TECHNISCHE HOCHSCHULE MITTELHESSEN
CAMPUS GIESSEN
MNI
Mathematik, Naturwissenschaften und Informatik

Abschlussprojekt im Modul
EINFÜHRUNG IN DEEP LEARNING

Tone Transfer

Sonntag, 04.02.2024

von
Jannis Müller
jannis.laurin.mueller@mni.thm.de
Matrikel Nr.: 5363340

und
Moritz Blum
moritz.blum@physik.uni-giessen.de
Matrikel Nr.: 8101649

Inhaltsverzeichnis

1 Einleitung

Mit „Tone Transfer“ lässt sich der Klangcharakter eines Instruments auf jede beliebige monophone Audioquelle übertragen. Das Ziel ist es, die aufgenommene Sequenz so klingen zu lassen, als würde sie von dem trainierten Instrument gespielt werden, in unserem Fall ist dies eine E-Gitarre.

Die Netze können mit fast jedem monophonen Instrument trainiert werden. Wir haben uns für eine E-Gitarre der Marke „Duesenberg“ entschieden, die gerade für über 2500 € erhältlich ist.

Mögliche Anwendung könnte das Modell in Online-Shops von Händlern finden, die Musikinstrumente anbieten. Für Musiker, die schon ein Instrument besitzen, aber ein Neues kaufen wollen, könnte durch das Modell das neue Instrument bequem von zu Hause aus testbar sein. Durch das Modell könnte die Audioeingabe vom eigenen Instrument auf den Klang des gewünschten Instruments übertragen werden, um dem Kunden eine Idee vom Klang zu geben und damit bei der Kaufentscheidung zu unterstützen. Des Weiteren wäre es möglich das Modell bei der Musikproduktion zu nutzen, um zu testen, welche E-Gitarre am besten in einen Song passen würde. Im Live-Einsatz könnte das Modell mit einer Implementierung als real-time-fähiges Audio-Plugin[1] ein fehlendes Instrument ersetzen.

2 Verwandte Arbeiten

Ein ähnliches Projekt unter dem gleichen Namen „Tone Transfer“ wurde 2021 von Google veröffentlicht, welches die grundlegende Idee sowie Namen für das Modell liefert [2]. Auf der Website[2] der Google-Forschungsgruppe kann das Modell getestet werden. Bei diesem Modell wurde die Python-Library „DDSP: Differentiable Digital Signal Processing“ verwendet, welches von Google 2020 veröffentlicht wurde und die Grundlage für verwendete Verarbeitungsmodule im Audiobereich liefert, die differenzierbar sind. Damit sind diesse für das Training von Neuronalen Netzen geeignet, weil diese zwischen Berechnung der Verlustfunktion und dem trainierbaren Netz geschaltet werden können [1]. Die Idee von Tone Transfer bildet zusammen mit den Modulen von DDSP die Basis für unser eigenes Modell. Da DDSP mit TensorFlow arbeitet, verwenden wir für unser eigenes Modell auch TensorFlow.

[1] Beispiel für die Implementierung als Audio-Plugin: https://magenta.tensorflow.org/ddsp-vst
[2] Website Tone Transfer: https://sites.research.google/tonetransfer

3 Datensatz

Der verwendete Datensatz wurde von einem professionellen E-Gitarristen (Josiah Lauer) aufgenommen und umfasst 20 min Audio. Bei der Länge wurde sich an der Länge der Datensätze des DDSP-Papers orientiert [1]. Die zwanzigminütige Aufnahme ist in drei Phasen aufgeteilt: Der Trainingsdatensatz besteht aus 70 %, der Validierungsdatensatz aus 20 % und der Testdatensatz aus 10 % der Originalaufnahme. Dabei ist zu erwähnen, dass der Trainings- und Validierungsdatensatz während dem Training nicht durchmischt werden.

