



Duale Hochschule Baden-Württemberg Mannheim

# Advanced Applied Machine Learning – Audio Similarity

***Wirtschaftsinformatik Schwerpunkt Data Science***

Verfasser:	Alisa Rogner (7894464), Florian Frey (7199749), Frederick Neugebauer (4521985), Philipp Dingfelder (8687786)
Kurs:	WWI 20 DSB
Modul:	Advanced Applied Machine Learning
Dozent:	Prof. Dr. Maximilian Scherer
Bearbeitungszeitraum:	22.05.2023 – 24.07.2023

# Ehrenwörtliche Erklärung

Wir versichern hiermit, dass wir die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.



---

23.07.2023, Florian Frey



---

23.07.2023, Alisa Rogner



---

23.07.2023, Frederick Neugebauer



---

23.07.2023, Philipp Dingfelder

# Abstract

In dieser Arbeit wurde die Fragestellung untersucht, was Musik ähnlich macht und im Zuge dessen eine Möglichkeit gesucht die Ähnlichkeit von Liedern mithilfe von Machine Learning Algorithmen und Methoden vorherzusagen. Dabei wurden zuerst verschiedene Features herauskristallisiert, die einen Einfluss auf die Ähnlichkeit von Musik haben können, sowie Möglichkeiten gesucht wurden Musik abseits von Audiosignalen darzustellen, bevor verschiedene Modelle trainiert wurden, die die Ähnlichkeiten von Liedern anhand der Features aus dem ausgewählten Datensatzes vorherzusagen.

Die Entscheidung fiel dabei auf ausgewählte Klassifikatoren, sowie verschiedene Autoencoder, Variational Autoencoder und ein Convolutional Neural Network. Diese Klassifikatoren wurden dabei zur Bestimmung von Genres genutzt, während die anderen Modelle verwendet wurden, um Ähnlichkeiten zwischen Liedern selbst herauszufinden, indem sie auf 30 Sekunden Abschnitten von Liedern trainiert wurden. Die Ergebnisse der verschiedenen Modelle wurden schließlich miteinander verglichen, wobei nach der graphischen und nach der statistischen Evaluation ein Autoencoder als das beste Modell ausgewählt wurde. Dieser wurde schließlich nochmals auf 3 Sekunden langen Liedabschnitten trainiert, was die Accuracy abermals verbesserte.

Die Ähnlichkeit der Lieder wird schließlich über eine Cosine-Similarity zwischen den verglichenen Liedern festgestellt. Dies geschieht innerhalb einer Redis Datenbank, in welcher die Vektoren und die Cosine-Similarity enthalten sind, welche schließlich an ein Frontend angebunden wurde.

Im Frontend selbst hat der Nutzer zwei Möglichkeiten sich ein Lied auszusuchen, zu welchem er ähnliche Lieder haben möchte, durch einen Song Upload oder durch eine Suche in der Redis Datenbank. Danach wird das ausgewählte Lied mit anderen aus der Datenbank verglichen werden, und die Ergebnisse der Ähnlichkeit werden angezeigt.

# Inhaltsverzeichnis

Abbildungsverzeichnis.....	v
1 Einleitung.....	1
2 Musiktheoretische Grundlagen.....	3
2.1 Ähnlichkeitsmerkmale von Musik .....	3
2.2 Darstellungsmöglichkeiten von Audiosignalen .....	5
3 Verwendete Algorithmen.....	7
3.1 Klassifikatoren .....	7
3.2 Vektordatenbanken und Kosinus-Ähnlichkeit.....	8
3.3 Dimensionsreduktionsalgorithmen .....	8
3.3.1 Autoencoder.....	9
3.3.2 Variational-Autoencoder.....	10
3.3.3 Convolutional Neural Network.....	11
3.4 Evaluierungsmetriken .....	12
3.4.1 PCA .....	12
3.4.2 TSNE.....	13
3.4.3 K-Nearest-Neighbours Klassifikation.....	13
4 Verwendeter Datensatz .....	14
5 Machine-Learning-Modelle und deren Auswertung.....	15
5.1 Genreklassifizierung zur Featureauswahl.....	15
5.2 Dimensionsreduktionsmodelle und deren Evaluierung .....	16
6 Applikation .....	19
6.1 Technische Umsetzung.....	19
6.2 Frontend .....	21
7 Fazit und Ausblick.....	24
Literaturverzeichnis.....	25

# Abbildungsverzeichnis

Abbildung 1: Visualisierung eines Convolutional Autoencoders .....	9
Abbildung 2: Variational Auto Encoder .....	11
Abbildung 3: Darstellung der CNN-Architektur zur Dimensionsreduktion .....	12
Abbildung 4: Vergleich der rekonstruierten Spektrogramme mit dem Original. ....	17
Abbildung 5: Modellvergleich durch PCA.....	17
Abbildung 6: Modellvergleich mit TSNE .....	18
Abbildung 7: Ergebnis des Vergleichs der verschiedenen Modelle mithilfe von kNN.....	18
Abbildung 8: Dateiupload.....	21
Abbildung 9: Songsuche mithilfe der Redis-Datenbank.....	22
Abbildung 10: Anzeige des ausgewählten Songs. ....	22
Abbildung 11 Ergebnis der Suche nach ähnlichen Liedern. ....	23

# 1 Einleitung

In der vorliegenden Arbeit soll mithilfe von Machine-Learning eine Ähnlichkeitsuntersuchung zwischen Audiosignalen stattfinden („Audio Similarity“). In klassischen Machine-Learning-Projekten und in der Lehre kommt die Verarbeitung und Auswertung von Audiosignalen selten zum Einsatz, obwohl sich, wie später zu sehen, auch Audiosignale in Form von Bildern und Zahlen darstellen lassen und somit für mathematische Algorithmen verwertbar sind.

Der Vergleich von Audiosignalen bietet dabei viele mögliche Anwendungsbereiche, die auch kommerziell relevant sind. So wurden im Jahr 2022 allein in Deutschland über 2 Milliarden Euro in der Musikbranche umgesetzt (vgl. [1])

Ein Anwendungsfall ist die Plagiatserkennung zur Feststellung von Urheberrechtsverstößen. Denn das Urheberrechtsgesetz führt „Werke der Musik“ unter §2 explizit als geschütztes Werk auf. Eine automatische Plagiatserkennung ist vor allem durch die ständig wachsende Menge an digitalen Inhalten (vgl. [1]) relevant, da sich hierdurch die manuelle Nachverfolgung von Urheberrechtsverstößen als nahezu unmöglich darstellt. Entgegen gängiger Meinung zählt auch das Covern eines Songs als Urheberrechtsverletzung, sofern man sich keine Lizenz dafür einholt (vgl. §23 Urheberrechtsgesetz). Trotzdem kann es für Künstler oder Plattenfirmen, die ein Cover erstellen oder beispielsweise einen Remix in Auftrag geben interessant sein, wie ähnlich das neue Werk dem Original ist.

Ein weitere Anwendungsfall von Audio Similarity ist eine Recommendation-Engine, also die Empfehlung von Songs anhand der Ähnlichkeit zu gegebenen Songs. Eine solche Recommendation-Engine ist heutzutage vor allem für die zahlreichen Streaming-Anbieter von digitalen Inhalten relevant, um den Benutzern eine möglichst Nutzerfreundliche Anwendung zu bieten. Auch für Künstler, vor allem noch unbekannte Künstler, kann dies hilfreich sein, da sie so von potenziellen Fans entdeckt werden können. Durch den Anwendungsfall ergibt sich auch die interessante Fragestellung welche Eigenschaften von Musik zur menschlichen Wahrnehmung von Ähnlichkeit beitragen.

Aufgrund des Zeitrahmens des Projekts und des ausgewählten Datensatzes, welcher später näher erläutert wird, wurde sich für die Umsetzung einer Recommendation-Engine entschieden. Dazu soll der Nutzer einer entworfenen Anwendung die Möglichkeit haben einen Song auszuwählen und basierend auf der durch Machine-Learning-Modelle berechneten Ähnlichkeit zu diesem Song Vorschläge zu weiteren Werken erhalten.

Dazu wird im Theorieteil dieser Arbeit zunächst darauf eingegangen, wie Audiosignale, unter anderem auch in digitaler Form, dargestellt werden können. Zudem soll geklärt werden ob und wie die Ähnlichkeit von Musik festgestellt und bewertet werden kann. Auch die in der Arbeit verwendeten Algorithmen werden erläutert.

Im Praxisteil wurden zu Beginn verschiedene Klassifikator implementiert, die anhand verschiedener Features des Datensatzes eine Vorhersage des Genres vornehmen, um so herauszufinden welche Features sich am besten für den Vergleich von Songs eignen und einen Vergleichswert für die danach entworfenen Machine-Learning-Modelle zu bieten. Die genannten Machine-Learning-Modelle sollen die Songs nicht anhand von Metadaten, sondern anhand der Audiosignale bewerten, um die Modelle auch auf neue, nicht im Datensatz enthaltene Songs anwenden zu können. Dazu wurden zwei Autoencoder und ein Variational-Autoencoder entworfen und evaluiert. Im letzten Teil der Arbeit wird auf die technische Umsetzung und die Gestaltung der entwickelten Anwendung eingegangen, bevor zum Schluss ein Fazit gezogen wird.

