

BACHELORARBEIT

Data Augmentation mit Variational Autoencodern

vorgelegt von

Paul Henri Iser

geboren am 02. Oktober 1998 in Bonn, Deutschland

1. Prüfer/-in: Dr. Pascal Welke
Universität Bonn

2. Prüfer/-in: Prof. Dr. Christian Bauckhage
Universität Bonn

Betreuer/-innen: Florian Seiffarth
Universität Bonn

24. April 2021

Zusammenfassung

In dieser Arbeit werden verschiedene Data Augmentation Ansätze basierend auf Variational Autoencodern untersucht. Im Fokus stehen die Analyse des erzeugten Latent-Space und die Augmentierung unterschiedlicher Datensätze, wie MNIST, CelebA und PROBEN1. Über reduzierte Varianten dieser Datensätze werden die vorgestellten Methoden auch in Few-Shot Learning Szenarios evaluiert. Die Ergebnisse offenbaren, dass Variational Autoencoder auf kleinen Datenmengen zu einer leichten Verbesserung der Performanz führen. Für größere Datensätze sind die generierten Beispiele jedoch weniger hilfreich. Hier erzielen andere Ansätze, wie Generative-Adversarial-Networks deutlich bessere Ergebnisse. Es wird sich außerdem herausstellen, dass die erzeugten Daten im Fall von Bilddaten eine starke Unschärfe aufweisen. Dennoch bietet der Variational Autoencoder viele Vorteile durch seine Latent-Space Struktur, denn dies erlaubt kontrollierbare Modifikationen der Daten.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Bonn, 24. April 2021

Inhaltsverzeichnis

1 Einführung	1
2 Wissenschaftlicher Hintergrund	3
3 Literatur	5
3.1 Few-Shot Learning	5
3.2 Autoencoder	6
3.2.1 Denoising Autoencoder	7
3.3 Variational Autoencoder	8
3.3.1 Latente Variablen	8
3.3.2 Kullback-Leibler Divergenz	9
3.3.3 Optimierung der VAE Zielfunktion	11
3.3.4 Reparametrisierungs-Trick	11
4 Datensätze	13
4.1 MNIST	13
4.2 PROBEN1	14
4.3 CIFAR-10	14
4.4 Large-scale CelebFaces Attributes	15
5 Methodik	17
5.1 Latent-Space Sampling	17
5.2 Single-VAE	17
5.3 Multi-VAE	18
5.4 Generative Classifiers	19
5.5 Details zu den VAE Modellarchitekturen	20
5.5.1 Hyperparameter für KL-Divergenz Gewichtung	20
6 Evaluation	23
6.1 Metriken	23
6.1.1 Few-Shot Szenario	24
6.2 Latent-Space Analyse	24
6.3 PROBEN1	28
6.3.1 Performanz auf dem gesamten Datensatz	29
6.3.2 Few-Shot Szenario	30

INHALTSVERZEICHNIS

6.4	MNIST	31
6.4.1	Few-Shot Szenario	31
6.4.2	Performanz auf nur generierten Daten	32
6.4.3	MNIST Few-Shot Szenario - Mehr generierte Daten	33
6.5	CelebA	34
6.6	Limitationen von VAE	36
7	Fazit	39
8	Ausblick	41
8.1	Few-Shot Learning	41
8.2	Weitere VAE Modelle	41
8.2.1	Denoising und Conditional VAE	41
8.2.2	VAE-GAN	42
Literatur		43

Kapitel 1

Einführung

Neuronale Netzwerke erwiesen sich in den letzten Jahren als eine der mächtigsten Ansätze im Bereich des maschinellen Lernens. Durch immer tiefere Modellarchitekturen konnten auch komplexere Aufgaben gelöst werden (z.B. AlphaFold, Senior u. a., 2020). Insbesondere für die Aufgabe der Klassifikation dominieren Neuronale Netze den *State-of-the-Art*. Dazu werden üblicherweise große Datenmengen benötigt, welche in vielen Szenarien jedoch nicht zur Verfügung stehen. Im Extremfall sind nur wenige Beispiele pro Klasse vorhanden (*Few-Shot Learning*). Die Gesichtserkennung, wie sie mittlerweile in den meisten modernen Smartphones zu finden ist, stellt ein Beispiel für einen solchen Fall dar. Ein Möglichkeit dieses Problem zu lösen bietet das künstliche Erweitern der gegebenen Daten.

Data Augmentation hat sich als eine der Standard Methoden im Umgang mit kleinen oder imbalancierten Datensätzen etabliert. Üblicherweise werden dazu Beispiele aus dem Datensatz über unterschiedliche Transformationen (z.B. Rotation, Translation, Skalierung) augmentiert. Mit Hilfe dieser Methode konnten bemerkenswerte Ergebnisse, insbesondere in der Bildverarbeitung, erzielt werden. Dieser Prozess baut jedoch häufig auf Expertenwissen darüber auf, welche Transformationen für das Erlernen der Aufgabe hilfreich sind. Ohne dieses Expertenwissen können unplausible Beispiele entstehen, welche die Performanz negativ beeinträchtigen.

Alternativ bieten Generative Modelle eine effiziente Methode für die automatisierte Erzeugung von Daten. Mittels tiefer konvolutionaler Netzwerk Architekturen wird eine immer bessere Qualität der generierten Daten erreicht. Somit stellen sie inzwischen eine effiziente Alternative zu den klassischen Data Augmentation Methoden dar. Zudem bieten sie den Vorteil, dass im Allgemeinen kein Expertenwissen benötigt wird. So zeigten sie in verschiedensten Bereichen beachtliche Ergebnisse (z.B. GPT-3, Brown u. a., 2020).

In der vorliegenden Arbeit wird der Variational Autoencoder (VAE) als generatives Modell betrachtet. Der übliche Nutzen von Autoencodern liegt in einer semantischen Repräsentation der Eingabe. Dies wird über eine Encoder-Decoder Architektur erreicht. Variational Autoencoder erweitern diese Struktur, um eine probabilistische Repräsen-

KAPITEL 1. EINFÜHRUNG

tation der Eingabe in einem *Latent-Space* zu erzeugen. Diese Arbeit teilt sich in einen theoretischen und einen praktischen Fokus auf. Der theoretische Fokus liegt in der Analyse der Latent-Space Struktur. Insbesondere wird sich mit der Fragestellung beschäftigt, wie diese für die Erzeugung neuer Daten verwendet werden kann. Der praktische Fokus liegt auf der Evaluation verschiedener VAE basierter Data Augmentation Ansätze auf unterschiedlichen Datensätzen. Diese beinhalten sowohl Bild-, als auch numerische Daten. Zudem wird der Einfluss der VAE basierten Ansätze in Few-Shot Szenarien untersucht. Abschließend werden diese Verfahren mit anderen generativen Methoden verglichen.

Kapitel 2

Wissenschaftlicher Hintergrund

In den letzten Jahren wurden bereits einige Arbeiten zu der Augmentation von Daten über Generative Modelle veröffentlicht. Einige dieser Arbeiten stellen Konzepte vor, welche auch in dieser Arbeit verwendet werden. Diese werden im Folgenden erläutert.

Jorge u. a., 2018 untersuchen empirisch, wie die Verwendung von Variational Autoencoder zur Data Augmentation, die Klassifikation auf dem MNIST und Omniglot Datensatz verbessert. Außerdem stellen sie einige Methoden des Latent-Space Samplings vor, welche auch in dieser Arbeit aufgegriffen werden. Garay-Maestre, Gallego und Calvo-Zaragoza, 2019 behandeln zusätzlich die Verbesserungen, die sich auf reduzierten Partitionen von MNIST erzielen lassen. Zudem wird die Verwendung von einem separaten Modell je Klasse motiviert. Dieser Ansatz wird in der vorliegenden Arbeit aufgegriffen und auf weitere, unter anderem numerische Daten angewendet.

Moreno-Barea, Jerez und Franco, 2020 beschäftigen sich in ihrer Arbeit mit der Anwendung verschiedener Generativer Modelle auf kleinen Datensätzen. Im Gegensatz zu den anderen Arbeiten in diesem Bereich, werden hier keine Bilddaten untersucht. Stattdessen werden numerische Daten mit teils diskreten Attributen behandelt. Außerdem stellen sie einen Data Augmentation Prozess vor, welcher zusätzlich eine Filterung der generierten Daten nutzt, um unplausible Beispiele auszusortieren. Damit soll die Performanz weiter verbessert werden. Da hier allerdings mehrere Generative Ansätze untersucht werden, macht der Variational Autoencoder nur einen Teil dieser Arbeit aus. Dieser wird in der vorliegenden Arbeit genauer analysiert.

Higgins u. a., 2017 schlagen eine β Erweiterung zum Variational Autoencoder vor, welche die Korrelation erlernter Merkmale beeinflusst. In der Arbeit werden ausschließlich Bilddaten untersucht. Der Fokus der Autoren liegt auf der Qualität der generierten Bilder und der unabhängigen Modifikation von Merkmalen. In der vorliegenden Bachelorarbeit wird darüber hinaus die Auswirkungen auf die Klassifikationsaufgabe für sowohl Bild-, als auch numerische Daten, behandelt.

KAPITEL 2. WISSENSCHAFTLICHER HINTERGRUND

I. J. Goodfellow u. a., [2014](#) stellen mit "Generative-Adversarial-Networks" eine weitere Architektur eines generativen Modells vor. Diese bietet einige Vorteile gegenüber Variational Autoencodern. Unter anderem werden sehr viel schärfere Bilder generiert, dafür erlauben sie jedoch keine Kontrolle über die generierten Beispiele. Ein Vergleich zu dieser Methode ist einer der zu untersuchenden Aspekte dieser Arbeit.

Kapitel 3

Literatur

Im Folgenden werden wichtige Begrifflichkeiten und Herleitungen, welche in dieser Arbeit verwendet werden, vorgestellt. Weitere theoretischen Grundlagen zu den vorgestellten Konzepten finden sich in den Arbeiten von I. Goodfellow, Bengio und Courville, 2016, Kingma und Welling, 2014.

3.1 Few-Shot Learning

Viele der heutigen *State-of-the-Art* Modelle bauen auf einer riesigen Anzahl von Trainings Beispielen auf. Diese Anforderung kann aber nicht immer erfüllt werden. Bei nur wenig Beispielen ist daher die Gefahr des "Overfittings", d.h. des Auswendig Lernens des Datensatzes, entsprechend groß. Der Forschungsstrang "Few-Shot Learning" beschäftigt sich mit der Aufgabe, eine gute Generalisierung aus nur wenigen Daten zu Erzielen. Für n Trainingsbeispiele pro k Klassen spricht man von n -way k -shot Learning. Sung u. a., 2017 erwähnen "Meta-Learning" als vielversprechenden Ansatz. Dieser geht, nach dem Paradigma "Lernen-zu-lernen" auf die Art zurück, wie Menschen lernen: Kinder haben kein Problem, aus einem einzelnen Bild das Objekt "Zebra" zu verstehen, da sie bereits das Konzept "Pferd" und "Farben" kennen. Kernidee in der von Sung u. a., 2017 veröffentlichten Arbeit ist es eine Ähnlichkeitsfunktion zwischen Bildobjekten zu lernen. Realisiert wird dies durch Lernen eines Embedding-Vektors mit dem Ziel, dass ähnliche Objekte in diesem Raum nahe beieinander liegen. So können ungewöhnliche Beispiele über einen "Nearest-Neighbor-Classifier" bekannten Beispielen zugeordnet und klassifiziert werden.

Ein alternativer Ansatz ist die Nutzung von Data Augmentation, um mehr Daten aus den vorhandenen zu erzeugen. Ein Vorteil dieses Weges besteht darin, dass für die Klassifikation wieder auf *State-of-the-Art* Modelle zurückgegriffen werden kann. Antoniou, Storkey und Edwards, 2017 zeigen, dass auch Generative Modelle dazu verwendet werden können, um Data Augmentation zu betreiben. Mit Hilfe von vielen generierten Daten können gute Generalisierungen erzielt werden.

3.2 Autoencoder

Der Autoencoder (AE) stellt eine Encoder-Decoder Architektur dar (siehe Abb. 3.1. Aufgabe des Encoders f ist es, die Eingabe x in einen Merkmalsraum abzubilden. Anschließend versucht der Decoder g die Eingabe zu rekonstruieren. Die Ausgabe des Autoencoders ist somit die Komposition $g(f(x))$). Um eine präzise Rekonstruktion zu erzielen, werden Fehlerfunktionen der folgenden Art verwendet:

$$\mathcal{L}(x, g(f(x))) \quad (3.1)$$

Demnach können Autoencoder in einem selbst-überwachten Szenario trainiert werden. Übliche Fehlerfunktionen zur Bewertung der Diskrepanz zwischen x und $g(f(x))$ sind "Binary-Cross-Entropy-Loss" und "Mean-Squared-Error". In Bezug auf die Dimension des Merkmalsraums unterscheidet man zwei Fälle.

Der erste Fall ist der sogenannte "undercomplete" AE. Hierbei wird die Dimension des

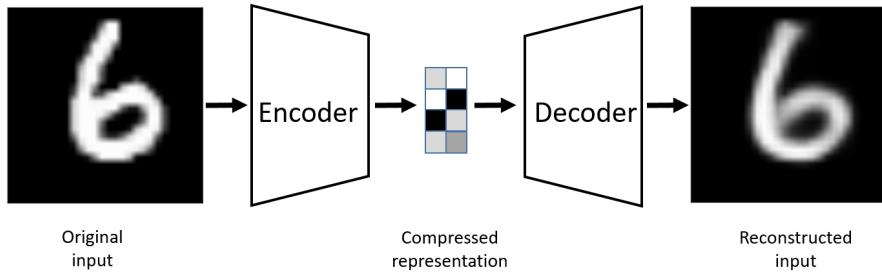


Abbildung 3.1: Die Encoder-Decoder Struktur des "undercomplete" Autoencoders.
Der Encoder berechnet eine komprimierte Repräsentation der Eingabe.
Der Decoder ist in der Lage diese wieder zu einem Bild zu rekonstruieren. Abb. entnommen aus Lopez Pinaya u. a., 2019.

Merkmalsraums kleiner als die der Eingabe gewählt, sodass dieser einen Flaschenhals für den Informationsfluss bildet. Der Encoder komprimiert die Eingabe und stellt so eine Dimensionsreduktion dar. Falls der Autoencoder ($f \circ g$) durch eine lineare Abbildung definiert ist, entspricht diese Dimensionsreduktion einer "Principle-Component-Analysis" (PCA) (siehe Ladjal, Newson und Pham, 2019). Eine Definition über nicht-lineare Abbildungen ist deutlich mächtiger. An dieser Stelle bieten sich Neuronale Netze an, da sie beliebige, insbesondere nicht-lineare Funktionen, approximieren können. Zudem sind (Konvolutionale-) Neurale-Netzwerke (CNN) eine weit verbreitete Methode der automatisierten Merkmalsextraktion. Ein üblicher Nutzen von "undercomplete" Autoencodern ist eben dieser komprimierte Merkmalsvektor als Repräsentant der Eingabe.

Der zweite Fall ist der "overcomplete" Autoencoder. Anders als zuvor wird hier die Dimension des Merkmalsraums größer als die der Eingabe gewählt. Dadurch bildet der Encoder keinen Flaschenhals und ist folglich auch nicht gezwungen, Merkmale zu extrahieren. Der "overcomplete" Autoencoder lernt den Datensatz auswendig ("Overfitting"). Dies führt zum Strukturverlust im Merkmalsraum. Um diesem Verhalten entgegen zu wirken, werden Regularisierungstechniken angewandt. Eine dieser Regula-

risierungen, Denoising Autoencoder, wird im Folgenden vorgestellt. Weitere Regularisierungstechniken sind z.B. "Sparse-Autoencoder", "Contractive-Autoencoder" und "Conditional-Autoencoder" (nachzulesen in Lopez Pinaya u. a., 2019). Man bemerke, dass diese Varianten auch bei "undercomplete" Autoencodern zu einem stabileren Training beitragen können.

