neuronen 2. Softplus Aktivierungsfunktion
7	<ol style="list-style-type: none"> 1. Die Repräsentationsschicht mit einem Sample aus der vorhergesagten Distribution von den Schichten 6.1 und 6.2
8	<ol style="list-style-type: none"> 1. Neuronale Schicht mit 16384 Neuronen 2. Umformung der neuronalen Schicht zu einem 4×4×1024 Tensor 3. Batchnormalization⁴⁹ 4. ReLU Aktivierungsfunktion
9	<ol style="list-style-type: none"> 1. Transpose konvolutionale Schicht⁵⁰ mit 5×5×512px stride 2 2. Batchnormalization 3. ReLU Aktivierungsfunktion
10	<ol style="list-style-type: none"> 1. Transpose konvolutionale Schicht mit 5×5×256px stride 2 2. Batchnormalization 3. ReLU Aktivierungsfunktion
11	<ol style="list-style-type: none"> 1. Transpose konvolutionale Schicht mit 5×5×128px stride 2 2. Batchnormalization 3. ReLU Aktivierungsfunktion
12	<ol style="list-style-type: none"> 1. Transpose konvolutionale Schicht mit 5×5×1px stride 2 2. Batchnormalization 3. Sigmoid Aktivierungsfunktion

⁴⁹ Batchnormalization bezeichnet eine Regulierungsmethode bei der die Aktivierungen der Neuronen einer statistischen Distribution unterstellt wird.

⁵⁰ Eine transpose konvolutionale Operation ist eine umgekehrte konvolutionale Operation. Anstelle einer Komprimierung der Informationen werden diese dekomprimiert.

Das Model besteht aus insgesamt 13 Schichten, bei dem jeweils 5 Schichten für den Kodierer und Dekodierer reserviert sind. Der Flaschenhals besteht aus insgesamt 25 Werten, die das aktuelle Bild repräsentieren sollen. Das oben beschriebene Model stellt dabei das Model dar, dass für die Praxis eingesetzt werden kann. Bei der Trainingsphase wird noch eine bzw. zwei Schichten benötigt, um den Fehlerwert zu bestimmen bestehend aus dem ELBO und der KL-Abweichung des Priors.

5.3 Durchführung

Dieses Kapitel soll einen Überblick über die Resultate geben, die hilfreich und notwendig sind um die späteren Ergebnisse auswerten zu können. Wir werden uns dabei bei der Durchführung auf einen vollständigen Datenbestand beschränken. Wir gehen zunächst auf die gelernte Mannigfaltigkeit des Videos ein und anschließend auf die einzelnen Komponenten der Repräsentation ein. Wir werden zusätzlich Vorüberlegungen über das Verhalten der Models treffen.

5.3.1 Die Mannigfaltigkeit der Hand

Sämtliche Bilder des Videos wurden als 25-dimensionale Datenpunkte abgespeichert nach 90 Trainingsepochen. Auf diesen Datenbestand wurde eine Hauptachsentransformation durchgeführt, um diese 25-Dimensionen auf 3 Dimensionen abzubilden. Diese 3 Dimensionen beschreiben 75,7% von Varianz der Daten. Das bedeutet das die 3 Komponenten 75,7% der Informationen der Daten repräsentieren. Wie in den Nachfolgenden Abbildungen zu erkennen ist haben sich eine konkave- und eine konvexe Kuppel gebildet deren Spitzen sich zu überlappen scheinen. Diese zwei Kuppeln umfassen einerseits das Spektrum der Faust samt verschiedener Rotationen inklusive der geschlossenen Handfläche mit gespreizten Daumen und die andere Kuppel beschreibt andere Operationen die mit der Handfläche durchgeführt werden können wie das spreizen der einzelnen Finger sowie die Überführung der ausgestreckten Handfläche zu einer Faust. Das Zentrum des Gebildes besteht aus Bildern des Handrückens mit nicht gespreizten Fingern. Es ist zu erkennen, dass die Öffnungen der beiden Kuppeln in einem Winkel von ca. 90° versetzt sind. Die X-Y-Z Dimensionen lassen sich wie folgt interpretieren. Die X-Dimension ist ein Indikator dafür wie breit das Bild der Hand ist. Befindet sich beispielsweise die Handfläche auf dem Tisch entspricht dies einer breiter Hand da diese mehr Raum im Bild einnimmt als eine Hand deren Unterkante auf dem Tisch liegt. Die Y-Dimension ist schwieriger zu bestimmen. Sie dient meiner Meinung nach dazu die zwei Archetypen der Hand zu trennen. Die Handfläche sowie die Faust. Es ist zu erkennen, dass auf der Unterseite des Gebildes sich eher Handflächen befinden und auf der Oberseite eher Fäuste. Die Z-Dimension scheint auf der Unterseite ein Indikator für

das Spreizen der Finger zu sein. Je gespreizter die Finger desto höher ist dieser Wert. Die Oberseite besitzt allerdings keine auffällige Interpretation, sie besteht aus einer Mischung Bildern, die eine Faust mit jeweils Vor- und Unterseite zeigen sowie die Oberkante der Hand. Diese Dimen-