## 2 Musiktheoretische Grundlagen

### 2.1 Ähnlichkeitsmerkmale von Musik

Es gibt viele Faktoren, die beeinflussen ob sich Musikstücke ähneln oder nicht. Der offensichtlichste Indikator für eine Ähnlichkeit ist dabei das Genre, dem ein Lied zugeordnet ist. Bei einem Genre handelt es sich per Definition um eine Kategorie in die Lieder eingeordnet werden, die durch spezifische musikalische Eigenschaften gekennzeichnet ist. Musikgenres sind somit eine oft verwendete Möglichkeit, Musik zu klassifizieren und zu organisieren, da sie es ermöglichen, ähnliche Musikstücke oder Künstler in Gruppen zusammenzufassen (vgl. [2]). Es existiert eine weite Bandbreite an verschiedenen Musikgenres, die auf bis zu 400 bis 500 verschiedene Genres geschätzt wird, die weltweit existieren. Viele davon sind allerdings sogenannte „Sub-Genres“ von den bekanntesten Musikgattungen wie Rock, Pop, Jazz, Heavy Metal etc. (vgl. [2]). Dies führt allerdings auch zu einem Problem: Lieder können mehr als ein Genre haben, und die Zuordnung zu einem Genre ist meist sehr subjektiv und sagt unter Umständen nichts über die Ähnlichkeit der Lieder aus, da beispielsweise das Genre Pop sehr breit gefächert ist. Deshalb muss ein anderes Maß gewählt werden, um Entscheidungen über die Ähnlichkeit von Liedern zu treffen.

Um im weiteren Verlauf also entscheiden zu können, was Musikstücke ähnlich macht oder sie unterscheidet, müssen zuerst verschiedene Bestandteile von Musikstücken erklärt werden, da diese maßgeblich für die vorliegenden Ähnlichkeiten verantwortlich sind. Dazu werden im Folgenden einige der wichtigsten Bestandteile von Musikstücken aufgeführt und erläutert: Tonart, Harmonik/Akkorde, Melodien, Rhythmus, Tempo, Taktart, Modulation, Dynamik und Instrumentierung.

Der erste wichtige Bestandteil eines Liedes, ist die Tonart. Eine Tonart ist jeweils nach ihrem sogenannten Grundton benannt, um welchen sich das Stück harmonisch aufbaut. Innerhalb dieser Tonart werden verschiedene Tonleitern und Skalen verwendet. Skalen sind dabei die Folge von Halbton und Ganztonschritten, die eine Tonleiter ausmachen, welche wiederum eine Visualisierung der Tonart darstellt (vgl. [3] S. 5). Da es nur 12 Tonarten gibt, sind diese also ein entscheidender Faktor für die Ähnlichkeit von Liedern. Lieder, welche in der



gleichen oder einer ähnlichen Tonart geschrieben sind, haben somit auch eine Tendenz zur Ähnlichkeit. Dieses Feature allein ist jedoch nicht für die Ähnlichkeit von Musik entscheidend.

Auch die sogenannte Harmonik eines Liedes ist ein wichtiger Indikator für die Ähnlichkeit von Liedern. Harmonien bestehen aus Akkorden, welche in der Musiktheorie das gleichzeitige Erklängen mehrerer Töne darstellen (vgl. [3] S. 8). In der traditionellen Harmonik versteht man dabei eine sogenannte Terzschichtung der Töne, also dass beispielsweise der erste, der dritte und der fünfte Ton einer Tonleiter gleichzeitig gespielt werden, je nach Genre kann dies allerdings variieren. Diese Akkorde und ihre Abfolge sind auch entscheidend, welche Stimmung das Musikstück beim Zuhörenden auslöst, da sie entscheiden, ob ein Stück eher melancholisch (Moll) oder fröhlich (Dur) klingt (vgl. [3] S. 8). Für jeden Akkord existieren ebenfalls andere Akkorde die harmonisch verwandt sind. Um dies verdeutlichen kann ein sogenannter Quintenzirkel verwendet werden, auf welchem verwandte Tonarten auch nah beieinander angezeigt werden. Je verwandter die Harmonien sind, desto ähnlicher sind sich also auch die Lieder, in welchen sie vorkommen (vgl. [3] S. 6).

Aus einer Abfolge von Tönen aus der Grundtonleiter zusammen mit den zugehörigen Akkorden, entstehen schließlich die verschiedenen Melodien aus denen Lieder zusammengesetzt sind. Diese sind oft der Hauptfokus der Zuhörer und somit zumindest bei einer subjektiven Bewertung maßgeblich entscheidend für die Ähnlichkeit von Liedern (vgl. [3] S. 9).

Wichtig bei der Ähnlichkeitsbestimmung von Musik sind jedoch nicht nur die Töne und Melodien selbst, sondern auch die Art wie sie gespielt werden: schnell, langsam, laut, leise oder mit Pausen. Dies wird durch den vorgegebenen Rhythmus, die Taktart und das Tempo definiert. Die Taktart ist dabei generell für das rhythmische Muster des Liedes verantwortlich. Diese wird je zu Beginn des Stückes definiert und kann sich während des Stückes ändern, was dann als Modulation bezeichnet wird (vgl. [4] S. 15). Es existieren viele verschiedene Taktarten, wie etwa beispielsweise ein 4/4 Takt,  $\frac{3}{4}$  Takt oder 2/3 Takt. Während der Takt auch Einfluss auf den Rhythmus hat, da er die grundlegende Struktur eines Musikstückes vorgibt, legt der Rhythmus die zeitliche Abfolge von Klängen und Pausen innerhalb dieses Rahmens fest. Ebenfalls eng verbunden damit ist das Tempo eines

Lieds, dass die Geschwindigkeit angibt, mit der das Lied gespielt wird (vgl. [4] S. 17). In der Musik existieren viele verschiedene Tempi, die je nach Genre auch variieren. Das alles zusammen ist ein wichtiger Indikator für die Ähnlichkeit von Musikstücken, da Lieder mit einer ähnlichen Metrik auch oft ähnlich klingen.

Nicht nur die Art wie schnell oder welche Töne gespielt werden ist entscheidend für die Ähnlichkeit von Liedern, sondern auch die Lautstärke der gespielten Töne, die sogenannte Dynamik, welche den Stil eines Lieds beeinflussen kann. Diese wird in der Musiktheorie oft verwendet, um innerhalb eines Liedes Schwerpunkte zu setzen und bestimmte Stellen hervorzuheben (vgl. [4] S. 20). Wenn eine ähnliche Dynamik bei Musikstücken vorliegt, kann dies somit ein Indiz für eine Ähnlichkeit sein.

Ein weiteres Indiz für eine Ähnlichkeit von Liedern ist schließlich die Instrumentierung, da Lieder, die auf denselben Instrumenten gespielt werden, auch eine gewisse Ähnlichkeit aufweisen können.

Während einer der genannten Faktoren jedoch noch keine Aussage über die Ähnlichkeit treffen kann, wird eine Ähnlichkeit immer wahrscheinlicher, je mehr der Faktoren auf beide verglichenen Lieder in ähnlicher Ausprägung zutreffen.

## 2.2 Darstellungsmöglichkeiten von Audiosignalen

Audiosignale sind jedoch nur schwer geeignet, um eine Ähnlichkeit zwischen Liedern vorherzusagen, da diese sich oft überlagern, weshalb andere Darstellungsformen gefunden werden mussten, um diese vergleichbar zu machen.

Eine Möglichkeit Audiosignale visuell darzustellen ist die sogenannte Notenschrift. Die Notenschrift besteht aus verschiedenen musikalischen Symbolen, sogenannten Noten, die spezifische Informationen über Tonhöhen, Rhythmen, Dynamik, Artikulation, Tempo und andere musikalische Elemente aus Kapitel 2.1 vermitteln (vgl. [5]).

Ebenfalls ist es möglich Audiodateien als Frequenzwellen darzustellen, die Schallwellen abbilden. Die Frequenz einer Schallwelle bestimmt ihre Tonhöhe, wobei höhere Frequenzen zu höheren Tönen und niedrigere Frequenzen zu tieferen Tönen führen (vgl. [6]). Diese Frequenzen kann man bildlich auf viele verschiedene Arten darstellen, wie durch

sogenannte Spektrogramme. Ein Spektrogramm ist dabei die visuelle Darstellung des zeitlichen Verlaufs der Zusammensetzung der einzelnen Frequenzen, die beispielsweise in einem Musikstück vorkommen. Dies geschieht mithilfe einer sogenannten Kurzzeit – Fourier – Transformation (vgl. [6]). Spektrogramme werden auch oft für die Analyse von Schallwellen und verschiedenen Frequenzen genutzt, da sie eine der wenigen Möglichkeiten sind, Audiosignale visuell abzubilden.

