OpenAI: Jukebox
Ein neuronales Netz, das Musik erzeugt

Tim Philipp
Matr.-Nr. 42678
Audiovisuelle Medien
Fakultät Electronic Media
tp057@hdm-stuttgart.de

Roland Ernst
Matr.-Nr. 42677
Audiovisuelle Medien
Fakultät Electronic Media
re033@hdm-stuttgart.de

24. März 2022

Das Team von OpenAI hat an einem neuronalen Netz gearbeitet, das in der Lage ist, spontan neue Musik zu erzeugen. Es gilt derzeit als das modernste Modell auf diesem Gebiet. Der Ansatz besteht darin, rohes Audiomaterial zu modellieren - eine Abkehr von früheren Versuchen, die Musik symbolisch erzeugte (z.B. aus Klaviernoten). Dies hat eine Reihe von Auswirkungen auf die Ergebnisse der Modelle, wobei die wichtigste die Fähigkeit ist, die menschliche Stimme beim Singen zu replizieren. Der gesamte Code und das trainierte Modell sind dabei öffentlich zugänglich, sodass wir es selbst ausprobieren konnten.

1 OpenAI

OpenAI beschäftigt sich mit der Erforschung von Künstlicher Intelligenz (KI) mit dem Ziel, Vorteile für die Gesellschaft zu generieren. Zu den Geldgebern von OpenAI zählen unter anderem Elon Musk und das Unternehmen Microsoft. Patente und Forschungsergebnisse von OpenAI sind für die Öffentlichkeit allgemein zugänglich. Ein weiteres bekanntes Projekt von OpenAI ist der Generative Pretrained Transformer (GPT) in seinen Generationen GPT-1, GPT-2 und GPT-3. Er ist in der Lage, selbstständig Texte zu erstellen.

2 Jukebox

Jukebox generiert Instrumentalmusik oder Musikstücke mit Gesang in vielen verschiedenen Stilrichtungen wie Pop, Rock, Reggae oder Blues. Als Vorgabe für die Erstellung der Musikstücke genügen der KI Stilrichtungen, Liedtexte oder Künstler:innen. Darüber hinaus ist Jukebox in der Lage, angefangene Musikstücke fertigzustellen. Sie basiert auf einem künstlichen neuronalen Netzwerk, das mit ca. 1,2 Millionen Musikstücken trainiert wurde. Die Hälfte dieser Musikstücke hat englische Texte. Neben den Musikstücken selbst wurden dem Modell auch Metadaten, wie die Namen der Künstler:innen, Stilrichtungen oder Veröffentlichungsdaten zugeführt.
Das Generieren der Musikstücke mit Jukebox ist ein rechenintensiver Vorgang und nimmt einiges an Zeit in Anspruch. Ein Minute Musik benötigt ca. neun Stunden Rechenzeit. Auf der Webseite von Jukebox haben die OpenAI Entwickler:innen zahlreiche Beispiele von Musikstücken veröffentlicht, die mit der KI generiert wurden:
`https://openai.com/blog/jukebox/`
Jukebox steht jedem zur Nutzung zur Verfügung. Beispielsweise lässt sich ein Jupyter Notebook von Google Colab verwenden, um mit Jukebox zu experimentieren. Das Jupyter Notebook läuft dann auf einer virtuellen Maschine oder auf einem Google-Server.
`https://colab.research.google.com`

2.1 Motivation

Die automatische Erzeugung von Musik gibt es schon seit mehr als einem halben Jahrhundert. Ein bekannter Ansatz sind *Symbolic Generators*. Dieses Format hat den Vorteil, dass es sowohl im MIDI-Format (Musical Instrument Digital Interface) interpretierbar (z.B. in Tonhöhe, Dauer und Lautstärke) ist als auch in standardmäßigen digitalen Audio-Workstations (DAWs) bearbeitet werden kann. *Sym-*