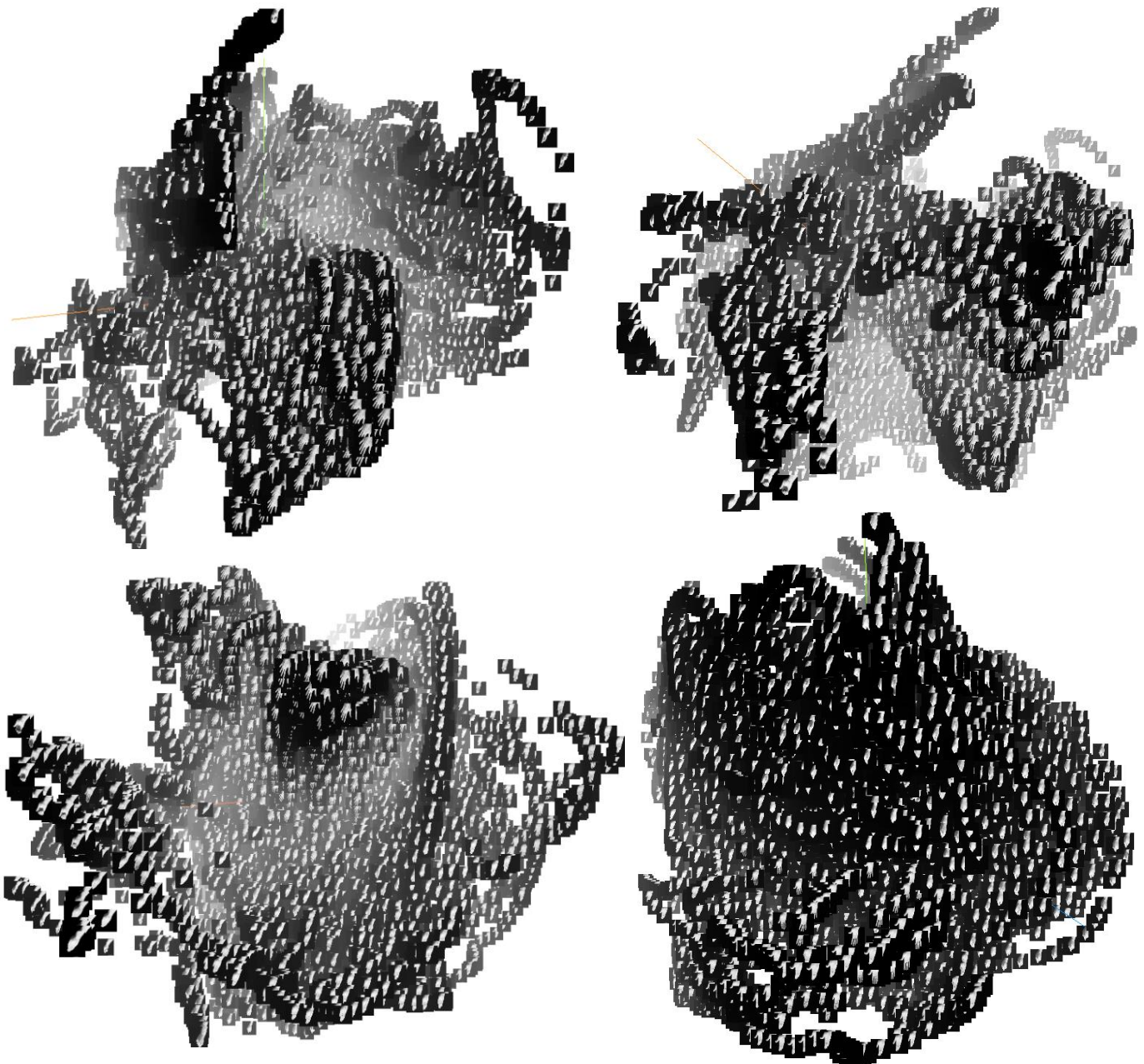


Abbildung 20: Die Mannigfaltigkeit der Hand

Quelle: Eigene Darstellung

sion scheint die schwächste zu sein, deren Ergebnisse weniger Relevanz besitzen. Bei der Hauptachsentransformation entsteht bei der Reduzierung der Daten zwangsläufig ein Datenverlust. Dieser schlägt sich anscheinend überwiegend auf der Z-Dimension aus.

Die Abbildung oben Links zeigt eine leicht versetzte Frontalansicht der Mannigfaltigkeit. Die Abbildung daneben das Seitenprofil. Die unteren Abbildungen zeigen einerseits die Unterseite des Gebildes und die Rückseite an denen sich die zwei Kuppeln überlagern.

5.3.2 Die Komponenten der Repräsentation

In dem vergangenen Abschnitt haben wir die Mannigfaltigkeit der Hand gesehen. Allerdings wurde die Repräsentation auf einen 3-dimensionalen Raum projiziert und somit wurde die Aussagekraft der einzelnen Komponenten unterschlagen. In diesen Abschnitt wollen wir uns näher mit den einzelnen gelernten Komponenten beschäftigen anhand von ausgewählten Beispielen. Präziser ausgedrückt werden wir die Veränderung des wiederhergestellten Bildes illustrieren, wenn wir die einzelnen Komponenten leicht modifizieren. Es sollte unbedingt noch erwähnt werden, dass die Ursprüngliche Annahme das die einzelnen Komponenten disjunkt sind sich nicht immer bewerkstelligen lässt. Die Komponenten sind mehr oder weniger abhängig voneinander und es entstehen viele spärliche Bereiche innerhalb der Mannigfaltigkeit. Mit anderen Worten das manipulieren von lediglich einer Komponente ist nicht zwingend aufschlussreich für das Endresultat da die Änderung einer Komponente häufig das ändern von weiteren Komponenten impliziert. Außerdem wird die Reihenfolge der Komponenten im Laufe des Trainingsprozess festgelegt. Es ist also sehr wahrscheinlich, dass die Komponenten bei mehrmaligen Trainingsdurchläufen miteinander vertauscht sind sofern der Zufallsgenerator bei der Initialisierung der Gewichte nicht näher spezifiziert wird.

Die folgende Abbildung zeigen die Ergebnisse, wenn die Komponenten leicht modifiziert wurden. Die einzelnen Zeilen zeigen die Veränderungen, die entstehen, wenn diese eine Komponente manipuliert wird und die anderen einen fixen Wert annehmen. Der Grad der Manipulation entspricht dabei einem Wertebereich von $[-4, 4]$ der mit insgesamt 15 Bildern veranschaulicht wird. Das entspricht einer Schrittweite von ungefähr 0,5 pro Bild. Die Bilder sind von einer negativen- hin zu einer positiven Veränderung hin angeordnet.