## 3 Verwendete Algorithmen

In diesem Kapitel werden die später verwendeten Algorithmen, Technologien und Evaluierungsmetriken beschrieben und erklärt.

### 3.1 Klassifikatoren

Um herauszufinden, welche Feature sich am besten zum Vergleich von Ähnlichkeiten verschiedener Songs eignen wurden verschiedene Klassifikator gebaut. Diese benutzen sowohl unterschiedliche Feature als auch unterschiedliche Technologien. Als Zielvariable wird das Top-Genre der Songs genutzt. Wie in Kapitel 2.1 beschrieben, sind Genre kein eindeutiger Indikator für die Ähnlichkeit von Songs. Die Ergebnisse der Klassifikatoren können aber als Richtwert dienen.

Bei zwei der verwendeten Klassifikatoren handelt es sich um XGBoost (eXtreme Gradient Boosting) [7] Ansätze. Hierbei wird mithilfe von Entscheidungsbäumen eine Klassifikation erstellt. Zunächst wird dafür ein Entscheidungsbaum aus einer Untermenge des Trainingsdatensatzes erstellt und die Klassen mithilfe dieses Baumes vorhergesagt. Anschließend werden die Residuen zwischen vorhergesagten und tatsächlichen Klassen berechnet. Im nächsten Schritt wird ein neuer Entscheidungsbaum trainiert, dessen Zielvariable jedoch die Vorhersagen des vorherigen Entscheidungsbaums sind. Die beiden Bäume werden nun mit einer Lernrate zu einem Entscheidungsbaum zusammengefügt. Dieser Prozess wird iterativ wiederholt, um progressiv die Fehler des vorhandenen Entscheidungsbaum auszubessern und somit einen robusten Klassifikator zu erstellen. (vgl. [8])

Ein dritter Klassifikator verwendet convolutional neural Networks (CNN), um anhand von Bildeingaben eine Klassifikation zu generieren. Der Unterschied von CNN zu herkömmlichen neuronalen Netzen ist die Verwendung von Convolution- und Pooling-Layern. Das Convolution-Layer bildet ein Skalarprodukt Ausschnitten des Eingabebilds und einer Matrix aus trainierbaren Parametern (Kernel), welche über das Eingabebild geschoben wird. So können die räumlichen Eigenschaften des Bildes beibehalten werden, während dennoch eine Dimensionsreduktion vorgenommen werden kann. Durch Pooling-

Layer können weitere Reduktionen ausgeführt werden, indem aus dem Output des Convolution-Layer eine zusammenfassende Statistik wie beispielsweise der Durchschnitt gebildet wird. (vgl. [9])

## 3.2 Vektordatenbanken und Kosinus-Ähnlichkeit

Um die Ähnlichkeit zweier Lieder letztendlich bestimmen zu können, wurden sich bei Audio Similarity für die Verwendung einer Vektordatenbank entschieden.

Vektordatenbanken speichern die eingegebenen Daten als Vektoreinbettungen. Das bedeutet, dass die Daten in numerische Vektoren übersetzt werden, die über die wesentlich wichtigen Eigenschaften der Daten verfügen. Jede Dimension des Vektors stellt dabei ein Merkmal der Daten dar. Der Vorteil dabei ist, dass eine Vektordatenbank ohne große Latenzzeiten Datenelemente abfragen kann, die einem bestimmten abgefragten Element ähnlich sind, da auch die Vektoreinbettungen dieser Elemente eine Ähnlichkeit haben (vgl. [10]).

Um die Ähnlichkeit zweier Vektoren in einem multidimensionalen Raum zu bestimmen, kann die Kosinus-Ähnlichkeit herangezogen werden. Die Kosinus - Ähnlichkeit bestimmt dabei den Kosinus des Winkels zwischen zwei Vektoren:

$$\text{Cosine similarity}(|X, Y|) = \frac{x \cdot y}{\|x\| \|y\|}$$

Das Ergebnis der Berechnung liegt zwischen -1 und 1. Hierbei impliziert 1 eine vollständige Ähnlichkeit und gleiche Richtung der Vektoren. Wenn die Kosinus-Ähnlichkeit -1 beträgt zeigen die beiden Vektoren genau in die entgegengesetzte Richtung. Wenn das Ergebnis 0 beträgt, stehen die Vektoren orthogonal zueinander (vgl. [11]).

## 3.3 Dimensionsreduktionsalgorithmen

Hochdimensionale Daten, wie Audio-, Bild-, oder Videodaten, stellen eine Herausforderung dar, wenn diese einer Datenbank gespeichert effizient verglichen werden sollen. Außerdem haben bei diesen Daten eine Vielzahl an Datenpunkten nur eine geringe Aussagekraft. Deswegen können Dimensionsreduktionsalgorithmen verwendet werden, um die Daten in

eine niedrigere Dimension zu konvertieren und die Aussagekraft einzelner Datenpunkte zu maximieren.

Wie bereits in Kapitel 2.2 erwähnt, sind Spektrogramme als Bilddaten eine beliebte Form Audiodateien darzustellen. Für Bilddaten eignen sich zur Dimensionsreduktion unter anderem der Autoencoder respektive dessen Weiterentwicklung des Variational Autoencoders. Außerdem können Convolutional Neural Networks (CNN) auf einer Zielaufgabe trainiert und im Anschluss die Ergebnisse einzelner Schichten als Approximation der Eingabe verwendet werden. Auch für eine Recommendation-Engine, bei der die Audiodaten verarbeitet in einer Datenbank abgefragt werden, um ähnliche Musikstücke zu identifizieren und vorschlagen zu können, ist eine Dimensionsreduktion essenziell. Deswegen werden im Folgenden die drei genannten Dimensionsreduktionsalgorithmen erläutert.

### 3.3.1 Autoencoder

Der Autoencoder stellt eine Konvertierungsmöglichkeit hochdimensionaler Daten in niedrigere Dimensionen mithilfe eines neuronalen Netzes dar. Diese besteht aus dem Encoder, der die Eingabedaten über Hidden-Layers in eine geringe Dimension überführt, einem Bottleneck, das die Stelle mit der geringsten Dimensionalität aufweist, und einem Decoder, der meist den Encoder spiegelt und für die Rekonstruktion der ursprünglichen Bilder aus der niedrigdimensionalen Latent-Space-Repräsentation zuständig ist (vgl. Abbildung 1).

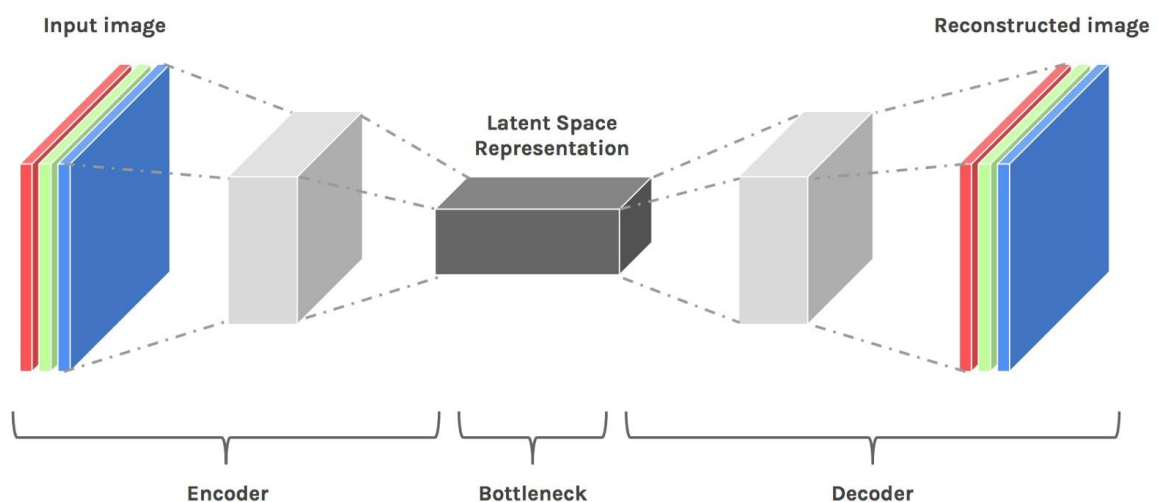


Abbildung 1: Visualisierung eines Convolutional Autoencoders – Quelle: [12]

Das Training erfolgt durch die Minimierung der Unterschiede der rekonstruierten Daten im Vergleich zu den Originaldaten, was der sogenannten Rekonstruktion-Loss angibt. Es werden also die Eingabedaten als Label verwendet, weswegen der Trainingsansatz auch als self-supervised Training bezeichnet wird.

Da das direkte Trainieren aller Schichten ineffizient sein kann und zu einer nur unzureichenden Rekonstruktion der Eingabedaten führen kann, wurde das Schichtenweise trainieren des Autoencoders durch Hinton et. al. eingeführt (vgl. [12]). Hierbei wird der Autoencoder vortrainiert, indem Stück für Stück weitere Layer hinzukommen. Im Anschluss können diese vortrainierten Gewichte verwendet werden, um als gute initiale Gewichte für den Autoencoder zu dienen. Im Anschluss daran findet das Fine-Tuning dieses vortrainierten Modells statt. Hierdurch kann sowohl eine deutlich bessere Performance als auch eine kürzere Trainingszeit erreicht werden.

