Variational Autoencoder

Proseminar
Softwareentwicklung in der Wissenschaft
Cindy Phung
29. August 2021
Betreuer: Tobias Finn, Jakob Lüttgau

Inhaltsverzeichnis

Was sind Autoencoder und wie funktionieren sie?

Autoencoder sind eine Form von Artifical neural network, also ein künstliches neurales Netzwerk. Sie gehören zu dem Unsupervised bzw. Self-supervised learning. Das bedeutet, dass sie zufällige Daten bekommen und ohne menschliche Aufsicht etwas daraus erschließen können.
Autoencoder werden zur Verringerung von Daten bzw. Dimensionen genutzt. Das heißt, dass sie ihre Input Daten verringern, indem sie diese auf eine kleinere Repräsentation abbilden. Dabei werden nur die wichtigsten Informationen extrahiert und beibehalten. Nachdem diese verringert wurden, kann man aus diesen komprimierten Daten einen neuen Output bilden, der dem Input ähnlich ist.
Ein Autoencoder besteht aus zwei Hauptkomponenten. Zum einen den Encoder und zum anderen den Decoder. Dabei bestehen Encoder und Decoder jeweils aus mehreren Schichten. Eine Schicht beschreibt eine bestimmte Anzahl von Neuronen, die miteinander verbunden sind. Beim Encoder ist die erste Schicht immer die sogenannte „Input layer“. Diese hat eine feste Anzahl an Neuronen, das heißt, alle Input Daten müssen, in einem Vektor codiert, genauso viele Dimensionen haben, wie die Input Schicht an Neuronen besitzt. Ist diese Voraussetzung nicht erfüllt, so kann der Encoder die Input Daten nicht lesen.
Alle weiteren Schichten des Encoders nennt man „Hidden layers“ und bestehen genauso wie die erste auch aus vernetzten Neuronen. Jedoch verringert sich die Anzahl der Neuronen immer weiter, je mehr Schichten existieren. Je nach dem wie der Autoencoder konstruiert wurde, kann es mehr oder weniger Schichten geben. Nachdem die Input Daten durch alle Schichten durchlaufen sind, kommt es zu dem sogenannten „Bottle neck“, welcher auch latenter Vektor oder latenter Bereich genannt wird. Dieser beschreibt den Vektor, der entsteht wenn man Input Daten durch den Encoder schickt. Er enthält die wichtigsten Informationen, die aus den Input Daten extrahiert wurden.
Die Größe des latenten Vektors beschreibt auch die Dimension des Autoencoders. So spricht man von einem 5-dimensionalen Autoencoder, wenn dieser einen latenten Vektor hat, der aus 5 Variablen besteht.
Bei dem Decoder ist dies ähnlich wie beim Encoder nur andersherum. Jede weitere Schicht des Decoders hat eine höhere Anzahl an vernetzten Neuronen als die letztere. Die aller letzte Schicht nennt man „Reconstruct layer“.
Der Decoder wird benutzt um die Input Daten zu rekonstruieren. Dies geschieht, indem man versucht einen Output zu kreieren mithilfe des latenten Vektors. Dieser sollte dem Input so ähnlich wie möglich sein.

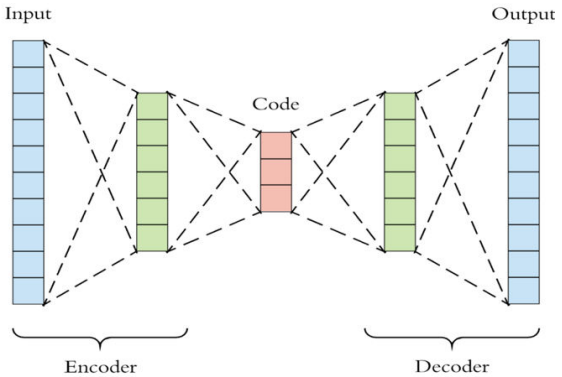

Abb.1: Modell eines Autoencoder
https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

Da es sich bei einem Autoencoder um unsupervised learning handelt, braucht es keine menschliche Aufsicht, die bestätigt, ob der Output richtig rekonstruiert wurde oder nicht. Dies macht der Autoencoder eigenständig.
Dazu vergleicht er erstmal den Output mit dem Input und berechnet den sogenannten Reconstruction loss. Danach nutzt der Autoencoder die Methode Backpropagation.
Das bedeutet, dass man rückwärts durch das neuronale Netzwerk traversiert und dabei die Gewichtungen der Neuronen ändert.
In der Trainingsphase werden dem Netzwerk jede Menge Daten gegeben. Dabei wird die Gewichtung der Neuronen bei jedem Input erneut geändert, sodass sich das Ergebnis immer weiter an die gewünschte Ausgabe annähert.