bolic Generators haben jedoch ihre Grenzen - sie können weder menschliche Stimmen noch viele der subtilen Klangfarben, Dynamiken und Ausdrucksmöglichkeiten erfassen, die für Musik wesentlich sind.
Die Generierung von Musik auf Audioebene ist eine Herausforderung, da die Sequenzen sehr lang sind. Ein typischer 4-Minuten-Song in CD-Qualität (44kHz, 16Bit) hat über 10 Millionen Zeitschritte. Um die hochauflösende Semantik von Musik zu erlernen, müsste ein Modell also mit extrem weitreichenden Abhängigkeiten umgehen. Eine Möglichkeit, das Problem des langen Inputs zu lösen, besteht in der Verwendung eines **Autoencoders**, der das rohe Audiomaterial auf einen nieder-dimensionalen Raum komprimiert, indem er einige der für die Wahrnehmung irrelevanten Informationsbits verwirft. Man kann dann ein Modell trainieren, welches Daten in diesem komprimierten Raum erzeugt und sie dann wieder in den rohen Audioraum hochrechnet.
Das Projekt Jukebox hat sich das Ziel gesetzt, die Grenzen generativer Modelle weiter auszuloten. Bereits in vorangegangenen Arbeiten wie dem *MuseNet* hat sich das OpenAI Team mit der Synthese von Musik auf Grundlage großer Mengen von MIDI-Daten beschäftigt. In der Audio-Rohdatenwelt müssen Modelle also lernen, sowohl mit einer großen Vielfalt als auch mit sehr weitreichenden Strukturen umzugehen. Audio-Rohdaten sind hierbei besonders empfindlich gegenüber Fehlern im kurz-, mittel- oder langfristigen Timing.

2.2 Vector Quantized Variational Autoencoders (VQ-VAE)

Das Autoencoder-Modell von Jukebox komprimiert Audio in einen diskreten Raum und verwendet einen quantisierungsbasierten Ansatz namens **VQ-VAE**. Hierarchische VQ-VAEs können kurze Instrumentalstücke aus einigen wenigen Instrumentenstücken erzeugen, leiden jedoch unter einem Zusammenbruch der Hierarchie aufgrund der Verwendung von aufeinanderfolgenden Encodern in Verbindung mit autoregressiven Decodern. Ein vereinfachtes Modell namens **VQ-VAE-2** meidet diese Probleme, indem es nur Feedforward-Encoder und -Decoder verwendet und dabei beeindruckende Ergebnisse bei der Erzeugung von Bildern mit hoher Wiedergabetreue zeigt. [1]
Die Grundidee dieses Modells ist also die Verwendung diskreter latenter Einbettungen für Variationsautokodierer.

2.2.1 Autoencoder

Ein Autoencoder besteht aus einem Encoder- und einem Decoder-Modul. Der Encoder bildet jede Eingabe **x** auf eine latente Repräsentation **y** ab, sodass das Decoder-Netzwerk in der Lage ist, eine **Rekonstruktion z** aus **y** zu berechnen, wobei **z** so nah wie möglich an der Eingabe **x** liegen muss. Die Gewichte des Encoder- und Decoder-Moduls werden durch Minimierung einer Verlustfunktion gelernt, die die Differenz zwischen der Rekonstruktion **z** und dem ursprünglichen **x** misst. Hierfür können standardmäßige Algorithmen des überwachten Lernens mit Gradientenabstieg, wie z.B. Backpropagation, verwendet werden. [3]

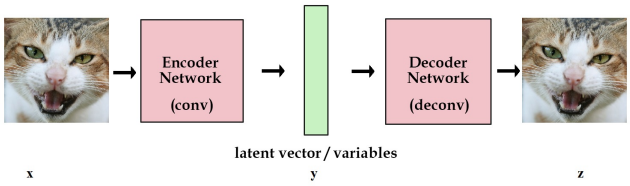

Abbildung 1: Frans, K. (2016): Autoencoder, *Variational Autoencoders Explained*, kevin frans blog

Normalerweise ist die latente Repräsentation y viel kleiner als das Original x und seine Rekonstruktion z. Der gesamte Prozess realisiert daher eine verlustbehaftete Standardkompression, wie z.B. die JPEG-Kodierung von Bildern.
Zwei wichtige Anwendungskategorien von Autoencodern sind:

- **Representation learning**: Die latente Repräsentation y kann als eine aussagekräftige und effiziente Repräsentation der Eingabe x betrachtet werden. Diese komprimierte Darstellung enthält alle relevanten Informationen, da das Original sonst nicht ausreichend rekonstruiert werden könnte.
- **Erkennung von Anomalien**: Der Autoencoder wird nur mit Normalfällen trainiert. In der Inferenzphase ergeben normale Fälle am Eingang einen kleinen Fehler zwischen dem Eingang und seiner Rekonstruktion. Nicht-normale Fälle können jedoch nicht gut rekonstruiert werden. Daher ist ein hoher Rekonstruktionsfehler ein Indikator für Anomalien.