Der Autoencoder stellt also eine Möglichkeit da, Daten in eine niedrigdimensionale, und im Gegensatz zur Principal Component Analysis nicht-lineare, Repräsentation zu bringen mithilfe von neuronalen Netzen (vgl. [12]).

### 3.3.2 Variational-Autoencoder

Der Variational-Autoencoder besteht ebenfalls aus Encoder und Decoder und kann zur Dimensionsreduktion oder Generierung von Daten verwendet werden. Hierbei transformiert der Encoder die Eingabedaten nicht nur in den latenten Raum, sondern gibt auch eine Wahrscheinlichkeitsverteilung darüber aus.

Der Encoder lernt also nicht nur einen einfachen Vektor, sondern ein Mittelwerts- und ein Standardabweichungsvektor der Verteilung. Aus diesen Vektoren wählt der „Sampled latent Vektor“ zufällige Werte aus, die dem Decoder übergeben werden (vgl. Abbildung 2).

Da über den Sampling Latent Vektor keine Backpropagation durchgeführt werden kann, wird der sogenannte Reparametrisierungstrick angewendet. Hierbei wird die zugrundeliegende Verteilung immer fest als Standardnormalverteilung, mit einem Mittelwert von null und einer Standardabweichung von eins, kodiert und nur der Mittelwerts- und Standardabweichungsvektor erlernt.

Damit die erlernte Verteilung nicht zu stark von der Normalverteilung abweicht, wird neben dem im Encoder verwendeten Reconstruction-Loss auch die Kullback-Leibler Divergenz angewendet, die ein Maß für die Ähnlichkeit zweier Wahrscheinlichkeitsverteilungen darstellt (vgl. [13]). Diese vergleicht die Normalverteilung mit dem latenten Raum und sorgt dafür, dass die erlernte Verteilung nicht zu weit von der Normalverteilung abweicht, wodurch ein gut strukturierter latenter Raum erlernt wird.

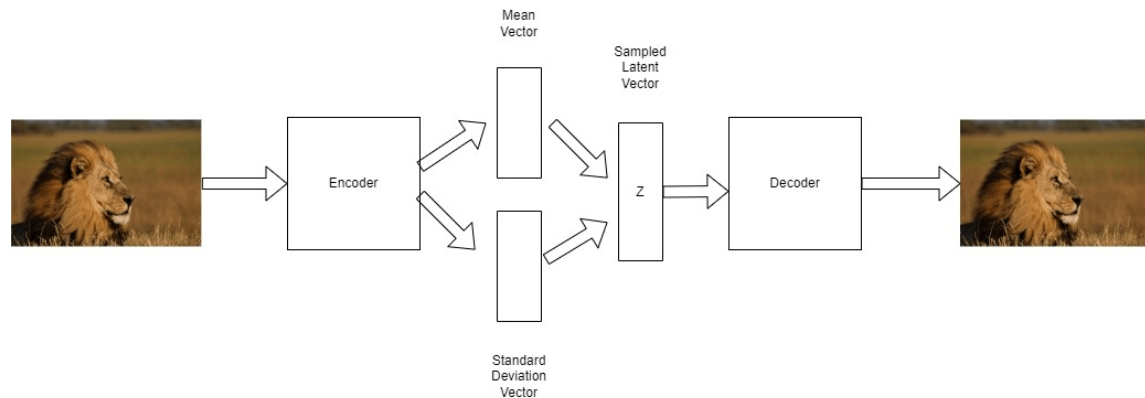


Abbildung 2: Variational Auto Encoder – Quelle: [14]

Der Variational-Autoencoder ist also eine Weiterentwicklung des Autoencoders, wobei neben der Rekonstruktion auch eine Wahrscheinlichkeitsverteilung über den latenten Raum erlernt wird. Dies führt dazu, dass vor allem zur Generierung von neuen Daten die VAEs damit den Autoencodern überlegen sind. Zum Beispiel ermöglichen sie Morphing, also den nahtlosen Übergang von einem Bild in ein anderes (vgl. [15]).

### 3.3.3 Convolutional Neural Network

Convolutional Neural Networks (CNNs) werden hauptsächlich zur Klassifizierung von Bilddaten verwendet. Unter bestimmten Bedingungen können diese auch zur Dimensionsreduktion verwendet werden, wie beispielsweise in [16] gezeigt. Wie das für Audio-Similarity funktioniert, wird im Folgenden vorgestellt.

Fathollahi et al. Trainieren hierfür das CNN zu Beginn auf einer Zielaufgabe. Im Fall von Audio-Similarity könnte dies zum Beispiel die Genre-Klassifizierung auf Basis von Spektrogrammen sein. Im Anschluss können verschiedenen Ausgabeschichten des Netzes verwendet werden, um die Eingabedaten in einer niedrigeren Dimension darzustellen, ohne jedoch den direkten Output des Modells zu verwenden. Dies ist beispielsweise bei



der Ähnlichkeit von Musik sinnvoll, da hier das Genre nur eine grobe Gliederung bietet, während die Dimensionalität der Spektrogramme zu groß ist, um diese performant für eine Ähnlichkeitssuche in einer Datenbank zu verwenden. Die verschiedenen Ausgabeschichten des Netzwerkes bieten hierbei einen Kompromiss. Während die unteren Schichten dabei meist eine niedrigere Dimension haben, wie beispielsweise Output 4 in der Abbildung 3 mit einer Dimension von 100, sind höhere Schichten näher am Original, haben jedoch auch mehr Datenpunkte.

Mithilfe eines kNN-Klassifizierer konnten die Ausgaben der unterschiedlichen Schichten evaluiert werden. Hierbei werden mit den Ausgabevektoren der einzelnen Schicht und den Genre-Labels des ursprünglichen Datensatzes ein kNN-Klassifizierer mit  $k=1$  trainiert und im Anschluss evaluiert. Die beste Genauigkeit erzielte dabei das Dense-100-Layer (vgl. Abb. 3 Output 4), das bis zu 87,92 % Accuracy erreichte. Das Ergebnis ist sogar noch leicht besser als die Klassifizierungsgenauigkeit des ursprünglichen Modells, das 86,5 % erzielte [16].

Es besteht also auch die Möglichkeit CNNs auf einer Aufgabe zu trainieren, die ähnlich zu der Zielaufgabe des dimensionsreduzierten Datensatzes ist, und unterschiedliche Ausgabeschichten des Netzwerkes zu verwenden, um die Daten auf eine bestimmte Dimension reduziert zu erhalten. Die Effektivität dieses Ansatzes wurde beispielsweise in [16] erläutert.

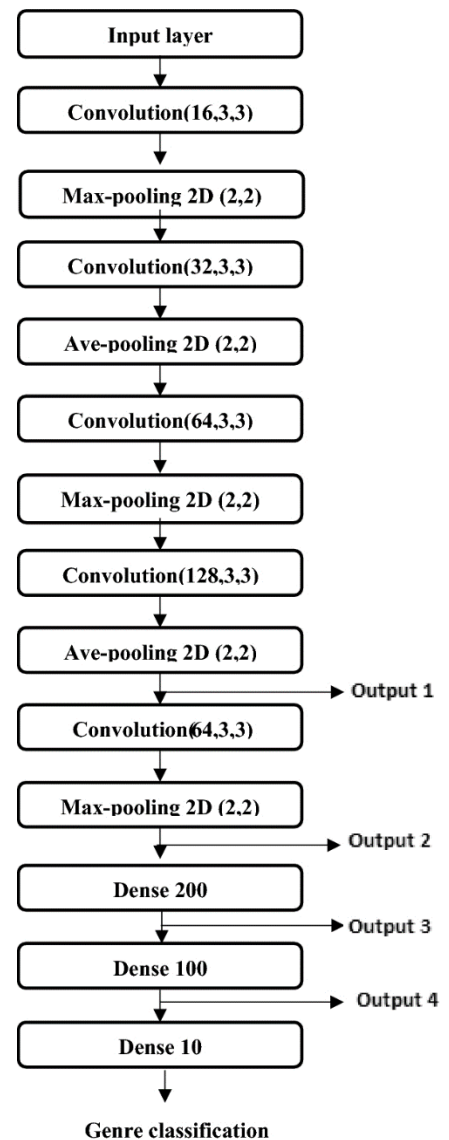


Abbildung 3: Darstellung der CNN-Architektur zur Dimensionsreduktion - Quelle: [16]

## 3.4 Evaluierungsmetriken

### 3.4.1 PCA

Um die Performance der Endmodelle vergleichen zu können, wurden verschiedene Tools und Methoden ausgewählt, um die Modelle vergleichen zu können.

