Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Projektarbeit Informatik

Deep Learning-basierte Voice Conversion mittels problemspezifischer Loss-Funktion

Autoren	Remo Knaus Stefan Schwizer
Hauptbetreuung	Thilo Stadelmann

Datum	22.12.2017

1 Zusammenfassung

Aktuelle Arbeiten haben aufgezeigt, dass ein direktes Voice Conversion System möglich ist. Es gibt aber zu diesem Zeitpunkt noch keine genauen Informationen über die Funktionsweise dieser Netzwerke. Um einen gesprochenen Satz umzuwandeln, als würde er von einer anderen Person gesprochen, müssen zwei Merkmale stimmen. Erstens der gesprochene Text darf sich in Inhalt nicht verändern. Zweitens die Sprachmerkmale des Satzes müssen mit den Sprachmerkmalen der Zielperson übereinstimmen. Mit diesen beiden Merkmalen kann eine problemspezifische *Loss Function* formuliert werden. Dies im Zusammenhang, dass die beiden Merkmale ebenfalls durch ein Neuronales Netzwerk berechnet werden.

Es werden zwei Subsysteme für die beiden Merkmale evaluiert. Speech Recognition, um den textuellen Inhalt sicherzustellen, und ein Voice Recognition Netzwerk um die Sprachmerkmale zu transferieren.

Das gesamte Voice Conversion System ist mit Tensorflow entwickelt und bietet das direkte Verarbeiten von Audiodateien an. Trainiert ist das Netzwerk mit dem TIMIT Datensatz. Die Performance der beiden Subsysteme ist mit ca. 70% Genauigkeit für das Speech Recognition System und einer minimierten *Loss-Function* vergleichbar mit anderen Implementationen dieser Art. Die Performance des gesamten Netzwerkes ist aufgrund von zeitlichen Problemen noch nicht vorhanden, wird aber Teil der weiterführenden Arbeiten.

2 Vorwort

Bereits während des Frühlingssemester 2017 stiessen wir auf die ausgeschriebene Arbeit «Deep Learning-basierte Voice Conversion mittels problemspezifischer Loss-Funktion» von Thilo Stadelmann (Aufgabenstellung im Anhang). Direkt begeistert und doch etwas eingeschüchtert versuchten wir die Arbeit abzuschätzen. Wir waren uns schnell bewusst, dass diese Arbeit sehr fordernd und kompliziert sein wird. Beide waren sich aber einig diese Herausforderung gerne in Angriff zu nehmen. Nach einigen Treffen mit Thilo Stadelmann und einer kurzen Auszeit über den Sommer hinweg, begannen wir mit dem Projekt.

«Voice Conversion», also gesprochen Text so zu verändern, als würde er von einer anderen Person gesprochen, war das vermerkte Ziel. Unsere Arbeit bestand also darin, den aktuellen Stand der Forschung zu verstehen, mögliche Modellierungsansätze zu erörtern und diese umzusetzen. Bereits der erste Teil, uns beide auf den aktuellen Stand der Forschung in diesem Gebiet zu bringen war sehr aufwändig und zeitintensiv. Es gab öfters eine Beratungsrunde über Begriffe und deren Bedeutung. Ebenso war das Level der zu lesenden Artikel über unserem damaligen Wissenstand, was den Prozess nicht gerade beschleunigte. Wir konnten für die Umsetzung auf bereits bestehenden Code früherer Projekt und Bachelorarbeiten zurückgreifen. Die Idee war klar, alles zu verwenden was uns die Arbeit erleichtern würde, damit schlussendlich mehr Zeit für die spannende Feinarbeit der Lösung besteht.

Mit Thilo Stadelmann als Betreuer haben wir eine sehr kompetente Begleitung für unsere Arbeit. Jemand der stets da war um ein umstrittenen Begriff zu erläutern und auf Fragen einzugehen. Die Sitzungen alle zwei Wochen halfen dem Verständnis immer auf die Sprünge und wir konnten mit neuen Ansätzen und anderen Blickwinkeln weiterarbeiten.

Inhalt

1 Zusammenfassung v
2 Vorwort v
3 Einleitung 1
3.1 Ausgangslage 1
3.2 Zielsetzung 2
3.3 Rahmenbedingungen 2
3.4 Motivation 2
4 Grundlagen 3
4.1 Neurale Netzwerke 3
4.2 Spracheigenschaften 4
4.3 Tensorflow 5
4.4 TIMIT 5
5 Konzept 6
5.1 Übersicht 6
5.2 Speech Recognition 7
5.3 Voice Recognition 9
5.4 Voice Conversion 9
6 Implementation 11
6.1 Speech Recognition 11
6.2 Voice Recognition 14
6.3 Voice Conversion 17
7 Fazit 18
7.1 Speech und Voice Recognition 18
7.2 Einschätzung 18
7.3 Anderer Ansatz 18
8 Ausblick 19
8.1 Weiterführende Arbeit 19
8.2 Reflektion 19
9 Literaturverzeichnis 20
10 Verzeichnisse 22
10.1 Abbildverzeichnis 22
10.2 Formelverzeichnis 22
11 Glossar 23
12 Anhang A: Aufgabenstellung 25

3 Einleitung

Dieses Kapitel beschreibt die Ausgangslage und die Zielsetzung für die Arbeit «Voice Conversion» zusätzlich werden Rahmenbedingungen erläutert die gelten. Es werden Fragen wie: «Welche Arbeit bestehen bereits?», «Wo liegt der Fokus der Arbeit?» beantwortet. Komplexe Begriffe die verwendet werden sind im gesamten Text *kursiv* geschrieben und im Glossar erklärt.

3.1 Ausgangslage

Zum Thema Voice Conversion wird zu jetzigem Zeitpunkt, Stand August 2017, viel geforscht und es gibt laufend neue spannende Artikel. Es werden einige Forschungen erwähnt die für den Denkprozess und die Umsetzung wegweisend waren. Wir fokussieren uns auf den Ansatz mit Neuralen Netzwerken.

Den Beweis, dass Voice Conversion funktioniert, hat wohl «Lyrebird.ai» [1] mit ihrer Demo und den Beispielen eindrücklich geliefert. Leider ist zu diesem Zeitpunkt noch keine wissenschaftliche Arbeit des Forscherteams nach der Veröffentlichung von «Lyrebird» erschienen.

Es gibt viele verschiedene Ideen die Voice Conversion ermöglichen sollen. Es werden unterschiedliche Eingaben und Netzwerkarchitekturen verwendet. Ein Ansatz besteht darin, Phoneme und Spektrogramme zu kombinieren und als Features im Zusammenspiel mit LSTM-RNNs zu verwenden [2], andere verwenden Convolutional Neural Networks und als Input zusätzliche zu der Sprache, das Klangbild des Sprechers [3].

Eines der grossen Probleme für die Umsetzung eines Voice Conversion Systems, ist die Bewertung des Resultats. Wie gut beziehungsweise wie nahe ist die generierte Ausgabe an der menschlichen Sprache. Es stellt sich mathematisch als sehr schwierig und umstritten heraus was ähnliche Stimmen sind und welche Parameter unsere Stimme eindeutig machen. [4] [5] [6]

Speech Recognition

Speech Recognition ist das Erkennen eines gesprochenen Wortes bzw. Satzes. An solchen Systemen wird schon seit Jahrzehnten gearbeitet. Bei der Spracherkennung wird zwischen sprecherabhängiger und sprecherunabhängiger Erkennung unterschieden. Mit einem sprecherabhängigen System erzielt man bessere Resultate, muss es jedoch erst auf die Stimme des Benutzers trainieren. In den meistverwendeten Implementierungen, wird der sprecherunabhängige Ansatz verwendet, da er besser generalisiert werden kann. Es gibt aber durchaus Systeme, die man zusätzlich auf die Stimme trainieren kann um diese zu verbessern.

Wir haben zwei verschiedene Open-Source Systeme ausprobiert. Die zwei Kandidaten sind Kaldi[7] und Deep Speech[8], beide Ansätze beruhen auf *Deep Learning* und sind mit Tensorflow[9] implementiert. Nach dem auswerten der Tests liefert Kaldi die besseren Resultate und auch kürzere Trainingszeiten als Deep Speech, somit ist Kaldi der erste Favorit.

Voice Recognition

Unter Voice Recognition versteht man das Erkennen einer Stimme. Das bedeutet, das System unterscheidet von welchem Sprecher eine Audiodatei stammt. Solche Ansätze sollen auch bei digitalen Assistenten eingearbeitet werden, um zu erkennen wer aktuell mit dem Assistenten spricht [10].

Für das Erkennen der Stimme, konnte auf die Software einer anderen Projektarbeit der ZHAW zurückgegriffen werden «Machine Learning for Speaker Clustering» [11]. Andere Kandidaten wären: «Learning embeddings for speaker Clustering» [12], «Speaker Clustering mit Metric Embeddings» [13], «Speaker Identification mit Reccurrent Neural Networks» [14]. Die Wahl fiel auf das Projekt «Machine Learning for Speaker Clustering», weil es die besten Resultate erzielte und da es neben dem *Speaker Clustering* auch einfach erlaubt ein *Embedding* des Sprechers, mit dessen vermeintlichen Features zu extrahieren.

3.2 Zielsetzung

Das Ziel ist mit Hilfe einer kombinierten *Loss-Function* aus Speech Recognition und Voice Recognition ein Neurales Netzwerk bauen, dass die identifizierenden Eigenschaften von gesprochenem Text so modifiziert, als spreche die Zielperson den Text. Das Kombinieren der *Loss-Function* ist notwendig um genau beschreiben zu können was das Ziel ist, nämlich der Text bleibt derselbe und die Stimme ändert sich auf die Zielstimme. Die beiden Subsysteme Speech Recognition und Voice Recognition werden so weit als mögliche nicht verändert. Als Voice Recognition System stehen mögliche Kandidaten aus früheren Projektarbeiten der ZHAW zur Verfügung. Das Speech Recognition System soll noch evaluiert werden.

3.3 Rahmenbedingungen

Das Projekt wird mit sechs ETCS Punkten bewertet, somit sind mindestens 180 Arbeitsstunden für das Projekt reserviert. Weitere Eingrenzungen seitens der ZHAW sind nicht gegeben. Für das trainieren der Netze stand ein GPU *Cluster* mit zwei Mal je acht GPU «Nvidia Titan» [15] zur Verfügung.

Es gab keine direkten technischen Rahmenbedingungen seitens der ZHAW, für die Umsetzung des Projekts. Trotzdem sind durch die verwendeten Subsysteme, ein grober Rahmen gegeben.

Abgabetermin für die Projektarbeit ist der 22.12.2017.

3.4 Motivation

In unserer heutigen Zeit, werden digitale Assistenten immer wichtiger und brauchbarer. Zunehmend mehr Leute haben zu Hause einen digitalen Assistenten wie: «Google Assistant» [16], Apples «Siri» [17], Microsofts «Cortana» [18] oder Amazons «Alexa». Doch so richtiges Vertrauen mag bei den synthetischen Stimmen nicht aufkommen. Was wäre nun, wenn Zuhause nicht «Siri», sondern die Lieblingsschauspielerin oder der Lieblingsschauspieler zu einem spricht? Dies ist nur eine von vielen Anwendungsmöglichkeiten eines guten Voice Conversion Systems. Zudem hilft es auch im Verständnis unserer eigenen Sprache um weitere Anwendungen, wie Simultanübersetzer und vieles weiteres zu entwickeln

Das «Voice Conversion» System wäre ein nächster Schritt, in der Erfolgsgeschichte von Neuralen Netzwerken bzw. *Maschine Learning* im Allgemeinen. Es wäre der Beweis, dass so etwas komplexes, wie die Sprache verarbeitet, verstanden und erzeugt werden kann.

4 Grundlagen

Es werden alle wichtigen Grundlagen für das Verständnis dieser Arbeit gegeben. Leser die bereits über ein gutes Vorwissen im Bereich Neurale Netzwerke und Spracheigenschaften besitzen können es überspringen.