2.2.2 Variational Autoencoder

Ein Variational Autoencoder ist ein **generatives Modell**, d.h. es kann durch Lernen von Trainingsdaten neue Daten (Samples) erzeugen. Wenn die Wahrscheinlichkeitsverteilung einer oder mehrerer Zufallsvariablen bekannt ist, können im Allgemeinen neue Samples erzeugt werden, die sich aus dieser Verteilung ergeben.
Sind zum Beispiel die Parameter Mittelwert und Standardabweichung einer Gauß-verteilten Zufallsvariablen bekannt, so können neue Samples erzeugt werden, die dieser Verteilung entsprechen.
Das Encodermodul bildet die Eingabe auf eine Wahrscheinlichkeitsverteilung ab. Es wird also ein Typ von Wahrscheinlichkeitsverteilung (z.B. Gaußsche Normalverteilung) angenommen und die Ausgabe des Encoders sind die konkreten Parameter dieser Verteilung (z.B. Mittelwert und Standardabweichung für den Typ Gauß'sche Normalverteilung). Dann wird ein konkretes Sample aus dieser Verteilung erzeugt, welche als Eingabe für den Decoder dient. [3]

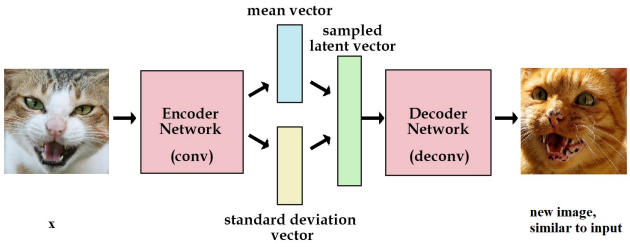

Abbildung 2: Frans, K. (2016): Variational Autoencoder, *Variational Autoencoders Explained*, kevin frans blog

2.2.3 Vektorquantisierung

Die Vektorquantisierung ist ein Verfahren zur Kompression oder Identifikation von Datensätzen.
Die Datensätze werden in Merkmalsvektoren zusammengefasst. Das Prinzip des Verfahrens besteht darin, diesen Merkmalsvektoren denjenigen Vektor aus einer Tabelle zuzuordnen, der dem betrachteten Merkmalsvektor am ähnlichsten ist. Statt alle Daten des Merkmalsvektors zu speichern, wird nur der Index dieses ähnlichsten Vektors benötigt.
Die Vektorquantisierung besteht dabei aus zwei Schritten:

- Im ersten Schritt, dem Training, muss eine Tabelle (**codebook**) mit häufig vorkommenden Merkmalsvektoren erstellt werden.
- Im zweiten Schritt wird für weitere Vektoren jeweils der Codebuchvektor mit dem **geringsten Abstand** bestimmt.

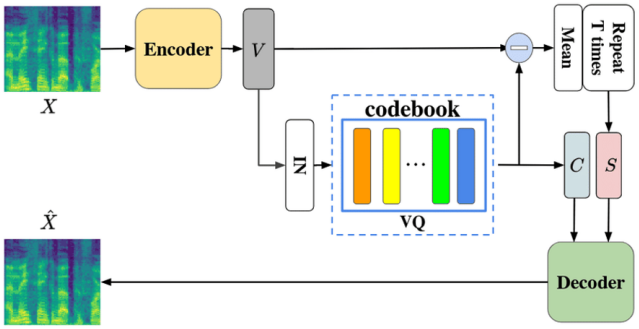

Abbildung 3: Wu, Chen, Lee (2020): The VQVC architecture, *VQVC+: One-Shot Voice Conversion by Vector Quantization and U-Net architecture*, College of Electrical Engineering and Computer Science, National Taiwan University

Zur Datenübertragung wird nur der Index des Codebuchvektors benötigt, der auch ein Vektor sein kann, wenn das Codebuch mehrdimensional ist. Der korrespondierende Decoder muss über das gleiche Codebuch verfügen und kann dann aus dem Index eine Schätzung des ursprünglichen Vektors erzeugen.