Was sind Variational Autoencoder und was unterscheidet sie von Autoencoder?

Variational Autoencoder sind prinzipiell ähnlich aufgebaut wie Autoencoder, jedoch unterscheidet sich hierbei der Bottleneck. Beim Autoencoder bestand dieser nur aus dem latenten Vektor. Beim Variational Autoencoder jedoch gibt es insgesamt drei Vektoren. Einmal den mean vector, den standard deviation vector und den sampled latent vector. Der mean vector spiegelt den konkreten Mittelwert wieder. Der standard deviation vector stellt die Standardabweichung dar und der sampled latent vector wählt zufällige Werte aus dem standard deviation vector und dem mean vector aus, die dann dem Decoder übergeben werden.

Dadurch das der sampled latent vector zufällig Werte wählt, können nicht nur Input Daten rekonstruiert werden, sondern auch neuer Inhalt produziert werden.

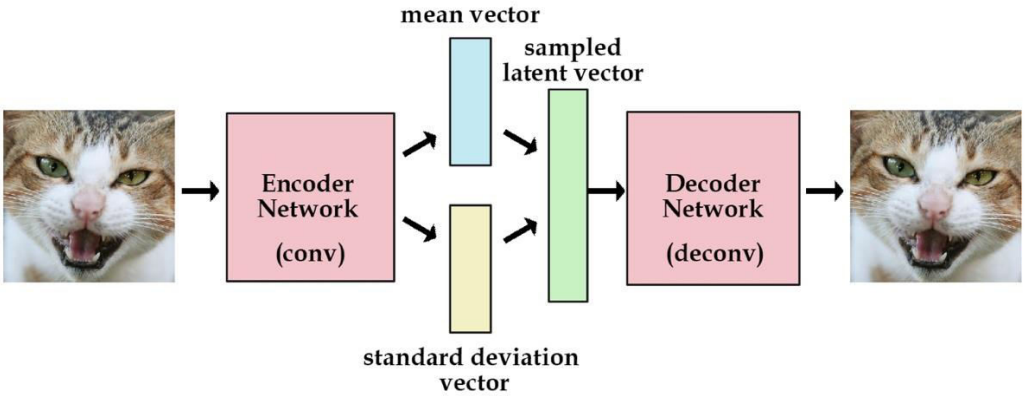

Abb.2: Modell Variational Autoencoder
https://wizardforcel.gitbooks.io/tensorflow-examples-aymericdamien/content/3.10_variational_autoencoder.html

Ein wichtiger Unterschied zur Konstruktion von Variational Autoencoder im Gegensatz zu normalen Autoencodern ist, dass bei Variational Autoencoder nicht nur der Reconstruction loss eine wichtige Rolle spielt, sondern auch die Regularization.

Was ist Regularization und warum ist sie notwendig?

Im Variational Autoencoder arbeiten wir nicht mit konkreten Werten, sondern mit einer Wahrscheinlichkeitsverteilung. Diese Wahrscheinlichkeitsverteilung wird aus den Informationen der Input Daten gebildet. Wie auch schon zuvor beschrieben, werden aus dieser Verteilung zufällige Werte ausgewählt, die dann dem Decoder übergeben werden. Regularization bedeutet, dass diese Wahrscheinlichkeitsverteilung so nah wie möglich an der Standardnormalverteilung sein sollte.
Angenommen, wir würden die Wahrscheinlichkeitsverteilung nicht regulieren, so kann es sein, dass der Decoder einen Output erzeugt, der nicht sinnvoll ist. Das liegt daran, dass es ohne Regularization dazu kommen kann, dass Punkte ausgewählt werden, die nach dem decodieren, keinen sinnvollen Inhalt haben. Um dies zu verhindern wird Regularization verwendet.
Dabei sind zwei wichtige Punkte zu beachten. Zum einen die Kontinuität und zum anderen die Vollständigkeit.
Kontinuität bedeutet, dass zwei naheliegende Punkte nach dem decodieren nicht komplett verschieden sein sollten, sondern einen ähnlichen Output haben sollten.