Die Audiodateien sind für die drei Phasen in jeweils vier Sekunden lange Abschnitte unterteilt. Mithilfe des bestehenden Netzes „CREPE“ [3] wird die Tonhöhe analysiert und zusammen mit der Lautstärke als TFRecord[3] abgespeichert. Da dieser Prozess mehr Rechenaufwand als das Training selbst benötigt, wird dies im Vorfeld berechnet und abgespeichert.

4 Entwicklung des Modells

4.1 Architektur

Viele frühere Ansätze versuchen Audio im Frequenz- oder Zeitbereich zu generieren. Damit ist die Menge der erzeugbaren Klänge enorm, wobei in der Regel nur ein ganz bestimmter Klang erwartet wird. Dazu kommt, dass das Netz die komplexe Kodierung von Audiosignalen lernen muss um diese erzeugen zu können. Beispiele dafür sind die Zusammenhänge zwischen Samples und Frequenz im Zeitbereich und von Grundfrequenz zu Oberschwingungen im Frequenzbereich. Dadurch geht ein Großteil der Trainingskapazitäten des Netzes für das Lernen der Kodierungen verloren. Für komplexe Signale, wie Sprache, ist dies zwar notwendig, aber bei Instrumenten handelt es sich meist um deutlich einfachere Klänge, nämlich um angeregte harmonische Schwingungen. Dieses Domänenwissen wird von DDSP ausgenutzt [1]. Im Paper wird eine Architektur vorgestellt, wo das Netz das Audiosignal nicht direkt erzeugt, sondern nur die Parameter von zwei verschiedenen Synthesizern steuert. Der erste Synthesizer, ein Additiver Synthesizer, dient der Erzeugung von Grund- und Obertönen der melodischen Klanganteile. Der zweite Synthesizer, ein Noise-Synthesizer, ist für die nicht-melodischen Klanganteile zuständig, wie das Anschlagen einer Saite oder das Blasen in eine Trompete.

Die Architektur umfasst eine Tonhöhenbestimmung durch CREPE, den trainierbaren Decoder, die beiden genannten Synthesizer und den Hall (Reverb), welcher hier nicht trainiert wurde. Eine Übersicht der Architektur ist in Abb. 1 dargestellt.

[3]Mit dem TFRecord-Format können Folgen von Binärdatensätzen gespeichert und von TensorFlow verarbeitet werden: https://github.com/google/tensorflow-recorder

Aus dem Original-Audio wird die Lautstärke und mithilfe von CREPE die Tonhöhe bestimmt, welche der Decoder als Inputs bekommt. Dieser liefert eine harmonische Verteilung der Obertöne und die Lautstärke an den Additiven Synthesizer (Harmonic Synth), welcher zusätzlich die Grundfrequenz als Input bekommt. Dadurch wird sichergestellt, dass das transformierte Audio dieselbe Gundtonhöhe wie das Original hat und nur die Klangfarbe verändert wird. Der Decoder liefert zusätzlich eine Frequenzantwort an den Noise-Synthesizer, der Anschlagsgeräusche und weitere nicht harmonische Geräusche erzeugt. Der Output beider Synthesizer wird dem Hall-Modul übergeben, welcher die Impulsantwort des Raums simulieren kann. Über das transformierte Audio und das Original-Audio kann letztendlich die Verlustfunktion berechnet und der Decoder trainiert werden.

