Identifikation gesprochener Sprache mit Deep Learning

Joel André

Maturaarbeit
Betreut von Beni Keller

Kantonsschule Zug

2019

Inhaltsverzeichnis

1 Einleitung

Sprachoberflächen sind der aktuelle Megatrend in der Tech-Branche. Ein Algorithmus, der natürliche Sprache zuverlässig versteht ist längst kein Sciene-Fiction mehr. Hight-Tech-Firmen rund um den Globus investieren Milliarden in die Entwicklung solcher Produkte. Wie das iPhone mit dem Touchscreen eine Revolution der Interaktion Mensch-Maschine lancierte, erhofft man sich mit Spracherkennung den nächsten Durchbruch. Sprachgesteuerte Programme sollen weitaus intuitiver zu bedienen sein als Texteingaben. In Zukunft ist die lästige Uhrzeitabfrage-Bewegung zum Handy wahrscheinlich Geschichte. Wir werden bequem danach fragen können.

Moderne Spracherkennung-Systeme sind auf einzelne Sprachen spezialisiert. Aus diesem Grund muss ein sprachunabhängiges System, in erster Linie die verwendete Sprache identifizieren, um anschliessend, das passende Sprachsystem anzuwenden. Dieser Schritt nimmt zum Beispiel die Grammatik der erkannten Sprache zur Hilfe. In dieser Arbeit geht es darum den Prozess der Sprachidentifikation zu implementieren: Aufnahmen mündlicher Äusserungen sollen der korrekten Sprache zugeordnet werden.
Die Idee zu diesem Thema entstand spontan. Für mich war aber seit Anfang an klar, dass ich meine Maturaarbeit im Fach Informatik beschreiten würde. Ich nahm bereits an der Informatikolympiade und dem Freifach *Begabtenförderung Informatik* teil. Mich weiter in diesen Bereich zu vertiefen, war eine logische Konsequenz. Überdies besitzen Informatik-Projekte den Vorteil äusserst ressourcenarm zu sein. Ein Computer mit Internet-Anschluss reicht, um ein innovatives Produkt zu entwickeln, dass die Welt verändern kann. Das nötige Wissen lässt sich ohne lange Ausbildung im Internet finden. Entsprechend erlaubt Informatik Jugendlichen bereits teil der Wirtschaft zu sein. Kein 14-Jähriger kann ohne teures Labor Medikamente entwickeln, hingegen für die Idee auf Blockchain hätte theoretisch jeder die Mittel gehabt.

Für die Sprachidentifikation beschränke ich mich auf die Methode, *Künstliche Intelligenz*. KI ist ein prominentes junges Teilgebiet der Informatik. In den letzten Jahren wurden enorme Fortschritte gemacht, was dazu führt, dass die Wirtschaft der Forschung weit hinterher hinkt. Dank fortgeschrittener Computer-Hardware und neuen Algorithmen können Maschinen heute Aufgaben lösen, die vor zehn Jahren unmöglich erschienen. Mich persönlich fasziniert diese Entwicklung. Künstliche Intelligenz beschäftigt den Menschen bereits seit Uhrzeiten. Das Prinzip, einer Maschine Teile unserer kognitiven Fähigkeiten beizubringen, ist für mich so wichtig, wie die Entdeckung des Feuers. Hier etwas beitragen zu können, ist meine grosse Ambition.
Die klassischen Projekte in KI-Arbeiten sind oft Spiele. Es wird dem Computer beigebracht besser zu spielen als jeder Mensch. So ist es zum Beispiel beim Schachcomputer *Deep blue*[5] oder dem Go-Programm *AlphaGo*[1]. Solche Projekte bieten zwar gute Forschungsprojekte, sind aber an sich keine Anwendungsfelder. Ich wollte in meiner Arbeit etwas programmieren, das tatsächlich einen realen Nutzen besitzt. Sprachidentifikation erfüllt das Kriterium. Sprachidentfikation kann nebst für die Spracherkennung, unter anderem im Anrufcenter angewendet werden. Eine Computer erkennt die Sprache des Kunden und leitet den Anruf an den passenden Mitarbeiter weiter.

Klassische Methoden für die Sprachidentifikation beruhen stark auf Expertenwissen. Analytisch entwickelte, handprogrammierte Verfahren erreichen mit wenig maschinellem Lernen sehr gute Leistungen. In dieser Arbeit wird ein anderer Ansatz gewählt. Der Computer soll möglichst viel selbst erlernen. Dafür beschränke ich mich auf die Technologie *Deep Learning*.
Das Ziel dieser Arbeit lässt sich in zwei Teile gliedern. Zuerst soll KI bzw. Deep Learning beschrieben werden und soweit wie möglich ein Verständnis dafür geschaffen werden. Anschliessend wird die Technologie auf das Problem Sprachidentifikation angewendet. Als zu identifizierende Sprachen beschränke ich mich auf Französisch, Englisch und Deutsch. Es werden verschiedene Ansätze probiert und verglichen. Das Produkt soll eine möglichst grosse Fehlerfreiheit besitzen. Um das Produkt zu demonstrieren, wird ein Web-Interface entwickelt, das einem ermöglicht, das Produkt selber zu testen.