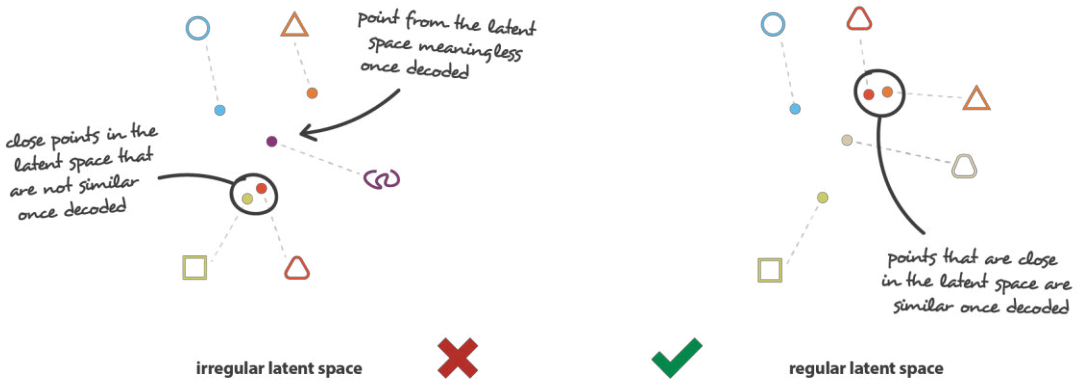

Abb.3: Unterschied zwischen regulärem und nicht regulärem latenten Bereich im Bezug auf Kontinuität
https://miro.medium.com/max/2000/1*83S0T8IEJyudR_I5rI9now@2x.png

Mit Vollständigkeit ist gemeint, dass ein ein zufällig ausgewählter Punkt im latenten Bereich auch zu einem sinnvollen Output führen sollte. Somit sollte es keine „Lücken“ geben, sondern es sollte einen fließenden Übergang bilden.

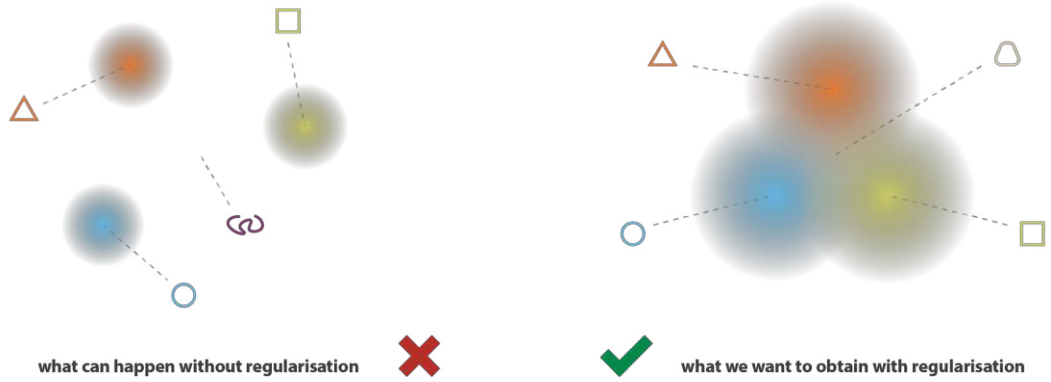

Abb.4: Unterschied zwischen regulärem und nicht regulärem latenten Bereich im Bezug auf Vollständigkeit
https://miro.medium.com/max/2000/1*9ouOKh2w-b3NNOVx4Mw9bg@2x.png

Worum handelt es sich beim Reparametrisierungstrick?