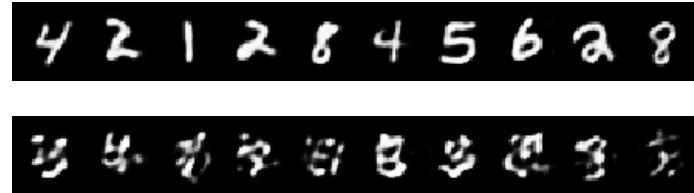


Abbildung 3.2: Zwei Rekonstruktionen von Vektoren aus dem Feature Space. Die Beispiele oben sind ein codierte Eingaben, welche anschließend decodiert wurden. Unten dargestellt sind zufällige decodierte Merkmalsvektoren.

3.2.1 Denoising Autoencoder

Der Denoising Autoencoder (DAE) betrachtet statt der Eingabe x ein modifiziertes \tilde{x} . Die Modifikation wird durch Addition von Rauschen z.B. aus einer uniformen oder einer Normalverteilung erreicht. Diese Regularisierungstechnik führt offensichtlich dazu, dass der DAE robust gegenüber Rauschen in den Eingabebildern wird. Außerdem erfolgt eine bessere Exploration des Merkmalsraumes. DAEs verwenden Fehlerfunktionen der folgenden Art:

$$\mathcal{L}(x, g(f(\tilde{x}))) \quad (3.2)$$

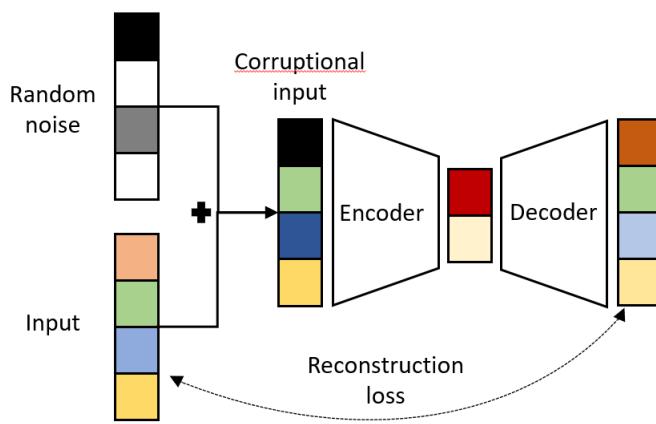


Abbildung 3.3: Der DAE minimiert den Fehler zwischen der Rekonstruktion des modifizierten Eingabevektors und der Eingabe. Abbildung entnommen aus Lopez Pinaya u. a., 2019

KAPITEL 3. LITERATUR

Abb. 3.2 zeigt zwei Rekonstruktionen von Merkmalsvektoren. Man beachte, dass ein zufälliges Sample aus dem Merkmalsraum mehr einem Rauschen als einer tatsächlichen Klasse aus dem Datensatz ähnelt. Denn die Räume zwischen den codierten Merkmalsvektoren sind im Allgemeinen nicht definiert. Der Merkmalsraum besitzt keine "geometrische" Struktur. Um dieses Problem zu lösen wurde eine weitere Klasse von Autoencodern entworfen: Variational Autoencoder.

3.3 Variational Autoencoder

Kingma und Welling, 2014 betrachten in ihrer Arbeit das Konzept des Autoencoders aus einer probabilistischen Sicht. Der Encoder bildet die Eingabe nicht auf einen eindeutigen Merkmalsvektor, sondern auf eine Wahrscheinlichkeitsverteilung über den Merkmalsraum ab. Damit bekommt der Merkmalsraum eine probabilistische Struktur. Die Herleitung des Variational Autoencoders erfolgt üblicherweise über das Konzept von latenten Variablen, welches im Folgenden näher erläutert wird. Die Herleitung Orientiert sich an den Arbeiten von Kingma und Welling, 2014 und Doersch, 2016.

3.3.1 Latente Variablen

Latente Variablen sind nicht beobachtbare Zufallsvariablen (ZV), welche aber aus einem mathematischen Modell heraus gefolgert werden können. Eine Latente Variable kann zum Beispiel eine codierte Merkmalsbeschreibung eines Bildobjektes sein. Diese gibt vor wie das Objekt aussieht, tatsächlich beobachtet werden, kann aber nur das Objekt selbst. Den Zufallsraum der latenten Variable wird Latent-Space genannt.

In den anschließenden Kapiteln werden die folgenden Bezeichnungen verwendet:

- \mathcal{D} , der Datensatz über einen Raum \mathcal{X} , respektive $x \sim \mathcal{D}$, mit $x \in \mathcal{X}$ ein beliebiger Datenpunkt aus dem Datensatz.
- $\mathcal{Z} = \mathbb{R}^d$, der Latent-Space mit Dimension d
- $P[\mathcal{Z}]$ eine "probability density function" (PDF) über \mathcal{Z} .
- $\mu_\phi : \mathcal{X} \rightarrow \mathcal{Z}$, eine beliebige Funktion parametrisiert durch ϕ . Wird im folgenden verwendet, um den Erwartungswert einer Normalverteilung \mathcal{N} zu approximieren.
- $\sigma_\phi : \mathcal{X} \rightarrow \mathcal{Z}$, analog zu μ_ϕ , die Varianz der Normalverteilung \mathcal{N} .
- $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$, beliebige Funktion parametrisiert durch Θ . Repräsentiert den Decoder.

$\mu_\phi(x)$ und $\sigma_\phi(x)$ repräsentieren gemeinsam die Ausgabe des Encoders für Eingabe x .

Sei die PDF $P[\mathcal{Z}]$ beliebig aber fest. Außerdem sei $z \in \mathcal{Z}$ eine Latente Variable, mit $z \sim P[\mathcal{Z}]$. Für ausreichend komplexe g_θ kann durch die Wahl von Θ jede beliebige Verteilung mittels g_θ modelliert werden (Inverse-Transform-Sampling). Sei nun $P_\theta(x|z) = P(g_\theta(z) = x)$ die bedingte Wahrscheinlichkeit des Auftretens von x . Über Marginalisierung erhalten wir:

$$P_\theta(x) = \mathbb{E}_{z \sim P(\mathcal{Z})} [P(x|z)] = \int_{\mathcal{Z}} P_\theta(x|z) \cdot P(z) dz \quad (3.3)$$

3.3. VARIATIONAL AUTOENCODER

Um den Datensatz möglichst gut zu repräsentieren, soll $P_\theta[\mathcal{X}]$ nach der "Maximum-Likelihood" Methode maximiert werden. Dazu sind folgende Ableitungen nötig:

$$\frac{\partial}{\partial \theta} P_\theta(x) = \int_{\mathcal{Z}} \frac{\partial}{\partial \theta} P_\theta(x|z) \cdot P(z) dz \quad (3.4)$$

Dieses Integral ist im Allgemeinen nicht lösbar. Dennoch kann es approximiert werden. Eine Möglichkeit dies zu tun ist Monte-Carlo-Integration. Dieser Ansatz ist allerdings nicht effizient, denn \mathcal{Z} ist überabzählbar. Erweiterungen der Monte-Carlo Integration (z.B. "Importance Sampling") sind auch nicht anwendbar, da keine Annahmen über die Verteilung $P_\theta(x|z)$ möglich sind. Bekannt ist aber, dass $P_\theta(z|x)$ die bedingte Wahrscheinlichkeit darstellt, dass z im Urbild (bzgl. g_ϕ) von x liegt, sprich z ist Erzeuger von x . Ziel ist es, $P_\theta(z|x)$ zu approximieren, um gezielt die z zu sampeln, welche x erzeugen. Gesucht ist also eine Verteilung $Q[\mathcal{Z}]$, welche ähnlich zu $P_\theta[\mathcal{Z}|x]$ ist. Dazu wird im anschließenden Abschnitt die Kullback-Leibler Divergenz eingeführt.

3.3.2 Kullback-Leibler Divergenz

Die Kullback-Leibler Divergenz, kurz KL Divergenz oder \mathcal{D}_{KL} , ist ein Maß für die Ähnlichkeit zweier Wahrscheinlichkeitsverteilungen. Für beliebige Verteilungen Q, P über eine Menge \mathcal{X} ist sie definiert als:

$$\mathcal{D}_{KL}[Q||P] = \mathbb{E}_{x \sim Q} [\log Q(x) - \log P(x)] \quad (3.5)$$

Mit der KL Divergenz und dem Satz von Bayes kann die Beziehung zwischen $P(x)$ und $\mathbb{E}_{z \sim Q}[P(x|z)]$ formuliert werden:

$$\begin{aligned} \mathcal{D}_{KL}[Q[\mathcal{Z}]||P_\theta[\mathcal{Z}|x]] &= \mathbb{E}_{z \sim Q} [\log Q(z) - \log P_\theta(z|x)] \\ &= \mathbb{E}_{z \sim Q} \left[\log Q(z) - \log \left(\frac{P_\theta(x|z) \cdot P(z)}{P_\theta(x)} \right) \right] \\ &= \mathbb{E}_{z \sim Q} [\log Q(z) - \log P_\theta(x|z) - \log P_\theta(z) + \log P_\theta(x)] \end{aligned} \quad (3.6)$$

Da $\log P_\theta(x)$ nicht von z abhängt kann dieser Term aus dem Erwartungswert gezogen werden.

$$\mathcal{D}_{KL}[Q(z)||P_\theta(z|x)] = \mathbb{E}_{z \sim Q} [\log Q(z) - \log P_\theta(x|z) - \log P(z)] + \log P_\theta(x)$$

Dies ist äquivalent zu:

$$\underbrace{\log P_\theta(x)}_{\text{Log-Likelihood}} - \underbrace{\mathcal{D}_{KL}[Q[\mathcal{Z}]||P_\theta[\mathcal{Z}|x]]}_{\text{Error Term}} = -\mathbb{E}_{z \sim Q} [\log Q(z) - \log P_\theta(x|z) - \log P(z)] \\ = \mathbb{E}_{z \sim Q} [\log(P_\theta(x|z))] - \mathcal{D}_{KL}[Q(z)||P(z)] \quad (3.7)$$

Zu sehen ist auf der linken Seite dieser Gleichung zunächst die logarithmierte Wahrscheinlichkeit von x (Log-Likelihood), welche maximiert werden soll. Von dieser wird die KL-Divergenz von Q und P subtrahiert. D.h. $P_\theta(x)$ wird maximiert und gleichzeitig $Q[\mathcal{Z}]$ dafür bestraft, zu weit von der tatsächlich unterliegenden Verteilung $P_\theta[\mathcal{Z}|x]$

KAPITEL 3. LITERATUR

abzuweichen. Die rechte Seite besteht aus dem Erwartungswert von $z \sim Q$ über folgende Terme: Die Log-likelihood $g_\phi(z) = x$ und die KL-Divergenz von $Q(z)$ und $P(z)$. Man beachte, dass $g_\phi(z)$ berechnet werden kann und $P[\mathcal{Z}]$ beliebig ist. Es wird nun $P[\mathcal{Z}]$ und Q spezifiziert. Üblicherweise wird $P[\mathcal{Z}]$ als $\mathcal{N}(0, I)$ gewählt ($I \in \mathbb{R}^{d \times d}$ ist die Einheitsmatrix). $Q[\mathcal{Z}]$ wird parametrisiert durch $Q_\phi[\mathcal{Z}|x] = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$. $\sigma_\phi(x)$ ist die Hauptdiagonale der Kovarianzmatrix $\Sigma_\phi(x)$. Eine Annahme für VAEs ist nähmlich, dass die Dimensionen in \mathcal{Z} unkorreliert sind und folglich Σ_ϕ diagonal ist. Vorteil dieser Parametrisierung von Q ist, dass sich die Ableitung der KL-Divergenz von Normalverteilungen effizient berechnen lässt. Einsetzen in 3.7 ergibt:

$$\begin{aligned} & \log P_\theta(x) - \mathcal{D}_{KL}[Q[\mathcal{Z}] \| P_\theta[\mathcal{Z}|x]] \\ &= \mathbb{E}_{z \sim \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))} \left[\underbrace{\log P_\theta(x|z)}_{\text{Decoder}} - \mathcal{D}_{KL}[\underbrace{\mathcal{N}(\mu_\phi(x), \sigma_\phi(x)) \| \mathcal{N}(0, I)}_{\text{Encoder}}] \right] \end{aligned} \quad (3.8)$$

Gleichung 3.8 zeigt direkt die Encoder-Decoder Struktur der VAE Zielfunktion.

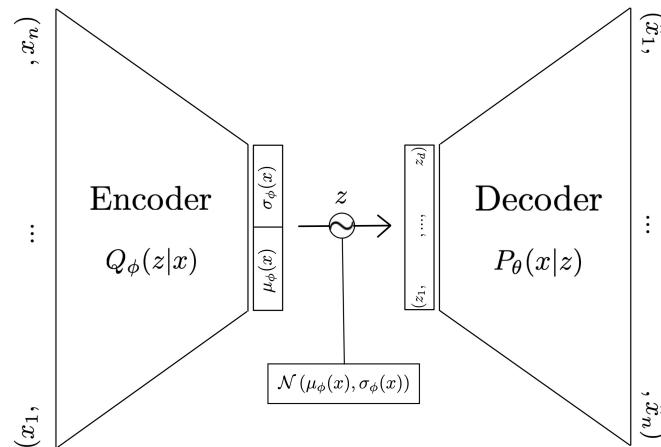


Abbildung 3.4: Der Vorwärtsdurchlauf beim VAE. Der Encoder die Verteilung $Q_\phi(z|x)$ parametrisiert durch μ_ϕ und σ_ϕ . Aus dieser wird z erzeugt. Anschließend liefert der Decoder die Rekonstruktionsverteilung $P_\theta(x|z)$ bzw. für festes x die Rekonstruktion \hat{x} . (siehe auch Doersch, 2016)

3.3.3 Optimierung der VAE Zielfunktion

Statt die obige Zielfunktion zu maximieren, wird folgende Fehlerfunktion minimiert (der Einfachheit wegen wird μ_ϕ statt $\mu_\phi(x)$ etc. geschrieben):

$$\mathcal{L} = - \left(\mathbb{E}_{z \sim \mathcal{N}(\mu_\phi, \sigma_\phi)} [\log(P_\theta(x|z) - \mathcal{D}_{KL}[\mathcal{N}(\mu_\phi, \sigma_\phi) \parallel \mathcal{N}(0, I)])] \right) \quad (3.9)$$

$$= - \mathbb{E}_{z \sim \mathcal{N}(\mu_\phi, \sigma_\phi)} [\log(P_\theta(x|z)] + \mathbb{E}_{z \sim \mathcal{N}(\mu_\phi, \sigma_\phi)} [\mathcal{D}_{KL}[\mathcal{N}(\mu_\phi, \sigma_\phi) \parallel \mathcal{N}(0, I)]] \quad (3.10)$$

Der erste Term entspricht dem komponentenweisen "Negative-Log-Likelihood" (NLL) Fehler der Eingabe x und der Rekonstruktion $g_\theta(z)$. z wird aus der Encoder Verteilung $\mathcal{N}(\mu_\phi, \sigma_\phi)$ gesampelt. Für die KL Divergenz zweier Normalverteilungen existiert eine geschlossene Form zur Berechnung (siehe Hershey und Olsen, 2007):

$$\begin{aligned} \mathcal{D}_{KL} [\mathcal{N}(\mu_0, \Sigma_0) \parallel \mathcal{N}(\mu_1, \Sigma_1)] &= \\ \frac{1}{2} \left(\text{Spur}(\Sigma_1^{-1} \Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) - k + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right) \right) \end{aligned} \quad (3.11)$$

Durch Einsetzen von $\mathcal{N}(\mu_\phi, \sigma_\phi)$ und $\mathcal{N}(0, I)$ vereinfacht sich die Gleichung zu:

$$\begin{aligned} \mathcal{D}_{KL} [\mathcal{N}(\mu_\phi, \Sigma_\phi) \parallel \mathcal{N}(0, I)] &= \\ = \frac{1}{2} \left(\sum_{i=1}^k (\sigma_\phi)_i + \sum_{i=1}^k (\mu_\phi)_i^2 - \sum_{i=1}^k 1 - \log\left(\prod_{i=1}^k (\sigma_\phi)_i\right) \right) \\ = \frac{1}{2} \sum_{i=1}^k ((\sigma_\phi)_i + (\mu_\phi)_i^2 - 1 - \log(\sigma_\phi)_i) \end{aligned} \quad (3.12)$$

Die Ableitung des KL-Terms ist somit berechenbar. Für die Ableitung des NLL Fehlers nach den Parametern ϕ des Encoders besteht jedoch folgendes Problem:

$$\frac{\partial}{\partial \phi} (-\mathbb{E}_{z \sim \mathcal{N}(\mu_\phi, \sigma_\phi)} [\log g_\theta(z)]) \quad (3.13)$$

Da z aus einer Zufallsverteilung gesampelt wird, kann keine Ableitung nach ϕ gebildet werden. Als Lösung wird der folgende Trick angewandt.