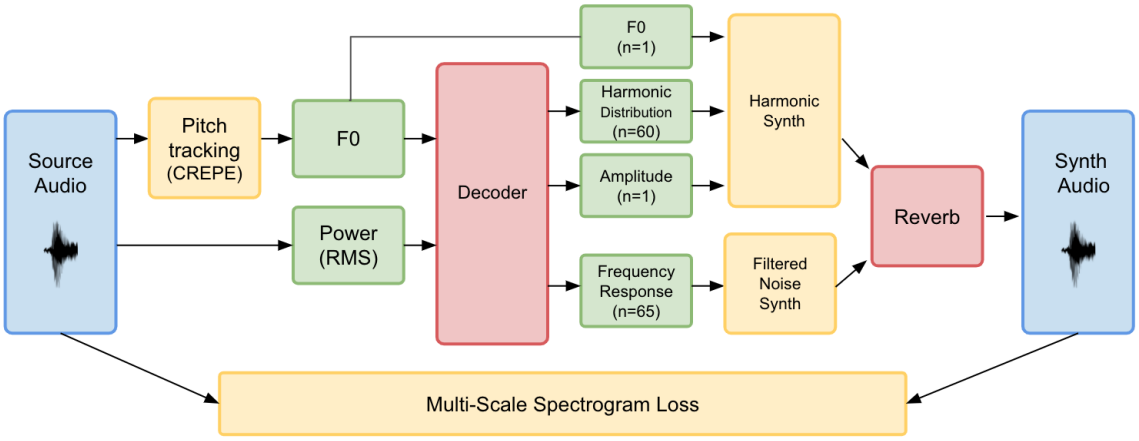

Abbildung 1: Übersicht der verwendeten Architektur. Die Module in gelb werden zur Vor- und Nachbearbeitung genutzt, die beiden roten Module sind trainierbare Netze und die grünen Module stellen verwendete Größen dar. Adaptiert von [2].

4.1.1 Decoder

Für den Decoder wird eine auf „Gated Recurrent Units“-basierende Architektur gewählt. Im Vergleich mit LSTMs erzielten GRUs mit weniger Parametern schneller bessere Ergebnisse. Ansätze basierend auf Dilated Convolutions, wie im Tone-Tranfer-Paper beschrieben [2], erzielten dabei schlechtere Werte bei den Metriken, als das RNN-basierte Netz.

Die Struktur des Decoders ist in Abb. 2 dargestellt. Die beiden Inputs Lautstärke und Tonhöhe werden durch ein Multi-Layered-Perceptron (MLP) verarbeitet und aneinander gehängt. Dabei besteht ein Layer des MLP aus einem Fully-Connected Layer, einer Normierung und der ReLU Aktivierungsfunktion. Die GRU liefert einen mehrdimensionalen Output, welcher zu einer Sequenz zusammengefügt wird und in

ein weiteres MLP gegeben wird. Der Output des MLP wird aufgeteilt und über je ein Fully-Connected Layer an die beiden Syntheziser übergeben.

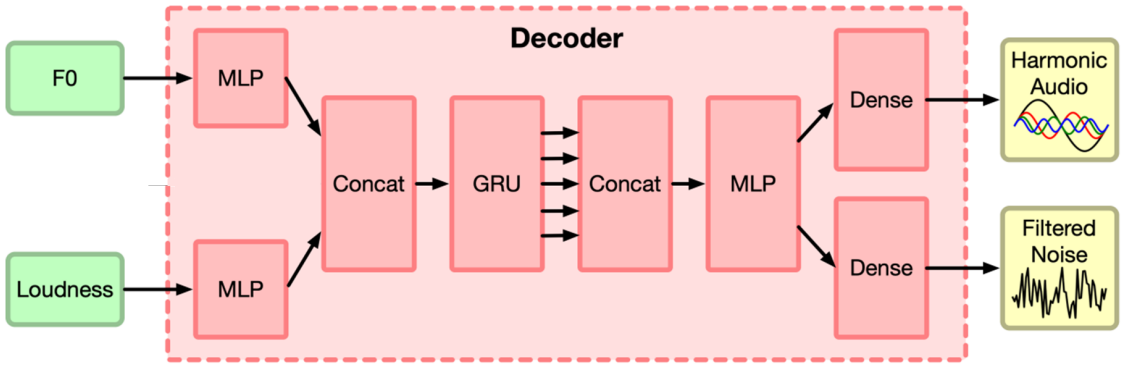

Abbildung 2: Schematischer Aufbau des Decoders. Hierbei steht „Dense“ für ein Fully-Connected Layer, F0 für die Tonhöhe und Concat für die Aneinanderreihung mehrdimensionaler Inputs. Adaptiert von [1].