2.2.4 Von VQ-VAE-2 inspirierte, modifizierte Architektur

Die Umsetzung von Jukebox wurde mit Hilfe der VQ-VAE-2 Architektur inspiriert und wendet den Ansatz mit folgenden Modifikationen auf Musik an:

- Um den bei VQ-VAE-Modellen üblichen Zusammenbruch des Codebuchs zu vermeiden, werden zufällige Neustarts getriggert, bei denen ein Codebuchvektor nach dem Zufallsprinzip auf einen der kodierten verborgenen Zustände zurückgesetzt wird, sobald seine Nutzung unter einen Schwellenwert fällt.
- Um die Nutzung der oberen Ebenen zu maximieren, werden separate Decoder verwendet und die Eingabe unabhängig voneinander aus den Codes jeder Ebene rekonstruiert.
- Damit das Modell höhere Frequenzen leicht rekonstruieren kann, wurde ein spektraler Verlust hinzugefügt, der die Norm der Differenz zwischen Eingabe- und rekonstruierten Spektrogrammen bestraft.

In VQ-VAE werden **drei Stufen** verwendet, die das 44-kHz-Rohsignal um das **8-fache, 32-fache bzw. 128-fache komprimieren**, mit einer **Codebuchgröße von 2048 für jede Stufe**. Bei diesem Downsampling gehen viele Audio-Details verloren und der Klang wird mit zunehmender Tiefe der Stufen deutlich verrauscht. Allerdings bleiben wesentliche Informationen über Tonhöhe, Klangfarbe und Lautstärke des Tons erhalten.

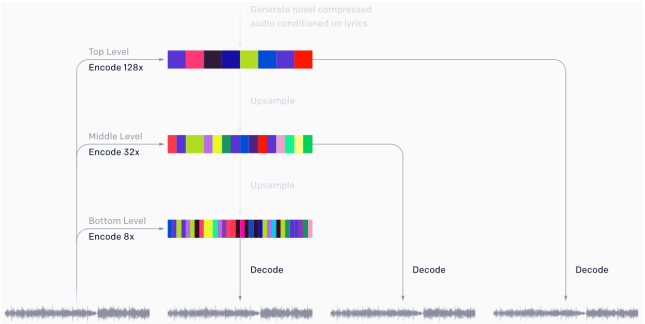

Abbildung 4: Dhariwal, Jun, Payne, Kim, Radford, Sutskever (2020): VQ-VAE encoding, *Jukebox: A Generative Model for Music*, OpenAI

Jede VQ-VAE-Stufe kodiert das Eingangssignal unabhängig. Die untere Kodierungsebene erzeugt die hochwertigste Rekonstruktion, während die obere Kodierungsebene nur die wesentlichen musikalischen Informationen beibehält.

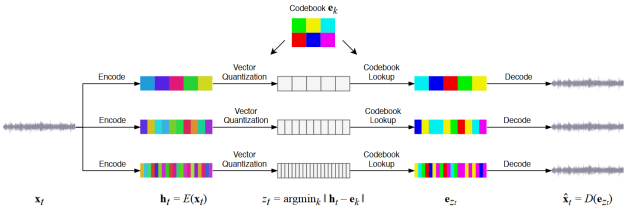

Abbildung 5: Dhariwal, Jun, Payne, Kim, Radford, Sutskever (2020): Three separate VQ-VAE models, *Jukebox: A Generative Model for Music*, OpenAI

2.2.5 Musik-Priori und Upsamplers

Um Samples zu erzeugen, muss nach dem Training des VQ-VAE eine **A-Priori-Verteilung** (kurz: Prior) über den komprimierten Raum **gelernt werden**. Das Prior-Modell

wird zerlegt und separate Modelle für den Top-Level-Prior trainiert. Jedes dieser Modelle ist ein autoregressives Modellierungsproblem im diskreten Token-Raum, der durch den VQ-VAE erzeugt wird. Dabei werden **Transformers mit Sparse Attention** verwendet.
Für die Upsampler müssen die autoregressiven Transformatoren mit Konditionierungsinformationen aus den Codes der oberen Ebenen versorgt werden.
Die konfigurierbaren Jukebox-Modi sind:

- **Ancestral sampling**: Erstellt Songs auf der Grundlage von Künstler- und Genre-Konditionierung:

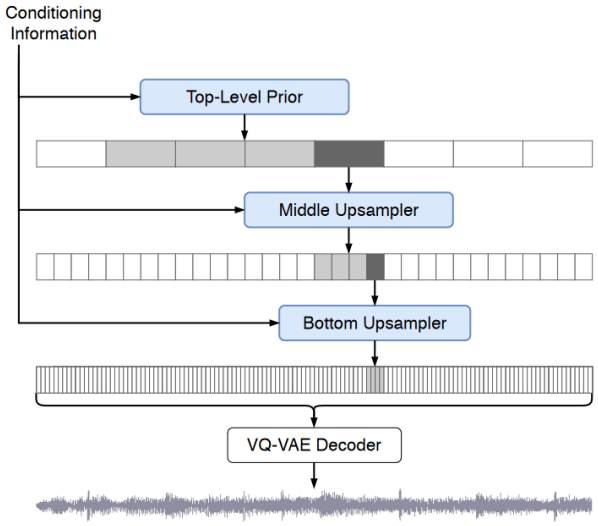

Abbildung 6: Dhariwal, Jun, Payne, Kim, Radford, Sutskever (2020): Ancestral sampling, *Jukebox: A Generative Model for Music*, OpenAI

- **Windowed sampling**: Um Musik zu erzeugen, die länger ist als die Kontextlänge des Modells, werden auf jeder Ebene wiederholt Unterbrechungen gesampelt, wobei überlappende Fenster früherer Codes als Kontext dienen:

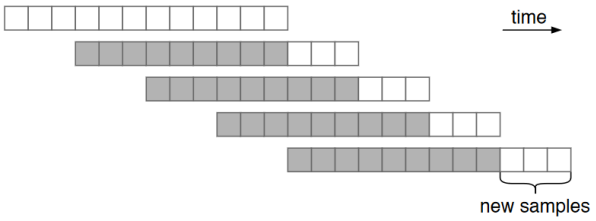

Abbildung 7: Dhariwal, Jun, Payne, Kim, Radford, Sutskever (2020): Windowed sampling, *Jukebox: A Generative Model for Music*, OpenAI

- **Primed sampling**: Neue Songerstellung unter Verwendung eines beliebigen Audio-Samples:

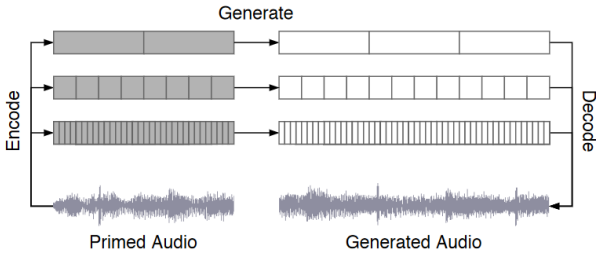

Abbildung 8: Dhariwal, Jun, Payne, Kim, Radford, Sutskever (2020): Primed sampling, *Jukebox: A Generative Model for Music*, OpenAI

- **Temperature**: Ein höherer Wert bedeutet mehr Zufall.

2.2.6 Lyrische Konditionierung

Um das generative Modell mit Liedtexten zu steuern, wird das Modell in jedem Audiosegment so auf Liedtexte konditioniert, dass das Modell gleichzeitig mit der Musik auch Gesang erzeugt:

- **Lyrics-to-singing (LTS) task**: Das Konditionierungssignal enthält nur den Text des Liedes, aber keine Zeit- oder Gesangsinformationen.
- **Bereitstellung von Liedtexten für einzelne Audioabschnitte**: Um die Aufgabe zu erleichtern, werden die Lyrics auf kürzere 24 Sekunden Audioabschnitte trainiert.
- Ein **Encoder-Decoder Modell** wird verwendet, um die Zeichen der Liedtexte zu konditionieren, wobei der Encoder Merkmale aus den Liedtexten erzeugt, die vom Decoder berücksichtigt werden, der die Musik-Token der obersten Ebene erzeugt.

Ziel ist es, die Zeichen des Liedtextes so auszurichten, dass sie die Dauer jedes Liedes linear überspannen. Bei bestimmten Genres mit schnellen Texten, wie z.B. Hip-Hop, funktioniert dies allerdings nicht optimal. Lösung hierbei ist die Verwendung spezieller Modelle:

- *Spleeter*, um den Gesang aus jedem Lied zu extrahieren und
- *NUS AutoLyricsAlign* auf den extrahierten Gesang auszuführen, um eine genaue Ausrichtung der Liedtexte auf Wortebene zu erhalten.