4.1 Neurale Netzwerke

Ein Neurales Netzwerk besteht grundsätzlich aus *Neuronen*, die wie ein Graph miteinander verbunden sind. Der Aufbau ist an das Gehirn angelegt siehe Abb. 1. Es wird verwendet um ein Problem, ohne spezifizierten Code für das Problem, zu lösen. Eingaben fliessen durch das Netzwerk und werden durch die *Neuronen* verarbeitet. Die Ausgabe löst die gestellte Aufgabe.

Neurale Netzwerke werden oft für Aufgaben verwendet, die ein Mensch intuitiv richtigmacht und es schwierig ist Regeln, für das Problem aufzustellen. Ein gutes Beispiel ist das Erkennen von Objekten in einem Bild. Für einen Menschen ist das identifizieren eines anderen Menschen in einem Bild keine Schwierigkeit, doch einem Computer diese Kompetenz beizubringen, ist sehr schwierig. Hier schaffen Neurale Netzwerke Abhilfe, da sie vermeintlich lernen wie ein Mensch. Deshalb werden sie oft in Gebieten der Bildverarbeitung, Sprachanalyse und sozialen Netzwerken verwendet.

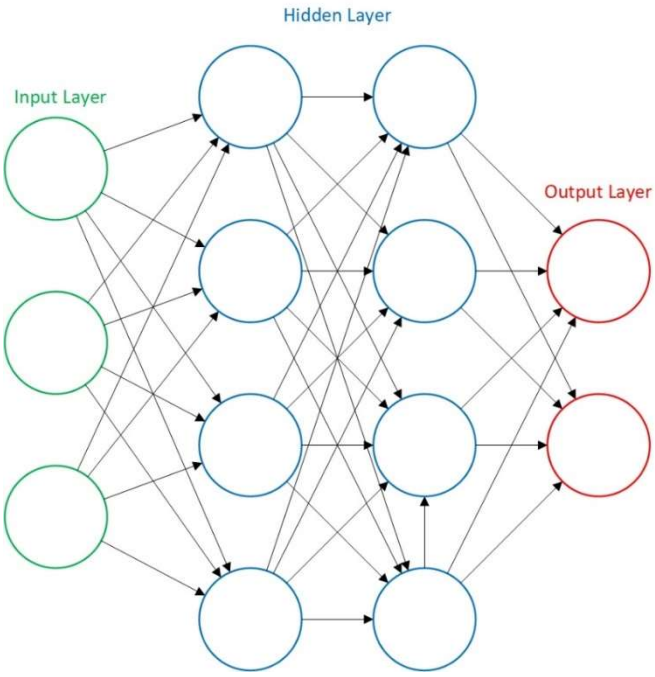


Abb. 1: Schema Neurales Netzwerk

Normalerweise gibt es zwei Phasen für ein Netzwerk, die Trainings- und die Evaluierungs-Phase. Also eine, wo das Netzwerk trainiert wird, die ihm gestellte Aufgabe zu «verstehen» und sie zu abstrahieren. Dies geschieht beim «*supervised learning*» [19] durch Trainingsdaten, die Aufgabe und Resultat beinhalten. Das Netzwerk lässt die Trainingsdaten durch die *Neuronen* fliessen und analysiert die Ausgabe. Danach werden die Gewichte innerhalb des Netzwerkes angepasst um dem gewünschten Resultat näher zu kommen. Für das berechnen der neuen Gewichte wird üblicherweise ein *Gradientenabstiegsverfahren* [20] verwendet.

Um das *Gradientenabstiegsverfahren* verwendet zu können, benötigen wir eine Funktion die das Resultat bewertet, diese nennet man «*Loss-Function*». Diese Funktion muss mathematisch differenzierbar sein, damit der Abstieg effizient durchführbar ist. Wenn die Loss-Funktion, hier auf drei Dimensionen vereinfacht, aufzeichnen können wir einen Abstieg und somit Verbesserung der Leistung des Netzwerkes erzielen. Zu sehen in Abb. 2. wie sich das Netzwerk der roten Linie entlang einen Weg zum tiefsten Punkt sucht, Epoche für Epoche.

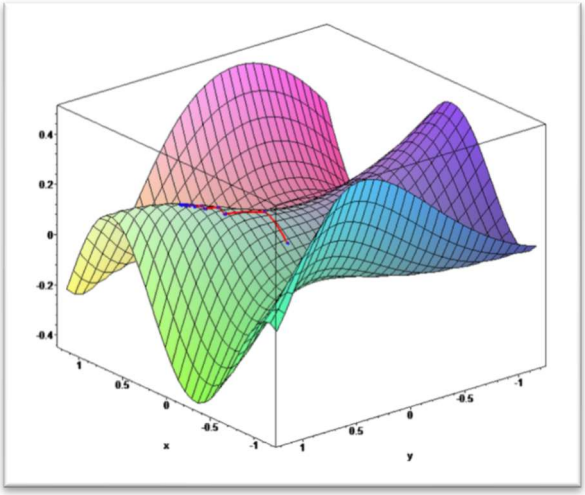


Abb. 2: Gradientenabstieg in 3 Dimensionen *[39]*

4.1.1 Convolutional Neural Networks

Ein Convolutional Neural Network kurz CNN ist eine Spezialform eines Neuronalen Netzes und besteht aus einem Inputlayer, einem Outputlayer und vielen *hidden layers*. Die versteckten Schichten, bestehen meist aus *convolutional*, *pooling* oder *fully connected layer*. Eine spezifischere Erklärung der Layer ist im Glossar vorhanden. Die *convolutional* und *pooling layer* ersetzen die dense layer eines normalen Neuralen Netzwerks. Es werden mehrere Featuremaps erstellt die alle unterschiedliche Gewichtung enthalten und nur mit einem kleinen Bereich des Inputs verbunden sind. Dies ermöglicht das Erkennen von Strukturen egal wo auf dem Bild sich diese befinden. In Abb. 3 wird der Roboterarm über eine Featuremap im CNN erkannt.

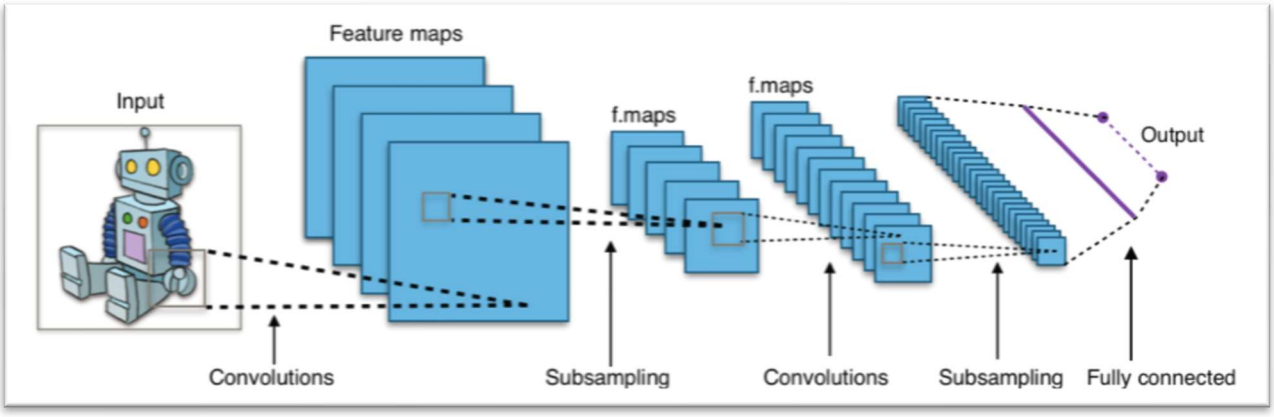


Abb. 3: CNN erkennt Roboterarm [21]

4.2 Spracheigenschaften

Um die menschliche Sprache für einen Computer verfügbar zu machen, werden Spektrogramme verwendet. Diese sind Diagramme der gesprochenen Sprache in Zeit, Frequenz und Lautstärke. Eine spezielle Version, die Mel-Spektrogramme, bewegen sich nur im Bereich, der für einen Menschen hörbar ist, zu sehen in Abb. 4.

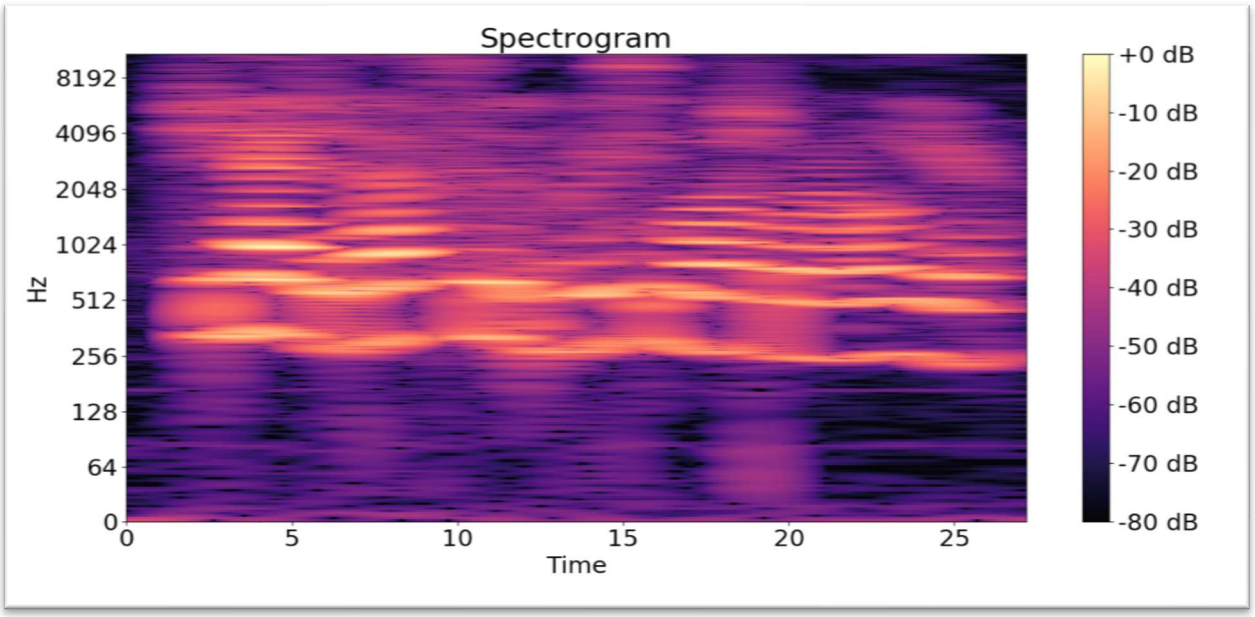


Abb. 4: Mel-Spektrogramm

Um aus der Sprache auf Spektrogramme zu kommen werden folgende Schritte absolviert:

1. Unterteilen des Sprachsignals in Fenster
2. Anwendung einer Diskreten Fourier Transformation
3. Logarithmieren des erstellten Betragsspektrums
4. Zusammenfassung auf von Menschen hörbaren Frequenzbandes
5. Dekorrelation mit einer diskreten Kosinus-Transformation

Die Frequenzachse wird auf die Mel Skala [22] abgebildet: Formel 1.

$$m(f) = 2595 * \log_{10}(1 + f/700)$$

Formel 1: Berechnung der Frequenzachse im Mel Spektrogramm

Eine weitere Methode die Sprache zu digitalisieren, ist sie in Phoneme aufzuteilen. Phoneme sind der atomare Baustein einer Sprache. Im englischen gibt es 44 Phoneme [23] die unterschieden werden.

4.3 Tensorflow

Tensorflow [9] ist ein *Open-Source Machine Learning Framework*, welches von Google entwickelt wurde. Daten werden als *Tensoren* dargestellt, fliessen durch die verschiedenen *layer* eines Netzwerkes und gelangen am Ende zur Ausgangschicht. Tensorflow findet vor allem in Google Produkten, wie z.B. Gmail, Google Fotos, oder Street View Verwendung. Durch die *Open-Source* Lizenz wir das *Framework* vermehrt in der Forschung und von privaten Personen für eigene Projekte verwendet.
Um die komplexen Vorgänge eines *Neural Network* zu veranschaulichen, gibt es Tensorboard [24]. Auf dieser Weboberfläche kann man den Graphen des Netzes, den Fortschritt des Trainings, die Entwicklung verschiedener Messwerte (Loss) und Daten, welche durch das Netz geflossen und verändert wurden, betrachten.