4.2 Hyperparameter

Im abgegebenen Modell wird ein Decoder mit einer 256 dimensionalen GRU verwendet. Die Fully-Connected Layer haben ebenfalls eine Dimension von 256, mit 2 Layers im MLP. Diese Parameter wurden im Trainingsverlauf variiert. Des Weiteren wird eine Lernrate von 0.001 und ADAM als Optimizer verwendet. Um Exploding-Gradients zu verhindern werden Gradient auf einen Wert von maximal 3.0 gekürzt (Gradient-Clipping). So konnte eine Überfixierung auf die stillen Teile der Trainingsdaten in den ersten Trainingsschritten vermindert werden. Diese Parameter sind an den Modellen der DDSP-Gruppe orientiert [1].

4.3 Lernstrategie

Das Modell wird auf Google-Colab trainiert, wobei der Datensatz im Google-Drive liegt. Mit Weights & Biases wird der Trainingsverlauf unter Verwendung der verschiedenen Metriken aufgezeichnet. Beim Training wird ein Forward-Pass mit Backpropagation als Step bezeichnet. Wir haben uns für eine Epochengröße von ungefähr 600 Steps entschieden. Nach jeder Epoche wird das Modell im Drive gespeichert und mit dem Validierungsdatensatz überprüft. Als Verlustfunktion wird der Spectral-Loss aus der DDSP-Library verwendet. Für das Training sind 50 Epochen angesetzt, wobei diese nicht immer vollständig durchgeführt werden, da bei schlechten Werten der

Metriken und der Verlustfunktion sowie durch Zeitmangel ein verfrühter Abbruch durchgeführt wird.

5 Ergebnisse

Es wurden verschiedene Netzarchitekturen und Hyperparameter im Trainingsverlauf getestet, welche zu verschiedenen Ergebnissen führen. Abb. 3 zeigt den Trainingsverlauf für verschiedene Modelle. Dabei fällt auf, dass Modelle mit Hall schlechtere Ergebnisse liefern als Modelle ohne Hall, was auf den Datensatz zurückgeführt werden kann, der ohne Hall aufgenommen wurde. Modelle mit GRU im Decoder, ohne Hall und mit nicht 256 Dimensionen im GRU und Fully-Connected Layer die besten Ergebnisse nach mehr als 10k Steps.