2.2.7 Datensatz

- 1,2 Millionen Lieder
- Entsprechende Liedtexte und Metadaten aus LyricWiki
- Metadaten umfassen Künstler, Genre, Album und Jahr, Stimmungen oder Playlist-Schlüsselwörter
- Training mit 32-Bit-, 44,1-kHz-Rohdaten

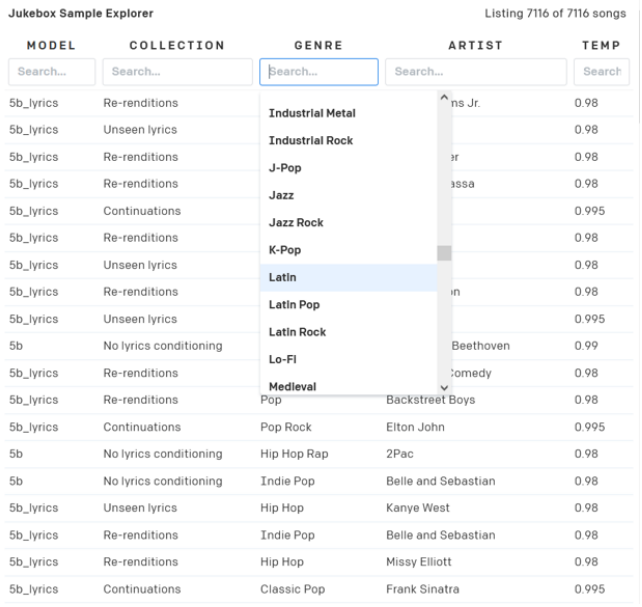

Abbildung 9: Dhariwal, Jun, Payne, Kim, Radford, Sutskever (2020): Jukebox Sample Explorer, *Jukebox: A Generative Model for Music*, OpenAI

3 Deep Learning @ HdM

Zur Erzeugung unserer eigenen Musikstücke haben wir aus dem Deeplearning Cluster der HdM die *TARDIS*-Maschine mit folgenden Spezifikationen zugewiesen bekommen:

Prozessor	i7-6950X (10-core) @ 3.0GHz
GPU	4 x NVIDIA RTX 2080 12GB
RAM	128GB (8 x 16GB) DDR4 PC2400
Speicher	1 x 1TB M.2 Samsung 970 Pro 1 x 4TB SATA-HDD WD Black

Tabelle 1: Spezifikationen der *TARDIS*-Maschine aus dem HdM Deeplearning Cluster

Um dem erhöhten Rechenaufwand gerecht zu werden, wurde uns außerdem eine NVIDIA RTX A6000 Grafikkarte mit 48GB dediziertem Speicher freigeschaltet. Zur Ausführung von Jukebox werden **16GB Grafikkartenspeicher empfohlen**.

4 Praxisanwendung

Jukebox ermöglicht es uns, durch die Angabe des Genres, der Künstler:innen und dem Hinzufügen eigner Lyrics ein vollkommen neues Sample zu generieren. Dabei ist eine Vielzahl von Ausgaben möglich, es können verschiedenste Genres und Styles aus den Eingaben erstellt werden. Durch ein Sprachmodell von Open AI ist es zudem möglich, auch „gesungene“ Lyrics zu erstellen, wobei diese oft schwer zu verstehen sind und teils beunruhigend klingen können.

4.1 Unseen lyrics

Bei dieser Ausgabe-Art sind Lyrics nicht Teil des Trainings. Das Ergebnis ist dementsprechend durchwachsen bis nicht als Lyrics erkenntlich.

Beispiel: Classic Pop in the Style of Frank Sinatra
`https://jukebox.openai.com/?song=788153929`

4.2 Re-renditions

Bei dieser Ausgabe-Art sind Lyrics Teil des Trainings, der Eingabe-Song wird neu interpretiert. So entstehen interessante Variationen.
Beispiel: Lose Yourself - Eminem
`https://jukebox.openai.com/?song=787882117`

4.3 Re-renditions in another Style

Bei dieser Ausgabe-Art wird der Eingabe-Song im Style einer/s anderen Künstler:in neu interpretiert. So entstehen „Cover“.
Beispiel: Lose Yourself – Eminem – Style of Kanye West
`https://jukebox.openai.com/?song=789016690`

4.4 Continuations/Completions

Bei dieser Ausgabe-Art werden die ersten 12 Sekunden des Eingabe-Songs analysiert, auf deren Basis die KI versucht, diesen fortzuführen. So entstehen „Fortführungen“.
Beispiel: Never Gonna Give You Up – Rick Astley
`https://jukebox.openai.com/?song=787729588`