4.4 TIMIT

TIMIT [25] ist eine Audio Datensatz, welcher auf der Arbeit vom Massachusetts Institute of Technology(MIT) und Texas Instruments(TI) beruht. Die Aufnahmen bestehen aus 630 verschiedenen Sprechern, welche acht weitverbreiteten Akzente der englischen Sprache repräsentieren. Jeder Sprecher hat zehn Sätze, welche viele unterschiedliche Phoneme besitzen, eingesprochen. Die Daten sind mit einer *Samplerate* von 16'000 Hertz und als WAV-Dateityp vorhanden. Diese Audiodateien besitzen keine Header, welche zum Abspielen in normalen Musik Programmen notwendig sind. Zusätzlich zu den Audiodateien sind die dazugehörigen Transkripte sowie die vorkommenden Phoneme vorhanden.

5 Konzept

Um ein perfektes Voice Conversion System zu erreichen, muss eine Äusserung mit der Stimme vom Sprecher A auf die gleiche Äusserung mit der Stimme von Sprecher B abgebildet werden. Wobei die Stimme A und B unterschiedlich sein müssen und das ohne diese Äusserung von Sprecher B gemacht wurde.

5.1 Übersicht

Die Übersicht Abb. 5 soll den Aufbau des kompletten Systems aufzeigen und verständlicher machen.

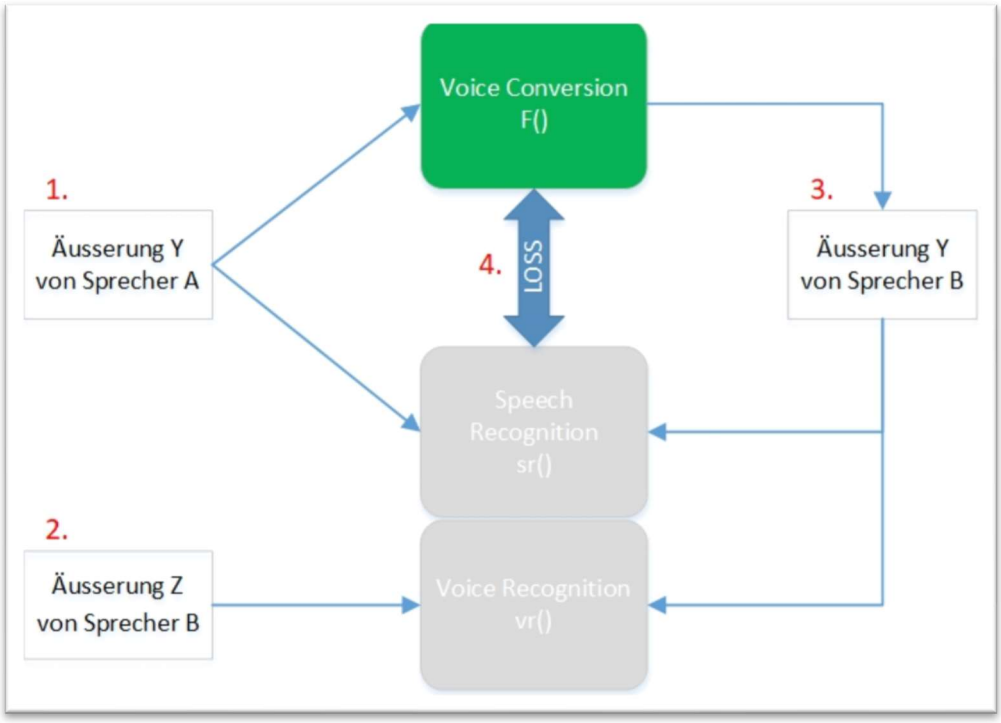


Abb. 5: Übersicht Konzept

Die Architektur besteht aus drei separaten *Neural Networks*: Speech Recognition, Voice Recognition und dem neuen Voice Conversion.

1. Als Input hat man eine Äusserung Y mit der Stimme von Sprecher A, welche in das Voice Conversion sowie das Speech Recognition Netz gespeist wird.
2. Eine Äusserung Z gesprochen von der Zielperson B wird dem Voice Recognition Netzwerk zum Vergleichen der Spracheigenschaften mitgegeben.
3. Daraufhin gibt das Voice Conversion Netz eine Äusserung Y mit der Stimme von Sprecher B aus.
4. Mit dieser neuen Kombination werden wiederum die zwei Subsysteme gespiesen und geben anschliessend die Abweichung der Äusserung sowie der Stimme aus.
5. Mit Hilfe dieser Abweichungen wird eine *Loss-Function* gebildet und schlussendlich das Voice Conversion Netzwerk trainiert.

5.2 Speech Recognition

Das Speech Recognition Netz wird für die *Loss-Function* im Voice Conversion Netz benötigt. Es soll sicherstellen, dass die Äusserung Y nach der «Conversion» immer noch die Aussage Y beinhaltet. Für diese Aufgaben gibt es mehrere mögliche Kandidaten die zu evaluieren wären. Ein guter Überblick verschafft die Arbeit «A Survey on Voice Conversion using Deep Learning» [26]. In Abb. 6 ist das Konzept als Blackbox ersichtlich.

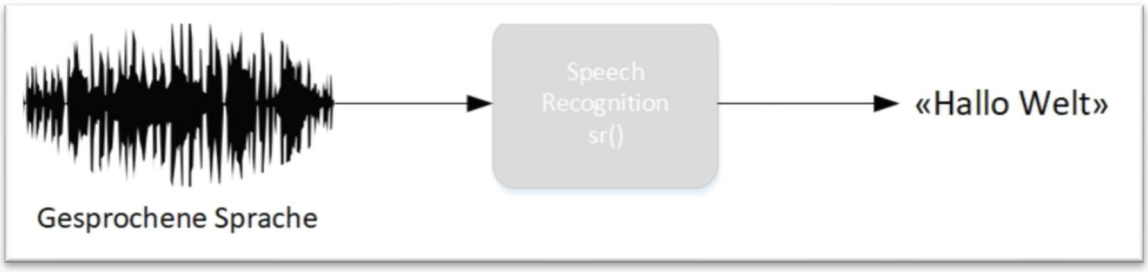


Abb. 6: Konzept Speech Recognition Kaldi

Die erste getestete Variante ist Kaldi [7]. Dies ist ein *Open-Source* Speech Recognition System basierend auf endlichen *Transduktoren*. Es bietet eine umfassende Lösung für das Übersetzen von Sprache auf Text. Kaldi bietet sehr gute Resultate mit geringem Aufwand und ist auch sehr performant [27].

Die eigenen Versuche mit Kaldi, wurden mit dem Unterprojekt «*aspire*» [28] unternommen und waren sehr vielversprechend. Die Beispiele die in einem Feldversuch gemacht wurden, wurden alle richtig und fehlerfrei übersetzt, auch wenn die Aufnahmen nicht dieselbe Qualität besassen, wie jene von TIMIT. Wie in Abb. 7 zu sehen, verwendet das «aspire»-Modell drei Stufen für die Übersetzung in Text. Als erstes werden die akustischen Eigenschaften aus dem Audio Input extrahiert, danach werden die akustischen Eigenschaften mit einem Wörterbuch verglichen und schlussendlich in einen Satz überführt.

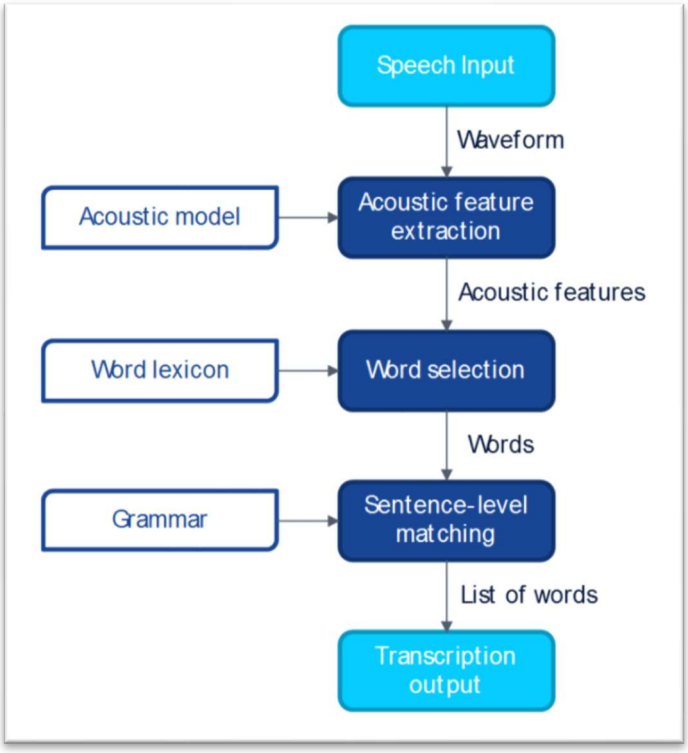


Abb. 7: Aspire Model mit Kaldi [28]

Problem Kompatibilität

Um nun aber dieses «aspire»-Modell weiter verwendet zu können, muss es auch ausserhalb von Kaldi lauffähig sein. Damit das «aspire»-Model verwendet werden kann, muss es in Tensorflow importierbar sein. Es gibt einige Dokumentationen betreffend der Integration in Tensorflow [29], leider sind keine wirklich vollständig und getestet.

Problem Differenzierbarkeit

Das Speech Recognition Netz hat das Ziel Text aus einer Aussage zu extrahieren, dies kann auf mehrere Arten geschehen. Die erste Methode ist bereits beschrieben in Abb.6, mit Text als Ausgabe. Text hat viele Vorteile und anderem, dass er direkt lesbar und verständlich ist. Leider hat er auch einen grossen Nachteil, er ist schwer zu vergleichen. Damit die verglichenen Texte auch in der *Loss-Function* verwendet werden können muss die gesamte mathematische Berechnung differenzierbar sein.
Nur wenn der gesamte Rechenweg ableitbar ist, kann Tensorflow durch «*Backpropagation*» und Gradientenabstiegsverfahren effizient trainiert werden [30]. Es wird also ein Verfahren benötigt, dass Texte vergleichen kann, aber trotzdem ableitbar ist. Ein sehr populärer und machtvoller Algorithmus ist die Levenshtein Distanz.

Das Problem an der Levenshtein Distanz Funktion ist, dass es sich um Text und nicht um Zahlen handelt. Damit eine Funktion ableitbar ist, muss sie mindestens komplexe Zahlen auf komplexen Zahlen abbilden. Deshalb kann die Levenshtein Distanz nicht effizient verwendet werden. Die mathematische Begründung in Formel 2 erklärt, dass alle Funktionen die von x nach y zeigen, wobei x und y Elemente der komplexen Zahlen sind, ableitbar sein können. Da es sich bei der Levenshtein Distanz aber um Buchstaben und Wörter handelt kann nicht von Ableitung gesprochen werden.

$$(x, y) \notin \mathbb{C} \Longrightarrow f \notin \{Abl.\ Funktionen\}, \qquad mit\ f: x \mapsto y$$

Formel 2: Teilbedingung Ableitbarkeit

Eine Alternative ist als Ausgabe nicht direkt ein Text zu verwenden, sondern eine Phonem-Verteilung. Diese hat den Vorteil, dass sie ein Vektor ist, somit kann mit der Kosinus-Ähnlichkeit eine Abweichung zwischen zwei Vektoren berechnet werden. Daraus ergibt sich folgender schematischer Aufbau des Speech Recognition Netzes in Abb.8. Die Schlussfolgerung daraus ist, dass Kaldi als Speech Recognition Netzwerk nicht verwendet werden kann. Eine öffentlich verfügbare Lösung die Phonem Verteilungen verwendet ist «deep voice conversion» [31]. Wir verwenden die Lösung der Phonem Verteilung.