Eine dieser Methoden war eine Principal Component Analyse (PCA), welche verwendet wird, um Lineare Dimensionsreduktionen durchzuführen. Dies geschieht durch die Verwendung von Singular Value Decomposition der Daten, um sie auf eine niedrigere Dimension projizieren zu können. Die Daten werden dabei lediglich zentriert, bevor die Singular Value Decomposition verwendet wird. Um bei einer PCA festzulegen, auf wie viele Dimensionen der Input reduziert werden soll, wird der Parameter „n\_components“ verwendet (vgl. [17]).

### 3.4.2 TSNE

Die nächste Methode, welche ebenfalls zur Dimensionsreduktion verwendet werden kann, ist ein sogenanntes TSNE, was für „T-distributed Stochastic Neighbor Embedding“ steht. TSNE ist ein Tool, welches oft verwendet wird, um hochdimensionale Daten darzustellen, was es gut geeignet macht für Musikdaten. Der Grund dafür ist die Funktionsweise von TSNE, welcher Ähnlichkeiten zwischen Datenpunkten zu gemeinsamen Wahrscheinlichkeiten konvertiert und versucht die Kullback Leibler Divergenz zwischen diesen Punkten und den ursprünglichen hochdimensionalen Daten (vgl. [18]).

Dabei können die Dimensionen, auf die reduziert werden soll, ebenso wie bei der PCA festgelegt werden, sowie die Perplexität, welche aussagt, wie viele Nachbarn eines Datenpunktes berücksichtigt werden. Die von sklearn vorgeschlagene Perplexität liegt dabei bei einer Zahl zwischen 5 und 50 (vgl. [18]).

### 3.4.3 K-Nearest-Neighbours Klassifikation

Als statistische Auswertungsmethode wurde sich schließlich für die Durchführung einer k-Nearest-Neighbours Klassifikation der Modelle entschieden. „Der K-Nächste-Nachbarn-Algorithmus, auch KNN oder k-NN genannt, ist ein nichtparametrischer, überwachter Lernklassifikator, der das Konzept der Nähe nutzt, um Klassifizierungen oder Vorhersagen über die Gruppierung eines einzelnen Datenpunktes zu treffen“ [19]. Dabei geht der Algorithmus davon aus, dass ähnliche Datenpunkte auch in der Nähe voneinander gefunden werden können. Das Ziel des kNN-Algorithmus ist es also, die nächsten Nachbarn eines Punktes ermitteln zu können, um ihnen eine Klasse zuzuweisen. Das Ergebnis zwischen den vorhergesagten Werten und den wahren Labeln wird dabei in Form einer Accuracy zurückgegeben (vgl. [19]).

## 4 Verwendeter Datensatz

Für die Umsetzung und Generalisierung des Projekts war die Verarbeitung von reinen Audiosignalen wichtig, weshalb ein Datensatz mit vorhandenen Audiodateien von Musik gefordert war. Der mit Abstand größte Datensatz, der diese Eigenschaft erfüllt ist der Datensatz des Free Music Archives (FMA) (vgl. [20]). Der FMA-Datensatz umfasst 106.574 Songs von 16.341 verschiedenen Künstlern und erreicht dabei eine Größe von 917 Gibibyte (vgl. [20]).

Neben den reinen Audiodateien der Songs stellt der Datensatz auch umfangreiche Metadaten zur Verfügung. So sind teilweise detaillierte Informationen über Song, Album und Künstler vorhanden. Aber auch die Zuordnung der Songs in verschiedene Genre wurde vorgenommen. Zudem gibt es bereits extrahierte und in statistische Werte zusammengefasste Features. Für 13.129 Songs sind außerdem Echonest-Daten vorhanden. Diese sind High-Level Features von Songs, welche von Echonest (heute Spotify) zur Verfügung gestellt werden und auch von der Spotify-API abrufbar sind. Sowohl die extrahierten Features als auch die Echonest-Features wurden zur Genre-Klassifikation verwendet.

Da es sich bei dem Datensatz um lizenzfreie Songs handelt, sind hauptsächlich unbekannte, von kleinen Künstlern gepflegte Werke enthalten. Für den Use-Case der Recommendation-Engine ist das nicht optimal, da Benutzer vermutlich auch bekannte Songs vorgeschlagen bekommen möchten. Würde das Projekt jedoch von einem Streaming-Anbieter wie Spotify verwendet werden, wäre der Zugriff auf eine viel umfangreichere Musik-Datenbank möglich. Die Funktionalität ist durch die Unbekanntheit der vorhandenen Songs nicht beeinträchtigt.

Der Datensatz ist zudem in 4 Varianten aufgeteilt. Neben dem vollständigen Datensatz gibt es eine große, mittlere und kleine Version (vgl. [20]). Für das Projekt wurde zunächst der kleine Datensatz verwendet, in dem 8.000 Songs, welche auf 30 Sekunden gekürzt wurden, verfügbar sind. Sie verteilen sich auf 8 balancierte Genres. Sollte das Projekt weiterverwendet werden, kann ein größerer Datensatz ausgewählt werden.

# 5 Machine-Learning-Modelle und deren Auswertung

Um die Zielaufgabe des Recommendation-Systems für Audiodaten zu erfüllen, erfolgte zuerst eine Vorauswahl, welche Datensätze respektive Features sich besonders für den Anwendungsfall eignen. Da das Genre die häufigste Methode ist, um Musikdaten nach Ähnlichkeit zu klassifizieren, wurde dies als zugrundeliegende Aufgabe verwendet, um den passendsten Datensatz auszuwählen. Dies wird in Kapitel 5.1 beschrieben.

Im Anschluss daran wurden verschiedene Dimensionsreduktionsalgorithmen trainiert, um die Daten performant in einer Vektordatenbank darstellen zu können. Die Ergebnisse davon sind in Kapitel 5.2 zusammengefasst.

## 5.1 Genreklassifizierung zur Featureauswahl

Zur Auswahl des Datensatzes zur Audio-Similarity wurden verschiedene Datensätze als Grundlage auf der ähnlichen Aufgabe Genre-Klassifizierung getestet. Diese und die Ergebnisse werden im Folgenden kurz vorgestellt.

Zum einen wurde ein XGBoost-Klassifikator sowohl auf den von der Spotify-API zur Verfügung stehenden Features als auch auf vom FMA-Datensatz gegebene Features klassifiziert. Dabei wurden die Hyperparameter mithilfe von RandomSearchCV optimiert. Die verschiedenen Daten zur Genre-Klassifikation zeigten alle ähnlichen Ergebnisse mit einer Accuracy von ca. 0,5. Bei den Features der Spotify-Daten wurde zusätzlich die Feature-Importance bestimmt, um Erkenntnisse über die Wichtigkeit dieser zu gewinnen. Dabei fiel auf, dass einige Features zwar wichtiger als andere zur Klassifikation sind, jedoch keine unwichtig erschienen.

Zum anderen wurde basierend auf Spektrogrammen aus den Audio-Rohdaten ein CNN-Klassifikator entwickelt. Hierfür wurden sowohl die Spektrogramme der 30-Sekunden Daten verwendet als auch Spektrogramme von 3-Sekunden Bruchstücken, wobei sich die einzelnen drei Sekunden Abschnitte jeweils um 1,5 Sekunden überschneiden, wie es auch in [20] geschieht. Für den ersten CNN-Klassifikator konnte hierbei eine Genauigkeit von ebenfalls circa 50 % erreicht werden. Das zweite Modell mit den drei Sekunden-Daten weist

eine Genauigkeit von 38,7% auf. Durch die Verwendung weiterer Spektrogramme neben dem Melspectrogramm (Chroma STFT und Spectral Contrast) kann die Genauigkeit auf 46,8% gesteigert werden. Dies ist in diesem Fall deutlich geringer als die Validation-Accuracy der letzten Epoche, die 80,1% beträgt.

Die trainierten Klassifikatoren zeigen keinen großen Unterschied hinsichtlich der Genauigkeit der Genre-Klassifizierung. Die 30-Sekunden Spektrogramme scheinen ein guter Indikator für das Genre zu sein, weswegen mit diesen zuerst weitere Dimensionsreduktionsalgorithmen trainiert werden. Auch die hohe Validation-Accuracy der 3-Sekunden Spektrogramme deutet an, dass diese repräsentativ für ein Musikstück sein können.

## 5.2 Dimensionsreduktionsmodelle und deren Evaluierung

Für die Dimensionsreduktion wurden zwei Convolutional-Variational-Autoencoder trainiert, die sich nur in Bezug auf die Trainingsfunktion leicht unterscheiden. Diese wurden mit einem Convolutional-Autoencoder verglichen. Im ersten Schritt wurden die Modelle auf den 30-Sekunden-Spektrogrammen trainiert. Da dies nur zu unzureichenden Ergebnissen führte, wurde das nach der ersten Evaluierung beste Modell, der Autoencoder, mit den 3-Sekunden-Melspektrogrammen erneut trainiert. Als Referenzmodell dient ebenfalls der letzte-Hidden-Layer des CNN-Genre-Klassifizierer, das wie im Grundlagenteil erläutert in ähnlichen Versuchen die besten Ergebnisse lieferte. Für die ersten drei Modelle findet zunächst eine visuelle Auswertung statt, bevor alle Modelle mithilfe eines kNN-Genre-Klassifizierer der encodierten Daten miteinander verglichen werden. Die Ergebnisse dieser Evaluierungen werden im folgenden Kapitel dargestellt.