Variational Autoencoder gehören, sowie Autoencoder allgemein zu den Unsupervised learning Methoden. Der Autoencoder lernt eigenständig, indem er den rekonstruierten Output mit dem Input vergleicht und den Reconstruction loss berechnet. Danach wird Backpropagation angewendet, um die Gewichtungen der einzelnen Neuronen zu verändern. Diese Methode wird auch beim Variational Autoencoder angewendet. Jedoch entsteht hierbei ein Problem, da wir hier immer zufällige Werte aus der latenten Verteilung wählen, die dann an den Decoder übergeben werden. Dadurch das diese zufällig sind kann man nicht Backpropagation anwenden. Als Lösung für dieses Problem nutzt man den Reparametrisierungstrick. Das heißt wir führen einen neuen Vektor ε ein, welcher zufällig zwischen den Werten -1 und 1 gewählt wird. Dieser wird dann mit dem standard deviation vector σ multipliziert und zu dem mean vector μ addiert.
Ohne den Reparametrisierungstrick sind mean vector und der standard deviation vector deterministisch, während der latente Vektor zufällig ist. Dadurch dass wir diesen neuen Vektor ε einführen, ist dieser zufällig und dadurch wird der latente Vektor auch deterministisch. Somit ist das Problem gelöst und man kann Backpropagation anwenden.

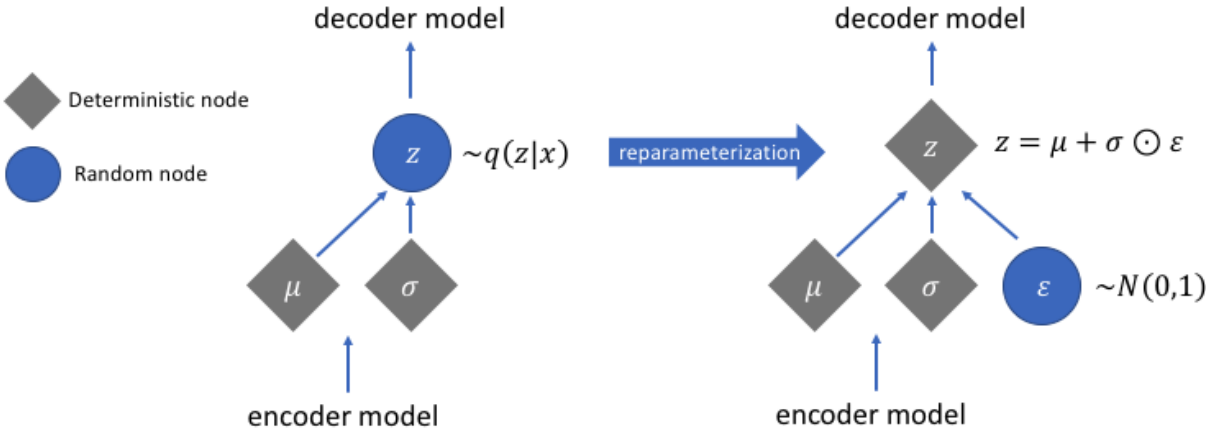

Abb.5: Reparametrisierungstrick
https://www.jeremyjordan.me/content/images/2018/03/Screen-Shot-2018-03-18-at-4.36.34-PM.png

Anwendungsbeispiel

Für unser Anwendungsbeispiel betrachten wir einen Variational Autoencoder mit 6-dimensionalen Encoding. Der Datensatz hierbei sind Bilder von Gesichtern. Da es ein 6-dimensionales Encoding ist, hat der latente Vektor 6 verschiedene Werte. In diesem Fall wären es das Lächeln, die Hautfarbe, das Geschlecht, wie viel Bart die Person besitzt, ob sie eine Brille trägt und die Haarfarbe. Bei einem Autoencoder würde man für jede Variable einen konkreten Wert bekommen, welcher dann dem Decoder übergeben wird.