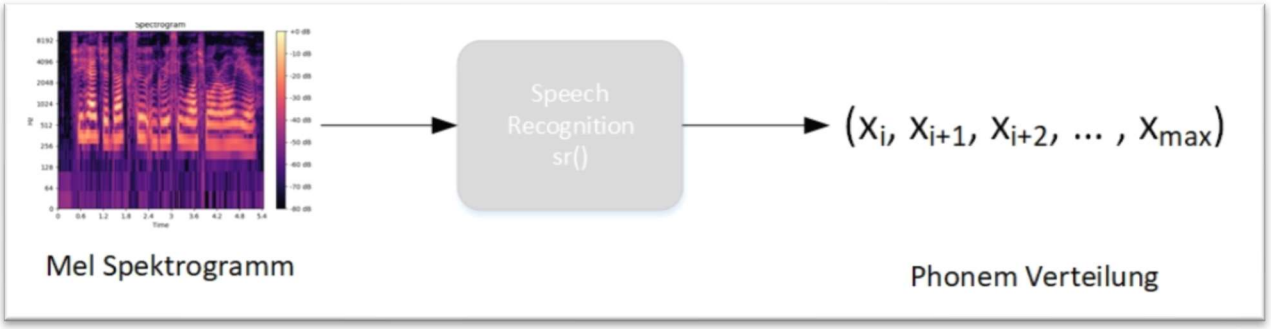


Abb. 8: Konzept Speech Recognition mit Phonem Verteilung

5.3 Voice Recognition

Das Voice Recognition System wird für die *Loss-Function* im Voice Conversion Netz benötigt. Das System stellt sicher, dass die Ausgabe aus dem Voice Conversion System, dieselben Spracheigenschaften besitzt, wie die des Zielsprechers. Für dieses System standen mehrere Kandidaten zur Verfügung, welche hauptsächlich auf *Speaker Clustering* bzw. Identification trainiert sind. Es wird also versucht zu bestimmen welcher Sprecher eine bestimmte Aussage gemacht hat. Je nach Trainingsdaten sind nur wenige Sprecher vorhanden und diese Systeme berechnen, welcher Sprecher am wahrscheinlichsten die Quelle der Aussage ist. Bei wenigen und sehr ähnlich klingenden Sprechern wird das ungenau werden. Jedoch bietet die Arbeit: «Machine Learning for Speaker Clustering» [11], die Möglichkeit auf einer früheren Schicht das *Embedding* der Stimme zu extrahieren. Somit kann man, diese Stimmeigenschaften direkt miteinander vergleichen und dadurch die Präzision verbessern.

Als Eingabe benötigt das Voice Recognition Netz ein Spektrogramm siehe Abb. 9. Aus diesem werden die Spracheigenschaften extrahiert und in ein *Embedding* gepackt.

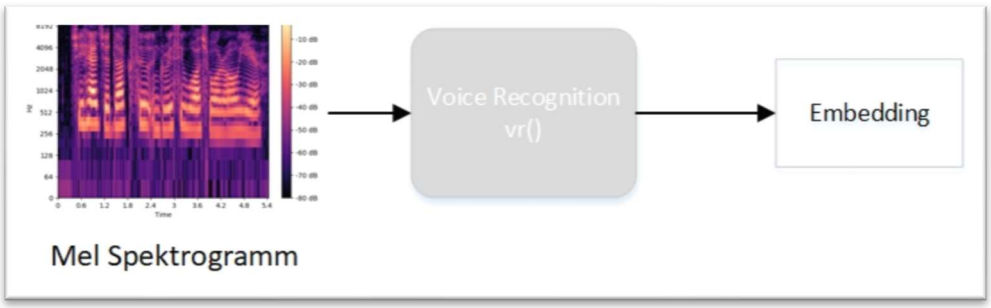


Abb. 9: Konzept Voice Recognition

5.4 Voice Conversion

Mit dem Speech Recognition System kann man nun die Äusserung, und mit dem Voice Recognition Netz den Sprecher vergleichen. Somit lässt sich die Ausgabe vom Voice Conversion System, auf die zwei elementaren Aspekte überprüfen und dadurch das Netz auf den gewünschten Effekt trainieren. Die daraus resultierende *Loss-Function* sieht man in Formel 3.

$$\alpha * \left(d_{\cos}\left(sr(x), sr\left(F(x,\theta)\right)\right)\right)^2 + \beta * \left(d_{\cos}\left(vr(x), vr\left(F(x,\theta)\right)\right)\right)^2$$

Formel 3: kombinierte Loss-Function des Voice Conversion Systems

Der linke Summand vergleicht die Äusserung und der rechte den Sprecher. Die Daten werden durch die Kosinus-Ähnlichkeit [32] verglichen. Welche eine im Machine Learning oft verwendete Möglichkeit ist, Vektoren miteinander zu vergleichen. Dies ist ein erster Ansatz, welcher in Experimenten weiter getestet werden muss. Es kommen auch andere Vergleichsvarianten in Frage, z.B. euklidischer Abstand. Entscheidend welche Art verwendet wird ist, wie gut die *Loss-Function* gegen Null konvergiert beziehungsweise wie gut die Ergebnisse klingen. Zusätzlich werden die beiden Summanden quadriert, um Fehler besonders stark zu bestrafen. Die beiden Parameter α und β erlauben es, einen Summanden stärker zu gewichten. Die genauen Werte müssen in Experimenten evaluiert werden. Initialisiert werden beide mit eins.

Netzaufbau

In einem ersten Ansatz wird das Voice Conversion Netz, ähnlich wie bei der Arbeit «Machine Learning for Speaker Clustering» [11], aufgebaut. Jedoch müssen die Ausgabe- sowie die Eingabegrösse der Elemente dieselben Dimensionen aufweisen. Um dies zu bewerkstelligen wird auf ein fully convolutional network (FCN) gesetzt. Die Abb.10 liefert eine schematische Darstellung eines FCN.

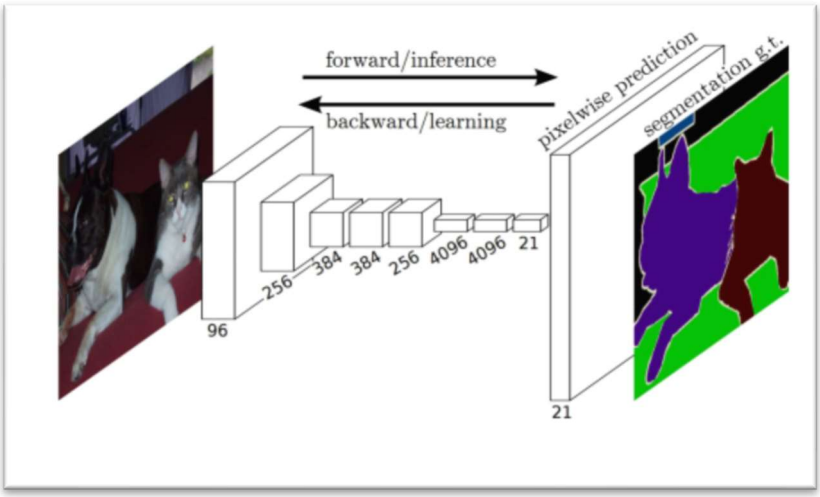


Abb. 10: Schema «fully convolutional network» adaptiert von [33]

Die Architektur eines FCN erinnert stark an ein CNN, bei dem Eigenschaften extrahiert und in sogenannten Featuremaps abgebildet werden. Um diese Features wieder in ein Spektrogramm zu bringen, werden am Ende, ein oder mehrere *fully connected layer* angehängt. Diese Architektur soll in Experimenten evaluiert und verbessert werden.

Eine Alternative, wäre ein Deconvolutional Network[34] in Abb. 11. Bei diesem werden nicht nur *fully connected layer*, sondern ein ähnlicher Aufbau wie eines CNNs nur in umgekehrter Richtung verwendet zusätzlich zu einem standardmässigen CNN. Featuremaps werden in unpooling-layer aufgelöst und somit ergibt sich langsam die richtige Dimension und hoffentlich ein gutes Ergebnis.

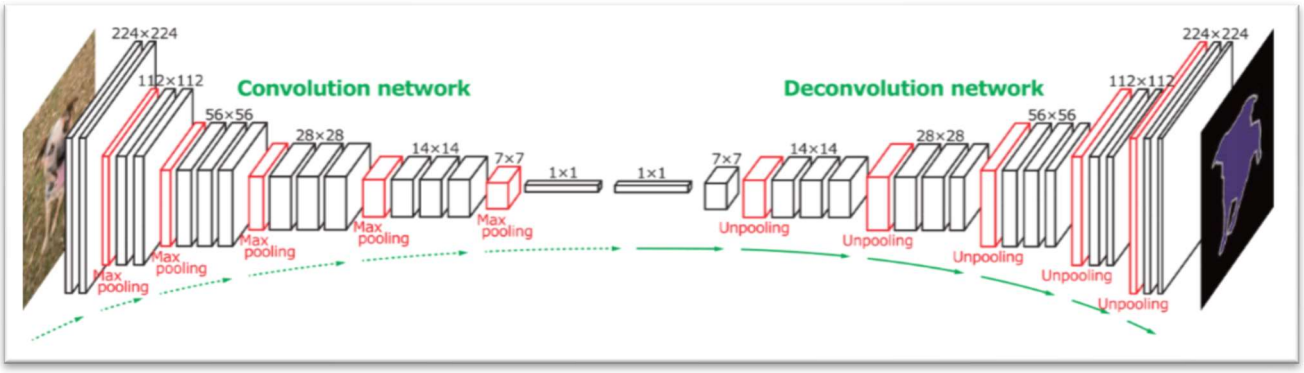


Abb. 11: Deconvolutional Network [33]

6 Implementation

Während der Implementation sind viele Kleinigkeiten und Probleme aufgetreten, einigen werden im folgenden Kapitel Rechnung getragen. Es wird auf den Aufbau und die genaue Spezifikation der Netze eingegangen und deren Ergebnisse so weit als möglich eingeschätzt.

6.1 Speech Recognition

6.1.1 Lösungsarchitektur

Für den Speech Recognition Part haben wurde gemäss Kapitel 5.2, die Lösung «deep voice conversion» ausgewählt [31]. Die Lösung von Dabi Ahn und Kyubyong Park ist eine vollumfängliche Voice Conversion Lösung ohne parallele Audiodatensätze der Zielperson. Sie verwenden zwei nacheinander angeordnete Netze. Auszug aus der Github Webseite in Abb. 12.

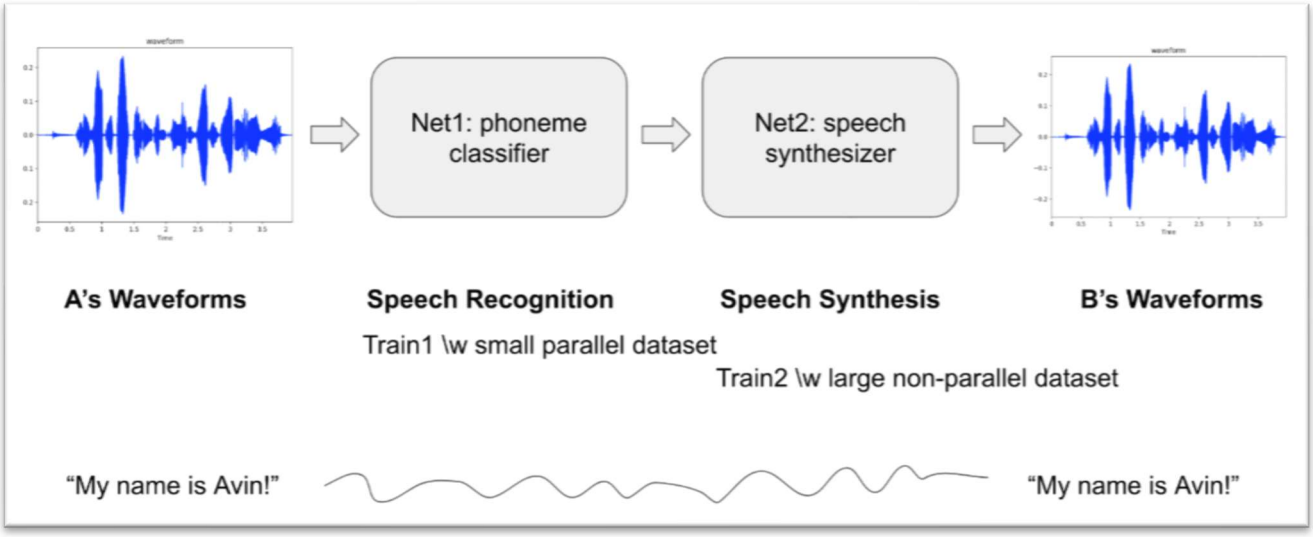


Abb. 12: Modell "deep voice conversion"[31]

Ihre Arbeit ist sehr interessant und funktioniert auch bereits erstaunlich gut. Für das Speech Recognition Netz haben wir Net1, den «phoneme classifier» adaptiert. Net1 wird mit den TIMIT Datensätzen trainiert und erzeugt eine Phonem Verteilung als Ausgabe.