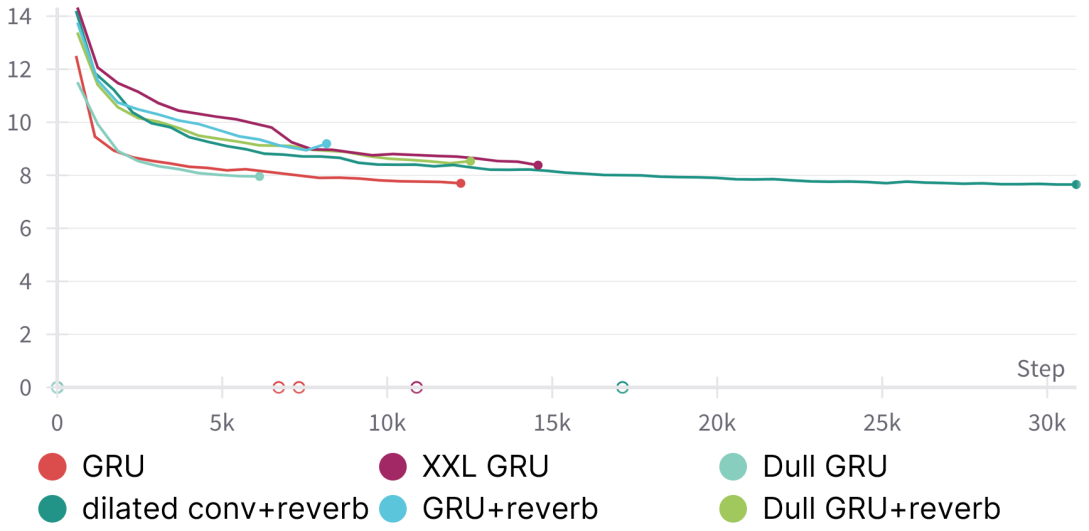

Abbildung 3: Verlustfunktion für verschiedene Modelle in Abhängigkeit der Trainingslänge (Steps). Das Modell GRU wurde abgegeben, es wird kein Hall verwendet. Das XXL GRU hat 512 dimensionen für Fully-Connected Layer und GRU, DULL GRU (+Hall) bezeichnet Modelle mit reduzierter Anzahl an trainierbaren Parametern und das Modell dilated conv+reverb hat eine andere Struktur basierend auf CNNs.

5.1 Metriken

Die Metriken wurden während dem Training nach jeder Epoche für den Validierungsdatensatz berechnet. Dabei haben wir uns für folgende Metriken entschieden.

Spektrogramm-Differenz: Dies ist die Differenz der Spektrogramme zum Quadrat in dB pro Frequenz- und Zeitschritt, welche auch als Verlustfunktion verwendet wird. Es bildet die Ähnlichkeit von zwei Signalen gut in einer Zahl ab und eignet sich deswegen Verlustfunktion. Jedoch ist dies nur ein grober Richtwert für die Analyse von zwei verschiedenen Vorhersagen, da sie nur wenig Aussage über Stärken und Schwächen des Netzes macht.

Gewichteter Lautstärkeunterschied: Hierbei handelt es sich um eine Schätzung für den Lautstärkeunterschied von zwei Signalen für das menschliche Ohr in dB pro Zeitschritt. Dies bildet ab, wie gut die Dynamik des Instruments vom Netz rekonstruiert werden kann z.B. charakteristische Lautstärkekurven des Instruments.

Wasserstein-Distanz: Dies ist eine genauere Metrik für den Klangcharakter der Vorhersage. Es repräsentiert die erforderliche Arbeit, die nötig wäre eine physische Repräsentation des Spektrogramms in das Original umzuwandeln. Hierbei wird die Ähnlichkeiten der Obertöne, also des Klangcharakters, abgebildet. Bei qualitativen Hörversuchen klangen Vorhersagen mit niedrigerer Wasserstein-Distanz authentischer, auch wenn der gewichtete Lautstärkeunterschied höher war.