Der erste Ansatz zur Auswahl des besten Modells war eine graphische Auswertung. Dazu wurden zuerst die Original-Spektrogramme mit den verschiedenen Modellen encodiert und wieder decodiert, bevor die dabei entstandenen Rekonstruktionen neben dem Original visualisiert wurden, um sie nebeneinander vergleichen zu können (siehe Abbildung 4).

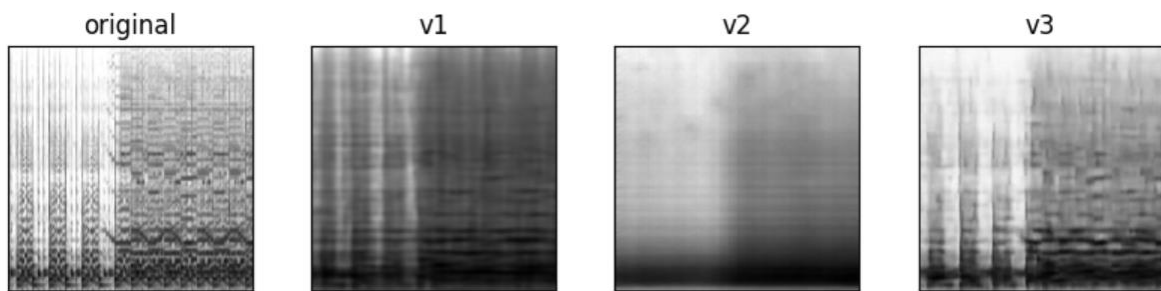


Abbildung 4: Vergleich der rekonstruierten Spektrogramme mit dem Original.

Dabei fiel bereits auf, dass rein visuell das dritte Modell am besten performt, da hier die ausgegebenen Bilder, die meiste Ähnlichkeit zum Original hatten. Das zweite Modell, der VAE, wiederum zeigte nur Durchschnittswerte.

Da diese Ausgabe jedoch nicht reichte, um eine fundierte Entscheidung treffen zu können, wurden die zurückgegebenen Werte der Modelle einer PCA unterzogen, da eine Dimensionsreduktion bei der Evaluierung der Modelle helfen könnte. Die 100 Dimensionen, die nach dem Training der verschiedenen Modelle noch übrig waren, wurden dabei auf 2 Dimensionen reduziert, welche schließlich mithilfe von Matplotlib visualisiert wurden (siehe Abbildung 5).

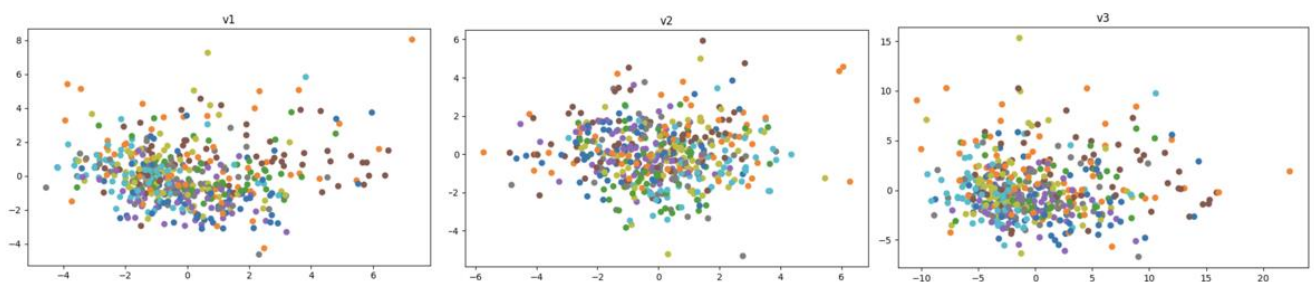


Abbildung 5: Modellvergleich durch PCA

Das Ergebnis der Principal Component Analyse war jedoch für die Bewertung der Modelle nicht zufriedenstellend, da bei keinem Modell die Streuung so gering war, dass man daraus graphisch etwas erkennen konnte. Es wurde lediglich entdeckt, dass die Streuung mancher Labels bei Modell drei geringer war als bei den anderen.

Im Anschluss daran wurde ein TSNE angewendet, um eine weitere Methode zur Dimensionsreduktion zu testen. Dazu wurde aufgrund einer Empfehlung von sklearn (vgl. [18]) zuerst eine PCA getätigt, um die Dimensionen auf 50 zu reduzieren, bevor schließlich TSNE angewendet wurde. Dazu wurde die Anzahl der Zieldimensionen auf 2 gesetzt und die Perplexität, die einen sehr hohen Einfluss auf TSNE hat, auf 20 gesetzt.

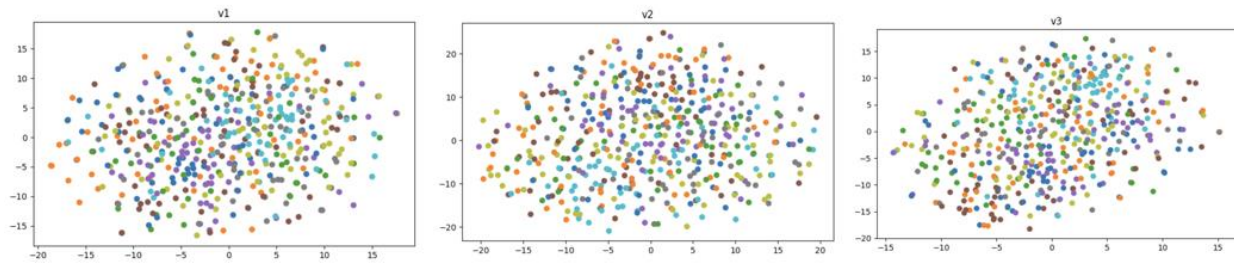


Abbildung 6: Modellvergleich mit TSNE

Wie in Abbildung 6 zu sehen ist, war das Ergebnis des TSNE jedoch genauso wenig aussagekräftig wie das Ergebnis der PCA, da abermals eine sehr große Streuung der Datenpunkte vorlag.

Da die graphischen Ansätze also keine zufriedenstellenden Ergebnisse lieferten, wurde eine statistische Methode angewendet, um herauszufinden, welches Modell am besten funktioniert: Ein kNN-Klassifikator, welcher genutzt wurde, um lediglich den Encoder zu bewerten.

Dieser wurde auf den dimensionsreduzierten Daten trainiert. Für diese Auswertung wurde auch das Convolutional Neural Network berücksichtigt. Für die Evaluierung wurden schließlich die vom kNN für  $k=1$  vorhergesagten Genres mit den tatsächlichen Genres verglichen.

CVAE (Modell 1) mit 30 Sekunden Audio-Daten	CVAE (Modell 2) mit 30 Sekunden Audio-Daten	CAE (Modell 3) mit 30 Sekunden Audio-Daten	CAE (Modell 3) mit 3 Sekunden Audio-Daten	CNN <u>letztes</u> <u>Hidden-Layer</u> mit 3 Sekunden Audio-Daten
25,8 %	21,6 %	26,8 %	41,5 %	12,4 %

Abbildung 7: Ergebnis des Vergleichs der verschiedenen Modelle mithilfe von kNN

Das beste Modell war dabei das Modell, dass die höchste Accuracy nach dieser Evaluierung hatte. Das war Modell Nummer drei, welches auch schon laut den graphischen Auswertungen leicht besser zu performen scheint als die anderen Modelle. Modell 1 schnitt dabei am zweitbesten ab, Modell 2 am drittbesten und das Convolutional Neural Network am schlechtesten.

Deshalb wurde der Autoencoder aus Modell Nummer 3 als finales Modell ausgewählt. Bei einem weiteren Versuch die Accuracy von Modell 3 zu verbessern, wurde dieses schließlich mit 3-Sekunden Audio Abschnitten anstatt 30 Sekunden Audio Abschnitten neu trainiert, wodurch sich die Accuracy des Modells nochmal maßgeblich verbesserte auf einen Wert von 41,5%.

# 6 Applikation

Um das ausgewählte Modell sinnvoll verwenden und den Use-Case des Projektes umsetzen zu können, wurde eine Applikation gebaut. Der Benutzer soll die Möglichkeit haben eine eigene Audiodatei hochzuladen oder einen vorhandenen Song aus der Datenbank auszuwählen, um basierend auf seiner Auswahl Empfehlungen von ähnlichen Songs zu erhalten. In diesem Kapitel wird die technische sowie visuelle Umsetzung der Applikation beschrieben.