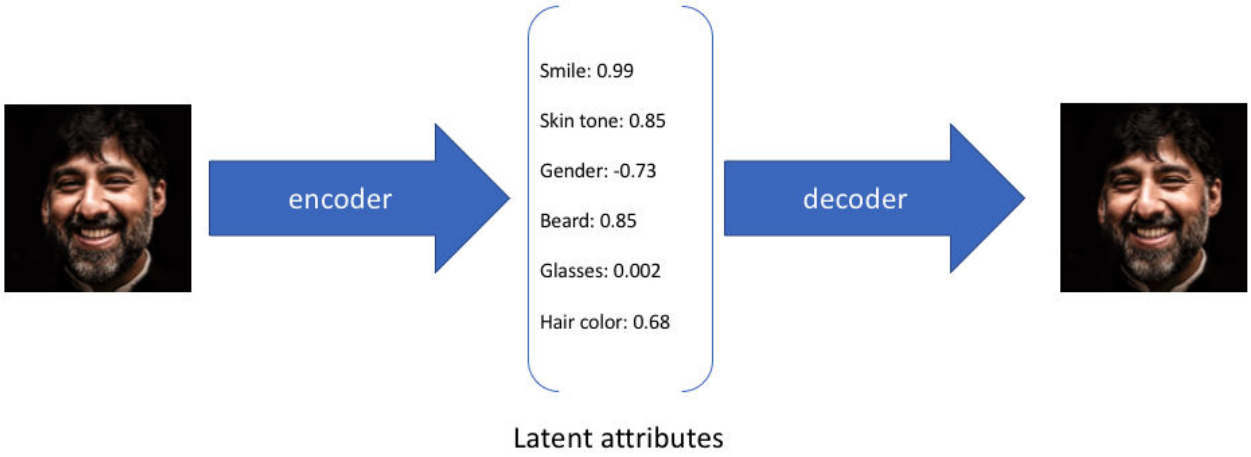

Abb.6: 6-dimensionaler Autoencoder für Bilder von Gesichtern
https://www.jeremyjordan.me/variational-autoencoders/

Beim Variational Autoencoder jedoch berechnet man für jeden Wert eine Wahrscheinlichkeitsverteilung, aus dem dann zufällige Werte gewählt werden.

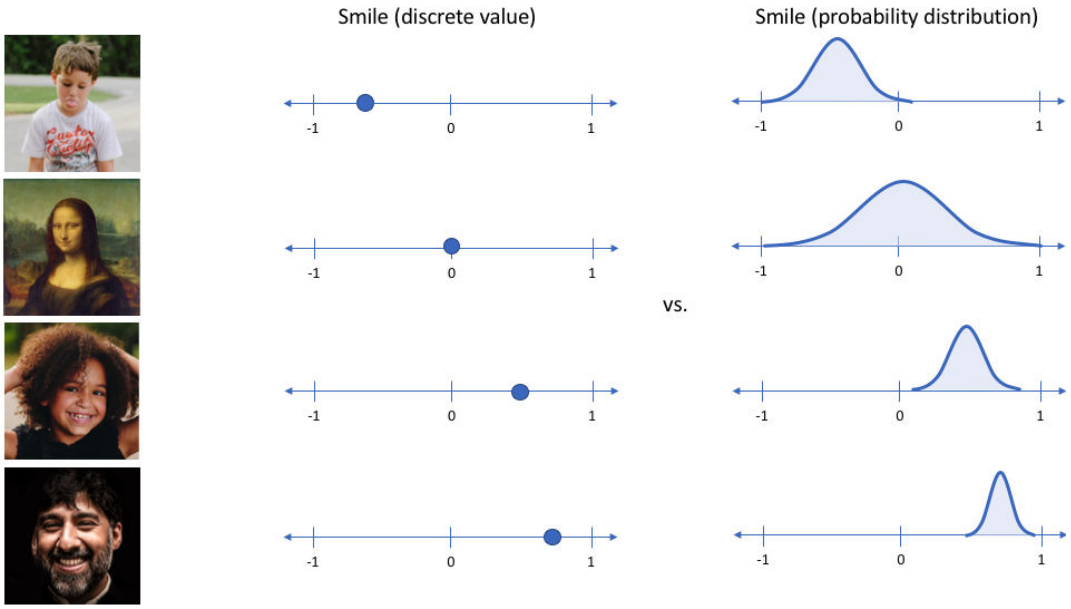

Abb.7: Wahrscheinlichkeitsverteilung bei Autoencoder
https://www.jeremyjordan.me/variational-autoencoders/

Dadurch entsteht dieser Autoencoder, welcher mit Wahrscheinlichkeitsverteilungen arbeitet.