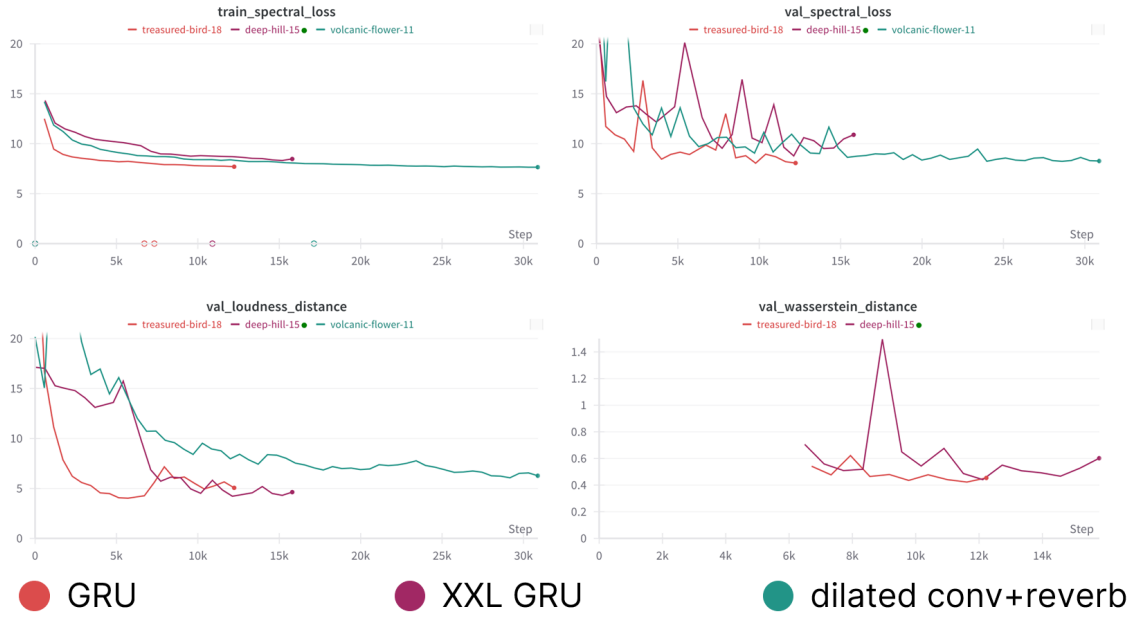

Abbildung 4: Trotz ähnlicher Spektrogramm-Differenz (spectral-loss) zwischen den Modellen, konnten die GRU-Modelle die Dynamik der E-Gitarre besser nachahmen (geringere 'loudness-distance'), besonders 'XXL-GRU'. Dafür konnte das 'GRU'-Modell mit einer etwas genaueren Obertonnachbildung gegenüber 'XXL-GRU' überzeugen.

5.2 MUSHRA-Test

Zusätzlich zu den Metriken wurde ein Multi Stimuli with Hidden Referenze (MUSHRA)-Test durchgeführt. Dieser liefert beim Vergleich von verlustbehafteter Audiokodierung schon mit einer kleiner Anzahl an Testpersonen statistisch signifikante Ergebnisse. Den Testpersonen hören pro Durchlauf vier Versionen einer E-Gitarrenaufnahme mit einer Länge von je 12 s. Eine davon stammt aus dem Testdatensatz, zwei davon wurden vom Modell generiert, wobei eins mit einem Match-Equalizer nachbearbeitet wurde eins von einem sehr kurz trainierten Modell. Letzteres dient als Anker und soll zusammen mit dem Original die Einordnung der vom Modell generierten Audios auf einer Skala von 0 bis 100 ermöglichen. Die Testpersonen kennen die Zuordnung der Audios nicht und die Audios unterliegen keiner bestimmten Reihenfolge. Insgesamt wurden drei Durchläufe pro Person durchgeführt. Es wurden 19 Personen befragt, von denen 12 aktiv mindestens ein Instrument spielen.

In Abb. 5 sind die Ergebnisse dargestellt. Es zeigt sich, dass das Modell mit Nachbearbeitung leicht besser abschneidet als das unbearbeitete Modell. Zudem wurde das vom Modell generierte Audio deutlich besser eingeordnet, als der Anker. Des Weiteren können aktive Musiker:innen das Original und den Anker besser einordnen als die restlichen Testpersonen.