## 6.1 Technische Umsetzung

Die Anwendung besteht aus zwei Komponenten. Zum einen die Webanwendung, welche dem Nutzer zur Verfügung steht und die visuelle Oberfläche sowie Funktionalität definiert. Zum anderen eine Redis-Datenbank, welche Metadaten des FMA-Datensatzes sowie die durch den Autoencoder berechneten Feature-Vektoren der Songs enthält. Die Redis-Datenbank ermöglicht sehr schnelle Berechnungen der Kosinus-Ähnlichkeiten, um so mit möglichst kurzen Ladezeiten dem Nutzer Empfehlungen zur Verfügung zu stellen.

Beide Komponenten werden durch docker-compose gestartet. Für die Webanwendung wurde zudem ein individuelles Dockerfile geschrieben. Die Redis-Datenbank wird in einem Container gestartet und anschließend von der Anwendung mit einem Skript („database\_init.py“) mit den benötigten Daten befüllt. Sie ist innerhalb des Clusters auf dem Port 6379 erreichbar. Dies stellt die Schnittstelle zwischen Webanwendung und Datenbasis dar.

Die Webanwendung wurde mithilfe des Python-Moduls Streamlit [21] erstellt. Streamlit ermöglicht das schnelle Erstellen von Websites und wird dabei vollständig in Python-Code definiert. Es gibt einige vordefinierte Elemente wie Textbausteine oder Inputfelder, aber auch sämtliche in Python definierte Funktionen können in die Applikation mit eingebaut werden. Die Applikation wird dabei in einem Python-Skript (hier „app.py“) definiert und anschließend mit dem Befehl „streamlit run app.py“ gestartet. Sie ist anschließend lokal auf dem Port 8501 zu erreichen.



Streamlit unterstützt zudem mehrseitige Applikationen. Dazu wird im Verzeichnis des Hauptskripts lediglich ein Verzeichnis mit dem Namen „pages“ benötigt, in welchem jedes weitere Skript eine Seite definiert. Standardmäßig werden die Seiten von Streamlit in einer Sidebar navigierbar gemacht. Für den vorliegenden Anwendungsfall ist es jedoch sinnvoller die Navigation lediglich über Buttons zu ermöglichen, welche erscheinen, nachdem der Benutzer einen Song ausgewählt hat. Daher wird die Sidebar durch einen Workaround mit CSS, welches in einem Markdown-Element von Streamlit definiert werden kann, versteckt. Für die Navigation über Buttons wird anschließend das Modul streamlit-extras [22] benötigt.

Die vorliegende Webanwendung besteht aus zwei Seiten. Auf der Startseite kann der Benutzer entweder eine eigene Audiodatei hochladen oder einen Song aus der Datenbank suchen. Der Upload einer Datei wird durch ein dafür vorgesehenes Streamlit-Element ermöglicht. Für die Suche in der Datenbank steht ein Suchfeld zur Verfügung, welches während der Suche vorhandene Songs anzeigt.

Nach Upload oder Auswahl eines Songs sieht der Nutzer nach einem Button-Klick die Ergebnisse auf einer dedizierten Seite. Sobald die Ergebnis-Seite aufgerufen wird, wird bei einer hochgeladenen Audiodatei zunächst die Feature-Extraktion mit dem Autoencoder durchgeführt. Bei einem ausgewählten Song aus der Datenbank wird der Feature-Vektor entsprechend aus der Datenbank abgerufen. Der erhaltene Vektor wird anschließend an eine Funktion übergeben, die die Redis-Datenbank abfragt. Diese berechnet die Kosinus-Ähnlichkeiten mit den restlichen Songs und gibt die ähnlichsten Songs zurück. Diese können nun für den Nutzer visualisiert werden. Standardmäßig werden ihm die 5 ähnlichsten Songs angezeigt. Er hat jedoch die Möglichkeit bis zu 50 Songs anzeigen zu lassen.

Da die API des Free Music Archives nicht mehr zur Verfügung steht (vgl. [23]), ist es ohne Abspeichern der Audiodateien nicht möglich die Songs in der Anwendung abzuspielen. Da diese sehr viel Speicherplatz (bei vollem Umfang 917 GiB (vgl. [20])) benötigen wurde darauf verzichtet. Stattdessen wird bei jedem Song eine Anfrage an die Spotify-API gestellt, um auf Spotify nach dem Song zu suchen. Ist dieser dort vorhanden wird der Spotify-Player mit einem Inlineframe in die Anwendung mit eingebunden. Allerdings lassen sich nur circa 10 Prozent der Songs auf Spotify finden.

Alle benötigten Funktionen – zur Feature-Extraktion, Datenbankabfrage und API-Requests – werden zur Übersichtlichkeit in einer separaten Datei („utils.py“) definiert und in den Skripten der Anwendung importiert.

## 6.2 Frontend

Das mit Streamlit entwickelte Frontend besteht wie erwähnt aus zwei Komponenten: Der Songauswahl und der Ausgabe der Ergebnisse. Das grundlegende Design der Webapplikation ist dabei schlicht gehalten, was auf die begrenzten Möglichkeiten von Streamlit zurückzuführen ist. Wichtige Features wie etwa der Titel von Songs oder wichtigen Details wurde mit einer lila Schrift hervorgehoben, während der Rest der Anwendung in schwarz-weiß gehalten ist.

Für die Songauswahl auf der Startseite kann zwischen zwei verschiedenen Varianten ausgewählt werden, indem ein entsprechende Tab ausgewählt wird.

In der ersten Variante kann der Nutzer selbst ein Lied in die Applikation laden, indem er entweder die Datei per ‚Drag and Drop‘ einfügt oder sein Gerät nach der gewünschten Audiodatei durchsucht:

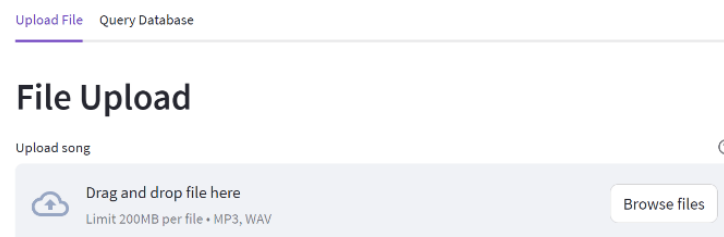


Abbildung 8: Dateiupload.

Bei Variante zwei hingegen kann der Nutzer den gewünschten Titel in einem Suchfeld angeben, woraufhin die in der Applikation verbaute Datenbank nach dem gewünschten Titel durchsucht wird. Während der Eingabe öffnet sich dabei ein Dropdown-Menü, in welchem Vorschläge anhand des eingegebenen Textes gemacht werden, um die Suche für den Nutzer zu erleichtern:

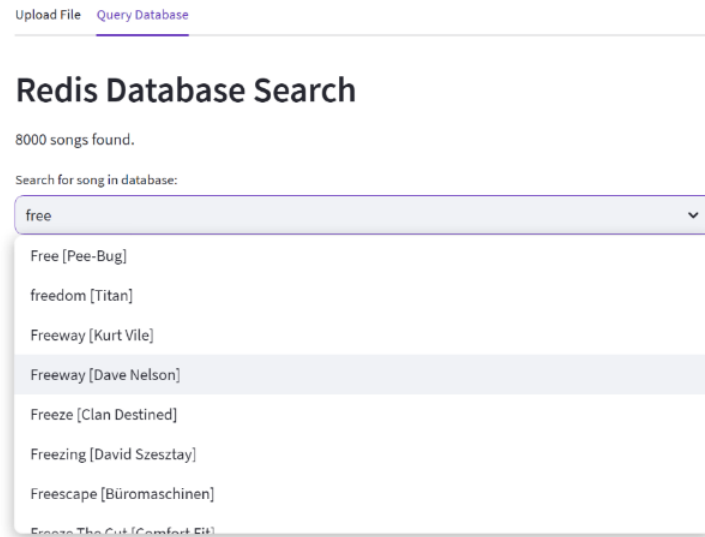


Abbildung 9: Songsuche mithilfe der Redis-Datenbank.

Nachdem der gewünschte Song ausgewählt wurde, kann der Nutzer auf einen Button drücken, der den Vergleich mit Songs aus der Datenbank startet, während der ausgewählte Song auf dem Bildschirm zu sehen ist und ein Ladescreen erscheint:

## Audio Similarity Results

### Your Song

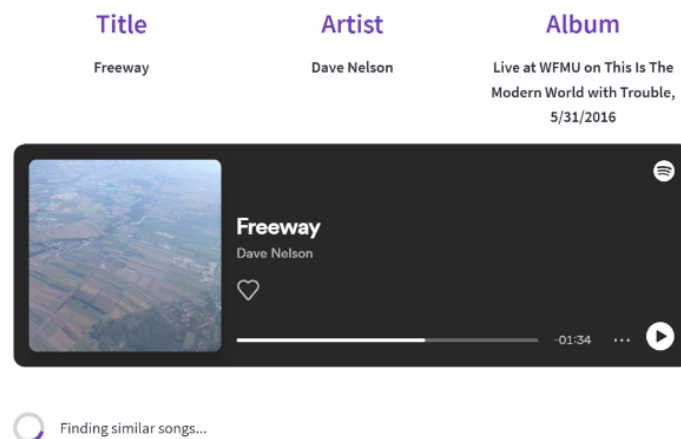


Abbildung 10: Anzeige des ausgewählten Songs.