Es ist zu erkennen, dass viele der Komponenten keine oder nur eine sehr geringe Auswirkung auf das Bild zu haben scheinen. Beobachtungen der latenten Plots⁵¹ von 12 verschiedenen Bildern zeigen, dass ins besonders die Komponenten 1, 11, 13, 14 und 19 aktiv sind. Es lässt also den Schluss zu, dass die Repräsentation weiter komprimiert werden könnte bzw. weniger Dimensionen vorhanden sind als eigentlich benötigt. Geringfügige Veränderungen sind beispielsweise leichte Schattierungen oder geringes wachsen bzw. schrumpfen der einzelnen Finger. Dieses Ergebnis deckt sich auch mit unserer vorherigen Observation, dass relativ wenig Dimensionen ausreichend sind, um die Informationen des Bildes akkurat widerzugeben. Eine Vermutung besteht, dass die überflüssigen Dimensionen Archetypen der Hand darstellen, um die Zufallskomponente innerhalb der Performancefunktion auszuhebeln. Es wäre beispielsweise denkbar, dass die Gewichte zwischen der nachfolgenden Schicht und der Repräsentation eine gewisse

⁵¹ Siehe Digitalen Anhang.

Ähnlichkeit aufweisen. Eine weitere Möglichkeit wäre es sich bei diesen Komponenten um Experten-Komponenten handelt, die nur sehr wenig verwendet werden. Diese Komponenten wären dann dafür zuständig Flags innerhalb des Generierungsprozesses zu verwenden. Die folgende Tabelle zeigt observierte Auffälligkeiten innerhalb der jeweiligen latenten Plots.

	0	1	2	3	4	5	6	7	8	9	10	11
1	x	x	x		x	x	x					x
2												
3												
4							x					
5												
6												
7												
8						x						
9						x						
10												
11	x	x	x			x	x	x	x			x
12												
13	x	x	x		x	x	x	x	x	x		x
14	x	x	x		x	x	x	x	x			x
15												
16			x		x	x	x					x
17												
18												
19	x	x	x	x	x	x	x	x	x	x	x	x
20												
21												
22												
23												
24												
25												

Abbildung 21: Auffälligkeiten der latenten Plots

Die nachfolgende Abbildung zeigt den latenten Plot mit den meisten Auffälligkeiten nach 90 Trainingsepochen mit einem vollständigen Datenbestand. Jede dieser Trainingsepoche besteht aus ungefähr 310 Minibatch Einheiten die zufällig zusammengestellt wurden. Das entspricht ungefähr 28000 Optimierungsschritte.