3.3.4 Reparametrisierungs-Trick

Eine Ableitung des NLL Fehlers ist nach den Parametern ϕ des Encoder Netzwerks wegen des Samplings von $z \sim \mathcal{N}(\mu_\phi, \sigma_\phi)$ nicht möglich. Statt aber direkt aus der Verteilung zu sampeln, kann ein $\epsilon \sim \mathcal{N}(0, I)$ gesampelt werden und die gleiche Verteilung durch Multiplikation und Addition berechnet werden. Für $z \sim \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$ gilt also:

$$z = (\sigma_\phi(x))^{\frac{1}{2}} \cdot \epsilon + \mu_\phi(x) \quad (3.14)$$

D.h. von folgender Gleichung kann die Ableitung nach ϕ berechnet werden:

$$\frac{\partial}{\partial \phi} \left(-\mathbb{E}_{x \sim D} \left[-\mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} \left[\log g_\theta \left(z = (\sigma_\phi(x))^{\frac{1}{2}} \cdot \epsilon + \mu_\phi(x) \right) \right] \right] \right) \quad (3.15)$$

Kapitel 4

Datensätze

Diese Arbeit wie bereits in der Einführung erläutert, ist ein Aspekt dieser Arbeit den VAE basierten Data Augmentation Ansatz auf unterschiedlichen Daten zu evaluieren. Dazu werden sowohl Bilddaten, als auch numerische Daten evaluiert. Zusätzlich wird die Datensatzgröße variiert. Im folgenden werden die verwendeten Datensätze und ihre Besonderheiten näher beschrieben.

4.1 MNIST

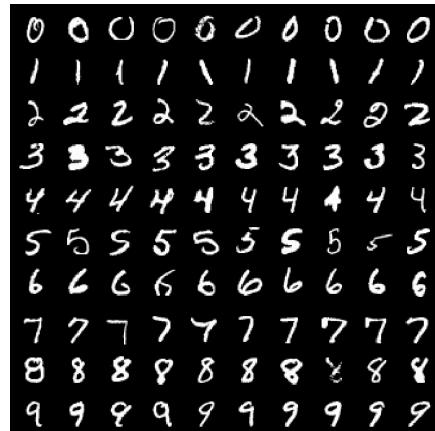


Abbildung 4.1: Der MNIST Datensatz beinhaltet annotierte Bilder der handgeschriebenen Zahlen $\{0, \dots, 9\}$

Der MNIST Datensatz von LeCun u. a., [1998](#) wird oft verwendet, um neue Ansätze im Bereich Visual Machine Learning auf ihre Nutzbarkeit hin zu untersuchen. Es handelt sich bei den Daten um (28×28) Pixel große Grauwertbilder handgeschriebener Zahlen $\{0, \dots, 9\}$. Es gibt zwei Partitionen des Datensatzes: Die Trainings-Partition besteht aus 60.000 Beispielen, also 6.000 pro Klasse. Die Test-Partition beinhaltet 10.000 Beispiele. Die Klassifikation von MNIST ist über die letzten Jahre nahezu fehlerfrei gelöst (z.B. von Byerly, Kalganova und Dear, [2021](#)), da die Daten klar voneinander unterscheidbar

KAPITEL 4. DATENSÄTZE

sind, der Datensatz balanciert ist und ausreichend Beispiele besitzt.

Da die MNIST Aufgabe wegen der genannten Eigenschaften schon durch einfache Modelle gut gelöst werden kann, werden in dieser Arbeit ausschließlich reduzierte Versionen des Originaldatensatzes betrachtet. Die Reduktion erfolgt durch zufälliges Wählen von Beispielen aus der Trainings-Partition des Datensatzes. Für die Evaluation wird der unveränderte Test-Teil verwendet.

4.2 PROBEN1

PROBEN1 ist eine Zusammenfassung mehrerer Datensätze, welche verschiedenste Diagnoseinformationen (Attribute) enthalten, die eine Klassifikation ermöglichen. Vorgeschlagen wurde dieser Datensatz von Moreno-Barea, Jerez und Franco, 2020 um verschiedene Generative Modelle auf kleinen teilweise imbalancierten Datensätzen zu evaluieren. Es ist ein rein numerischer Datensatz, bei dem jede Dimension des Eingabevektors ein Attribut (z.b. Alter, BMI, etc.) beschreibt. Nicht alle diese Attribute entsprechen kontinuierlichen Werten, daher wurden alle Daten, kontinuierliche und diskrete, normalisiert. Fehlende Daten (im Original durch ein "?" repräsentiert) wurden durch Nullen ersetzt. Im Gegensatz zu MNIST und anderen Bild-Datensätzen spielt die relative Position der Attribute im Eingabevektor keine Rolle. Dies wird bei der Wahl der Modellarchitektur für diesen Datensatz berücksichtigt. Tabelle 4.1 gibt einen Überblick über die Dimension der Eingabevektoren (# Attribute), die Anzahl an Klassen und die Größe und Balance der einzelnen Datensätze. Für diese Arbeit wurde jeder Datensatz in

Datensatz	# Attribute	kontinuierlich	# Klassen	# Beispiele	Balancing
card	15	0.40	2	690	0.99
diabetes	8	1.00	2	768	0.93
geneN	60	0.00	3	3175	0.93
glass	9	1.00	6	214	0.84
horse-colic	20	0.70	3	364	0.84
thyroid	21	0.29	3	7200	0.28

Tabelle 4.1: PROBEN1 Datensatz Infos.

Trainings und Test-Partition aufgeteilt, wobei 80% für das Training und 20% für die Evaluation verwendet wurden. Zusätzlich werden reduzierte Trainings-Partitionen, analog zum Abschnitt 4.1 betrachtet.

4.3 CIFAR-10

CIFAR-10 erstellt von Krizhevsky, 2012 ist eine annotierte Teilmenge des "80 million tiny images" Datensatzes. Der Datensatz ist analog zu MNIST (vgl. Abschnitt 4.1) aufgeteilt in 60.000 Trainingsbeispiele und 10.000 Testbeispiele. Er enthält 10 verschiedene Klassen (siehe Abb. 4.2). CIFAR-10 stellt als Klassifikaionsaufgabe, wie MNIST, keine

4.4. LARGE-SCALE CELEBFACES ATTRIBUTES

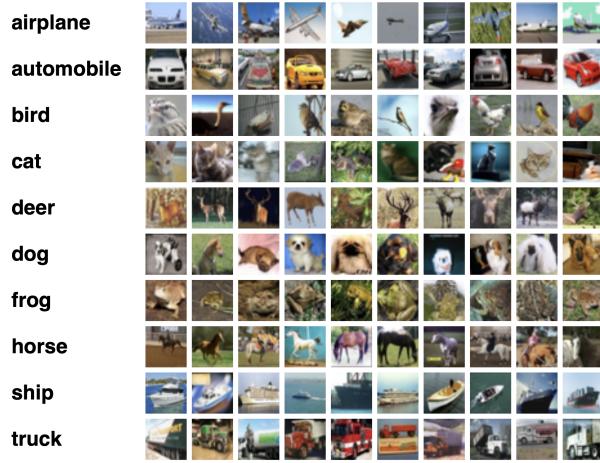


Abbildung 4.2: CIFAR-10 Datensatz: Die einzelnen Klassen schließen sich gegenseitig aus, d.h. es gib z.b. keine Mischform aus "truck" und "automobile" wie SUV.

große Herausforderung da (siehe Foret u. a., 2021). Daher fokussiert sich diese Arbeit vor allem auf die Qualität der Rekonstruktionen auf diesem Datensatz.

4.4 Large-scale CelebFaces Attributes



Abbildung 4.3: Die Gesichtsdaten des CelebA Datensatzes

Large-scale CelebFaces Attributes (CelebA), von Liu u. a., 2015, ist eine Sammlung von über 200.000 Bildern von Gesichtern. Es gibt 10.177 verschiedene Identitäten. Jedes Bild ist mit 40 Attributen wie Haarfarbe, Brille, etc. annotiert. In dieser Arbeit wird insbesondere untersucht, wie sich Manipulationen im Merkmalsraum auf die decodierten Bilder auswirken. Da CelebA ein Multi-Label Datensatz ist (d.h. Attribute schließen

KAPITEL 4. DATENSÄTZE

sich gegenseitig nicht aus), liegt der Fokus auf der Trennung dieser Attribute im Latent-Space.

Kapitel 5

Methodik

Im Abschnitt 3.3 wurde bereits die Funktionsweise und das Optimieren des VAE Modells beschrieben. Im folgenden wird erklärt, wie neue Beispiele generiert werden. Außerdem werden Details zu den Modellarchitekturen gezeigt.

5.1 Latent-Space Sampling

Der Variational Autoencoder besitzt einen strukturierten Latent-Space. Daher kann zwischen den Erwartungswerten $\mu_1 = \mu_\phi(x_1)$ und $\mu_2 = \mu_\phi(x_2)$ vom Encoder berechneten Normalverteilung interpoliert werden. Es werden im Folgenden vier verschiedene Sampling Methoden vorgestellt, um neue Latent-Vektoren \hat{z} zu erzeugen. Diese wurden in leicht anderer Form von Jorge u. a., 2018 vorgeschlagen. Die erste Methode ist aus der Standardnormalverteilung $\mathcal{N}(0, 1)$ zu sampeln. Dies ist sinnvoll, da der Encoder durch den KL-Term dazu gezwungen ist, nicht zu sehr von $\mathcal{N}(0, 1)$ abzuweichen. Eine weitere Methode ist, den Erwartungswert eines original Beispiels $\mu_\phi(x)$, welcher das Zentrum der dazugehörigen Normalverteilung repräsentiert, zu modifizieren. Betrachtet werden drei Modifikationen: Addieren von Gauss-Rauschen (Noise), Interpolation und Extrapolation. Für die Noise Variante wird zufälliges Rauschen einer Normalverteilung $\mathcal{N}(0, \alpha)$ auf den Eingabe- Erwartungswert addiert.

$$\hat{z}_i = z_i + \epsilon, \epsilon \sim \mathcal{N}(0, \alpha) \quad (5.1)$$

Für die Wahl Interpolations Partner wird eine k -Nächster-Nachbar Suche im Latent-Space durchgeführt. Gesucht wird unter allen Erwartungswerten $\mu_\phi(x_i)$ für Eingaben x_i . \hat{z}_i repräsentiert den modifizierten Latent-Vektor. Das Vorzeichen vor α bestimmt, ob interpoliert oder extrapoliert wird. α ist ein zu setzender Hyperparameter und definiert die Stärke der Inter-/Extrapolation

$$\hat{z}_i = \pm \alpha \cdot [\mu_\phi(x_k) - \mu_\phi(x_i)] + \mu_\phi(x_i) \quad (5.2)$$

5.2 Single-VAE

Dieses Verfahren trainiert ein VAE Modell auf dem gesamten Datensatz. Das Training erfolgt selbst-überwacht, d.h. es werden keine Informationen über die Klassen benötigt.

KAPITEL 5. METHODIK

Vorteil dieses Ansatzes ist, dass große Datenmengen genutzt werden können, welche nicht annotiert sein müssen. Es erschwert allerdings das Sampling neuer Beispiele, denn für einen Latent-Vektor \hat{z}_i ist nicht definiert, welcher Klasse er angehört. Der Single-VAE wird deshalb nur mit den Sampling Methoden Noise, Interpolation und Extrapolation für die Data Augmentation genutzt. Das Originalbeispiel, welches für Generation modifiziert wird, definiert dabei die Klasse des generierten Beispiels. Weitere Ansätze, wie das Nutzen eines Klassifikators, welcher die Klasse zu \hat{z}_i bestimmt, wurden nicht weiter verfolgt.

5.3 Multi-VAE

Ein von Moreno-Barea, Jerez und Franco, 2020 angeführtes Problem mit dem Single-VAE Ansatz besteht darin, dass ein einzelner VAE bei imbalancierten Datensätzen dazu tendiert, die größere Klasse im Latent-Space zu überrepräsentieren. Um dem entgegen zu wirken, schlagen die Autoren vor, nur einen VAE pro Klasse zu trainieren. Dazu werden die Daten in k Submengen aufgeteilt, wobei k die Anzahl der verschiedenen Klassen bezeichnet. Jede dieser k Klassen besteht aus den der Klasse zugehörigen Daten. Außerdem wird eine Variante untersucht, bei der zusätzlich noch zufällige Beispiele der anderen Klassen mit einem Anteil von 20% hinzugefügt werden (Bezeichnet mit "Mix Data"). Falls Klassen bestimmte Merkmale teilen, kann dies dem Encoder helfen, diese zu extrahieren. Die Abbildung 5.1 veranschaulicht die Aufteilung des Datensatzes. Das Training der einzelnen VAEs erfolgt auf den klassenweisen Datensätzen. Neue Beispiele können somit leicht über Sampling aus einer Standard Normalverteilung generiert werden. Die Klasse des neuen Beispiels ist die zugehörige Klasse des VAEs.

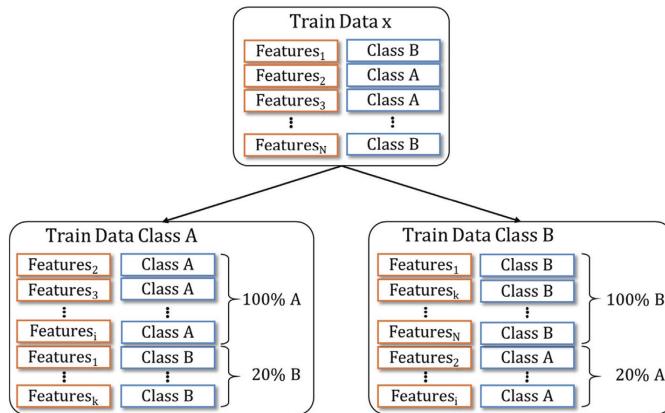


Abbildung 5.1: Die Aufteilung der Datensätze beim Multi-VAE Ansatz. Der vollständige Datensatz wird in klassenweise partitioniert. Zusätzlich wird der Ansatz betrachtet, zu den Daten der Klasse i auch Daten der anderen Klassen beizumischen. Bei mehr als 2 verschiedenen Klassen werden die 20% zusammen aus zufälligen Beispielen der jeweils anderen Klassen gebildet. Abbildung entnommen aus Moreno-Barea, Jerez und Franco, 2020

5.4 Generative Classifiers

Da der VAE darauf trainiert wird, nicht Merkmalsvektoren sondern Latent-Vektoren, welche aus einer Distribution gesampelt werden, zu decodieren, sind die generierten Beispiele häufig unscharf oder entsprechen nicht-interpretierbaren Mischformen zweier Klassen. Moreno-Barea, Jerez und Franco, 2020 schlagen daher ein weiteres Neuronales Netz, den "Generative Classifier" (GC), vor, um diese Beispiele zu erkennen und zu verwerfen. Dazu wird zu jedem Eingabebeispiel x eine verrauschte Variante \tilde{x} erzeugt. Das Netzwerk wird auf die binäre Klassifikation von Rauschen und Originalbeispiel trainiert. Moreno-Barea, Jerez und Franco, 2020 verwenden Rauschen aus zwei Quellen:

- Uniformes Rauschen: Repräsentiert, dass dieses Beispiel "nur Rauschen" beinhaltet
- Normalverteiltes Rauschen: Repräsentiert verrauschte Originalbeispiele. Der Erwartungswert der Normalverteilung ist der Sample-Mean über die Originalbeispiele und die Varianz ist ein einstellbarer Hyperparameter σ_{gc}

Mit σ_{gc} kann die Sensitivität des Generative Classifiers beeinflusst werden.

Da auf Bilddaten die verrauschten Beispiele \tilde{x} nicht den verrauschten Beispielen des VAEs entsprechen, wird in dieser Arbeit eine weitere Art von Trainings Beispielen für den GC vorgeschlagen. Während des VAE Trainings wird eine frühe Version des Modells zwischengespeichert. Diese frühe Modellversion erzeugt in der Regel noch sehr verrauschte Rekonstruktionen. Diese können genutzt werden, um den GC darauf zu trainieren. Diese Art des GC wird auf dem MNIST Datensatz verwendet.