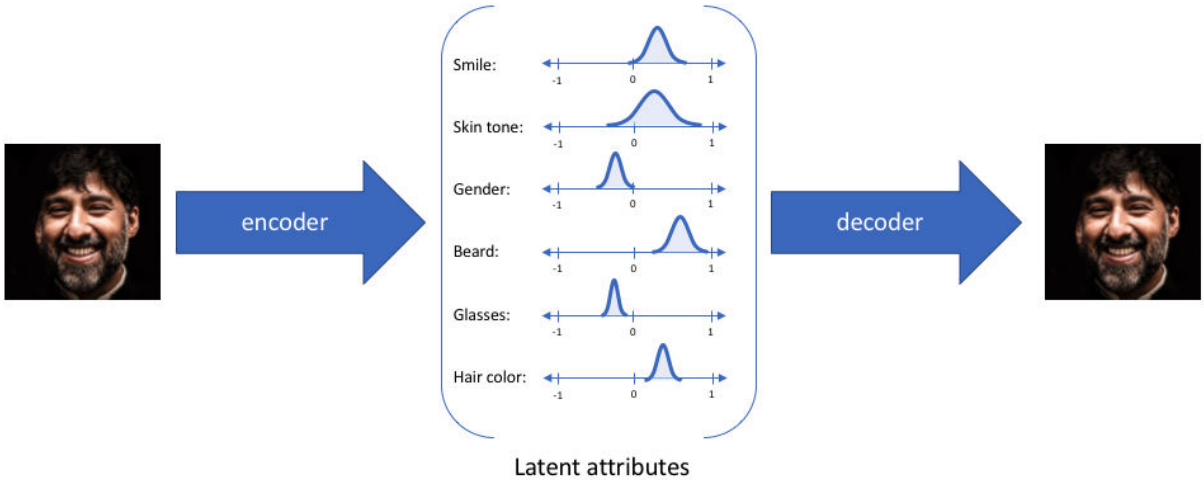

Abb.8: Wahrscheinlichkeitsverteilung im Autoencoder
https://www.jeremyjordan.me/variational-autoencoders/

Wenn man nun zufällige Werte aus diesen Verteilungen auswählt und sie dem Decoder übergibt, sollten ähnliche Bilder rekonstruiert werden.

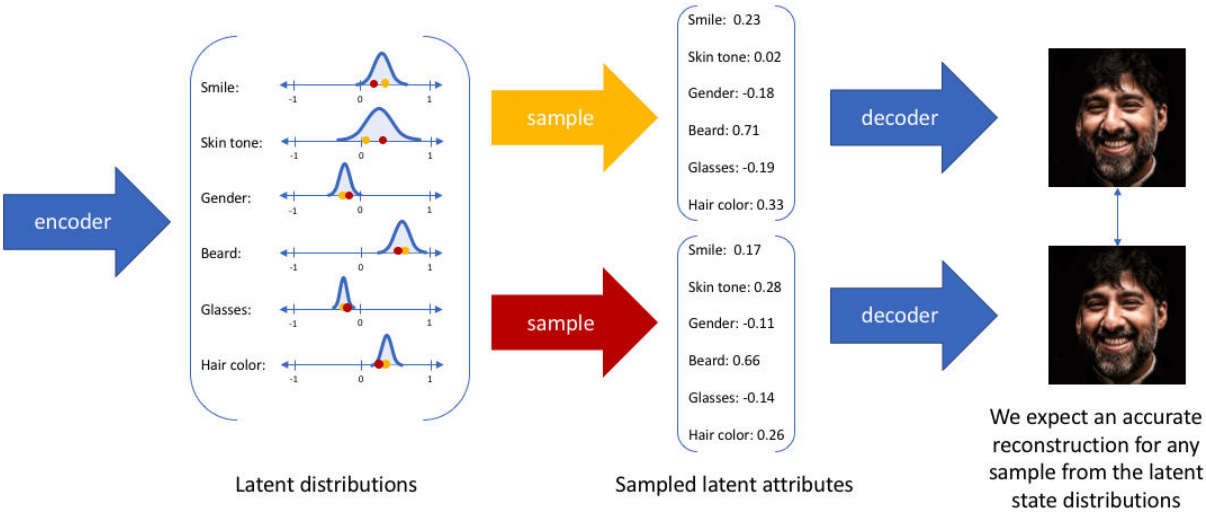

Abb.9: 2 Samplings aus einer latenten Verteilung
https://www.jeremyjordan.me/variational-autoencoders/

Wie werden Autoencoder und Variational Autoencoder genutzt?