6.1.2 Schwierigkeiten & Stolpersteine

Während der Implementation sind noch einige Schwierigkeiten aufgetreten, die es erschwerten das Netz zu trainieren. Am wichtigsten ist, dass die Voraussetzungen für das Netzwerk gegeben sind, das heisst alle Softwarepakete sind installiert und die Versionen stimmen genau. Es werden «requirements» vorgegeben, leider sind sie nicht vollständig. Dabi Ahn und Kyubyong Park haben einen sehr aufgeräumten und gut verständlichen Code geschrieben, jedoch haben Sie nicht angegeben mit welcher Linux/Windows Version sie gearbeitet haben. Es kommen zum Beispiel noch fehlende Audio Codecs zu den «reqirements» hinzu. Die entsprechende Fehlermeldung sieht folgendermassen aus:

```
>>> import librosa
>>> a = librosa.load('SA1.WAV',sr=16000)

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\librosa\core\audio.py", line 107, in load
    with audioread.audio_open(os.path.realpath(path)) as input_file:
  File "C:\User\__init__.py", line 116, in audio_open
    raise NoBackendError()
    audioread.NoBackendError
```

Diese Meldungen sieht man leider während des Trainingsprozesses nicht. Die Auswirkung ist lediglich, dass alle Batches in der Batchqueue leer sind. Sofern die Batchqueue leer ist, kommt das oft wegen fehlerhafter Verarbeitung der Eingabedaten. Das Programm beendet mit einem Fehler das die Batchqueue leer ist.

Um nun die benötigten Codecs zu installieren gibt es mehrere Varianten, eine ist das Modul «ffmpeg» zu installieren. Mit «ffmpeg» werden im Hintergrund einige Audio Codecs installiert, darunter auch derjenige der benötigt wird. Hier der nötige Befehl für eine Anacoda Umgebung:

```
conda install -c menpo ffmpeg
```

6.1.3 Parameter

Das Speech Recognition Netzwerk verwendet übergreifende Parameter die sehr essentiell für die Funktionsweise des Netzwerkes sind. Die wichtigsten dieser Parameter sind hier mit einer Erklärung aufgelistet:
Auszug aus dem Parameterfile «.\VoiceConversion\src\speech_recognition\hparams.py»

sr = 16000 Die *Samplerate* ist global gegeben durch die Inputdaten, sie wiederspiegelt wie hoch die Abtastrate der Aufnahme war.

frame_shift = 0.005 Der «frame_shift» wird für die Berechnung der hop_length verwendet. Er muss durch die ausführende Person definiert werden.

frame_length = 0.025 Der «frame_length» wird für die Berechnung der win_length verwendet. Er muss durch die ausführende Person definiert werden.

n_fft = 512 «n_fft» beschreibt die «window length» der verwendeten Fourier Transformation. 512 Samples werden pro Berechnung beachtet.

hop_length = int(sr*frame_shift) Ist die Anzahl ausgelassener Frames zwischen den verschiedenen Windows.

win_length = int(sr*frame_length) Ist die effektive «window length» und wird mit Nullen auf die spezifizierte Grösse von «n_fft» aufgefüllt.

preemphasis = 0.97 Die Präemphase wird verwendet um hohe Frequenzen anzuheben und tiefe abzusenken. Dieser Prozess wird nach der Verarbeitung wieder Rückgängig gemacht. Die Präemphase soll das Rauschen reduzieren.

n_mfcc = 40 Anzahl berechnete Punkte pro Frame, wiederspiegelt die Genauigkeit der Übersetzung von Audio zu einem Spektrogramm.

n_mels = 80 Anzahl der Mel-Bänder die generiert werden.

6.1.4 Evaluierung

Sofern alle Pfade auf die Eingabedaten stimmen kann das Training gestartet werden. Dem Trainingsaufruf wird noch zusätzlich ein Parameter mitgegeben, welcher bestimmt wie das Experiment heisst. Damit können zu einem späteren Zeitpunkt alle Checkpoint Daten einfach wiedergefunden werden.

```
python .\VoiceConversion\src\speech_recognition\train1.py [case]
```

Während des Trainings auf dem *Cluster* mit einer GPU und 16 GB Arbeitsspeicher dauert das Training ca. einen Tag, danach ist das Netzwerk ausreichend trainiert. Nach ca. 90 Epochen wird das Resultat nicht mehr relevant besser. Wir erreichen also wie in Abb. 13 zu sehen, einen minimalen Loss Wert von 0.81.

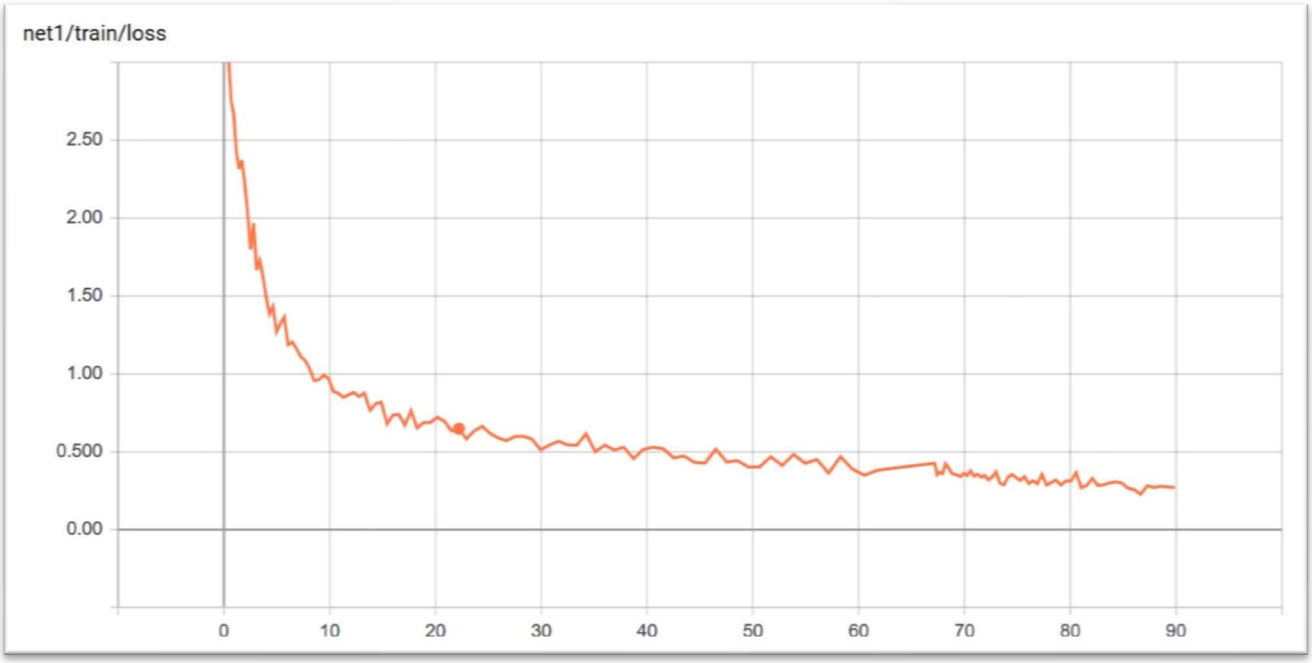
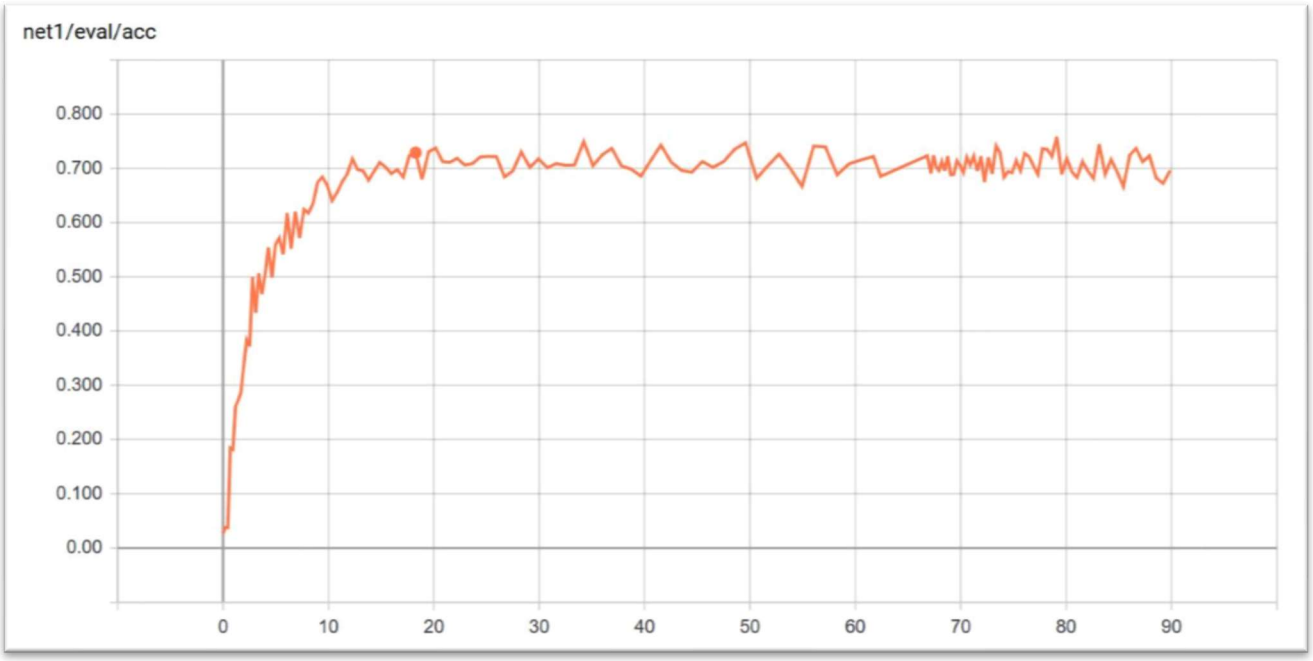


Abb. 13: Train-Loss Speech Recognition Netz

Dies sagt noch nicht viel über die Funktionstüchtigkeit des Netzwerkes aus aber es bedeutet zumindest, dass die Implementation funktioniert. Ein interessanter Punkt dabei ist, dass die Epochen kontinuierlich länger dauern und das ohne ersichtlichen Grund. Dies ist ein Problem das man in einer weiteren Arbeit beachten bzw. analysieren müsste. Die Genauigkeit des Netzwerks verhält sich gemäss Abb. 14.



Wir erreichen eine Genauigkeit von 70%. Diese Zahl scheint auf den ersten Blick relativ ernüchternd aus, doch beim Erkennen von Phonemen ist dieser Wert mehr als genügend. Im Vergleich, ein System basierend auf einer «Supported Vector Machine» erreicht in ähnlicher Umgebung 71,4% Genauigkeit [35] oder ein Ansatz mit Naive Bayes erreicht 61.59% [36].

Das Speech Recognition Netzwerk ist fertig trainiert und kann nun im Voice Conversion Netzwerk als Teil der *Loss-Function* verwendet werden.