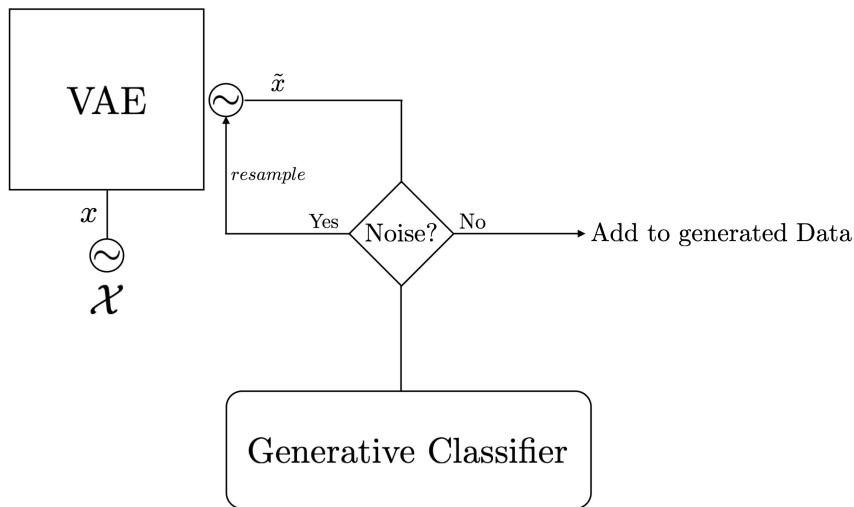


Abbildung 5.2: Der Generative Classifier dient dazu, verrauschte und verschwommene Daten während des Generations Prozesses auszusortieren. Wenn ein generierter Datenpunkt \tilde{x} als *Noise* klassifiziert wird, erzeugt der VAE ein neues Beispiel.

5.5 Details zu den VAE Modellarchitekturen

Wie schon im Abschnitt 3.3 eingeführt besteht der Variational Autoencoder aus einem Encoder und einem Decoder Teil. Der Encoder extrahiert Merkmale aus der Eingabe x und bildet diese auf eine d -dimensionale Normalverteilung $Q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$ ab. Die Verteilung ist durch die Ausgaben $\mu_\phi(x)$ und $\sigma_\phi(x)$ des Encoders parametrisiert. Man beachte, dass aus numerischen Gründen tatsächlich die logarithmische Varianz $\log \sigma_\phi(x)$ berechnet wird. Das in Abschnitt 3.3.4 beschriebene Sampling findet nur während des Trainings statt. Für das Generieren neuer Beispiele wurden die in 5.1 erklärten Sampling Methoden genutzt. Wie im Abschnitt 3.2 motiviert, wurden auf Bilddaten Konvolutionen für die Merkmalsextraktionen im Encoder verwendet. Der Decoder besteht respektive aus den entsprechenden transponierten Konvolutionen, welche den Latent-Vektor auf Bildgröße hochskalieren. Für die numerischen Daten des Datensatzes PROBEN1 werden keine Konvolutionen verwendet, da die Eingabeattribute dieses Datensatzes eindimensional und unabhängig voneinander sind (vgl. Abschnitt 4.2).

Für unsere Experimente wurden die in Tabelle 5.1 spezifizierten Modell Architekturen gewählt. "Batch-Normalization" wurde zwischen allen Layer genutzt und "LeakyRelu" als Aktivierungsfunktion verwendet, um das Training zu stabilisieren (siehe Garay-Maestre, Gallego und Calvo-Zaragoza, 2019). Der Log-Likelihood Fehler wird über die "Binary-Cross-Entropy" (BCE) zwischen Eingabe und Rekonstruktion minimiert. Für den KL-Term wurde die in Abschnitt 3.3.3 angegebene Formel genutzt. Als Reduktion über die Batches für den Binary-Cross-Entropy Term und die KL-Divergenz wurde "Mean" benutzt, allerdings wird der BCE Term mit der Eingabegröße skaliert, um diesen höher zu gewichten.

5.5.1 Hyperparameter für KL-Divergenz Gewichtung

Higgins u. a., 2017 zeigen empirisch, dass es sinnvoll ist, für das Training des VAE einen Hyperparameter β einzuführen. Dieser gewichtet den KL-Term in der Fehlerfunktion und ermöglicht es, einen Trade-Off zwischen einem strukturierten Merkmalsraum und der Qualität der Rekonstruktion zu steuern. Größere Werte lenken den Encoder Output $\mu_\phi(x)$ näher gegen 0. Mit Werten < 1 sind die $\mu_\phi(x)$ weniger beschränkt, d.h. Verteilung $\mu[\mathcal{X}]$ liegt nicht mehr zentral bei 0 und kann beliebige Strukturen annehmen. Da beliebig verteilte Erwartungswerte weniger überlappende Verteilungen zur Folge haben, ermöglicht ein kleinerer Wert für β schärfere Rekonstruktionen.

Desweiteren geben die Autoren eine Berechnungsvorschrift für ein normalisiertes β an, welches die Gewichtung in Abhängigkeit der Dimension der Eingabe N und der Größe des Latent-Spaces kontrolliert.

$$\beta_{norm} = \beta \cdot \frac{d}{N}, \quad (5.3)$$

mit Latent-Space Dimension d .

5.5. DETAILS ZU DEN VAE MODELLARCHITEKTUREN

Datensatz / VAE-Ansatz	Netzwerk	Konfiguration	
MNIST Single-VAE	Encoder	<i>Conv</i> (64, 4 × 4) <i>Conv</i> (128, 4 × 4) <i>Conv</i> (256, 4 × 4)	<i>Flatten()</i> <i>Linear</i> (128) $\mu = \text{Linear}(d), \sigma = \text{Linear}(d)$
	Decoder	<i>Linear</i> (128) <i>Reshape</i> (256 × 4 × 4) <i>ConvT</i> (128, 4 × 4)	<i>ConvT</i> (64, 4 × 4) <i>ConvT</i> (1, 4 × 4) <i>Sigmoid()</i>
MNIST Multi-VAE	Encoder	<i>Conv</i> (32, 4 × 4) <i>Conv</i> (64, 4 × 4)	<i>Flatten()</i> $\mu = \text{Linear}(d), \sigma = \text{Linear}(d)$
	Decoder	<i>Linear</i> (64 · 4 · 4) <i>Reshape</i> (64 × 4 × 4) <i>ConvT</i> (32, 4 × 4)	<i>ConvT</i> (1, 4 × 4) <i>Sigmoid()</i>
MNIST Classification	Classifier	<i>Conv</i> (8, 5 × 5) <i>MaxPool2D</i> (2) <i>Conv</i> (16, 5 × 5) <i>Dropout</i> (0.5) <i>MaxPool2D</i> (2)	<i>Flatten()</i> <i>Linear</i> (64) <i>Dropout</i> (0.5) <i>Linear</i> (10) <i>LogSoftmax()</i>
PROBEN1 Multi-VAE	Encoder	<i>Linear</i> (512) <i>Linear</i> (256) <i>Linear</i> (128)	<i>Linear</i> (64) $\mu = \text{Linear}(d), \sigma = \text{Linear}(d)$
	Decoder	<i>Linear</i> (64) <i>Linear</i> (128) <i>Linear</i> (256)	<i>Linear</i> (512) <i>Linear</i> (input_size) <i>Sigmoid()</i>
PROBEN1 Classification	Classifier	<i>Linear</i> (512) <i>Dropout</i> (0.1) <i>Linear</i> (256) <i>Dropout</i> (0.1)	<i>Linear</i> (128) <i>Dropout</i> (0.1) <i>Linear</i> (n_classes) <i>LogSoftmax()</i>
CIFAR-10 Single-VAE	Encoder	<i>Conv</i> (64, 4 × 4) <i>Conv</i> (256, 4 × 4) <i>Conv</i> (512, 4 × 4)	<i>Flatten()</i> <i>Linear</i> (128) $\mu = \text{Linear}(d), \sigma = \text{Linear}(d)$
	Decoder	<i>Linear</i> (128) <i>Reshape</i> (512 × 4 × 4) <i>ConvT</i> (256, 4 × 4)	<i>ConvT</i> (64, 4 × 4) <i>ConvT</i> (1, 4 × 4) <i>Sigmoid()</i>
CelebA-10 Single-VAE	Encoder	<i>Conv</i> (64, 4 × 4) <i>Conv</i> (128, 4 × 4) <i>Conv</i> (256, 4 × 4) <i>Conv</i> (512, 4 × 4)	<i>Flatten()</i> <i>Linear</i> (128) $\mu = \text{Linear}(d), \sigma = \text{Linear}(d)$
	Decoder	<i>Linear</i> (128) <i>Reshape</i> (512 × 4 × 4) <i>ConvT</i> (256, 4 × 4) <i>ConvT</i> (128, 4 × 4)	<i>ConvT</i> (64, 4 × 4) <i>ConvT</i> (1, 4 × 4) <i>Sigmoid()</i>

Tabelle 5.1: Architektur der verwendeten VAE Modelle (spaltenweise angegeben). Mit $\text{Linear}(m)$ wird ein "Fully-Connected"-Layer mit m Ausgabe-Features beschrieben. $\text{Conv}(m, n \times n)$ bezeichnet eine 2D Konvolution mit Kernelgröße $n \times n$, ConvT analog für 2D transponierte Konvolution. m bezeichnet hierbei die Anzahl an Ausgabe Featuremaps der jeweiligen Konvolution. In allen Konvolutionen wurde Zero-Padding der Größe 1 angewandt, ebenso wie eine Stride von 2. Dies führt zu einer Halbierung der Bildgröße in jedem Conv -Layer. Ausnahme stellt das MNIST Klassifikations Netzwerk dar: Hier wurde Stride 2 verwendet und MaxPooling2D um eine Dimensionsreduktion zu erreichen. Für die Ausgabelayer μ und σ ist d die Dimension des Latent-Spaces.

Kapitel 6

Evaluation

Im anschließenden Kapitel wird die Güte des VAE basierten Data Augmentation An-satzes auf den in Kapitel 4 beschriebenen Datensätzen evaluiert. Außerdem wird eine umfassende Analyse des Latent-Spaces vorgeführt. Alle Zufallskomponenten wurden mit Seeds verwendet, um zwischen den Methoden vergleichbare Resultate zu erzielen. Dabei wurden alle Experimente stets für 3 verschiedene Seeds durchgeführt. Anschlie-ßend wurden die Ergebnisse der Metriken gemittelt. Alle Hyperparameter wurden innerhalb der Experimente identisch gewählt und für die Gewichtung β des KL-Terms wurde keine β -Normalisation angewandt, wenn nicht anders vermerkt. Trainiert wur-de auf einer GeForce RTX 2070 SUPER mit 8 GB VRAM.

6.1 Metriken

		ground truth	
		True	False
predicted	True	true-positive (tp)	false-positive (fp)
	False	false-negative (fn)	true-negative (tn)

Tabelle 6.1: Üblicherweise beschreibt man in der Klassifikation die Beziehung zwischen Vorhersage eines Modells (*predicted*) und der tatsächlichen Klasse (*ground truth*) über eine Konfusionsmatrix. Basierend darauf ergeben sich mehrere Bewertungs Metriken.

In der binären Klassifikation dient meist die *Accuracy Metric* als Bewertungsgrundla-ge. Diese teilt die Anzahl richtig-positiv (*tp*) vorhergesagter Klassen durch die Gesamt-anzahl an Beispielen. In der Multi-Class Klassifikation bringt die *Accuracy* jedoch den Nachteil mit, dass sie falsch-positive (*fp*) und falsch-negative (*fn*) nicht berücksichtigt. Daher betrachten wird in dieser Arbeit der F-Score als Vergleichsmetrik betrachtet. Der

KAPITEL 6. EVALUATION

F-Score stellt das harmonische Mittel von "Recall" und "Precision" dar (siehe Gl. 6.1).

$$\begin{aligned}
 precision &= \frac{tp}{tp + fp} \\
 recall &= \frac{tp}{tp + fn} \\
 f\text{-score} &= 2 \cdot \frac{precision \cdot recall}{precision + recall} \\
 &= \frac{tp}{tp + \frac{1}{2}(fp + fn)}
 \end{aligned} \tag{6.1}$$

Im Fall von mehr als zwei Klassen werden verschiedene Durchschnittsmethoden benutzt, um die klassenweisen F-Scores zusammenzurechnen (siehe Gleichung 6.2). Da auch imbalancierte Datensätze Teil der Experimente sind, wird der *weighted-F-Score* als Vergleichsmetrik genutzt. Dieser gewichtet die klassenweisen F-Scores nach der Anzahl an Beispielen pro Klasse. k bezeichnet die Anzahl an Klassen und n_c die Anzahl an Beispielen in Klasse c . n gibt die Gesamtanzahl Beispiele an.

$$\begin{aligned}
 micro\text{-}f\text{-score} &= \frac{\sum_{c=1}^n tp_c}{\sum_{c=1}^n tp + fp} = Accuracy \\
 macro\text{-}f\text{-score} &= \frac{\sum_{c=1}^n f_c}{n} \\
 weighted\text{-}f\text{-score} &= 2 \frac{\sum_{c=1}^n f_c \cdot n_c}{n}
 \end{aligned} \tag{6.2}$$

6.1.1 Few-Shot Szenario

Für die Evaluation in Few-Shot Szenarien wird die Größe der Trainings-Partition jedes Datensatzes reduziert. Gewählt werden dabei jeweils n Beispiele pro Klasse, mit $n \in \{2, 3, 4, 5, 10, 20, 30, 50, 100, 200, 500, 1000, 2000\}$. Für 1-Shot Learning ist der Multi-VAE basierte Ansatz nicht praktikabel, da aus einem Beispiel keine Generalisierung über den Datensatz erzielt werden kann. In diesem Fall erzeugt der VAE keine ungewöhnlichen Beispiele. Da ebenfalls die Interpolations- und Extrapolations Methoden auf einer nächsten-Nachbar Suche basieren, welche bei nur einem Beispiel nicht möglich ist, werden nur Few-Shot Szenarien ab $n = 2$ untersucht.

6.2 Latent-Space Analyse

Für die Analyse des Latent-Space wird aus Gründen der Visualisierung im Folgenden eine Latent-Space Dimension von $d = 2$ betrachtet. Analysiert wird zunächst die Test-Partition des MNIST Datensatzes. Abb. 6.1a stellt eine Visualisierung des erlernten Latent-Spaces dar. Zu sehen ist, dass die Encoder Verteilung nahe einer Normalsverteilung $\mathcal{N}(0, 1)$ liegt. Gleichzeitig ist eine Trennung der verschiedenen Klassen in einzelne Cluster sichtbar. Es fällt auf, dass die beiden Ziffern 4 und 9 weniger deutlich als z.B. 0 und 1 getrennt werden. Offenbar ähneln sich die Merkmale einer 4 und einer 9 also mehr. Dies ist auch in den Rekonstruktionen in Abb. 6.2 erkennbar. Auffallend ist, dass

einige Klassen einen größeren Teil des Latent-Spaces einnehmen, obwohl die Anzahl der Beispiele pro Klasse im Training identisch war.