4.5 Novel Artists and Styles

Bei dieser Ausgabe-Art können mehrere Stile miteinander kombiniert werden. So entstehen theoretisch neue Genres und unübliche Cover (auf Text-Basis).
Beispiel: Country Jazz
`https://jukebox.openai.com/?song=794020879`

4.6 Unsere Samples

Folgende Parameter haben wir bei der Erstellung unserer Songs gesetzt:

```
In [7]: metas = [dict(artist = "50 Cent",
                      genre = "Country",
                      total_length = hps.sample_length,
                      offset = 0,
                      lyrics = """Mustard on the beat ho!
I was good on my own, that's the way it was, that's the way it was
You was good on the low for a faded fuck, on some faded love
Shit, what the fuck you complaining for?
Feeling jaded huh?
Used to trip off that shit I was kickin' to you
Had some fun on the run though I give it to you
But baby, don't get it twisted
You was just another nigga on the hit list
Tryna fix your inner issues with a bad bitch
Didn't they tell you that I was a savage
Fuck your white horse and a carriage
Bet you never could imagine
Never told you you could have it
You needed me
Oooh, you needed me
To feel a little more, and give a little less
Know you hate to confess
But baby ooo, you needed me
                      """,
                      ),
                 ] * hps.n_samples
labels = [None, None, top_prior.labeller.get_batch_labels(metas, 'cuda')]
```

Link zur Audiodatei: `https://bit.ly/3qmVPl0`

```
In [8]: metas = [dict(artist = "bob marley & the wailers",
                      genre = "Reggae",
                      total_length = hps.sample_length,
                      offset = 0,
                      lyrics = """There ain't no gold in this river
That I've been washing my hands in forever
I know there is hope in these waters
But I can't bring myself to swim
When I am drowning in this silence
Baby, let me in

Go easy on me, baby
I was still a child
Didn't get the chance to
Feel the world around me
I had no time to choose what I chose to do
So go easy on me
                      """,
                      ),
                 ] * hps.n_samples
labels = [None, None, top_prior.labeller.get_batch_labels(metas, 'cuda')]
```

```
In [9]: sampling_temperature = .99

lower_batch_size = 16
max_batch_size = 3 if model == "5b_lyrics" else 16
lower_level_chunk_size = 32
chunk_size = 16 if model == "5b_lyrics" else 32
sampling_kwargs = [dict(temp=.99, fp16=True, max_batch_size=lower_batch_size,
                        chunk_size=lower_level_chunk_size),
                   dict(temp=0.99, fp16=True, max_batch_size=lower_batch_size,
                        chunk_size=lower_level_chunk_size),
                   dict(temp=sampling_temperature, fp16=True,
                        max_batch_size=max_batch_size, chunk_size=chunk_size)]
```

Link zur Audiodatei: `https://bit.ly/3ijtWWz`

5 Limitierungen von Jukebox

Die Lücke zu menschlich-erstellter Musik ist noch deutlich zu spüren, so interessant und gut die Ergebnisse auch sein mögen. Gerade der Teil, der „wirklich“ menschlich ist, nämlich die Stimme und das Vermögen, sinnvolle Lyrics zu schreiben, fehlen gänzlich. Zwar ist es beachtlich, dass Jukebox bereits in der Lage ist, mehr als Akkordfolgen zu schreiben, beispielsweise auch Soli sind möglich, doch genau dieser Baustein macht einen Song menschlich. Zusammenhangslose Wortfetzen und syntaktisch falsche Satzstrukturen allein können nicht mit der Kreativität menschlicher Individuen mithalten. Uns bekannte Strukturen wie Refrains oder Breaks fehlen ebenso.

Hinzu kommt, dass Jukebox zum aktuellen Zeitpunkt lediglich englischsprachige Titel verarbeiten und wiedergeben kann. Nicht nur das, auch die Datenbank ist lediglich mit westlicher Musik gefüllt; große Teile kulturell einflussreicher Musikstücke fehlen vollkommen und schränken die Funktionsmöglichkeit der KI bedeutend ein.