6.2 Voice Recognition

6.2.1 Lösungsarchitektur

Die Lösungsarchitektur Abb. 15 ist sehr nahe an der beschriebenen Konzeption aus Kapitel 5.3. Es wird das Netzwerk von Patrick Gerber und Sebastian Glinski-Haefeli verwendet aus ihrer Bachelorarbeit 2017. Wir geben trotzdem einen kurzen Überblick über die verwendete Architektur.

Auszug aus Machine Learning for *Speaker Clustering* Kapitel 3.1 Seite 10[11]:

(Refferenzen wurden angepasst)
«Für die LSTM Netzwerk Architektur wurde entschieden, sich an dem BLSTM Netzwerk von Gerber [14] zu orientieren, da dieses für die Sprecheridentifikation sehr gute Resultate erzielt hat. Allerdings wurden aufgrund der Experimente aus Abschnitt 4 dem BLSTM Layer noch zwei dense layer angefügt. Die beiden dense layer verwendeten 1000 respektive 500 Units und die ReLU Aktivierungsfunktion. Als Output des L3, dem letzten BLSTM Layer, wurde jeweils der letzte Output des vorwärts und rückwärts laufenden LSTMs genommen und zusammengehängt. So entstand ein Output Vektor der Länge 512. Dieser diente in den Experimenten auch als Embedding Layer für das Clustering. Das Netzwerk wurde mithilfe der Python-Bibliothek Keras [37] implementiert.»

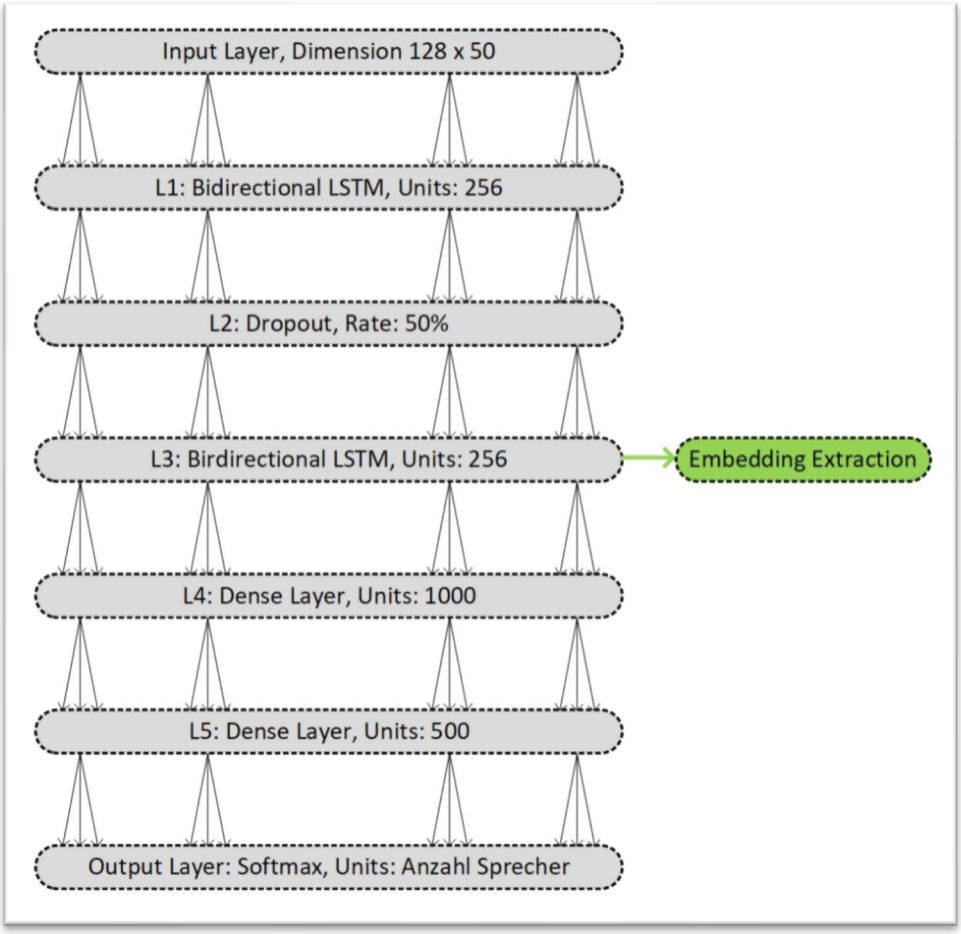


Abb. 14: Voice Conversion Architektur

Das Netzwerk ist auf *Speaker Clustering* ausgelegt, da wir für die *Loss-Function* des Voice Conversion Netzes aber nur das Erkennen eines Sprechers benötigen, entziehen wir die *Embeddings* aus dem Layer 3. Diese können dann verglichen werden.

6.2.2 Schwierigkeiten & Probleme

Anstatt den Standard Datensätzen, verwendet das Voice Recognition Netzwerk angepasste WAV-Audiodateien. Die Dateien benötigen einen Header und werden mit «RIFF» versehen. Dies erschwert die Kompatibilität zwischen den Netzwerken.

Das Voice Recognition Netzwerk wurde auf Python 2.7 programmiert. Da alle anderen Teile mit Python 3.5 programmiert sind gibt es teilweise Probleme bei der Kompatibilität die aber gelöst wurden. Diese entstehen zum Beispiel bei der Verarbeitung von Strings, so gibt es in Version 3.5 einen Unterschied zwischen byte-wise oder direkt String. Der linke String muss zuerst mithilfe von «UTF-8» decodiert werden.

```
b'Hallo Welt' ≠ 'Hallo Welt'
```

Einige Funktionen die verwendet wurden haben, sich in der neusten Version der verwendeten Softwarepakete verändert. So ist es wichtig, dass die Keras und Numpy Versionen genau stimmen. Falls dies nicht der Fall ist oder zwischen dem Training und der Verwendung ein Versionswechsel stattgefunden hat, können die Gewichte des Models nicht geladen werden.

6.2.3 Evaluierung

Damit ein Training erfolgreich gestartet werden kann müssen einige Voraussetzungen gegeben sein. Die Training- und Test-Daten müssen aus dem TIMIT Datensatzes extrahiert werden. Dies geschieht über das Skript:
«.\VoiceConversion\src\voice_recognition\spectrogram_generation\main_data_extractor.py»
Hier müssen folgende Parameter richtig gesetzt werden:

SENTENCES_PER_SPEAKER = 10 Anzahl an Sätzen pro Sprecher, bei den TIMIT Datensätzen sind das immer 10 Sätze

MAX_SPEAKERS = 100 Anzahl Sprecher die in den Trainings bzw. Test Daten vorhanden sein sollen.

WITH_SPLIT = True Soll bereits in Test und Trainings Daten aufgeteilt werden

Wenn das Skript erfolgreich beendet ist hat es ein pickle-File mit allen Daten erstellt. Diese Datei wird nun dem eigentlichen Training mitgegeben:

```
import nets.bilstm_2layer_dropout_plus_2dense as lstm2_dense

lstm2_dense.bilstm_2layer_dropout('[Name des Experiments]',
                                  '[Pickle File mit Traings-Datensätzen]',
                                  n_hidden1=256,
                                  n_hidden2=256,
                                  n_classes=100,
                                  segment_size=50)
```

Mit den oben verwendeten Parametern trainiert das Netzwerk auf einer mittelschnellen CPU ca. 16h. Die Performance des Netzwerks wird auf dem *Cluster* mit einer GPU und 16 GB Arbeitsspeicher nicht besser, was vermutlich daran liegt, dass die GPU nicht richtig verwendet wird. Leider war es nicht mehr möglich dieses Problem zu verfolgen und aufzulösen.

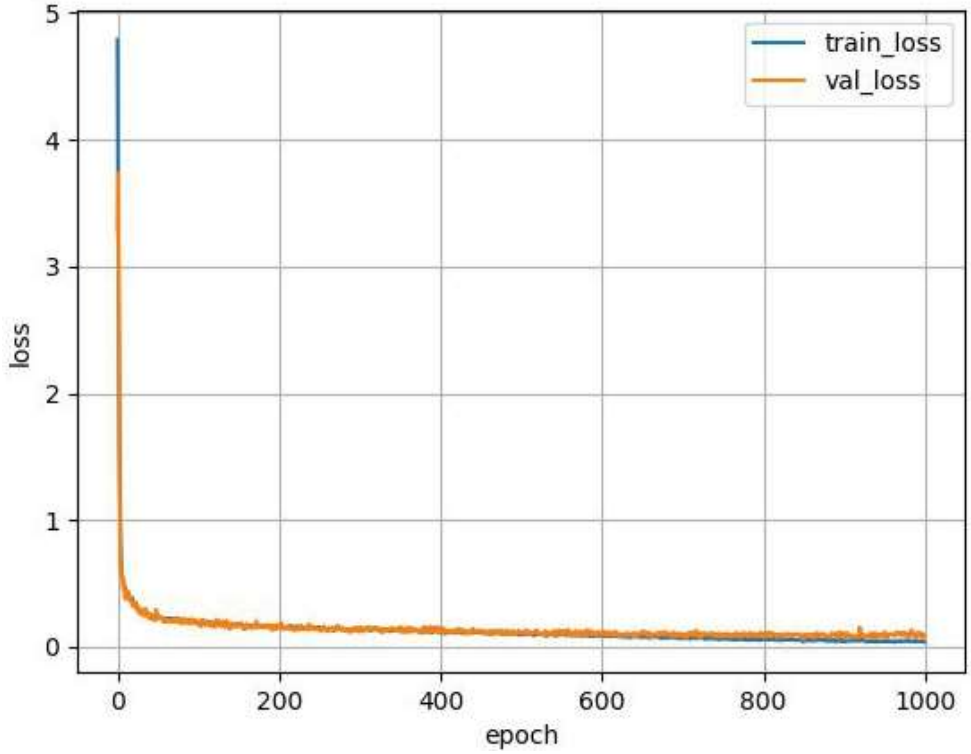


Abb. 15: Performance des Voice Recognition Netzwerks

Die beiden Graphen in Abb. 16 liegen sehr nahe zusammen, was ein Zeichen für ein gutes Verhältnis zwischen Training und Test Daten ist. Doch gegen Ende der 1000 Epochen differenzieren sich die beiden Loss Kurven leicht voneinander, dies könnte ein Anzeichen für ein «*overfitting*» sein. In diesem kleinen Ausmass kann die Verschiebung aber ignoriert werden.
Kritisch ist, dass sich dieser Graph und die beiden Loss Kurven auf *Speaker Clustering* spezifizieren und nicht auf das extrahieren des *Embeddings* eines Sprechers. Da das Eine ohne das Andere nicht funktioniert kann davon ausgegangen werden, dass auch das Extrahieren des *Embeddings* gut funktioniert.
Das Voice Recognition Netzwerk kann nun in der *Loss-Function* des Voice Conversion Netzwerk verwendet werden um die Übereinstimmung zwischen Stimmen zu bestimmen.

6.3 Voice Conversion

6.3.1 Lösungsarchitektur

Das Voice Conversion Netzwerk ist mit Tensorflow implementiert. Der Aufbau ist an die Architektur des Speech Recognition Netzwerks angelehnt mit dem Unterschied, dass wir ein Fully Convolutional Neural Network verwenden. Dies hat den wie bereits in Kapitel 5.4 erwähnten Vorteil, dass die Ausgabe dieselben Dimensionen besitzt wie die Eingabe Abb. 17.