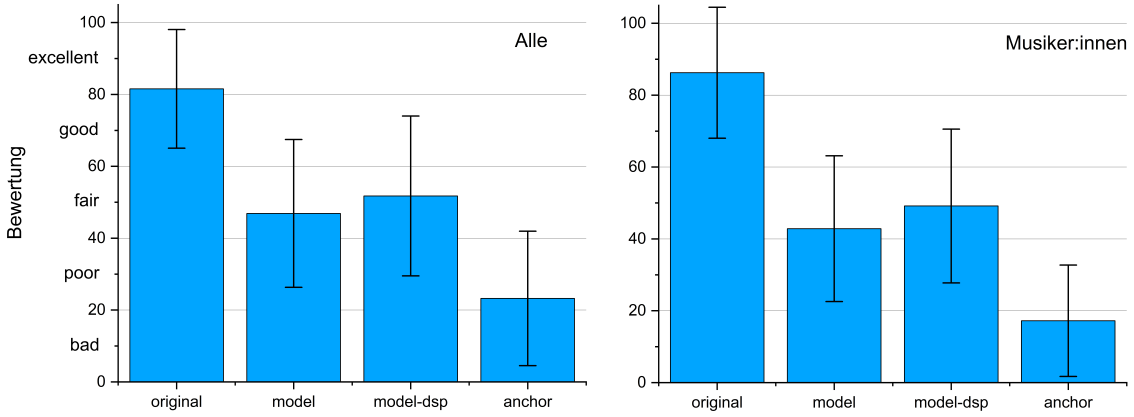

Abbildung 5: Vergleich der Bewertungen aller Testpersonen (links) und der aktiven Musiker:innen (rechts). Das vom Modell generierte Audio wird mit 'model' bezeichnet und das Nachbearbeitete mit 'model-dsp'. Die Fehlerbalken entsprechen der Standardabweichung.

5.3 Vergleich mit anderen Modellen

Ein Vergleich mit „Tone Transfer“ zeigt bei dessen Modell eine bessere Audio-Transformation, d.h. das transformierte Audio entspricht mehr dem trainierten Instrument. Der Ver-

such eine Aufnahme einer Trompete in die E-Gitarre mit unserem Modell zu transformieren erwies sich als weniger erfolgreich im Vergleich zu „Tone Transfer“. Zwar klingt das Audio im Groben nach einer E-Gitarre, weist aber noch Charakteristiken, wie Anschlagsdynamik, von der Trompete auf. Die Instrumentenwahl von „Tone Transfer“ ist dabei ausschlaggebend. Instrumente mit klarem Anschlagscharakter begrenzen die Wahl des Eingabeinstruments auf Instrumente mit ähnlicher Dynamik.

6 Fazit

6.1 Lessons learned

Der verwendete Datensatz ist komplex und beinhaltet verschiedene Spielweisen, wodurch das Training komplizierter wird, da das Modell versucht die Spielweisen zu lernen. Außerdem hat eine E-Gitarre eine spezielle Anschlagsdynamik, wo der Ton nach dem Anschlagen der Seite klingt, aber nur noch gedämpft und nicht verändert werden kann im Gegensatz z.B. zu einer Trompete. Es ist somit wichtig die Eigenschaften des Datensatzes vor dem Training zu kennen und sich über die Einflüsse auf das Training im Klaren zu sein. Die Wahl des Instruments spielt auch eine Rolle, da verschiedene Instrumente durch ihre Klangeigenschaften unterschiedlich gut andere Audios durch Modelle transformieren können.

Die unterschiedlichen Trainingsverläufe sind teilwiese recht kurz und könnten bei längeren Trainingszeiten bessere Ergebnisse liefern. Eine Vermischung des Trainingsdatensatzes mit dem Validierungsdatensatz liegt nicht vor, ist aber durchaus möglich.

6.2 Ausblick

Längere Trainings mit ausgewählten Datensätzen könnten zu besseren Ergebnissen führen, wo durch das Trainieren von mehreren Modellen mit je einem anderen Instrument auch mehrere Intstrumente zur Verfügung stehen könnten.

Für ähnliche Projekte könnte in der Zukunft ein Ansatz mit Transformern bzw. Attention verwendet werden, was bei MusicLM[4] der Fall ist. Dort wird die Generierung von Tokens zur Erzeugung von Audio verwendet.

[4]Mit MusicLM kann Musik mittels einer KI generiert werden: https://musiclm.com/

Literaturverzeichnis

[1] J. Engel, L. Hantrakul, C. Gu, A. Roberts. DDSP: Differentiable Digital Signal Processing, ICLR 2020, Google Research, Brain Team, Mountain View, CA 94043, USA

[2] M.Carneya, C. Lia, E. Toha, N. Zadaa, P. Yu, J. Engela. Tone Transfer: In-Browser Interactive Neural Audio Synthesis (2021), Google Inc., 1600 Amphitheatre Pkwy, Mountain View, CA, 94043, USA

[3] J. W. Kim, J. Salamon, P. Li, J. P. Bello. CREPE: A Convolutional Representation For Pitch Estimation (2018), New York University