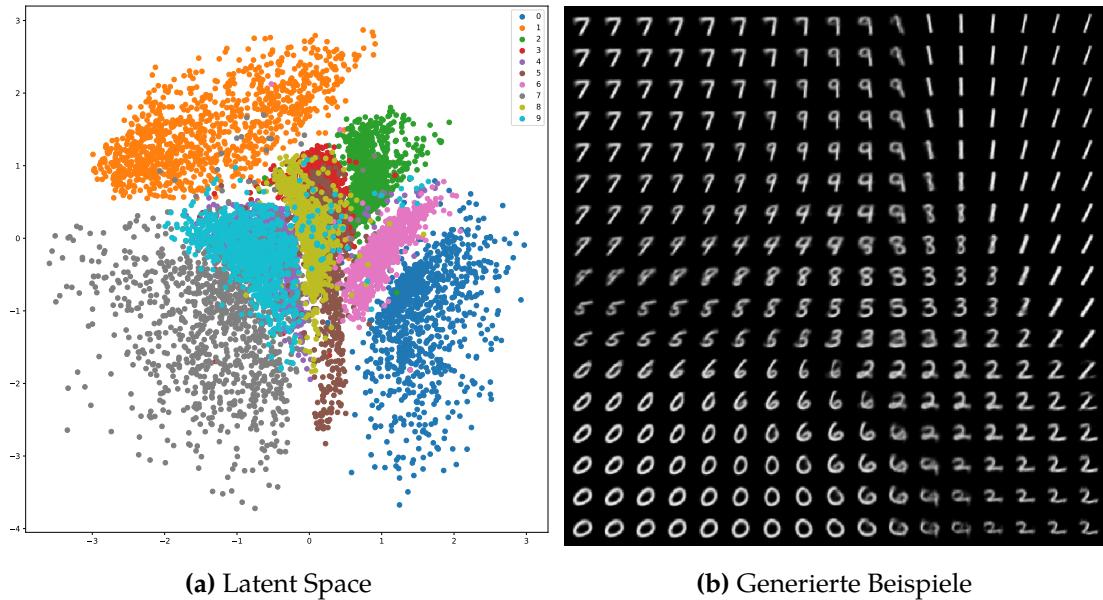


Abbildung 6.1: (links) Der Latent Space eines Single-VAE auf dem Datensatz MNIST. Trainiert wurde mit $\beta = 0.5$. Zu sehen ist die klare Trennung der Klassen im äußeren Bereich. Nahe 0 sind die Klassen weniger deutlich getrennt. (rechts) Die Rekonstruktionen von Latent-Vektoren aus dem Intervall $(-3, 3)$. Auch hier sind die Rekonstruktionen im äußeren Bereich deutlich schärfer.

Abb. 6.1b zeigt die Rekonstruktionen von 2D Latent-Vektoren aus dem Intervall $(-3, 3)$ in beiden Dimensionen. Es ist dieselbe Trennung wie schon im Latent-Space sichtbar. An den Grenzen zwischen zwei Clustern sind außerdem Mischformen von Klassen zu sehen.



Abbildung 6.2: Rekonstruktionen verschiedener Original Ziffern 4 beim Single-VAE. Trainiert wurde mit $\beta = 0.5$. Zu sehen ist das Problem bei Überlappungen von Klassen im Latent-Space. Die Rekonstruktionen sind nicht eindeutig einer 4 oder 9 zuordbar, da durch die geringe Latent-Space Dimension ein zu hoher Informationsverlust mit einhergeht. Höheren Latent-Space Dimensionen verringern diesen Effekt.

In Abb. 6.3 ist die Funktionsweise des Interpolations- bzw. Extrapolations-Samplings gezeigt. Eine Erkenntnis ist, dass bei diesen Sampling Methoden immer garantiert ist, aus der Nähe von Originalbeispielen zu sampeln. Damit ergeben sich mehr plausible

KAPITEL 6. EVALUATION

Rekonstruktionen.

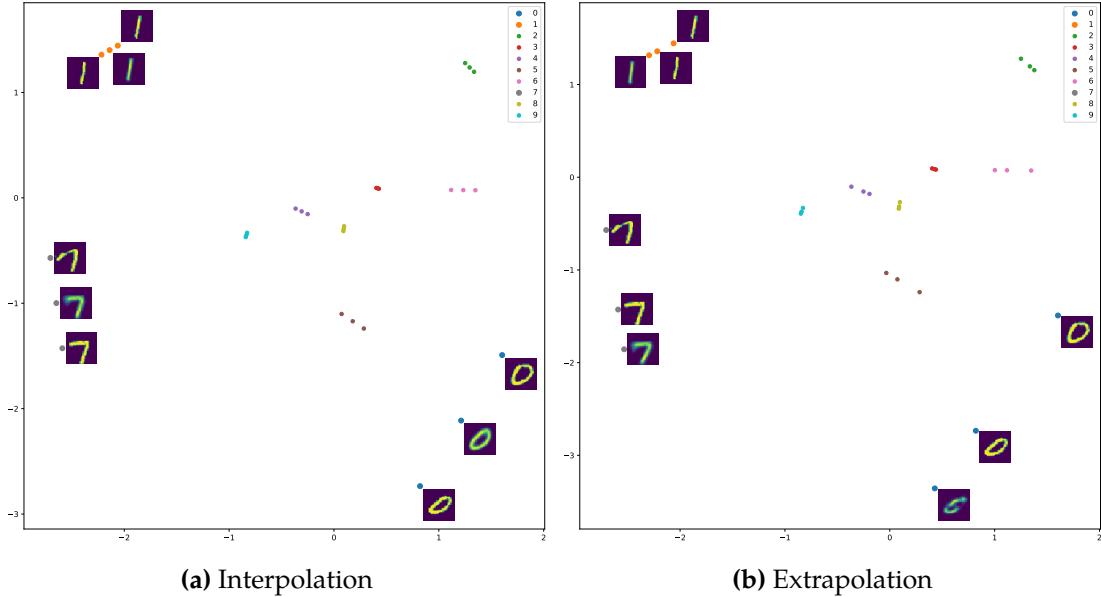


Abbildung 6.3: Die Interpolations und Extrapolationsmethode. Zu sehen sind das Ergebnis der nächsten-Nachbar Suche im Latent-Space und der inter- / extrapolierte Punkt für jeweils ein Beispiel jeder Klasse. Der α Parameter wurde auf 0.5 gesetzt, was bei einer Interpolation genau dem Mittelpunkt entspricht. Zusätzlich sind die Rekonstruktionen der korrespondierenden Punkte zu sehen.

In Abb. 6.4 wird der Einfluss des β Faktors in der Fehlerfunktion des VAEs gezeigt. Für kleine β Werte ähnelt die Struktur des Latent-Space weniger einer Standardnormalverteilung. Er hat zwar immernoch eine probabilistische Form, welche durch das Sampling im Training (siehe Reparametrisierungstrick, Abschnitt 3.3.4) erreicht wird, die Latent-Vektoren weichen aber immer weiter von einem Erwartungswert 0 ab. Der KL-Term ist also essentiell wichtig, um eine bekannte Struktur im Latent-Space zu erzeugen. Gleichzeitig kann eine zu große Gewichtung dazu führen, dass der Latent-Space kollabiert und alle Eingaben auf Erwartungswert 0 und Varianz 1 abbildet. Für Datensätze mit eindeutigen Klassen ist dies von großem Nachteil, da die Merkmale nun von einander entkoppelt sind (*Disentanglement*, siehe Higgins u. a., 2017). Der Latent-Vektor verliert die Informationen über seine eindeutigen Merkmale. Für Multi-Label Klassifikation kann sich diese Eigenschaft jedoch zu Nutze gemacht werden, um gezielt Merkmale zu dekorrelieren. In den Rekonstruktionen in Abb. 6.5 sehen wir, dass eine größere Gewichtung der KL-Divergenz eine verschwommenere Rekonstruktion mit sich bringt.

6.2. LATENT-SPACE ANALYSE

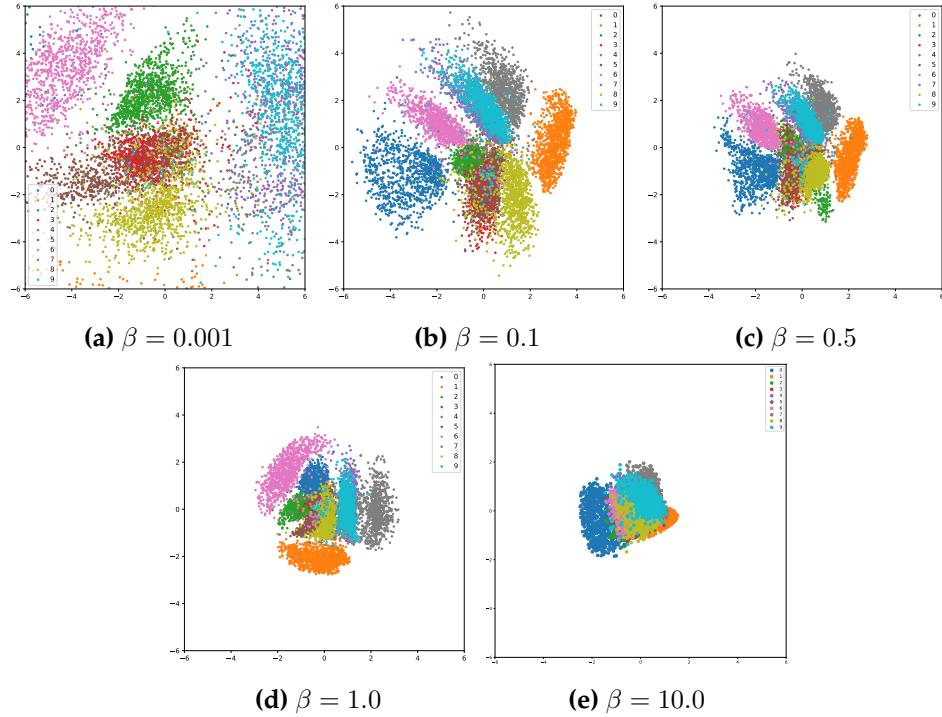


Abbildung 6.4: Der Latent-Space eines Single-VAEs (MNIST) für verschiedene Werte von β . Erkennbar ist, dass zu kleine Werte für β weniger Annahmen über die Latent-Vektoren ermöglichen. Zu große Werte haben dagegen einen vollständigen Strukturverlust zur Folge.

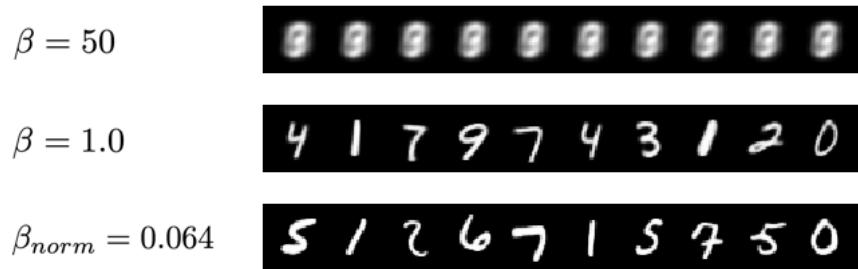


Abbildung 6.5: Rekonstruktionen zufälliger MNIST Beispiele eines Single-VAEs. Höhere Werte für β haben deutlich unschärfere rekonstruierte Bilder zur Folge. Extrem kleine Werte beeinträchtigen die Rekonstruktionsqualität dagegen nicht.

Für numerische Daten in den PROBEN1 Datensätzen hat es sich als außerordentlich schwierig erwiesen, geeignete Hyperparameter für einen strukturierten Latent-Space zu finden. In vielen Fällen kollabierte der Latent-Space schon für Werte für $\beta > 0.5$. Ein Grund dafür ist das Verhältnis des BCE-Fehlers zur KL-Divergenz. Für MNIST ist der BCE-Fehler entsprechend groß, da die Bildgröße mit 28×28 Pixeln einen hohen BCE-

KAPITEL 6. EVALUATION

Fehler bewirkt (vgl. Reduktion des BCE-Fehlers, Abschnitt 5.5.1). Im Falle von PROBEN1 jedoch ist die Eingabe nur 8-60 dimensional. Über β -Normalisierung konnte diesem Problem in den meisten Fällen entgegen gewirkt werden. Bei numerischen Daten hatten auch Batch-Größen von über 32 einen negativen Effekt auf die Struktur des Latent-Spaces.

Ein weiteres Problem bei den numerischen Daten des PROBEN1 Datensatzes sind die diskreten Attribute. In Abb. 6.6 ist eine Clusterbildung im Latent-Space zu erkennen.

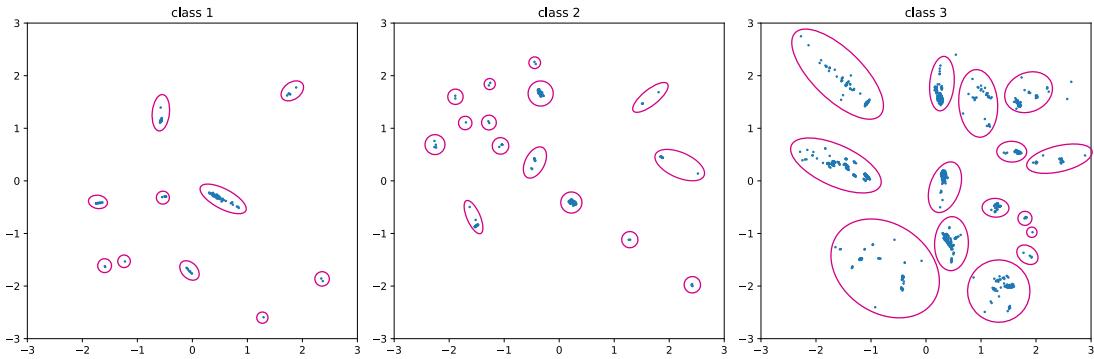


Abbildung 6.6: Latent-Spaces der Klassenweisen VAEs auf dem "thyroid" Datensatz. Der Datensatz besitzt 15 diskrete Attribute und 6 kontinuierliche Attribute. Besonders im Latent-Space zur Klasse 2 führen diese 15 diskreten Attribute zu 15 Clustern.

Es zeigt sich eine Relation zwischen der Anzahl an diskreten Attributen und Clustern. Für jedes Attribut entsteht ein Teil-Cluster für jeden Wert aus dem Wertebereich des Attributes. In der Abbildung ist dies besonders in Klasse 2 zu sehen. Man beachte, dass die Attribute des Datensatzes "thyroid" alle binär sind, also 2 Teil-Cluster erkennbar sind. Im Latent-Space der Klasse 3 ist dieses Clustering weniger scharf, aber ebenfalls erkennbar. Grund hierfür ist, dass die Klasse 3 insgesamt 5333 Beispiele, die Klasse 2 dagegen nur 294 Beispiele enthält. So wird der Zwischenraum innerhalb der Cluster in Klasse 3 mehr exploriert. Dieses Clustering ist problematisch für das Generieren neuer Beispiele. Die Grundannahme für das Sampling im Latent-Space besteht darin, dass die Verteilung ähnlich zu einer Standard Normalverteilung ist. Erzielt die KL-Divergenz Regularisierung diese Struktur nicht, so werden Rekonstruktionen erzeugt, welche stark verrauscht sind. Dies hat einen negativen Einfluss auf das Training eines Klassifikators auf diesen generierten Daten.

6.3 PROBEN1

Die Experimente auf den PROBEN1 Datensätzen stellten sich als sehr Hyperparameter sensitiv heraus. So hatte beispielsweise der β Parameter, die Batch-Größe des VAEs und die Trainingsdauer des VAEs einen erheblichen Einfluss auf die Resultate. Eine Erkenntnis war, dass die Gefahr des Overfittings beim VAE relativ gering ist. Generell galt: Je

länger der VAE trainiert wurde, desto besser wurden die Rekonstruktionen und damit die Ergebnisse im Training mit generierten Daten. Ab ca. 6000 Trainingsschritten (je nach Größe des Datensatzes), verbessert sich die Qualität der Rekonstruktionen nicht mehr. Dies ist nicht überraschend, da Regularisierung des KL-Terms und das Sampling während des Trainings Overfitting verhindern. In den Experimenten zu PROBEN1 wurde sowohl der Multi-VAE, als auch Single-VAE auf der selben Datenmenge trainiert. Hierbei stellte sich heraus, dass sowohl auf dem gesamten Datensatz, als auch im Few-Shot Szenario der Single-VAE Ansatz keine Verbesserung in der Klassifikation erzielen konnte. Grund dafür ist dass die Eigenschaft des selbst-überwachten Lernens nicht durch zusätzliche nicht-annotierte Daten ausgenutzt werden kann. Ebenso übertraf das zufällige Sampling aus der Normalverteilung $\mathcal{N}(0, 1)$ die anderen Sampling Methoden im Multi-VAE Fall. Die hier beschriebenen Ergebnisse beschränken sich daher auf den Multi-VAE Ansatz mit Sampling aus der Standardnormalverteilung.

6.3.1 Performanz auf dem gesamtem Datensatz

Für das Training eines Klassifikators auf jeweils dem gesamten Datensatz konnten Verbesserungen des *weighted-f-score* in fast allen Datensätzen erzielt werden. Für das Training mit generierten Daten wurden genauso viele echte Trainingsbeispiele wie generierte benutzt. Mehr generierte Daten hinzuzunehmen, erzielte auf diesen Datensätze keine Verbesserungen. Die Anzahl an Trainingsschritten für den VAE und die Sensitivität des Generative Classifiers wurde jeweils an den Datensatz angepasst. Alle VAEs wurden mit Latent-Space Dimension $d = 2$ trainiert. Es stellte sich heraus, dass eine höhere Dimension auf diesen Datensätzen keine Verbesserung zur Folge hatte. Abb. 6.2 zeigt den *weighted-f-score* des Klassifikationsnetzwerks auf den Originaldaten (Baseline), den generierten Daten (VAE) und den gefilterten Daten des Generative Classifiers (VAE + GC).