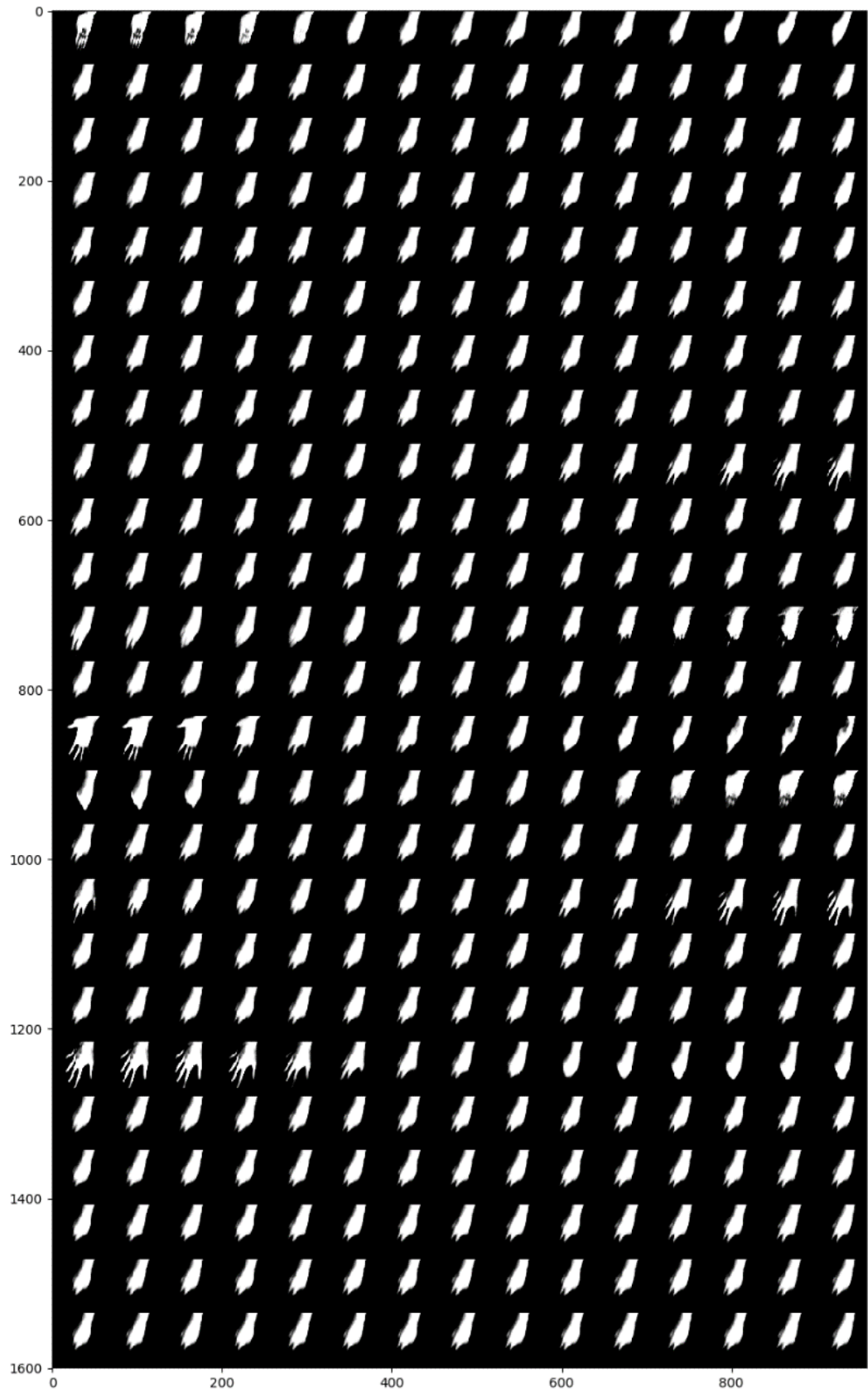


Abbildung 22: Latenter Plot Nr. 5

Quelle: Eigene Darstellung

5.4 Optimierung von Videos

Wie Anfangs beschrieben besteht die Aufgabe dieser Arbeit bei der Optimierung von Videos mithilfe von VAE. Im vergangenen Abschnitt haben wir uns mit der gelernten Repräsentation beschäftigt, die der VAE lernt um das Bild wiederherzustellen. In diesem Abschnitt gehen wir auf den Versuchsaufbau, die Probleme und die Ergebnisse der Experimente ein. Abschließend geben wir ein Fazit über die Eignungsfähigkeit dieser Methode.

5.4.1 Versuchsaufbau

Das Experiment sieht vor den vollständigen Datenbestand auszudünnen und nur einen Bruchteil des Datenbestandes im Trainingsprozess zu verwenden. Das Video dieses ausgedünnten Datenbestandes wieder vollständig zu rekonstruieren stellt das zentrale Ziel dieses Projekts dar. Mit vorheriger Überlegung zu der Mannigfaltigkeit sollte es Möglich sein zwischen den einzelnen Datenpunkten zu Interpolieren und neue bis dato unbekannte Bilder zu generieren. Wir werden insgesamt 9 verschiedene VAE-Models mit einem reduzierten Datenbestand trainieren. Das Verhältnis zwischen verworfenen- und dem verwahrten Bild lässt sich anhand der Nummer bestimmen. Die einzelnen Bilder wurden vor dem mischen des geordneten Datenbestandes entfernt um das simulieren eines nicht flüssigen Videos zu simulieren. Das Ursprüngliche Video besteht aus 30 Bildern pro Sekunde. Es werden ungefähr 6000 Optimierungsschritte pro Model angesetzt, um die Ergebnisse besser miteinander vergleichen zu können. Die spezifischen Merkmale der einzelnen Trainingseinheiten lassen sich aus der folgenden Tabelle entnehmen:

Model	Datenbestand	Frames/s	Minibatches	Trainingsepochen
2	9930	15	155	40
3	6620	10	103	60
4	4965	7,5	78	80
5	3972	6	62	100
6	3310	5	52	120
7	2837	4,3	44	140
8	2483	3,8	39	160
9	2207	3,3	34	180
10	1986	3	31	200

Die Bezeichnung des Models ist aufgrund der Übersichtlichkeit um einen Wert verschoben.

Anhand der Tabelle und der zugrundeliegenden Formel lässt sich ablesen das die einzelnen Modelle sich nicht linear unterscheiden und die Funktion gegen Null konvergiert. Der Sprung zwischen den einzelnen Modellen nimmt immer weiter ab. Beispielsweise unterscheidet sich die Ausgangslage von Model 10 und 9 nur geringfügig wohingegen der Sprung von 2 zu 3 gravierender ausfällt.

Die folgende Abbildung zeigt die Leistungen der einzelnen VAE-Models in dem Trainingszeitraum anhand der Performancefunktion. Die einzelnen Werte wurden jeweils an den Epochenenden gespeichert. Die Models einer höheren Nummer besitzen also eine verbesserte Präzision allerdings lassen sich die stochastischen Verläufe erkennen.