Sobald ein Ergebnis bereitsteht, kann der Nutzer auswählen wie viele der vorgeschlagenen Lieder er angezeigt bekommen möchte. Dies geschieht mithilfe einer Leiste, bei der der Nutzer einen Wert zwischen 1 und 50 Liedern wählen kann.

Im Anschluss daran werden die von der Anwendung vorgeschlagenen Titel angezeigt. Diese Anzeige beinhaltet die errechnete Ähnlichkeit zwischen dem vorgeschlagenen Song und

dem vom Nutzer ausgewählten Song. Zusätzlich werden noch wichtige Informationen über das vorgeschlagene Lied, wie etwa der Interpret, das Album oder das Hauptgenre, angezeigt. Sofern das Lied über die Spotify-API zu finden ist, wird unterhalb der eben beschriebenen Informationen ein Fenster eingeblendet, über welches das vorgeschlagene Lied direkt abgespielt werden kann.

### Recommended Songs

Number of results to display:

5

150

1

50

1) Niveau Supérieur

Similarity ⓘ

17.62%

Artist

Chase Alan Willis

Top Genre

Song Duration

201


Album

The Kingdom of Back

Album Tracks

19

Instrumental



Niveau Supérieur

Chase Alan Willis

03:21

⋮

▶

Abbildung 11 Ergebnis der Suche nach ähnlichen Liedern.

## 7 Fazit und Ausblick

Trotz der teils guten Ergebnisse der Ähnlichkeitsbestimmung von Audio Similarity, gibt es dennoch einige Potenziale zur Weiterentwicklung der Anwendung.

Da sehr lange unklar war, was eine gute Möglichkeit für die Suche nach Ähnlichkeiten von Liedern ist, wurde sich bei der Modellentwicklung eher in die Breite orientiert und viele verschiedene Modelle getestet, anstatt die vorhandenen Modelle weiter zu optimieren. Somit existiert mit der Optimierung des Modells eine Möglichkeit die Genauigkeit in Zukunft weiter zu verbessern.

Auch könnte das Modelltraining zur weiteren Verbesserung der Anwendung auf einem größeren Datensatz trainiert werden. Aktuell wird dazu nur der kleine FMA-Datensatz verwendet, da für die Verwendung des gesamten Datensatzes zu viel Speicher und Trainingszeit benötigt wird und dies im Rahmen des Projekts nicht umsetzbar war.

Zudem muss angemerkt werden, dass die im Datensatz vorhandenen Lieder tendenziell eher unbekannt sind und die Genres nur vom Nutzer zugeordnet wurden, was dafür sorgt, dass sie nicht immer einheitlich waren. Dieselben Modelle könnten auf bekannteren Datensätzen teils deutlich bessere Ergebnisse erzielen, was abermals an der Genrezuordnung liegen kann.

Auch die Evaluierung der finalen Anwendung ist schwierig. Ob die vorgeschlagenen Songs tatsächlich Ähnlichkeiten aufweisen, und somit eine gute Empfehlung darstellen, bleibt weiterhin ein subjektives Empfinden. Besonders schwierig war es eine geeignete Evaluierungsmetrik für die Modelle zu finden, da eine graphische Evaluierung keine klaren Ergebnisse liefern konnte. Dieses Problem wurde jedoch durch die Anwendung des kNN-Klassifikators gelöst.

Die Nutzeranwendung bietet weiterhin viele Verbesserungspotenziale. Beispielsweise die Möglichkeit eine genauere Analyse zwischen einem vorgeschlagenem und dem ausgewählten Song anzuzeigen, könnte in einer zukünftigen Version der Anwendung ergänzt werden. Aufgrund des strengen Zeitplans zur Umsetzung musste dieses Feature jedoch vorerst außen vorgelassen werden.

# Literaturverzeichnis

- [1] „MUSIKINDUSTRIE IN ZAHLEN 2022,“ Bundesverband Musikindustrie, [Online]. Available: <https://www.musikindustrie.de/wie-musik-zur-karriere-werden-kann/markt-bestseller/musikindustrie-in-zahlen-2022>. [Zugriff am 22 Juli 2023].
- [2] F. Landwehr, „Musikflx,“ 17 September 2021. [Online]. Available: <https://www.musicflx.de/musikrichtungen/>. [Zugriff am 20 Juli 2023].
- [3] J. B. u. N. Opgenoorth, Keyboardtabelle - Akkorde, Skalen & Modi, Wachtberg: Voggenreither Verlag OHG, 2020.
- [4] D. C. Reineke, Musiktheorie - Ein inhaltlicher Leitfaden für den musiktheoretischen Unterricht an Musikschulen in, Magdeburg: Landesverband der Musikschulen Sachsen-Anhalt e.V., 2008.
- [5] H. Riemann, Musik-Lexikon. Theorie und Geschichte der Musik, die Tonkünstler alter und neuer Zeit mit Angabe ihrer Werke, nebst einer vollständigen Instrumentenkunde, Leipzig: Hugo Riemann, 1882.
- [6] U. Kiencke, Signalverarbeitung, De Gruyter Oldenbourg, 2008.
- [7] dmlc, „XGBoost Documentation,“ [Online]. Available: <https://xgboost.readthedocs.io/en/stable/>. [Zugriff am 23 7 2023].
- [8] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016.
- [9] M. Mishra, „Convolutional Neural Networks, Explained,“ Medium, 26. August 2020. [Online]. Available: <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>. [Zugriff am 23. Juli 2023].

- [10] T. Warnecke, „DATENTRENDS: WIE VEKTORDATENBANKEN NEUE HERAUSFORDERUNGEN MEISTERN,“ Camelot Management Consultants, 12 Juni 2023. [Online]. Available: <https://blog.camelot-group.com/de/2023/06/datentrends-wie-vektordatenbanken-neue-herausforderungen-meistern/>. [Zugriff am 23 Juli 2023].
- [11] V. K. a. B. Deshpande, Data Science - Concepts and Practice, Elsevier Inc., 2019.
- [12] G. E. Hinton, „Papers with Code,“ 2006. [Online]. Available: <https://paperswithcode.com/paper/reducing-the-dimensionality-of-data-with>. [Zugriff am 23 Juli 2023].
- [13] Scipy Developers, „Scipy,“ 2023. [Online]. Available: [https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.kl\\_div.html#scipy-special-kl-div](https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.kl_div.html#scipy-special-kl-div). [Zugriff am 21 Juli 2023].
- [14] Baeldung, „Baeldung,“ [Online]. Available: <https://www.baeldung.com/wp-content/uploads/sites/4/2022/03/LionVAE.png>. [Zugriff am 23 Juli 2023].
- [15] D. P. Kingma, „An Introduction to Variational Autoencoders,“ in *Foundations and Trends in Machine Learning*, Boston, 2019.
- [16] M. S. Fathollahi, „Music similarity measurement and recommendation system using convolutional neural networks,“ International Journal of Multimedia Information Retrieval, 2021.
- [17] Scikit Learn Developers, „Scikit Learn,“ 2023. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>. [Zugriff am 21 Juli 2023].
- [18] Scikit Learn Developers, „Scikit Learn,“ 2023. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html>. [Zugriff am 21 Juli 2023].
- [19] IBM, „K-Nächste Nachbarn Algorithmus,“ IBM, 2023. [Online]. Available: <https://www.ibm.com/de-de/topics/knn>. [Zugriff am 23 Juli 2023].

- [20] M. Defferrard, K. Benzi, P. Vandergheynst und X. Bresson, „FMA: A Dataset For Music Analysis,“ arXiv, 2017.
- [21] Streamlit, [Online]. Available: <https://streamlit.io/>. [Zugriff am 19 7 2023].
- [22] streamlit-extras. [Online]. Available: <https://github.com/arnaudmiribel/streamlit-extras>. [Zugriff am 19 7 2023].
- [23] Free Music Archive, „App developers,“ [Online]. Available: <https://freemusicarchive.org/app-developers>. [Zugriff am 22 7 2023].
- [24] VFR Verlag für Rechtsjournalismus GmbH, „Urheberrecht.de,“ 10. Mai 2023. [Online]. Available: <https://www.urheberrecht.de/musik/>.
- [25] M. Schmahl, „Urheberrechtsreform: Ab sofort könnt ihr 15 sek. Musik legal nutzen!,“ 21. Mai 2021. [Online]. Available: <https://www.gearnews.de/urheberrechtsreform-ab-sofort-koennt-ihr-15-sekunden-legal-samplen/>.
- [26] Zebrafish Labs, „imgix,“ 2023. [Online]. Available: [https://hackernoon.imgix.net/hn-images/1\\*op0VO\\_QK4vMtCnXtmigDhA.png](https://hackernoon.imgix.net/hn-images/1*op0VO_QK4vMtCnXtmigDhA.png). [Zugriff am 23 Juli 2023].