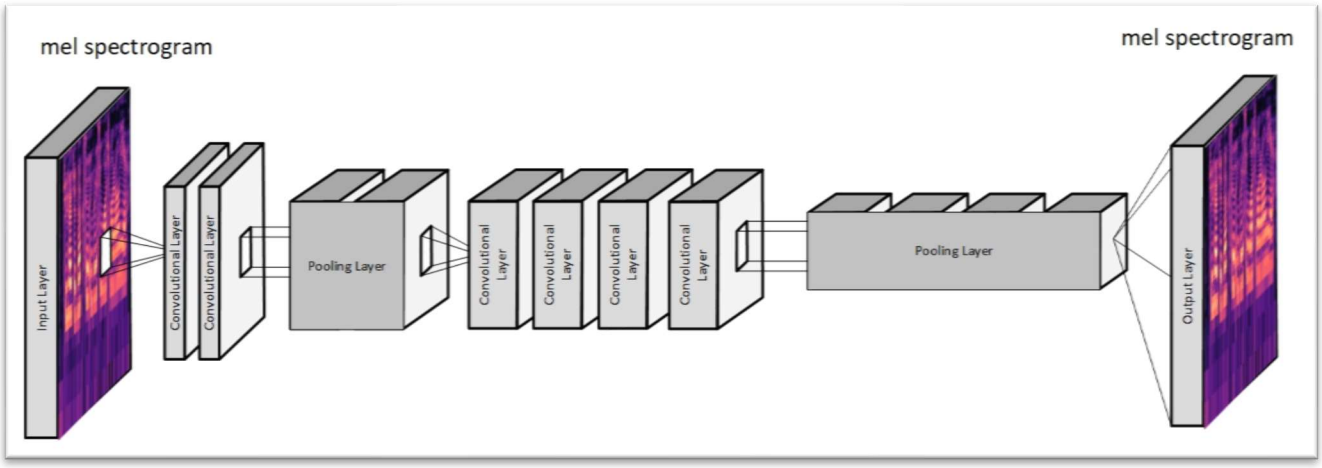


Abb. 16: Voice Conversion Architektur

Damit Tensorflow nur das neue Netz trainiert und die anderen ausschliesslich ausführt, ohne die Gewichte zu verändern, muss man diese einfrieren. In diesem Fall wurde Tensorflow mitgeteilt, welche Gewichte trainiert werden dürfen, die restlichen lässt das Framework unverändert.

6.3.2 Schwierigkeiten & Probleme

Die grösste Schwierigkeit besteht darin die beiden vorausgegangenen Netzwerke, also Speech Recognition und Voice Recognition zu verwenden. Die Problematik liegt in den Dimensionen. Die beiden Netzwerke haben unterschiedliche Eingabegrössen. Das ist problematisch, weil jeder Input des Voice Conversion Netzwerk ebenfalls durch die beiden Subnetzwerke fliessen können muss. Nur so kann garantiert werden, dass das Netzwerk richtig trainiert. Wenn also der Input «Hallo Welt» von Sprecher A auf «Hallo Welt» von Sprecher B verändert werden soll, müssen alle Netzwerke z.B. nur Hallo verwenden. Falls Speech Recognition den gesamten Text also «Hallo Welt» in diesem Batchdurchgang verwendet, kann es zu Inkonsistenzen kommen.

Leider ist es uns zeitlich nicht mehr gelungen diese Normalisierung zwischen den Netzwerken vorzunehmen.

7 Fazit

In diesem Kapitel wird auf die Arbeit zurückgeblickt und erklärt, was den Erflog des Projektes verhinderte und die dabei gemachten Erfahrungen aufzeigt.

7.1 Speech und Voice Recognition

Wie zuerst gehofft, konnte man die zwei Subsysteme, Speech und Voice Recognition, nicht als Blackbox betrachten und nur mit einer Audiodatei speisen. Damit das Voice Conversion Netz funktioniert, sollten alle Systeme gleich lange Spektrogramme verwenden, was bedeutet die importierten Netze müssten umgeschrieben werden. Dies war in der kurzen Zeit leider nicht möglich. Zudem muss man darauf achten, dass der Speech Recognition Teil eine optimale Länge der Ausschnitte erhält, damit die Phoneme gut erkannt werden. Für das Voice Recognition System wiederum, darf der Ausschnitt nicht zu klein sein, sonst wird das extrahierte *Embedding* ungenau.

7.2 Einschätzung

Während diesem Projekt konnten wir einiges zu Neuronalen Netzen und Spracheigenschaften (Phoneme, Spektrogramme, ...) lernen. Mit diesem Vorwissen und dem Verständnis der zwei Subsysteme sind wir überzeugt, dass die Idee der Arbeit «Deep Learning-basierte Voice Conversion mittels problemspezifischer Loss-Funktion» umsetzbar ist und nach einigem experimentieren gute Resultate erzeugen könnte.

7.3 Anderer Ansatz

Statt eine Änderung an dem Spektrogramm zu machen, um die Stimme anzupassen aber die Äusserung gleich zu lassen, ist die Idee von Andabi [31] sehr elegant. Bei diesem Ansatz wird aus einem Spektrogramm das dazugehörige Phonem bestimmt und ein weiteres Netz lernt, dieses Phonem wieder in ein Spektrogramm mit der Zielstimme umzuwandeln. Dieser Ansatz funktioniert gut, nur das beim Stand Heute, die erzeugte Stimme mechanisch klingt. Eine Ergänzung mit dem Voice Recognition Netz, könnte bessere, menschenähnlichere Stimmen generieren.

8 Ausblick

Im Kapitel Ausblick beantworten wir Fragen wie: «Was ist noch offen?», «Welcher Aufwand wäre noch nötig?» und mehr. Wir geben ebenfalls ein persönliches Statement ab und reflektieren uns.

8.1 Weiterführende Arbeit

Damit eine Voice Conversion effizient und gut verständlich funktioniert fehlen noch einige Implementationen, die in dieser Arbeit zeitlich nicht mehr möglich waren.

Normalisierung der Inputdimensionen
Die beiden Subnetzwerke haben unterschiedliche Eingabedimensionen, diese müssen aneinander angeglichen werden. Weil das Voice Conversion System auf dem Speech Recognition System basiert, sind diese kompatible. Das Voice Recognition System hingegen sollte auf die anderen beiden Systeme angeglichen werden.

Keras vs. Tensorflow
Die beiden Frameworks Keras und Tensorflow sind beide sehr mächtige Hilfen beim Bau eines Neuronalen Netzwerks. Auch wenn Keras auf Tensorflow aufsetzt, gibt es Kompatibilitätsprobleme. Wenn die beiden Umgebungen gemischt werden kann es zu einigen sehr mühsamen Kleinigkeiten kommen, obschon die beiden Systeme einander unterstützen [38]. Es benötigt viel Erfahrung und sollte gut abgewogen sein, welches Framework zu verwenden ist.

***Pickle* Dateien**
Um ein aufgeräumtes und effizientes Voice Conversion System zu bauen, ist die im Voice Recognition Netzwerk verwendete Pickle-Architektur hinderlich. Die Verarbeitung und Vorbereitung der Daten als *Pickle* Dateien verhindert den direkten Fluss der Daten. Das Netzwerk sollte umgebaut werden, damit es direkt Spektrogramme verwenden kann analog dem Voice Conversion und Speech Recognition Netzwerk.

Balance der kombinierten Loss-Function
Wenn die Integration der beiden Subnetzwerke erfolgreich war muss die richtige Balance zwischen Voice Recognition Teil und Speech Recognition Teil gefunden werden. Diese Balance ist wichtig da nicht beide Teile gleich viel Anteil an der Sprache haben. Vielleicht ist in einer späteren Applikation wichtiger das die Aussage stimmt als das die Stimme auf die Zielstimme passt. Weiter kann die Loss-Function auch noch hinsichtlich anderer Vergleichsalgorithmen und Normalisierung verändert werden, falls diese bessere Resultate liefern.

8.2 Reflektion

Wir haben diese Arbeit gewählt, weil uns die Idee eines Voice Conversion System mit Hilfe von Neuronalen Netzwerken sehr fasziniert. Obschon wir keinerlei Erfahrung im Bereich KI oder Sprachanalyse hatten, haben wir diese Herausforderung gerne angenommen.

Nach dem Studium von etlichen Stunden an KI-Material war es für uns leider immer noch schwierig, die richtige Flughöhe zwischen unserem Netzwerk und den beiden Subnetzwerken zu finden. Wir haben viel Zeit investiert die beiden Subsysteme zu verstehen und sie zu trainieren. Das wir zu Beginn entschieden haben unsere Umgebung als Python Version 3.5 aufzusetzen hat uns nachträglich sehr gebremst, oft mussten solche Probleme gelöst werden anstatt and der Architektur zu testen. Leider haben wir es in der vorgegebenen Zeit nicht geschafft ein direktes «ent to end» Voice Conversion System zu bauen, trotzdem haben wir sehr viel gelernt.

Oft gegen Ende der Arbeit sprachen wir davon, mit dem jetzigen Wissen die Arbeit nochmals zu beginnen, dies hätte einen enormen Schub an Leistung zur Folge. Wir hoffen unsere Arbeit hilft einem nächsten Team einen lauffähigen Prototyp zu erstellen.

9 Literaturverzeichnis

[1] A. de Brébisson, J. Sotelo, and K. Kumar, “Lyrebird - Create a digital copy of voice,” 2017. [Online]. Available: https://lyrebird.ai/. [Accessed: 14-Sep-2017].

[2] J. Lai, B. Chen, T. Tan, S. Tong, and K. Yu, “Phone-aware LSTM-RNN for voice conversion,” in *International Conference on Signal Processing Proceedings, ICSP*, 2017, pp. 177–182.

[3] S. Mobin and J. Bruna, “Voice Conversion using Convolutional Neural Networks,” 2016.

[4] D. Huang, L. Xie, Y. Lee, J. Wu, … H. M.-9th I. S., and undefined 2016, “An automatic voice conversion evaluation strategy based on perceptual background noise distortion and speaker similarity,” *nwpu-aslp.org*.

[5] L. Sun, S. Kang, K. Li, H. M.- Acoustics, S. and Signal, and undefined 2015, “Voice conversion using deep bidirectional long short-term memory based recurrent neural networks,” *ieeexplore.ieee.org*.

[6] L. Chen, Z. Ling, L. Liu, L. D.-I. T. on Audio, and undefined 2014, “Voice conversion using deep neural networks with layer-wise generative training,” *dl.acm.org*.

[7] D. Povey *et al.*, “The Kaldi speech recognition toolkit,” in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011, pp. 1–4.

[8] “DeepSpeech.” [Online]. Available: https://github.com/mozilla/DeepSpeech. [Accessed: 01-Dec-2017].

[9] “TensorFlow.” [Online]. Available: https://www.tensorflow.org/. [Accessed: 01-Dec-2017].

[10] Jack Purcher, “Apple Patent Reveals a New Security Feature Coming to Siri - Patently Apple,” *04.04*, 2017. [Online]. Available: http://www.patentlyapple.com/patently-apple/2017/04/apple-patent-reveals-a-new-security-feature-coming-to-siri.html. [Accessed: 15-Dec-2017].

[11] P. Gerber and S. Glinski-haefeli, “Bachelorarbeit (FS17 Studiengang Informatik) Machine Learning for Speaker Clustering,” ZHAW - Zürcher Hochschule für Angenwandte Wissenschaften, 2017.

[12] Y. X. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, “LEARNING EMBEDDINGS FOR SPEAKER CLUSTERING BASED ON VOICE EQUALITY,” pp. 25–28, 2017.

[13] T. Gygax and J. Egli, “Speaker Clustering mit Metric Embeddings,” p. 69, 2017.

[14] P. Gerber, “Speaker Identification mit Reccurrent Neural Networks,” p. 44, 2016.

[15] Nvidia Corp., “Grafikkarte NVIDIA TITAN X mit Pascal | GeForce.” [Online]. Available: http://www.nvidia.de/graphics-cards/geforce/pascal/titan-x/. [Accessed: 04-Dec-2017].

[16] L. Page, “Google Assistant - Your own personal Google.” [Online]. Available: https://assistant.google.com/. [Accessed: 29-Nov-2017].

[17] S. Jobs, “iOS - Siri - Apple.” [Online]. Available: https://www.apple.com/de/ios/siri/.

[18] B. Gates, “Cortana.” [Online]. Available: https://www.microsoft.com/de-de/windows/cortana. [Accessed: 29-Nov-2017].

[19] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, “Supervised, unsupervised, and semi-supervised feature selection: A review on gene selection,” *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 13, no. 5, pp. 971–989, 2016.

[20] M. Andrychowicz *et al.*, “Learning to learn by gradient descent by gradient descent,” *NIPS*, pp. 1–16, 2016.