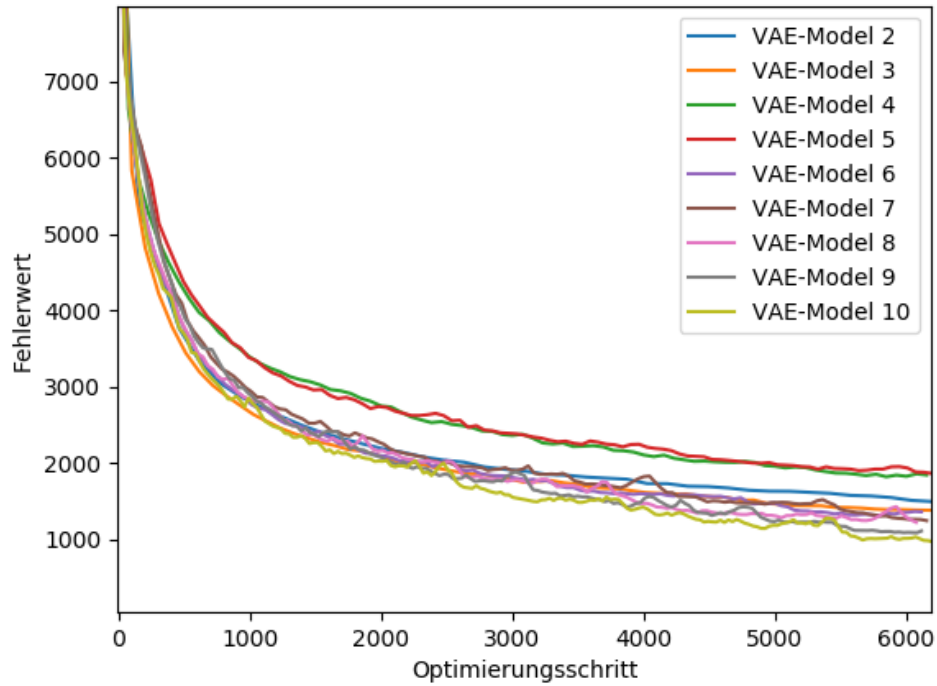


Abbildung 23: Performancemessungen der einzelnen Models

Quelle: Eigene Darstellung

Jedes der VAE-Modelle erstellt ein Video, dass sich in 4 Areale einteilen lässt. Die obere linke Ecke stellt die „Ground Truth“ dar bzw. das vollständige Video, das der VAE vorhersagen soll. Die untere Zeile entspricht dem reduzierten Video. Auf der linken Seite kann man bei VAE mit einer höheren Nummer sehr gut das stottern beobachten. Die rechte Seite hingegen bildet den eigentlichen Kern dieser Arbeit. Es ist das vorhergesagte Video, das eine künstliche höhere Bildwiederholungsrate besitzt. Um die Bildwiederholungsrate zu erhöhen wurde zwischen den einzelnen Bildern linear interpoliert. Es wird der Differenzvektor berechnet der anschließend mit einem Bruchteil multipliziert wird in Abhängigkeit des vorhergesagten Frames und anschließend zu dem Ausgangsframe hinzuaddiert:

$$Frame_{Neu} = Frame_i + (Frame_{i+1} - Frame_i) * \frac{Interpolationsschritt}{VAE_{Nr}}$$

Bei der linearen Interpolation handelt es sich um eine der einfachsten Varianten der Interpolation. Es existieren zahlreiche weitere Verfahren, die je nach Anwendungsgebiet geeigneter sind. Beispielsweise entstehen der bei der linearen Interpolation knicke an dem Graphen an den einzelnen Datenpunkten. Es handelt sich also um einen nicht differenzierbaren Graphen.

Kompliziertere Verfahren sind allerdings in der Lage die Interpolation weicher durchzuführen. Allerdings soll die Interpolation nicht das Hauptaugenmerk dieser Arbeit werden.

5.4.2 Beobachtbare Probleme

Wir wollen uns zunächst einen Überblick verschaffen über generelle Probleme bei dieser Methode sowie die Ursachen diskutieren. Wir werden uns im darauffolgenden Abschnitt die Videos nach genau diesen Kriterien hin untersuchen und erfassen ob die Probleme amplifiziert werden mit steigender Nummer oder ob die Effekte erst ab einer bestimmten Bildwiederholungsrate observierbar werden.

Die folgende Abbildung zeigt einen beispielhaften Ausschnitt eines Videos:



Abbildung 24: Unschärfe des Videos

Quelle: Eigene Darstellung

Es lassen sich bei den Videos generell eine gewisse Unschärfe des generierten Bildes feststellen. Die Schattierungen sowie die Adern sind nur leicht sichtbar. Dieses Problem war wie im Abschnitt über VAE beschrieben zu erwarten. Allerdings sind auch Verbesserungen zu beobachten. Beispielsweise haben die Finger auf der linken Seite einen eher gestuften Verlauf wohingegen auf der rechten Seite Glättungseffekt eintritt. Der Unschärfe-Effekt könnte reduziert werden indem wir die Performancefunktion leicht anpassen würden. Beispielsweise könnten wir, mit der Annahme das der Kodierer präzise ist, sämtliche Werte in der Repräsentation abspeichern und einen neuen Trainingsdatenbestand bilden. Dieser Datenbestand kann daraufhin verwendet werden, um den Generator erneut zu trainieren und nur den Rekonstruktionsfehler in Betracht zu ziehen. Mit anderen Worten die KL-Abweichung bekommt keine Relevanz mehr und es wird nur noch nach dem Rekonstruktionsfehler optimiert.

Es lassen sich noch zwei weitere deutliche Effekte beobachten. Es entstehen bei einigen Szenen einige Verunreinigen. Diese Verunreinigungen könnten Artefakte des Maskierungsprozesses sein. Die einzelnen Pixelpunkte sind schwer zu optimieren, da es sich bei der Performancefunktion um quadratische Natur handelt. Je größer die Differenz desto größer ist der Beitrag an der Performance. Zusätzlich ist zu dem einzelnen Pixel die Koordinate notwendig, um exakt an dieser Stelle eine Pigmentierung durchzuführen. Allerdings treten diese Effekte in größeren Arealen auf und lässt den Schluss zu, dass der VAE schwarze Farbe auf einen weißen Hintergrund aufträgt als umgekehrt. Die Erklärung könnte auch das bilden von Schattierungen erklären, da die einzelnen Schattierungen überlagert werden können und somit die Intensität des Pixels bzw. des Pixelbereichs reduzieren können. Das Problem könnte vermutlich mit einem professionellem Videosetting reduziert werden allerdings ist gerade diese Herausforderung für den VAE interessant zu beobachten. Eine weitere Problematik, die häufig bei Models mit einer höheren Nummer auftreten, sind teilweise groteske Verformungen der Hand. Dieser Effekt lässt sich häufig bei abrupten Veränderungen der Hand beobachten. Vermutlich ist dieser Effekt eng an der Interpolationsmethode gekoppelt da wir, mit respektive der Mannigfaltigkeit, unzulässige Bereiche nutzen. Das ist vergleichbar mit dem Einsetzen eines Wertes, das sich nicht im Definitionsbereich befindet. Das Problem bei der Interpolation ist, dass zwischen den Einzelbildern keine weiteren Informationen bestehen und nur ein Pfad von einer willkürlichen Menge an Pfaden verwendet wird. Beispielsweise wenn Bild 1 und Bild 3 identisch wären und wir das Bild 2 verworfen hätte das ein komplett anderen Sachverhalt darstellt hätten wir keine Möglichkeit an diese Informationen zu gelangen. Bei der Interpolation würden die Bilder 1, 2 und 3 identisch sein ist aber ein Trugschluss. Dieser Effekt tritt insbesondere bei schnellen Veränderungen sowie einer geringen Bildwiederholungsrate ein.