Es ist eine leichte Verbesserung des F-Scores für die mit VAE Daten trainierten Modelle sichtbar. Die höchste Verbesserung konnte im Datensatz "horse" mit 0.016 erzielt werden. Die Datensätze "thyroid" und "geneN" erreichten eine Verbesserung nur, wenn kein Balancing erfolgte. Das Mischen der klassenweisen Datensätze führte hierbei zu einem höheren F-Score. In den anderen Datensätzen konnten bessere Ergebnisse erzielt werden, wenn nicht gemischt, dafür aber die kleineren Klassen ausbalanciert wurden. Die identischen F-Score Werte für den Datensatz "glass" lassen sich dadurch begründen, dass der F-Score für 2 der 3 Seeds bei 1.00 lag, die Baseline also bereits nahezu fehlerfrei war. Generell ist die Verbesserung mit generierten Daten sehr gering und hängt stark vom Datensatz und der Wahl des Seeds ab. Es ist nicht auszuschließen, dass Zufalls Effekte hier eine Rolle spielen. Imbalancierte Datensätze wie "thyroid" erzielen tendenziell weniger Verbesserungen mit dem VAE Ansatz. Der Generative Classifier führt in einigen Fällen zu einem höheren F-Score als der VAE ohne GC. Jedoch fiel beim Trainieren auf, dass je nach Seed die Sensitivität für "schlechte" Beispiele unterschiedlich hoch war. Seed übergreifend Hyperparameter zu finden, erwies sich als äußerst schwierig.

Im Vergleich zu den generativen Modellen der Autoren Moreno-Barea, Jerez und Franco, 2020 liefert der hier verwendete VAE Ansatz eine geringere Modellgüte. Insbe-

KAPITEL 6. EVALUATION

sondere die dort verwendeten Generative-Adversarial-Networks erzielten bessere Resultate.

Datensatz	β	Baseline	VAE	VAE + GC
card	norm=0.133	0.871	0.873	0.869
	0.5	0.871	0.868	0.864
	1.0	0.871	0.876	0.871
diabetes	norm=0.25	0.823	0.831	0.826
	0.5	0.823	0.830	0.822
	1.0	0.823	0.827	0.833
geneN	norm=0.033	0.813	0.802	0.814
	0.5	0.813	0.818	0.813
	1.0	0.813	0.812	0.813
glass	norm=0.222	0.985	0.977	0.985
	0.5	0.985	0.977	0.985
	1.0	0.985	0.984	0.985
horse	norm=0.1	0.828	0.837	0.844
	0.5	0.828	0.833	0.835
	1.0	0.828	0.836	0.828
thyroid	norm=0.095	0.954	0.952	0.956
	0.5	0.954	0.927	0.956
	1.0	0.954	0.925	0.954

Tabelle 6.2: F-Scores auf den PROBEN1 Datensätzen. Das beste Ergebnis ist jeweils **fett** gedruckt. Bis auf den "glass" Datensatz verbessert sich der F-Score durch den VAE um ca. 0.007. Es wurden 3 verschiedene Werte für β getestet: 1.0, 0.5 und 1.0 mit β -Normalisation (mit norm angegeben). Höhere Werte für β führen zum Kollaps im Latent-Space und daraus resultierend schlechterer Güte.

6.3.2 Few-Shot Szenario

Abbildung 6.7 zeigt, dass die Verbesserung im F-Score mit zunehmender Anzahl an Trainingsbeispielen abnimmt. Ab 50 Beispielen pro Klasse ist sogar keine Verbesserung mehr sichtbar. Der Baseline Score liegt dann schon entsprechend hoch. Es scheint, als würden die verrauschten Beispiele, die der VAE generiert, die Klassifikationsaufgabe erschweren. Lediglich bei sehr wenig Beispielen pro Klasse sind die Beispiele hilfreich für die Klassifikation. Der Generative Classifier scheint das Ergebnis nicht konsequent zu verbessern.

# Trainingsbeispiele	# Generierte Beispiele	Baseline	VAE	VAE + GC
5	5	0.619	0.693	0.675
10	10	0.727	0.778	0.761
20	20	0.735	0.753	0.778
30	30	0.772	0.786	0.796
50	50		0.818	0.808
100	100		0.935	0.922

Tabelle 6.3: Der F-Score auf dem Datensatz "thyroid" im Few-Shot Szenario. Für kleine Trainingsgrößen ergibt sich ein verbesserte Wert von bis zu 0.074. Die Verbesserung nimmt mit der Anzahl originaler Trainingsbeispiele ab.

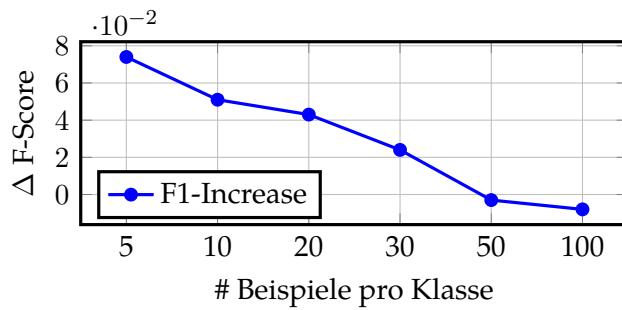


Abbildung 6.7: Differenz des Baseline F-Scores und des Multi-VAEs auf dem Datensatz "thyroid" im Few-Shot-Szenario. Es wurde jeweils der höchste Wert von VAE und VAE + GC verwendet

6.4 MNIST

In unseren Experimenten zum MNIST Datensatz wurden die klassenweisen VAEs für den Multi-VAE Ansatz stets auf der angegebenen Menge an Beispielen pro Klasse trainiert. Als Latent-Space Dimension wurde $d = 2$ gewählt und die KL-Divergenz wurde mit $\beta = 0.5$ gewichtet, um schärfere Bilder zu generieren. Die Multi-VAE Modelle wurden alle für 6000 Schritte trainiert. Für den Single-VAE Ansatz wurde ein Modell selbstüberwacht auf dem gesamten Trainingsdatensatz (60000 Bilder) trainiert. Die Latent-Space Dimension dieses Modells wurde auf $d = 50$ gesetzt. Der β Faktor lag bei 1.0. Das verwendete Modell wurde für 93 000 Schritte trainiert. Die Trainingszeit lag bei $\approx 15\text{min}$. Da die Klassifikation des MNIST Datensatzes mit 60000 Trainingsbeispielen als gelöstes Problem gilt, wird dieser Fall nicht betrachtet. Untersuchen jedoch zusätzlich zum Few-Shot Fall das Training auf nur generierten Daten des VAEs. In allen Experimenten zum MNIST Datensatz wurden dieselben Seeds verwendet.

6.4.1 Few-Shot Szenario

Für das Few-Shot Szenario wurden die in Abschnitt 5.1 beschriebenen Generationsmethoden für den Single-VAE sowie das Sampling aus einer Standardnormalverteilung für

KAPITEL 6. EVALUATION

den Multi-VAE Ansatz evaluiert. Ebenso wurden diese Methoden mit und ohne Generative Classifier betrachtet. Tabelle 6.4 zeigt die Güte der verschiedenen Ansätze für unterschiedliche Trainingsgrößen. Für dieses Experiment wurden original Beispiele und Generierte im Verhältnis 1:1 verwendet. Zu sehen ist, dass mit zunehmender Datengröße

# Beispiele pro Klasse	Baseline	Noise	Interpolation	Extrapolation	Interpolation + Noise	Extrapolation + Noise	Multi-VAE
2	0.660	0.619	0.669	0.654	0.674	0.653	0.678
3	0.699	0.662	0.706	0.700	0.703	0.706	0.706
4	0.734	0.703	0.740	0.732	0.744	0.734	0.734
5	0.776	0.721	0.782	0.756	0.787	0.768	0.770
10	0.849	0.825	0.855	0.861	0.859	0.849	0.854
20	0.905	0.878	0.905	0.911	0.905	0.909	0.904
30	0.929	0.898	0.924	0.930	0.927	0.923	0.923
50	0.941	0.920	0.939	0.941	0.942	0.938	0.937
100	0.954	0.940	0.951	0.953	0.951	0.950	0.949
200	0.962	0.950	0.958	0.960	0.956	0.958	0.956
500	0.964	0.955	0.962	0.963	0.962	0.963	0.959
1000	0.965	0.952	0.963	0.964	0.964	0.963	0.960
2000	0.967	0.953	0.964	0.966	0.963	0.964	0.963

Tabelle 6.4: Evaluation des Klassifikators auf reduzierten Partitionen des MNIST Datensatzes. Die Werte stellen jeweils das Maximum aus VAE ohne GC und VAE mit GC dar.

die Verbesserungen durch die generierten Daten abnimmt. Bei mehr als 100 Trainingsbeispielen pro Klasse ist der VAE Ansatz sogar schlechter als die Baseline. Trotzdem verbessert der VAE den F-Score bei wenig Beispielen um bis zu 0.018. Die Interpolation und Extrapolationsmethoden mit Noise erwiesen sich als die besten Ansätze. Nur Noise lieferte keine Verbesserung. Der Multi-VAE Ansatz zeigt nur im 2-3-Shot Fall eine gute Performance. Im Vergleich zu den Resultaten der Autoren Garay-Maestre, Gallego und Calvo-Zaragoza, 2019 schneidet das hier evaluierte Modell bei höheren Datensätzen schlechter ab. Für über 100 Beispielen pro Klasse konnten die dort erzielten Ergebnisse nicht reproduziert werden. Dies kann aber auch durch das in der vorliegenden Arbeit verwendete Evaluationsverfahren begründet sein, da über 3 unterschiedliche Seeds der Mittelwert gebildet wird.

6.4.2 Performanz auf nur generierten Daten

Da die erzielten Verbesserungen in der Modellgüte gering ausfallen, stellt sich die Frage: Wie mächtig ist der VAE Ansatz? Darum wurden die Resultate des Klassifikators bei einem Training auf nur generierten Daten mit der Baseline verglichen. Ausgangspunkt für das VAE Modell und die Baseline bilden 5 Originalbeispiele. Abb. 6.8 zeigt die Entwicklung des F-Scores mit der Zunahme an generierten Beispielen.

# Generierte Beispiele pro Klasse	Baseline	Noise	Interpolation + Noise	Extrapolation + Noise	Multi-VAE
2	0.776	0.402	0.636	0.542	0.635
3	0.776	0.518	0.681	0.637	0.666
4	0.776	0.542	0.699	0.616	0.697
5	0.776	0.555	0.686	0.574	0.712
10	0.776	0.618	0.740	0.696	0.764
20	0.776	0.670	0.748	0.740	0.773
30	0.776	0.700	0.765	0.744	0.769
50	0.776	0.726	0.774	0.754	0.763
100	0.776	0.750	0.797	0.771	0.768
200	0.776	0.765	0.790	0.761	0.771
500	0.776	0.788	0.792	0.770	0.774
1000	0.776	0.775	0.788	0.773	0.767
2000	0.776	0.777	0.792	0.765	0.770

Tabelle 6.5: F-Score bei Training auf ausschließlich generierten Daten. Die Baseline wurde auf 5 Originalbeispielen trainiert. Für den VAE stehen dieselben 5 Beispiele zur Verfügung um Daten zu generieren. Anschließend wurden nur die generierten Daten für die Evaluation der jeweiligen Methode verwendet. Die Methode Interpolation + Noise erzielt insbesondere für viele generierte Beispiele eine hohe Modellgüte, teilweise sogar besser als die Baseline.

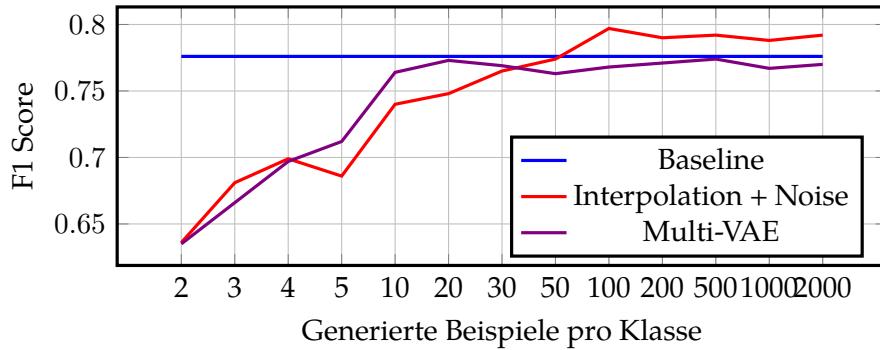


Abbildung 6.8: Der Verlauf der Modellgüte bei verschiedenen Mengen von generierten Daten. Die Methode Interpolation + Noise übertrifft die Baseline für eine hohe Anzahl. Die Multi-VAE Methode nähert sich asymptotisch der Baseline Qualität.

Erkennbar ist, dass ab 50 Beispielen die Methode Interpolation mit Noise die Baseline sogar übertrifft. Die Modellgüte nimmt der Anzahl an generierten Beispielen zu.

6.4.3 MNIST Few-Shot Szenario - Mehr generierte Daten

Da Abschnitt 6.4.2 zeigt, dass sich die Modellgüte erhöht, wenn mehr Daten generiert werden, wurde das Few-Shot Szenario zusätzlich mit diesmal 2000 generierten Daten pro Klasse evaluiert. Das Setup ist ansonsten das gleiche wie in Abschnitt 6.4.1. In Abb.

6.6 ist zu sehen, dass die F-Scores weiter verbessert werden konnten. Der VAE Ansatz erzielt nun einen bis zu 0.022 höheren F-Score als die Baseline. Außerdem zeigt sich, dass die Noise Methode vor allem im 2-4-Shot Fall eine höhere Bewertung erzielt. Für das Multi-VAE Verfahren scheint die Zunahme an generierten Daten kontraproduktiv zu sein.

# Beispiele pro Klasse	Baseline	Noise	Interpolation + Noise	Extrapolation + Noise	Multi-VAE
2	0.660	0.679	0.668	0.638	0.634
3	0.699	0.716	0.715	0.711	0.687
4	0.734	0.748	0.747	0.740	0.724
5	0.776	0.779	0.795	0.772	0.768
10	0.849	0.851	0.868	0.871	0.839
20	0.905	0.894	0.907	0.919	0.897
30	0.929	0.905	0.922	0.936	0.916
50	0.941	0.914	0.935	0.945	0.925
100	0.954	0.923	0.946	0.951	0.936
200	0.962	0.929	0.952	0.960	0.934
500	0.964	0.939	0.957	0.960	0.924
1000	0.965	0.946	0.960	0.961	0.913
2000	0.967	0.953	0.963	0.964	0.905

Tabelle 6.6: F-Score auf dem MNIST Datensatz im Few-Shot Szenario. Mit 2000 generierten Beispielen erzielen alle Methoden höhere Werte als bei einer 1:1 Zusammensetzung des Datensatzes.

6.5 CelebA

Die übliche Klassifikationsaufgabe auf dem CelebA Datensatz ist die Multi-Label Klassifikation. Die Aufgabe besteht darin alle Attribute eines Bildobjektes korrekt vorherzusagen. Da dies den Rahmen dieser Arbeit sprengen würde, konzentriert sich unsere Analyse auf Modifikationen im Latent-Space und deren Auswirkung auf die Rekonstruktionen. Da es keine eindeutigen Klassen gibt, wird nur der Single-VAE Ansatz betrachtet. Der Datensatz ist mit ca. 200.000 Bildern und 10.177 Identitäten sehr groß, daher wurde auch die Latent-Space Dimension mit $d = 50$ entsprechend groß gewählt. Trainiert wurden 2 Modelle mit β -Normalisation: Eines mit $\beta = 1.0$, das zweite mit $\beta = 150$. In Abb. 6.9 sehen wir, dass die Rekonstruktionen deutlich verrauscht sind. Außerdem ist zu erkennen, dass dies bei $\beta = 150$ stärker der Fall ist. Der hohe β Faktor hat also einen negativen Einfluss auf die Qualität der Rekonstruktionen. Der β Faktor hat aber auch einen Einfluss auf die Representationen der extrahierten Merkmale im Latent-Space.