Es gibt viele verschiedene Arten von Autoencodern, wie den Variational Autoencoder oder den Denoising Autoencoder. Jeder diese Arten kann auch auf verschiedene Weise genutzt werden.
Im Folgenden werden 3 verschiedene Beispiele vorgestellt in Bezug auf die Nutzung von Autoencoder.

1. Denoising von Bildern
   Bei diesem Beispiel betrachten wir den Denoising Autoencoder. Diese sind dazu fähig aus Bildern mit Noise, das Bild ohne Noise zu konstruieren. Dadurch kann man zum Beispiel Wasserzeichen von Bildern entfernen ohne das originale Bild komplett zu verändern.

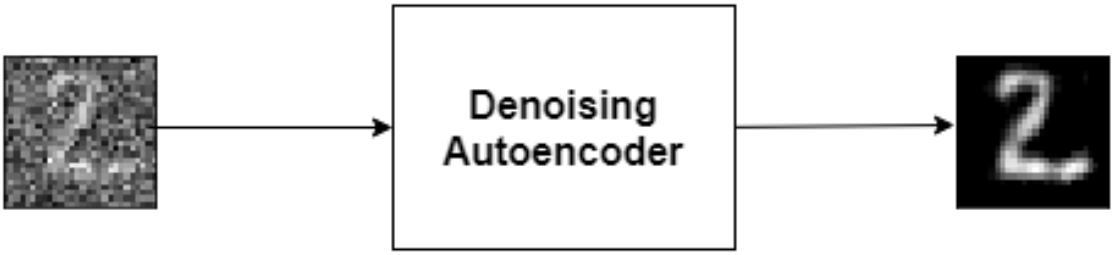

Abb.10: Denoising Autoencoder
https://theailearner.com/tag/denoising-autoencoder/

2. Generieren von neuen Charakteren oder Menschen
   Dieses Beispiel bezieht sich auf den Variational Autoencoder, da dieser nicht nur Dimensionen verringern und Daten rekonstruieren kann, sondern auch neuen Inhalt erschaffen kann. So kann man einen Variational Autoencoder mit einem Datensatz an Bildern von Gesichtern trainieren und kann dadurch auch Bilder mit künstlich erzeugten Gesichtern kreieren.
   Diese Methode kann man bei einer Videospiel Entwicklung nutzen, um neue Charaktere zu erschaffen.

Abb.11: Künstlich generierte Gesichter
https://jaan.io/images/variational-autoencoder-faces.jpg

3. Empfehlungssysteme
   Beim letzten Beispiel handelt es sich um die Empfehlungssysteme, die sehr häufig auftreten. Sie kommen in verschiedenen Bereichen wie Online-Shopping, Filme oder Musik Streaming und ähnlichen vor. Hierbei wird ein Autoencoder genutzt, um die Interessen des Nutzers herauszufinden und ähnlich Elemente vorzuschlagen.
   Angenommen, wir betrachten ein Empfehlungssystem einer Streaming Plattform für Serien und Filme. Zuerst erfasst der Encoder des Autoencoder die Interessen des Nutzers.
   Der Autoencoder versucht alle Filme und Serien zu kategorisieren. Dies kann sich immer variieren, denn ob man die Filme und Serien nach Genre, Schauspieler oder Regisseur kategorisiert ist nicht klar vorgegeben.
   Nehmen wir an der Nutzer interessiert sich für Komödien, so versucht der Decoder des Autoencoders diese Information auf andere Inhalte zu projizieren.
   Je mehr der Nutzer streamt, desto ähnlicher werden die vorgeschlagenden Inhalte.

Was ist MNIST?