Die folgende Abbildung zeigen eine mittlere Verunreinigung des Bildes:



Abbildung 25: Bildverunreinigung
Quelle: Eigene Darstellung

5.4.3 Ergebnisse

Einige kritische interessante Höhepunkte des Videos deren es einer genauen Beobachtung bedarf sind die folgenden:

- 3:01 – Bewegung des Daumens bei vertikaler Fauststellung
- 4:23 – Bewegung des Daumens bei horizontaler Fauststellung
- 4:31 – Auf- und Abwärtsbewegungen der Faust
- 6:10 – Bewegungen der unteren Fingerglieder
- 6:55 – Diverse Auf- und Abwärtsbewegungen der Hand
- 7:13 – Position- und Winkelveränderung der Hand
- 8:05 – Öffnen und Schließen der ausgestreckten Hand
- 8:33 – Schnelle Handbewegungen
- 9:45 – Daumenbewegungen und einzelne Fingerbewegungen

In der folgenden Tabelle sind die einzelnen Beobachtungen der jeweiligen VAE-Models aufgelistet:

VAE-Nr.	Beobachtungen:
2	<ul style="list-style-type: none"> • Die Unschärfe hält sich in Grenzen. • Geringes flimmern • Sehr wenige grotesken Veränderungen • Unterschied der Bildwiederholungsrate des Videos kaum wahrnehmbar • Sämtliche Bewegungen wurden richtig erfasst
3	<ul style="list-style-type: none"> • Die Unschärfe hält sich in Grenzen.

	<ul style="list-style-type: none"> • Geringes flimmern • Sehr wenige grotesken Veränderungen • Unterschied der Bildwiederholungsrate des Videos wahrnehmbar • Ein paar Bewegungen wurden nicht richtig erfasst
4	<ul style="list-style-type: none"> • Die Unschärfe ist deutlicher wahrnehmbar als in vorherigen Versuchen • Wenig flimmern • Groteske Veränderungen sind deutlich häufiger als zuvor • Unterschied der Bildwiederholungsrate des Videos deutlich wahrnehmbar • Ein paar Bewegungen wurden nicht richtig erfasst bzw. lassen sich nicht eindeutig zuordnen • Besondere Notiz: Bei 7:19 wird eine ausgestreckte Hand mit einer Faust verwechselt. Es macht den Eindruck als würden zu der Faust ein paar Finger hinzugefügt werden da die Formen sich zu ähneln scheinen.
5	<ul style="list-style-type: none"> • Besondere Notiz: Am Anfang des Videos entstehen oft die falschen Handgesten. Es scheint als ob die Mannigfaltigkeit eine komplett andere Struktur zu haben scheint als die anderen Models. • Geringes Flimmern • Bewegungen werden teilweise falsch erfasst • Teilweise Groteske Veränderungen treten häufig ein • Die Daumenbewegungen werden korrekt erfasst. • Einige Handgesten sind nicht mehr eindeutig zuordbar • Unterschied der Bildwiederholungsrate des Videos ist deutlich spürbar allerdings werden teilweise falsche Bilder gezeigt. Es ist nicht bekannt ob die Ursache die Interpolation ist oder ob der Generator falsch liegt. • Es handelt sich bei dem Video keinesfalls um eine Optimierung. Es werden zwar einige Handgesten flüssiger dargestellt allerdings stehen diese aber in keinem akzeptablen Verhältnis zu der Anzahl der falsch gezeigten Bilder
6	<ul style="list-style-type: none"> • Deutliche Verbesserung gegenüber dem Model-Nr. 5 • Normale Unschärfe feststellbar • Wenig flimmern • Groteske Veränderungen sind seltener als zuvor • Unterschied der Bildwiederholungsrate des Videos deutlich wahrnehmbar • Es werden nahezu alle Gesten richtig erfasst
7	<ul style="list-style-type: none"> • Leichtes Flimmern • Teilweise Unschärfe und nicht direkt zuordnungsbarer Bilder • Sämtliche Gesten wurden richtig erfasst • Deutliche verbesserte Bildwiederholungsrate die Verläufe werden richtig eingeschätzt
8	<ul style="list-style-type: none"> • Wenige unscharfe verwechselte Bilder • Es entstehen ein paar nicht zuordnungsbarer Bilder • Es entsteht ein deutlich flüssigeres Video deren mit einem meist klaren Verlauf • Kaum flimmern • Nahezu sämtliche Bewegungen wurden richtig erfasst
9	<ul style="list-style-type: none"> • Teilweise unscharfe nicht zuordnungsbarer Bilder • Relativ häufiges Flimmern • Nahezu alle Bilder wurden richtig erfasst

	<ul style="list-style-type: none"> • Es entsteht ein deutlich flüssigeres Video deren mit einem meist klaren Verlauf. • Die Interpolationsschritte sehen den ursprünglichen Bildern erstaunlich ähnlich
10	<ul style="list-style-type: none"> • Teilweise unscharfe nicht zuordnungsbare Bilder • Kaum flimmern • Nahezu alle Handgesten wurden richtig erfasst. • Es entsteht ein deutlich flüssigeres Video deren mit einem meist klaren Verlauf. • Die Interpolationsschritte sehen den ursprünglichen Bildern erstaunlich ähnlich • Besondere Notiz: Schöner Bewegungsfluss an der Stelle 7:18