In Abb. 6.10 sind Interpolationen von beiden Modellen jeweils in einer Dimension zu sehen. Im oberen Bild stellt das interpolierte Merkmal eine Rotation von rechts nach links dar, im unteren Bild die Farbe der Haut. Man sieht, dass das Modell mit $\beta = 1$ Schwierigkeiten hat, die Merkmale "Hautfarbe" und "Lächeln" zu trennen. Dies ist bei dem Modell mit $\beta = 150$ nicht der Fall. Ein höherer β Wert führt also zu Entwirrung von Merkmalen in den Dimensionen des Latent-Space. Erklären lässt sich das folgendermaßen (vgl. Higgins u. a., 2017): Durch den hoch gewichteten KL-Term wird



Abbildung 6.9: CelebA - Originalbilder oben, Rekonstruktionen mit verschiedenen Werten für β unten. β_{norm} (links), $150 \cdot \beta_{norm}$ (rechts)

die vom Encoder gebildete Normalverteilung Richtung Erwartungswert 0 und Varianz 1 gezwungen. Dies führt dazu, dass die Trennung von Merkmalen nun mehr über Dimensionen erfolgen muss. Daher sehen wir, dass verschiedene Dimensionen unterschiedliche Merkmale kontrollieren. Diese Repräsentation wird auch "Disentangled-Feature-Representation" genannt Higgins u.a., 2017. In Abb. 6.11 sind weitere unabhängige Merkmale visualisiert.

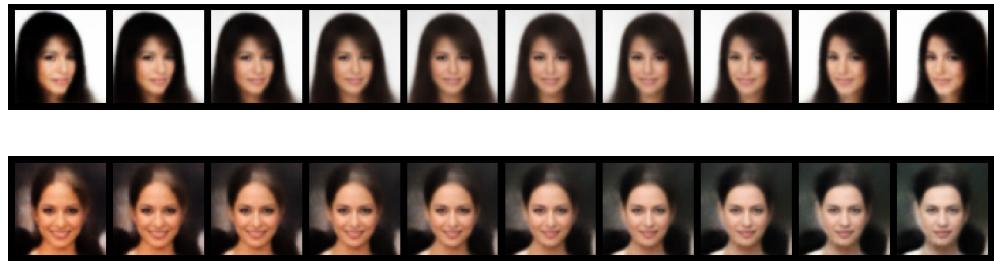


Abbildung 6.10: Sweep durch eine Dimension im Latent-Space für ein gegebenes Originalbeispiel (Die anderen Dimensionen sind fest). Oben zu sehen der VAE mit β_{norm} für $\beta = 150$, unten β_{norm} mit $\beta = 1$. Der höhere β Wert erzielt eine bessere Entkopplung der Merkmale. Unten zu sehen ist, dass das Merkmal "Hautfarbe" und "Lächeln" gekoppelt sind.

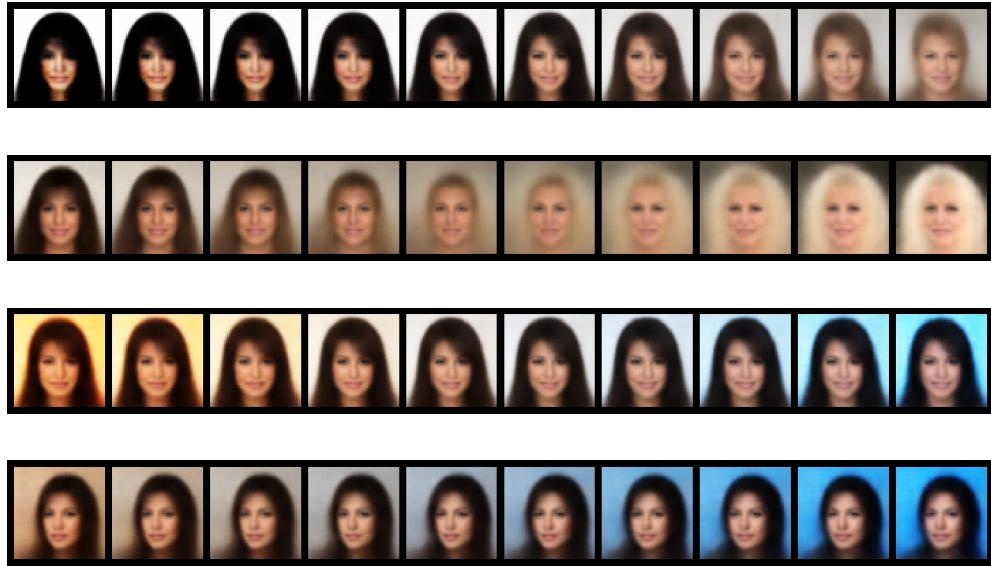


Abbildung 6.11: Weitere entkoppelte Dimensionen des *Disentangled-VAE*. Oben: Das Merkmal "Haarfarbe". Unten: Die Hintergrundfarbe.

6.6 Limitationen von VAE

In den durchgeföhrten Experimenten ist aufgefallen, dass die Rekonstruktionen, welche der VAE erzeugt, unscharf sind. Dies betrifft vor allem den Hintergrund bei den Daten des CelebA Datensatzes. Für Datensätze, welche keine organischen Merkmale wie Gesichtszüge besitzen, ist dieser Effekt noch stärker ausgeprägt. Dies ist z.B. beim CIFAR-10 Datensatz der Fall (siehe Abb. 6.13).

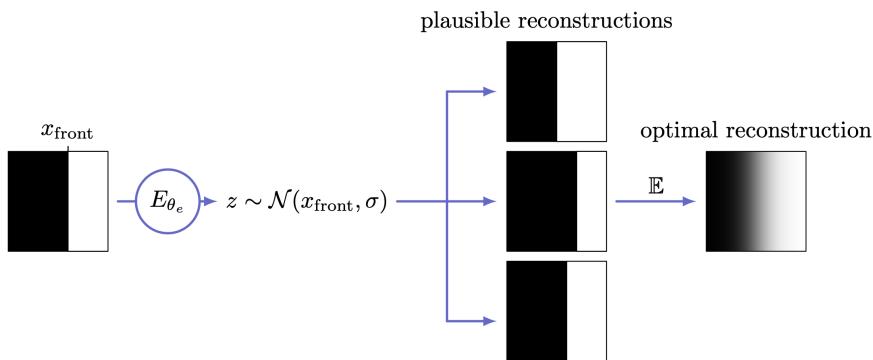


Abbildung 6.12: Grund für die unscharfen Rekonstruktionen des VAEs ist die Modellierung der Ausgabe über eine Verteilung im Latent-Space. Damit sind mehrere Rekonstruktionen plausibel und ergeben eine unscharfe optimale Rekonstruktion.

Eine Erklärung für dieses Phänomen bei VAEs liefern Plumerault, Borgne und Hudelot, 2020 (siehe Abb. 6.12). Betrachtet wird eine Kante im Eingabebild i und der Latentvektor z , welcher die Position x_{front} der Kante beschreibt. Wenn nun

$$Q_\phi(z|i) = \mathcal{N}(x_{front}, \sigma)$$

, dann ist die Verteilung der plausiblen Rekonstruktionen gegeben durch

$$P_\theta(x_{front}|z) = \mathcal{N}(z, \sigma)$$

. Die optimale Rekonstruktion des betrachteten Pixels ist durch den Erwartungswert über diese Verteilung gegeben, was einem weichen Übergang entspricht.

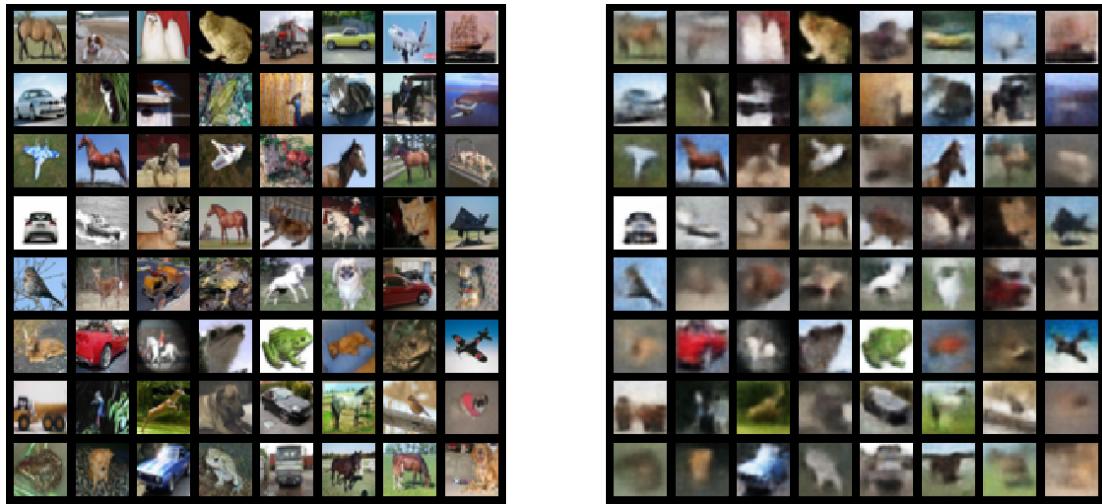


Abbildung 6.13: Die Rekonstruktionen auf dem Datensatz CIFAR-10 sind besonders stark von der unschärfe des VAEs betroffen. In einigen Fällen ist die Klasse nicht mehr ableitbar.

Kapitel 7

Fazit

In dieser Arbeit wurden der Ansatz der Data Augmentation mit Variational Autoencoder auf unterschiedlichen Datensätzen untersucht. Die Verfahren wurden unter verschiedenen Szenarien evaluiert, welche in der Menge, Balance und Art der Daten variieren. Außerdem wurde die Repräsentation der Daten im Latent-Space durch die verschiedenen Modelle analysiert. Die wichtigsten Erkenntnisse werden im Folgenden zusammengefasst.

In der Evaluation konnten Schwierigkeiten des VAE Ansatzes im Umgang mit diskreten Attributen beobachtet werden. Dies machte diesen Ansatz auf numerischen Daten sehr Hyperparameter sensitiv. Außerdem wurden Probleme beim generieren neuer Daten festgestellt, da der Latent-Space eine kontinuierliche probabilistische Struktur hat. Die erzeugten Beispiele verbessern die Klassifikation daher nur geringfügig. Auf imbalancierten Datensätzen fiel die Verbesserung zusätzlich kleiner aus, da die stärker repräsentierte Klasse bessere Rekonstruktionen zur Folge hat. Im Vergleich zu den in der Arbeit von Moreno-Barea, Jerez und Franco, 2020 genutzten generativen Methoden schneidet der VAE Ansatz schlechter ab. Im Few-Shot Szenario kann dagegen eine geringe, aber konsistente Verbesserung erreicht werden. Diese nimmt jedoch mit der Anzahl an original Beispielen ab.

Im Fall von Bilddaten zeigt der VAE eine, für die Modifikation von Bildern nützliche, Latent-Space Struktur. Insbesondere weiche Übergänge und organische Formen, wie Gesichtszüge, werden gut repräsentiert und erlauben Augmentierungen. Ein höher gewichteter KL-Term führte außerdem zu einer besseren Dekorrelation unterschiedlicher Merkmale in den Dimensionen des Latent-Spaces (*Disentangled-VAE*). Im Allgemeinen sind die Rekonstruktionen des VAEs aber unscharf. Dies zeigt sich besonders bei harten Übergängen. Im Few-Shot Szenario zeigte sich auch hier eine konsistente Verbesserung des F-Scores. Außerdem konnte beobachtet werden, dass das selbst-überwachte Trainieren eines Single-VAEs bessere Ergebnisse erzielt, da wesentlich mehr Daten genutzt werden können.

KAPITEL 7. FAZIT

Zusammenfassend haben Variational Autoencoder ein großes Potenzial, reichen alleine jedoch nicht aus um herkömmliche Data Augmentation Techniken vollständig zu ersetzen. Das Verfahren ist im Allgemeinen äußerst Hyperparameter sensitiv. Die Encoder-Decoder Struktur bietet jedoch enorme Vorteile, da kontrollierbare Augmentationen im Latent-Space durchführbar sind. Mögliche Lösungsansätze für die Probleme und weitere Verbesserungen sind im folgenden Ausblick ausgeführt.

Kapitel 8

Ausblick

Es wurden bereits die Vorteile und Nachteile des VAE basierten Ansatzes der Data Augmentation vorgeführt. Im anschließenden Kapitel wird ein Ausblick über Weiterentwicklung der in dieser Arbeit vorgestellten Konzepte gegeben.

8.1 Few-Shot Learning

In der vorliegenden Arbeit wurde beobachtet, dass die verwendeten Augmentierungs Techniken, besonders im Few-Shot Szenario zu konsistenten Verbesserungen führten. Man beachte, dass diese Verbesserungen ausschließlich über Data Augmentation erzielt und keine Few-Shot Learning spezifischen Ansätze verwendet wurden. Eine vielversprechende Idee ist daher, die VAE basierte Data Augmentation mit *State-of-the-Art* Few-Shot Learning Ansätzen, wie "Transfer-Learning" und "Meta-Learning", zu kombinieren.

8.2 Weitere VAE Modelle

8.2.1 Denoising und Conditional VAE

Für die Anwendung von VAE Modellen auf den numerischen Daten des PROBEN1 Datensatzes, wurden von Moreno-Barea, Jerez und Franco, 2020 zwei Weiterentwicklungen vorgestellt: Der Denoising-VAE (DVAE) und der Conditional-VAE (CVAE). Der DVAE hat dieselbe Funktionsweise wie schon der in Abschnitt 3.2.1 angeführte Denoising Autoencoder. Das Eingabebeispiel x wird durch eine modifizierte Variante \tilde{x} ersetzt. Der CVAE gibt dem VAE zusätzlich zum gesampelten z die Klassen Information y mit. Die Ausgabeverteilung des Encoders und Decoders hängt somit jeweils auch von der Klasse des gesampelten Latent-Vektors ab. Die Fehlerfunktion von DVAE und CAE haben folgende Form:

$$\mathcal{L}_{DVAE} = \mathbb{E} [\log P_\theta(x|z) - \mathcal{D}_{KL} [Q_\phi(z|\tilde{x}) \| \mathcal{N}(0, 1)]] \quad (8.1)$$

$$\mathcal{L}_{CVAE} = \mathbb{E} [\log P_\theta(x|z, y) - \mathcal{D}_{KL} [Q_\phi(z|x, y) \| \mathcal{N}(0, 1)]] \quad (8.2)$$

8.2.2 VAE-GAN

Larsen u. a., 2016 schlagen in ihrer Arbeit eine Kombination des VAE Modells mit einem Generative-Adversarial-Network (GAN) vor. GANs erzeugen wesentlich schärfere Bilder, haben aber keinen Encoder. Dies hat zur Folge, dass sie weniger Kontrolle über die Rekonstruktionen bieten, da keine Manipulationen im Latent-Space möglich sind. Das vorgeschlagene VAE-GAN Modell kombiniert den Encoder des VAEs mit der Generator-Discriminator Architektur des GANs, indem der Decoder durch den Generator und Discriminator ersetzt wird (siehe Abb. 8.1). VAE-GANs haben die gleichen Latent-Space Eigenschaften wie VAEs, erzeugen aber wesentlich schärfere Bilder, wie in Abb. 8.2 zu sehen ist. Damit führen die Autoren die Vorteile dieser beiden generativen Ansätze in einem Modell zusammen.