[21] “Typical_cnn.png (PNG-Grafik, 593 × 182 Pixel).” [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/6/63/Typical_cnn.png/593px-Typical_cnn.png. [Accessed: 14-Dec-2017].

[22] S. S. Stevens, J. Volkmann, and E. B. Newman, “A Scale for the Measurement of the Psychological Magnitude Pitch,” *J. Acoust. Soc. Am.*, vol. 8, no. 3, pp. 185–190, 1937.

[23] “The 44 Sounds (Phonemes) of English.”

[24] Google Inc., “TensorBoard: Visualizing Learning | TensorFlow.” [Online]. Available: https://www.tensorflow.org/get_started/summaries_and_tensorboard. [Accessed: 15-Dec-2017].

[25] Z. Victor, S. Seneff, and J. Glass, “TIMIT acoustic-phonetic continuous speech corpus,” *Speech Commun.*, vol. 9, no. 4, pp. 351–56, 1990.

[26] B. Meier, “A Survey on Voice Conversion using Deep Learning.”

[27] C. Gaida, P. Lange, R. Petrick, P. Proba, A. Malatawy, and D. Suendermann-Oeft, “Comparing Open-Source Speech Recognition Toolkits.”

[28] K. Varga, “Kaldi ASR: Extending the ASpIRE model – Research Stories.” [Online]. Available: https://chrisearch.wordpress.com/2017/03/11/speech-recognition-using-kaldi-extending-and-using-the-aspire-model/. [Accessed: 06-Dec-2017].
[29] R. Alvarez and Y. Carmiel, “Kaldi now offers TensorFlow integration,” 2017. [Online]. Available: https://developers.googleblog.com/2017/08/kaldi-now-offers-tensorflow-integration.html. [Accessed: 06-Dec-2017].
[30] D. Bahdanau *et al.*, “TASK LOSS ESTIMATION FOR SEQUENCE PREDICTION.”
[31] D. Ahn and K. Park, “Voice Conversion with Non-Parallel Data,” 2017. [Online]. Available: https://github.com/andabi/deep-voice-conversion. [Accessed: 08-Nov-2017].
[32] Wikipedia, “Cosine similarity,” *Wikipedia*, pp. 1–4, 2013.
[33] Leonardo Araujo dos Santos, “Image Segmentation · Artificial Inteligence,” 2017. [Online]. Available: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html. [Accessed: 15-Dec-2017].
[34] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2528–2535.
[35] J. Salomon, “Support Vector Machines for Phoneme Classification,” 2001.
[36] J. Ye, R. J. Povinelli, and M. T. Johnson, “PHONEME CLASSIFICATION USING NAIVE BAYES CLASSIFIER IN RECONSTRUCTED PHASE SPACE.”
[37] T. Keras, “Keras,” 2015. [Online]. Available: https://keras.io/.
[38] R. Thomas, “Big deep learning news: Google Tensorflow chooses Keras · fast.ai,” *03*, 1AD. [Online]. Available: http://www.fast.ai/2017/01/03/keras/. [Accessed: 15-Dec-2017].
[39] “Gradient Descent.” 2016.

10 Verzeichnisse

10.1 Abbildverzeichnis

Abb. 1: Schema Neurales Netzwerk 3
Abb. 2: Gradientenabstieg in 3 Dimensionen [39] 3
Abb. 3: CNN erkennt Roboterarm [21] 4
Abb. 4: Mel-Spektrogramm 4
Abb. 5: Übersicht Konzept 6
Abb. 6: Konzept Speech Recognition Kaldi 7
Abb. 7: Aspire Model mit Kaldi [28] 7
Abb. 8: Konzept Speech Recognition mit Phonem Verteilung 8
Abb. 9: Konzept Voice Recognition 9
Abb. 10: Schema «fully convolutional network» adaptiert von [33] 10
Abb. 11: Deconvolutional Network [33] 10
Abb. 12: Modell "deep voice conversion"[31] 11
Abb. 13: Train-Loss Speech Recognition Netz 13
Abb. 14: Voice Conversion Architektur 15
Abb. 15: Performance des Voice Recognition Netzwerks 16
Abb. 16: Voice Conversion Architektur 17

10.2 Formelverzeichnis

Formel 1: Berechnung der Frequenzachse im Mel Spektrogramm 5
Formel 2: Teilbedingung Ableitbarkeit 8
Formel 3: kombinierte Loss-Function des Voice Conversion Systems 9

11 Glossar

Loss-Function eine Funktion, welche den Fehler eines Neuronalen Netzes misst und die Genauigkeit zu verbessern versucht. Die am häufigsten verwendete Loss-Function in Neuronalen Netzwerk Bereich ist Cross Entropie.

Deep Learning bezeichnet Optimierungsmethoden künstlicher neuronaler Netze, die mit zahlreiche Zwischenlagen zwischen Eingabe- und Ausgabeschicht haben. Deep Learning wird oft in Umgebungen mit Sprache oder Bildern verwendet.

Speaker Clustering beschreibt das maschinelle Erkennen zu welcher Sprechergruppe eine Audio Aussage gehört. Dies ermöglicht über Gruppen bzw. Dialekte zu generalisieren.

Embedding beschreibt in diesen Zusammenhang, eine Liste von Stimmeigenschaften in vektorieller Form. Jeder Sprecher besitzt seine eigenen Stimm-Embeddings.

Cluster sind im IT-Bereich ein Verbund von mehreren CPUs bzw. GPU zur effizienteren Berechnung von mathematischen Problemen.

Neuron ist die kleinste Verarbeitungseinheit eines Neuronalen Networks, inspiriert von einer Nervenzelle des Gehirns. Es verbindet alle Eingaben und entsprechend ihrer Aktivierung leitet die Zelle ein Resultat an das nächste Neuron weiter.

machine learning beschreibt eine künstliche Maschine die mit Hilfe von Beispielen Zusammen hänge erlernen kann. Es werden keine Beispiele auswendig gelernt, sondern versucht zu abstrahieren und dem Problem auf den Grund zu gehen.

supervised learning ist machine learning mit einer Datenquelle die Rohdaten wie auch die Lables dazu beinhaltet. z.B. Audiodateien mit passenden Transkripten oder Phonemen.

Gradientenabstiegsverfahren ist ein Verfahren, um bei einer Funktion das Minimum oder das Maximum zu finden, dies auch über lokale Minima hinweg. Ein Gradientenabstieg bedient sich der Ableitung der jeweiligen Funktion um einen Abstieg festzustellen.

layer ist eine Schicht eines Neuronalen Netzes, welche aus vielen Neuronen besteht. Es gibt viele verschiedene Arten und Grössen von Layern, in dieser Arbeit geht es mehrheitlich um die «convolutional layer» und «hidden layers».

Framework ist ein Programmgerüst, welches vordefinierte Funktionen zur Verfügung stellt. Es kann in eigenen Code eingearbeitet werden, damit die vordefinierten Funktionen verwendet werden können.

Tensor ist in diesem Zusammenhang, ein Objekt von Tensorflow, welches die Eingabe- bzw. Ausgabedaten und sämtliche Messgrössen eines Netzwerkes beinhaltet. Alles was verarbeitet wird, ist ein Tensor.

Open-Source ist Software, deren Quelltext eingesehen, bearbeitet und verändert werden und meistens kostenlos genutzt werden darf. Diese Programme stehen meist unter öffentlichen Lizenzen wie der General Public License.

Samplerate ist in der Signalverarbeitung der Wert, wie oft ein Audiosignal pro Sekunde abgetastet bzw. gemessen wird. Je höher die Samplerate desto genauer kann das Audiosignal digitalisiert werden, was natürlich zu mehr Speicherplatzbelegung führt.

Transduktoren sind in der theoretischen Informatik eine Art von spezieller endlichen Automaten. Diese zeichnen sich dadurch aus, dass sie im Gegensatz zu einem Akzeptor direkt eine Ausgabe erzeugen.

fully connected layer ist eine Schicht, welche jedes Neuron der letzten Schicht mit jedem Neuron dieser Schicht verbindet. Dies ist ein einfacher Weg nichtlineare Kombinationen von Features zu lernen.

convolutional layer ist eine Schicht, welche die wichtigsten Eigenschaften auf mehrere Featuremaps abbildet. Jede Map fokussiert sich auf einen speziellen Bereich der Inputdaten, damit eignen sich conolutional layer für das Erkennen von spezifischen Merkmalen.

pooling layer ist eine Schicht, welche die Dimension der Daten reduziert und die wichtigsten Informationen übernimmt. Diese Schicht hat mehrere theoretische Vorteile, nämlich verbesserte Laufzeit, Prävention gegenüber Overfitting und verringerter Platzbedarf.

hidden layer ist eine innere Schicht eines Netzes, welche normalerweise gegen aussen nicht sichtbar sind.

Backpropagation Ein Spezialfall des Gradientenabstiegsverfahren. Die Ausgabe wird mit dem gewünschten Ergebnis verglichen und die Abweichung rückwärts durch das Netz geleitet, um die verantwortlichen Gewichte anzupassen.

Overfitting Ein Netz lernt die Training-Daten «auswendig». Die Trainings- ist im Gegensatz zur Testgenauigkeit sehr hoch.

Pickle Ist ein Programm, welches das abspeichern und Laden von Objekten erlaubt. So kann z.B. ein Spektrogramm einfach und schnell zwischengespeichert werden.

12 Anhang A: Aufgabenstellung

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Deep Learning-basierte Voice Conversion mittels problemspezifischer Loss-Funktion
PA17_stdm_5

BetreuerInnen: Thilo Stadelmann, stdm
Fachgebiete: Datenanalyse (DA)
Software (SOW)

Studiengang: IT
Zuordnung: Institut für angewandte Informationstechnologie (InIT)

Gruppengrösse: 2

Kurzbeschreibung:
Hintergrund und Ziel

Nicht erst seit der Aufsehen erregenden Demo des AI-Startups "Lyrebird" (https://lyrebird.ai/demo) ist Voice Conversion ein Thema. An der ZHAW wird der Einsatz dieser Technologie für neuartige Trainingsmöglichkeiten in der Spracherkennung von Nischendialekten seit 2016 untersucht. Ziel dieser Arbeit ist die Ausgestaltung, Implementierung und Evaluation eines Voice Conversion Prototypen basierend auf der Idee, dass am Ende am besten ein automatisches Spracherkennungssystem (ASR) plus ein automatisches Sprecheridentifikationssystem (SI) beurteilen können, wie gut der Prototyp sein Ziel erlernt hat; folglich sollten die Outputs zweier solcher Systeme Teil der zum Training verwendeten Loss-Funktion sein.

Vorgehen

Aufsetzen eines Deep Learning-basierten State of the Art SI und ASR Systems aus dem Open Source Bereich als differenzierbare "Black Box"

- Implementierung eines Voice Conversion Prototypen in Python / TensorFlow basierend auf Open Source Code und unter Einbeziehung obiger Systeme in der Loss Function
- Hierzu Ausgestaltung und -Formulierung der Trainingsidee
- Experimentelle Evaluation und Optimierung des Prototypen
- Darstellen von Vorgehen und Ergebnissen in Berichtsform; ggf. Publikation im Nachgang in geeigneter Form

Voraussetzungen:

Es wird kein Vorwissen über Deep Learning oder TensorFlow vorausgesetzt. Alles kann im Laufe der Arbeit erlernt werden. Notwendig sind gute Programmierkenntnisse sowie Lust und die Fähigkeit, sich eigenständig in wissenschaftliche Literatur einzuarbeiten, diese zu verstehen und umzusetzen. Am wichtigsten ist Freude und Leidenschaft für das Thema. Ein Besuch Data Science-relevanter Vorlesungen wie STDM, IE1/2 oder KI ist von Vorteil.

Die Arbeit ist vereinbart mit:

Remo Knaus (knausrem)
Stefan Schwizer (schwist1)

Weiterführende Informationen:

https://www.zhaw.ch/no_cache/de/forschung/personen-publikationen-projekte/detailansicht-projekt/projekt/3012/

Dienstag 6. Juni 2017 17:41