Generell lässt sich eine Unschärfe in den einzelnen Videos feststellen. Allerdings in Kombination mit dem falschen Erfassen von Gesten entstehen teilweise nicht zuordnungsbare Bilder. Es ist erstaunlich das diese Effekte so stark von den einzelnen Models abweichen. Prinzipiell wurden die Modelle nur im geringen Maße trainiert. Bei einem längeren Training sollten diese Effekte deutlich reduziert werden. Anders sieht es allerdings bei einer falschen Mannigfaltigkeit aus diese nachträglich zu korrigieren würde sich meiner Meinung nach als deutlich schwieriger erweisen. Besonders auffällig scheint das Model Nr. 5 zu sein. Die Mannigfaltigkeit scheint sich Grundlegend von den anderen zu unterscheiden. Ein Blick auf den latenten Plot wie beispielsweise latentplot_5_z25_e100 zeigen sehr häufig redundante Informationen. Es besteht die Möglichkeit das diese redundanten Informationen im Verlauf des Trainingsprozess weniger werden. Die letzten beide Modelle veranschaulichen allerdings das es sehr wohl möglich ist die Bildwiederholungsrate künstlich zu erhöhen für den Preis eines unschärferen Videos. Die Performance der einzelnen Models scheinen lassen sich anhand der Performancemessungen bereits erahnen da beispielsweise das Model Nr. 5 den schlechtesten Performancewert liefert.

5.5 Fazit des Experiments

Wir haben gesehen, dass es sehr wohl möglich ist die Bildwiederholungsrate künstlich zu erhöhen. Diese setzt allerdings voraus das eine geeignete Struktur der Mannigfaltigkeit gelernt wird. Für das lernen der Mannigfaltigkeit ist der VAE allerdings prädestiniert und stellt meiner Meinung nach das Haupteinsatzgebiet des VAE dar. Ich denke das die Nachteile des VAE den Vorteilen überwiegen und würde von der Verwendung abraten.

Das Projekt würde ich dennoch als erfolgreich bezeichnen da nur relativ wenig Trainingsproben ausreichend sind, um vorzeigbare Ergebnisse zu erzielen. Allerdings lässt sich über die realen Anwendungsgebiete streiten da sich das Video nur auf den Aspekt der Hand beschränkt. Würden andere Elemente, verschiedene Hintergründe und diese in einer unterschiedlichen

Häufigkeit gezeigt werden hätte der VAE deutlich mehr Probleme. Das gesamte Experiment beschränkt sich nur ein relativ niedrig aufgelöstes Bild. Es ist nicht klar ob sich die Beobachtungen auch auf ein hochaufgelöstes Video übertragen lässt. Meiner Meinung nach wäre eine Lösung, wenn wir ein Video mit einer sehr geringen Bildwiederholungsrate hätten, dass wir ein weiteres Video aufnehmen, das wir zum lernen der Mannigfaltigkeit verwenden können. Die Kombination von einzelnen Datenpunkten und einer detaillierten Mannigfaltigkeit würde zu hervorragenden Ergebnissen führen. Mit einer besseren Interpolationsmethode würden sich vermutlich auch Videos mit einer deutlich geringeren Bildwiederholungsrate restaurieren lassen indem wir einen geeigneten Pfad auf der Mannigfaltigkeit finden.

6 Ausblick

Ich halte das maschinelle Lernen und insbesondere das ungestützte Lernen für ein Forschungsgebiet mit weitreichender Zukunft. Die Menge an produzierten Daten wird in der Zukunft deutlich zunehmen insbesondere durch die wachsende Anzahl an Computern und dem „Internet of things“. Die Technologie, um diese Datenmengen auszuwerten ist meiner Meinung nach allerdings noch nicht vollständig ausgereift. Viele der Modelle des maschinellen Lernens leben von der schieren Fülle an Daten. Dies ist meiner Meinung nach allerdings der falsche Ansatz da beispielsweise Menschen durchaus in der Lage sind aus vergleichbar minimalen Informationen trotzdem noch geeignete Informationen zu ziehen. Es existieren anscheinend noch nicht entdeckte Algorithmen die deutlich leistungsfähiger als die heutigen sind.

Die Auswirkungen dieser neuen Technologien werden die Welt grundlegend verändern. Sei es die Arbeit, die Kommunikation sowie soziale Systeme. Allerdings werden diese neuen Technologien sich auch in einer emotionalen, noch nicht einschätzbaren Weise, auf die Menschen auswirken. Ich denke das die Arbeit sehr an das Wohlergehen des Menschen geknüpft ist und sehe daher sowohl Potenzial für herausragende positive Dinge für die Menschheit wie eine verbesserte medizinische Versorgung sowie auch die negativen Potenziale wie ein Machtungleichgewicht zwischen der Bevölkerung und einzelnen Individuen.

Abschließend lässt sich sagen das, auch wenn in der Zukunft der VAE mit einer anderen Technologie abgesetzt wird, es sich um eine herausragende Technologie handelt, die einen weiten Anwendungsspielraum besitzt.

BESCHREIBUNG DES COST PLOTS

GEHE NÄHER AUF DIE VIDEOS EIN AUFBAU AREALE

BESCHREIBE DIE EFFEKTE

Probleme bei der Interpolation

FAZIT ZIEHEN eventuell festsetzen und weiter optimieren

Vorderseite Seitenansicht

Unterseite Rückseite anbindungspunkt der Kupeln

Projekt

Projektbeschreibung

Data Preprocessing

Model architecture

Durchführung

Ergebnisse

Evaluation

Dasd

Autoencoder

Mannigfaltigkeit

Hautachsentransformation

Autoencoder

Variational Autoencoder

Projekt

Datensatz

Problemstellung

Vorgehensweise

...

Text $\underbrace{\hspace{1.5cm}}_{below}$

Literaturverzeichnis

// Wird später ausgefüllt... Es sind noch die alten Daten von dem Muster drin.