Auch aus technischer Sicht ist Jukebox noch nicht auf einem Level angekommen, das für die Masse tauglich ist und dadurch nur Technik-affinen Menschen vorbehalten bleibt. Wünschenswert wäre eine Applikation, die keine Highend-Hardware auf Seiten der Nutzer erfordert. Zwar ermöglicht es Google Colab, deren Notebooks kostenfrei zu nutzen, dennoch wäre eine Web-basierte Applikation oder Ähnliches deutlich bedienungsfreundlicher und würde es so einer breiteren Masse ermöglichen, sich mit KI-generierter Musik auseinanderzusetzen und sie zu erleben. Doch selbst mit solch einer für die Nutzenden freundlicheren Bedienungsoberfläche benötigt die KI schlichtweg zu viel Zeit zum Rendern, selbst mit Hochleistungsrechnern. Hier kann man lediglich auf den technischen Fortschritt und das mooresche Gesetz hoffen.

Davon einmal abgesehen erreicht die KI bei weitem noch nicht die technische Raffinesse menschlich erzeugter Musik. Selbst die besten Samples, die Jukebox ausgeben kann, sind von Glitches durchzogen und besitzen selbst auf den höchsten Einstellungen noch ein deutlich hörbares Rauschen. Dies könnte durch eine Verbesserung des VQ-VAE beseitigt werden.

6 Zukunft von Jukebox

Derzeit wird versucht, sich auf verschiedene Information beim Priming zu konzentrieren, um neue Ergebnisse zu erhalten. Ebenso werden neuerdings auch MIDI-Dateien nun Stems verwendet, um daraus besser strukturierte Samples zu erzeugen. Solch eine Einbindung würde es den Anwendenden auch erlauben, das Endergebnis zu beeinflussen. Eine Zusammenarbeit von Künstler:innen und der KI sind geplant und sollen der Forschung und Entwicklung förderlich sein.

7 Diskussion: Auswirkung auf Industrie und Kunst

Im Rahmen unserer Forschung haben wir uns gefragt, inwiefern sich diese KI-basierte Musik auf die aktuelle oder zukünftige Musikindustrie auswirken könnte. Beispielsweise könnte man sich überlegen, unvollendete Stücke von verstorbenen Künstler:innen fertigzustellen oder schlichtweg neue Alben in deren Stil zu produzieren und sie somit künstlich zu immortalisieren. Dabei kamen wir zu dem Ergebnis, dass das Potenzial vorhanden ist, aber noch nicht ausreichend genutzt werden kann. So wurde besonders auf die niedrige Qualität sowie die fehlende Struktur eingegangen, die der generierten Musik einen unangenehmen Charakter verleiht, die dem sogenannten Uncanny Valley zugeordnet werden kann. Auch im bewussten Einsatz von Harmonie und Rhythmik ist der Mensch der KI noch deutlich voraus.

Auch Open AI übergab die KI an 10 ausgewählte Künstler:innen, um neues Wissen aus deren Arbeit zu erhalten. Diese gaben jedoch an, die KI zu ihrem jetzigen Zeitpunkt nicht nutzen zu können, da deren Funktion noch zu sehr eingeschränkt sei.

Ebenfalls fragten wir bei einem Major-Label wir an, bekamen jedoch lediglich die Antwort, dass es momentan eine eher untergeordnete Rolle spielen würde.

8 Fazit

Das Potenzial, neue Musik zu erschaffen ist bereits vorhanden. Jukebox schafft es bisher jedoch nicht, konsistent menschlich anmutende Musik zu generieren oder Stile exakt nachzuahmen. Ob sich dies in naher Zukunft ändern kann, wird sich noch zeigen. Hinzu kommt die Frage, ob das künstliche Immortalisieren bereits verstorbener Künstler:innen gesellschaftlich erwünscht und ethisch vertretbar wäre. Als Inspirationsquelle oder eben doch nur zur Unterhaltung funktioniert Jukebox jedoch schon heute

und kann teils beeindruckende Ergebnisse vorweisen.

9 Referenzen

1 Razavi, Ali and van den Oord, Aaron and Vinyals, Oriol (2019): *Generating Diverse High-Fidelity Images with VQ-VAE-2*, Curran Associates, Inc.,
https://proceedings.neurips.cc/paper/2019/file/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Paper.pdf

2 Dhariwal, Prafulla and Jun, Heewoo and Payne, Christine and Kim, Jong Wook and Radford, Alec and Sutskever, Ilya (2020): *Jukebox: A Generative Model for Music*,
https://arxiv.org/abs/2005.00341

3 Kingma, Diederik P and Welling, Max (2013): *Auto-Encoding Variational Bayes*,
https://arxiv.org/abs/1312.6114