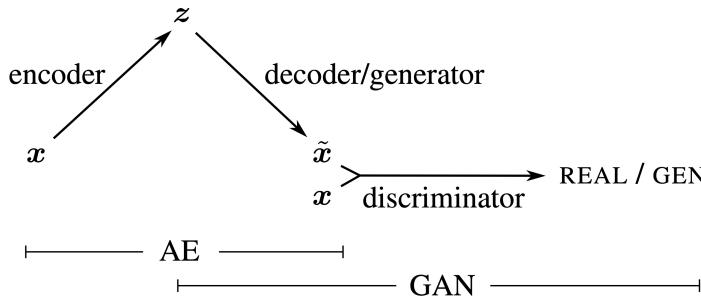


Abbildung 8.1: Das VAE-GAN Modell. Der Decoder des VAEs wird durch den Generator und den Discriminator ersetzt. Abbildung entnommen aus Larsen u. a., 2016

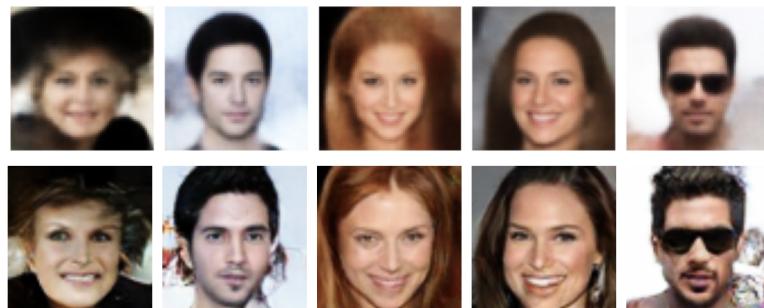


Abbildung 8.2: Die Rekonstruktionen unseres VAE Modells (oben). Die Rekonstruktionen des VAE-GAN Modells (unten) (VAE-GAN Bilder entnommen aus Larsen u. a., 2016)

Literatur

- Antoniou, Antreas, Amos Storkey und Harrison Edwards (2017). „Data augmentation generative adversarial networks“. In: *arXiv*, S. 1–14. arXiv: [1711.04340](https://arxiv.org/abs/1711.04340).
- Brown, Tom B. u. a. (2020). *Language Models are Few-Shot Learners*. arXiv: [2005 . 14165 \[cs.CL\]](https://arxiv.org/abs/2005.14165).
- Byerly, Adam, Tatiana Kalganova und Ian Dear (2021). *A Branching and Merging Convolutional Network with Homogeneous Filter Capsules*. arXiv: [2001.09136 \[cs.CV\]](https://arxiv.org/abs/2001.09136).
- Doersch, Carl (2016). „Tutorial on Variational Autoencoders“. In: S. 1–23. arXiv: [1606 . 05908](https://arxiv.org/abs/1606.05908). URL: <http://arxiv.org/abs/1606.05908>.
- Foret, Pierre u. a. (2021). „Sharpness-aware Minimization for Efficiently Improving Generalization“. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=6Tm1mposlrM>.
- Garay-Maestre, Unai, Antonio Javier Gallego und Jorge Calvo-Zaragoza (2019). „Data augmentation via variational auto-encoders“. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11401 LNCS. April 2020, S. 29–37. ISSN: 16113349. doi: [10 . 1007 / 978 - 3 - 030 - 13469 - 3 _ 4](https://doi.org/10.1007/978-3-030-13469-3_4).
- Goodfellow, Ian, Yoshua Bengio und Aaron Courville (2016). *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press.
- Goodfellow, Ian J. u. a. (2014). *Generative Adversarial Networks*. arXiv: [1406.2661 \[stat.ML\]](https://arxiv.org/abs/1406.2661).
- Hershey, John und Peder Olsen (Mai 2007). „Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models“. In: Bd. 4, S. IV–317. ISBN: 1-4244-0728-1. doi: [10 . 1109 / ICASSP . 2007 . 366913](https://doi.org/10.1109/ICASSP.2007.366913).
- Higgins, Irina u. a. (2017). „B-Vae : Learning Basic Visual Concepts With a Constrained Variational Framework“. In: *Iclr 2017*.
- Jorge, Javier u. a. (2018). „Empirical evaluation of variational autoencoders for data augmentation“. In: *VISIGRAPP 2018 - Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* 5.Visigrapp, S. 96–104. doi: [10 . 5220 / 0006618600960104](https://doi.org/10.5220/0006618600960104).
- Kingma, Diederik P. und Max Welling (2014). „Auto-encoding variational bayes“. In: *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings* Ml, S. 1–14. arXiv: [1312 . 6114](https://arxiv.org/abs/1312.6114).
- Krizhevsky, Alex (Mai 2012). „Learning Multiple Layers of Features from Tiny Images“. In: *University of Toronto*.
- Ladjal, Saïd, Alasdair Newson und Chi-Hieu Pham (2019). *A PCA-like Autoencoder*. arXiv: [1904.01277 \[cs.CV\]](https://arxiv.org/abs/1904.01277).

LITERATUR

- Larsen, Anders Boesen Lindbo u. a. (2016). „Autoencoding beyond pixels using a learned similarity metric“. In: *33rd International Conference on Machine Learning, ICML 2016* 4, S. 2341–2349. arXiv: [1512.09300](https://arxiv.org/abs/1512.09300).
- LeCun, Y. u. a. (Nov. 1998). „Gradient-Based Learning Applied to Document Recognition“. In: *Proceedings of the IEEE* 86.11, S. 2278–2324.
- Liu, Ziwei u. a. (Dez. 2015). „Deep Learning Face Attributes in the Wild“. In: *Proceedings of International Conference on Computer Vision (ICCV)*.
- Lopez Pinaya, Walter Hugo u. a. (2019). „Autoencoders“. In: *Machine Learning: Methods and Applications to Brain Disorders*, S. 193–208. doi: [10.1016/B978-0-12-815739-8.00011-0](https://doi.org/10.1016/B978-0-12-815739-8.00011-0). arXiv: [2003.05991](https://arxiv.org/abs/2003.05991).
- Moreno-Barea, Francisco J., José M. Jerez und Leonardo Franco (2020). „Improving classification accuracy using data augmentation on small data sets“. In: *Expert Systems with Applications* 161, S. 113696. ISSN: 09574174. doi: [10.1016/j.eswa.2020.113696](https://doi.org/10.1016/j.eswa.2020.113696). URL: <https://doi.org/10.1016/j.eswa.2020.113696>.
- Plumerault, Antoine, Hervé Le Borgne und Céline Hudelot (2020). „AVAE: Adversarial Variational Auto Encoder“. In: January, S. 1–16. arXiv: [2012.11551](https://arxiv.org/abs/2012.11551). URL: [http://arxiv.org/abs/2012.11551](https://arxiv.org/abs/2012.11551).
- Senior, Andrew W u. a. (2020). „Improved protein structure prediction using potentials from deep learning“. In: *Nature* 577.7792, S. 706–710. ISSN: 1476-4687. doi: [10.1038/s41586-019-1923-7](https://doi.org/10.1038/s41586-019-1923-7). URL: <https://doi.org/10.1038/s41586-019-1923-7>.
- Sung, Flood u. a. (2017). „Learning to compare: Relation network for few-shot learning“. In: *arXiv*, S. 1199–1208. ISSN: 23318422.

Abbildungsverzeichnis

3.1	Die Encoder-Decoder Struktur des "undercomplete" Autoencoders. Der Encoder berechnet eine komprimierte Repräsentation der Eingabe. Der Decoder ist in der Lage diese wieder zu einem Bild zu rekonstruieren. Abb. entnommen aus Lopez Pinaya u. a., 2019	6
3.2	Zwei Rekonstruktionen von Vektoren aus dem Feature Space. Die Beispiele oben sind ein codierte Eingaben, welche anschließend decodiert wurden. Unten dargestellt sind zufällige decodierte Merkmalsvektoren.	7
3.3	Der DAE minimiert den Fehler zwischen der Rekonstruktion des modifizierten Eingabevektors und der Eingabe. Abbildung entnommen aus Lopez Pinaya u. a., 2019	7
3.4	Der Vorwärtsdurchlauf beim VAE. Der Encoder die Verteilung $Q_\phi(z x)$ parametrisiert durch μ_ϕ und σ_ϕ . Aus dieser wird z erzeugt. Anschließend liefert der Decoder die Rekonstruktionsverteilung $P_\theta(x z)$ bzw. für festes x die Rekonstruktion \hat{x} . (siehe auch Doersch, 2016	10
4.1	Der MNIST Datensatz beinhaltet annotierte Bilder der handgeschriebenen Zahlen $\{0, \dots, 9\}$	13
4.2	CIFAR-10 Datensatz: Die einzelnen Klassen schließen sich gegenseitig aus, d.h. es gib z.b. keine Mischform aus "truck" und "automobile" wie SUV.	15
4.3	Die Gesichtsdaten des CelebA Datensatzes	15
5.1	Die Aufteilung der Datensätze beim Multi-VAE Ansatz. Der vollständige Datensatz wird in klassenweise partitioniert. Zusätzlich wird der Ansatz betrachtet, zu den Daten der Klasse i auch Daten der anderen Klassen beizumischen. Bei mehr als 2 verschiedenen Klassen werden die 20% zusammen aus zufälligen Beispielen der jeweils anderen Klassen gebildet. Abbildung entnommen aus Moreno-Barea, Jerez und Franco, 2020 . . .	18
5.2	Der Generative Classifier dient dazu, verrauschte und verschwommene Daten während des Generations Prozesses auszusortieren. Wenn ein generierter Datenpunkt \tilde{x} als <i>Noise</i> klassifiziert wird, erzeugt der VAE ein neues Beispiel.	19

ABBILDUNGSVERZEICHNIS

6.1 (links) Der Latent Space eines Single-VAE auf dem Datensatz MNIST. Trainiert wurde mit $\beta = 0.5$. Zu sehen ist die klare Trennung der Klassen im äußeren Bereich. Nahe 0 sind die Klassen weniger deutlich getrennt. (rechts) Die Rekonstruktionen von Latent-Vektoren aus dem Intervall $(-3, 3)$. Auch hier sind die Rekonstruktionen im äußeren Bereich deutlich schärfer.	25
6.2 Rekonstruktionen verschiedener Original Ziffern 4 beim Single-VAE. Trainiert wurde mit $\beta = 0.5$. Zu sehen ist das Problem bei Überlappungen von Klassen im Latent-Space. Die Rekonstruktionen sind nicht eindeutig einer 4 oder 9 zuordbar, da durch die geringe Latent-Space Dimension ein zu hoher Informationsverlust mit einhergeht. Höheren Latent-Space Dimensionen verringern diesen Effekt.	25
6.3 Die Interpolations und Extrapolationsmethode. Zu sehen sind das Ergebnis der nächsten-Nachbar Suche im Latent-Space und der inter- / extrapolierte Punkt für jeweils ein Beispiel jeder Klasse. Der α Parameter wurde auf 0.5 gesetzt, was bei einer Interpolation genau dem Mittelpunkt entspricht. Zusätzlich sind die Rekonstruktionen der korrespondierenden Punkte zu sehen.	26
6.4 Der Latent-Space eines Single-VAEs (MNIST) für verschiedene Werte von β . Erkennbar ist, dass zu kleine Werte für β weniger Annahmen über die Latent-Vektoren ermöglichen. Zu große Werte haben dagegen einen vollständigen Strukturverlust zur Folge.	27
6.5 Rekonstruktionen zufälliger MNIST Beispiele eines Single-VAEs. Höhere Werte für β haben deutlich unschärfere rekonstruierte Bilder zur Folge. Extrem kleine Werte beeinträchtigen die Rekonstruktionsqualität dagegen nicht.	27
6.6 Latent-Spaces der Klassenweisen VAEs auf dem "thyroid" Datensatz. Der Datensatz besitzt 15 diskrete Attribute und 6 kontinuierliche Attribute. Besonders im Latent-Space zur Klasse 2 führen diese 15 diskreten Attribute zu 15 Clustern.	28
6.7 Differenz des Baseline F-Scores und des Multi-VAEs auf dem Datensatz "thyroid" im Few-Shot-Szenario. Es wurde jeweils der höchste Wert von VAE und VAE + GC verwendet	31
6.8 Der Verlauf der Modellgüte bei verschiedenen Mengen von generierten Daten. Die Methode Interpolation + Noise übertrifft die Baseline für eine hohe Anzahl. Die Multi-VAE Methode nähert sich asymptotisch der Baseline Qualität.	33
6.9 CelebA - Originalbilder oben, Rekonstruktionen mit verschiedenen Werten für β unten. β_{norm} (links), $150 \cdot \beta_{norm}$ (rechts)	35
6.10 Sweep durch eine Dimension im Latent-Space für ein gegebenes Originalbeispiel (Die anderen Dimensionen sind fest). Oben zu sehen der VAE mit β_{norm} für $\beta = 150$, unten β_{norm} mit $\beta = 1$. Der höhere β Wert erzielt eine bessere Entkopplung der Merkmale. Unten zu sehen ist, dass das Merkmal "Hautfarbe" und "Lächeln" gekoppelt sind.	35

ABBILDUNGSVERZEICHNIS

6.11	Weitere entkoppelte Dimensionen des <i>Disentangled</i> -VAE. Oben: Das Merkmal "Haarfarbe". Unten: Die Hintergrundfarbe.	36
6.12	Grund für die unscharfen Rekonstruktionen des VAEs ist die Modellierung der Ausgabe über eine Verteilung im Latent-Space. Damit sind mehrere Rekonstruktionen plausibel und ergeben eine unscharfe optimale Rekonstruktion.	36
6.13	Die Rekonstruktionen auf dem Datensatz CIFAR-10 sind besonders stark von der unschärfe des VAEs betroffen. In einigen Fällen ist die Klasse nicht mehr ableitbar.	37
8.1	Das VAE-GAN Modell. Der Decoder des VAEs wird durch den Generator und den Discriminator ersetzt. Abbildung entnommen aus Larsen u. a., 2016	42
8.2	Die Rekonstruktionen unseres VAE Modells (oben). Die Rekonstruktionen des VAE-GAN Modells (unten) (VAE-GAN Bilder entnommen aus Larsen u. a., 2016)	42

Tabellenverzeichnis

4.1	PROBEN1 Datensatz Infos.	14
5.1	Architektur der verwendeten VAE Modelle (spaltenweise angegeben). Mit $Linear(m)$ wird ein "Fully-Connected"-Layer mit m Ausgabe-Features beschrieben. $Conv(m, n \times n)$ bezeichnet eine 2D Konvolution mit Kernelgröße $n \times n$, $ConvT$ analog für 2D transponierte Konvolution. m bezeichnet hierbei die Anzahl an Ausgabe Featuremaps der jeweiligen Konvolution. In allen Konvolutionen wurde Zero-Padding der Größe 1 angewandt, ebenso wie eine Stride von 2. Dies führt zu einer Halbierung der Bildgröße in jedem $Conv$ -Layer. Ausnahme stellt das MNIST Klassifikations Netzwerk dar: Hier wurde Stride 2 verwendet und $MaxPooling2D$ um eine Dimensionsreduktion zu erreichen. Für die Ausgabelayer μ und σ ist d die Dimension des Latent-Spaces.	21
6.1	Üblicherweise beschreibt man in der Klassifikation die Beziehung zwischen Vorhersage eines Modells (<i>predicted</i>) und der tatsächlichen Klasse (<i>ground truth</i>) über eine Konfusionsmatrix. Basierend darauf ergeben sich mehrere Bewertungs Metriken.	23
6.2	F-Scores auf den PROBEN1 Datensätzen. Das beste Ergebnis ist jeweils fett gedruckt. Bis auf den "glass" Datensatz verbessert sich der F-Score durch den VAE um ca. 0.007. Es wurden 3 verschiedene Werte für β getestet: 1.0, 0.5 und 1.0 mit β -Normalisation (mit norm angegeben). Höhere Werte für β führten zum Kollaps im Latent-Space und daraus resultierend schlechterer Güte.	30
6.3	Der F-Score auf dem Datensatz "thyroid" im Few-Shot Szenario. Für kleine Trainingsgrößen ergibt sich ein verbesserte Wert von bis zu 0.074. Die Verbesserung nimmt mit der Anzahl originaler Trainingsbeispiele ab. . .	31
6.4	Evaluation des Klassifikators auf reduzierten Partitionen des MNIST Datensatzes. Die Werte stellen jeweils das Maximum aus VAE ohne GC und VAE mit GC dar.	32

TABELLENVERZEICHNIS

6.5 F-Score bei Training auf ausschließlich generierten Daten. Die Baseline wurde auf 5 Originalbeispielen trainiert. Für den VAE stehen dieselben 5 Beispiele zur Verfügung um Daten zu generieren. Anschließend wurden nur die generierten Daten für die Evaluation der jeweiligen Methode verwendet. Die Methode Interpolation + Noise erzielt insbesondere für viele generierte Beispiele eine hohe Modellgüte, teilweise sogar besser als die Baseline.	33
6.6 F-Score auf dem MNIST Datensatz im Few-Shot Szenario. Mit 2000 generierten Beispielen erzielen alle Methoden höhere Werte als bei einer 1:1 Zusammensetzung des Datensatzes.	34