Mitchell, Tom M. Mitchell (1997): Machine Learning, McGraw-Hill Science/Engineering/Math.

Goodfellow, Ian Goodfellow, Yoshua Bengio und Aaron Courville (2016): Deep Learning, The MIT Press.

Kriesel, David Kriesel (2007): A Brief Introduction to Neural networks, available at <https://www.dkriesel.com>

McCulloch, Warren S. McCulloch und Walter H. Pitts (1943): A logical calculus of the ideas immanent in nervous activity, in: Bulletin of Mathematical Biophysics, Band 5, S. 115-133.

Rey, Günter Daniel Rey und Karl F. Wender (2011): Neuronale Netze: Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung, 2. Auflage, Bern, Huber.

MacKay, David J. C. MacKay (2005): Information Theory, Inference, and Learning Algorithms, 7.2 Auflage, Cambridge University Press.

Blei, David M. Blei, Alp Kucukelbir und Jon D. McAuliffe (2018): Variational Inference: A Review for Statisticians, Online im Internet: <https://arxiv.org/pdf/1601.00670.pdf>, abgerufen am: 3.12.2018.

Salimans, Tim Salimans, Diederik P. Kingma und Max Welling (2014): Markov Chain Monte carlo and Variational Inference: Bridging the Gap, Online im Internet: <http://proceedings.mlr.press/v37/salimans15.pdf>, abgerufen am: 4.12.2018.

Raschka, Sebastian Raschka (2017): Machine Learning mit Python: Das Praxis-Handbuch für Data Science, Predictive Analytics und Deep Learning, 1. Auflage, Frechen, mitp.

Kingma, Diederik P. Kingma und Max Welling (2013): Auto-Encoding Variational Bayes, Online im Internet: <https://arxiv.org/pdf/1312.6114.pdf>, abgerufen am: 4.12.2018.

Higgins, Irina Higgins et al. (2016): Early Visual Concept Learning with Unsupervised Deep Learning, Online im Internet: <https://arxiv.org/pdf/1606.05579.pdf>, abgerufen am: 4.12.2018.

Britz, Denny Britz (2015): Understanding Convolutional Neural Networks for NLP, Online im Internet: <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>, abgerufen am: 4.12.2018.

Rey, Günter Daniel Rey (2017): Methoden der Entwicklungspsychologie – Datenerhebung und Datenauswertung – Neuronale Netze, Online im Internet: http://www.methoden-psychologie.de/neuronale_netze.html, abgerufen am: 4.12.2018.

Ganesh, Tinniam V Ganesh (2011): Simplifying Machine Learning: Bias, Variance, regularization and odd facts – Part 4, Online im Internet: <https://gigadom.in/2014/01/03/simplifying-machine-learning-bias-variance-regularization-and-odd-facts-part-4/>, abgerufen am: 4.12.2018.

Fumo, David Fumo (2017): A Gentle Introduction To Neural Networks Series — Part 1, Online im Internet: <https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc>, abgerufen am: 4.12.2018.

Vgl. Higgins (2016), S. 2 f. <https://arxiv.org/pdf/1606.05579.pdf>

Vgl. Kingma (2013), S. 1-4. <https://arxiv.org/pdf/1312.6114.pdf>

<https://arxiv.org/pdf/1601.00670.pdf> S8

Vgl. Blei (2018) <https://arxiv.org/pdf/1601.00670.pdf> S. 6f.

Albers, S./Peters, H. (2007): Emission Trading – Strategy Implications for Airlines, in: Association for European Transport (Hrsg.): Proceedings of the 2007 European Transport Conference, Noordwijkerhout.

Brack, D./Grubb, M., Windram, C., et al. (2000): International trade and climate change policies, London, Earthscan.

Bauchmüller M./Gammelin, C. (2010): Deutscher Streit blockiert Klimaplan/Emissionshandel für Flugzeuge auf der Kippe, in: Süddeutsche.de, 17.05.2010, Online im Internet: <http://www.sueddeutsche.de/wirtschaft/emissionshandel-fuer-flugzeuge-auf-der-kippe-deutscher-streit-blockiert-klima-plan-1.219438>, abgerufen am 17.02.2015.

Brack, D./Grubb, M., Windram, C., et al. (2000): International trade and climate change policies, London, Earthscan.

Coase, R. H. (1960): The Problem of Social Cost, in: Journal of Law and Economics Vol. 3. (Oct. 1960), S. 1-44.

o.V. (2011): Opel kämpft um Bochum, in: Frankfurter Allgemeine Zeitung, Nr. 15, 18.03.2011, S. 5.

Europäische Union (2009): Amtsblatt der Europäischen Union ABl. L 103 vom 23.4.2009, S. 10-29 (DE), Online im Internet: <http://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CLEX:32009D0339&rid=1>, abgerufen am 17.02.2015.

Fees, E. (2015): Art. Coase-Theorem, in: Gabler Wirtschaftslexikon/Das Wissen der Experten, Online im Internet: <http://wirtschaftslexikon.gabler.de/Definition/coase-theorem.html>, abgerufen am 17.02.2015.

International Civil Aviation Organisation (2007): ICAO Environmental Report 2007, Online im Internet: http://www.icao.int/environmental-protection/Documents/Env_Report_07.pdf, abgerufen am 18.02.2015.

Umweltbundesamt (2015): Der Europäische Emissionshandel, Online im Internet: <http://www.umweltbundesamt.de/daten/klimawandel/der-europaeische-emissionshandel>, abgerufen am 19.02.2015.

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt und mich anderer als der in den beigefügten Verzeichnissen angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Der Durchführung einer elektronischen Plagiatsprüfung stimme ich hiermit zu. Die eingereichte elektronische Fassung der Arbeit entspricht der eingereichten schriftlichen Fassung exakt. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen hat.

Ort, den

Vorname, Name