MNIST ist ein Akronym und steht für Modified National Institute of Standards and Technology. Es enstand 1998 aus NIST (National Institute of Standards and Technology) und ist eine Datenbank an handgeschriebenen Ziffern. Diese werden genutzt, um neuronale Netzwerke zu trainieren und handgeschriebene Ziffern zu erkennen. Insgesamt gibt es 60.000 Beispiele im Trainingssatz und 10.000 Beispiele im Testdatensatz.

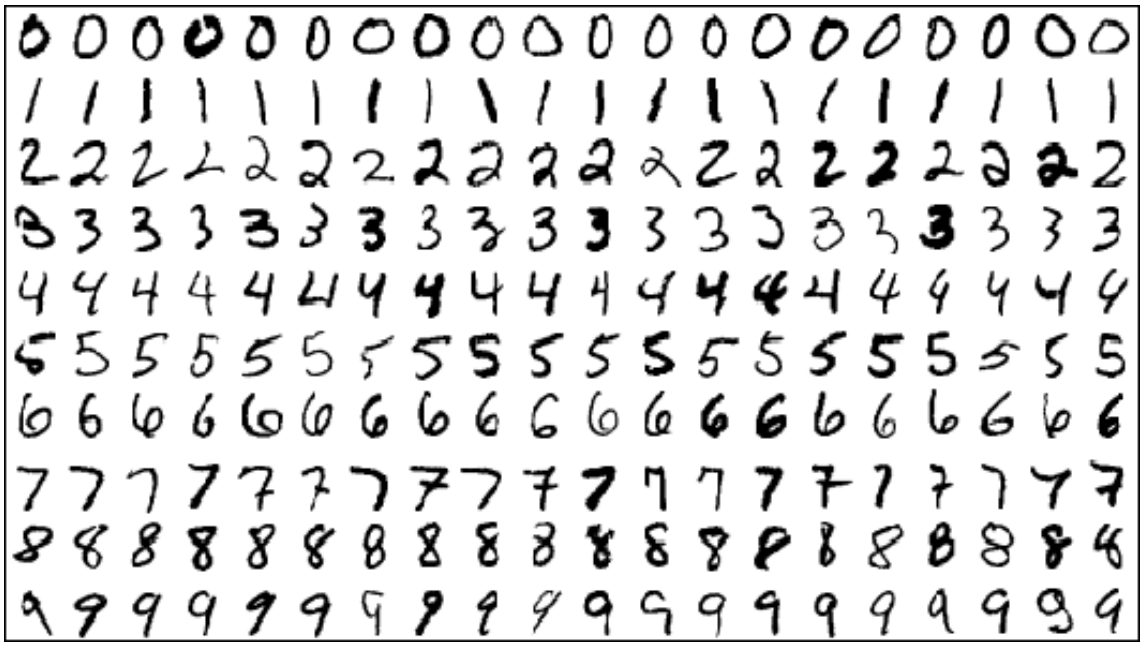

Abb.12: MNIST Datensatz
https://miro.medium.com/max/530/1*VAjYygFUinnygIx9eVCrQQ.png

Eine ähnlich Datenbank, die zum Trainieren von neuronalen Netzwerken erschaffen wurde ist das Fashion-MNIST. Diese entstand durch Zalando und enthält verschiedene Bilder von Kleidungsstücken. Sie soll neuronale Netzwerke trainieren, Kleidungsstücke zu erkennen, sodass diese Netzwerke für ein Empfehlungssystem genutzt werden können.

Quellenverzeichnis

Informationsquellen:
https://arxiv.org/abs/1312.6114
https://arxiv.org/abs/1606.05908
https://www.jeremyjordan.me/variational-autoencoders/
https://medium.com/analytics-vidhya/generative-modelling-using-variational-autoencoders-vae-and-beta-vaes-81a56ef0bc9f
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73
https://en.wikipedia.org/wiki/Autoencoder
https://iq.opengenus.org/applications-of-autoencoders/

Anwendungsbeispiel:
https://www.jeremyjordan.me/variational-autoencoders/

MNIST:
https://en.wikipedia.org/wiki/MNIST_database
https://de.wikipedia.org/wiki/MNIST-Datenbank