Um die Realisarbarkeit dieses Vorhabens zu beurteilen, hatte ich im Vorfeld der Arbeit, bereit das Buch *Neuronale Netze selbst programmieren*[19] aus dem Info-Z gelesen. Dies ermöglichte mir erst, meine Fragestellung so genau einzuschränken und den nötigen Aufwand abzuschätzen. Die enthaltenen Informationen waren aber noch lange nicht ausreichend um mit dem Programmieren zu beginnen. Ausschlaggebend war erst später das Buch *Deep learning with Python*[4], das als meine theoretische Grundlage dient.
Parallel fing ich an das Web-Interface zu entwickeln. Das nötige Know-How besass ich bereits grösstenteils als Vorwissen. Das früh realisierte Interface erlaubte mir, mein Produkt in der Entwicklungsphase live zu testen.
Der nächste Schritt war endlich das Programmieren am Produkt. In dieser Phase gab mir das Paper *Practical Applications of Multimedia Retrieval. Language Identification in Audio Files*[23] die Richtung vor. Um die Unterschiede zwischen den Sprachen zu erlernen, benötigt mein Programm viele Trainings-Daten. Die Beschaffung dieser grossen Menge Daten und das effiziente Umgehen damit, sind ein nicht zu unterschätzender Aufwand.
Die letzte Phase, das Entwickeln der künstlichen Intelligenz, war am spannendsten. Die Komplexität des Themas erlaubte mir allerdings oft nur oberflächlich, ein Verständnis zu entwickeln, zumal mir die mathematischen Grundlagen auf Universitäts-Stufe fehlten.

Im Kapitel **2 Deep Learning** versuche ich, das Verständnis für die verwendeten Technologien der künstlichen Intelligenz weiterzugeben. Das Kapitel bildet das Herz des theoretischen Teils. Das nächste Kapitel **3 Daten** befasst sich mit den Quellen der Daten, den Methoden zur Datenbeschaffung und den Algorithmen zur Datenbearbeitung. Anschliessend wird in Kapitel **4 Web Interface** das Interface und die verwendeten Frameworks vorgestellt. Kapitel **5 Resultate** präsentiert die entwickelten Modelle und vergleicht ihre Fehlerfreiheit. In **6 Diskussion** werden die Ergebnisse in Zusammenhang gestellt und Kapitel **7 Anhang** erlaubt Interessierten einen Einblick in den nötigen Code.

2 Deep learning

Die Begriffe *Deep Learning*, *maschinelles Lernen* und *künstliche Intelligenz* werden oft fälschlicherweise auswechselbar verwendet. Es gibt allerdings eine ganz klare, und für das Verständnis wichtige Hierarchie zwischen den Wörtern. Um Klarheit zu verschaffen werden darum alle Gebiete aufgeführt.

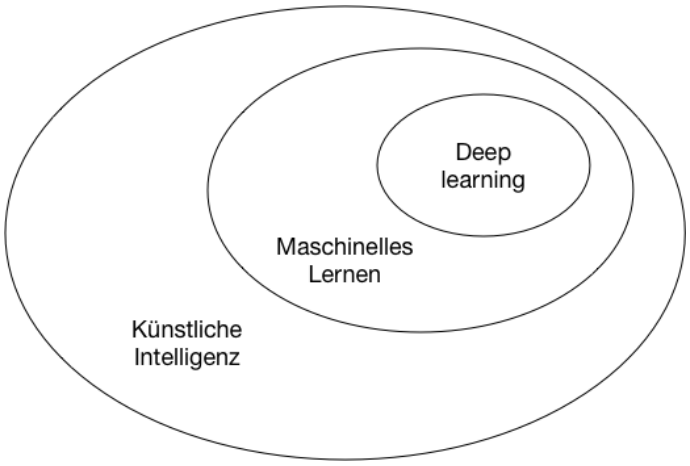

Abbildung 1: Künstliche Intelligenz, Maschinelles Lernen und Deep Learning

Das Gebiet der künstlichen Intelligenz gibt es schon so lange wie den Computer selbst. Die Frage, wie schlau ein Computer werden kann, beschäftigt uns bis heute. Als anerkannte Definition für KI gilt, das Bestreben intellektuelle Aufgaben, die normalerweise von Menschen gelöst werden, zu automatisieren.

Erste Erfolge erreichte man zum Beispiel mit Schachcomputern, die handgeschriebene Regeln befolgten. Diese Form von künstlicher Intelligenz hat aber schnell Grenzen, da viele Prozesse schlicht zu komplex sind, um sie unter angemessenem Aufwand mit Regeln zu beschreiben. Um dieses Problem zu lösen, erfand man maschinelles Lernens.

Der Ablauf von maschinellem Lernen ist grundlegend anders als konventionelles Programmieren. Der Entwickler muss keinen Programmcode mit festen Regeln schreiben, im Gegenteil: Er liefert dem Computer Eingabe und Ausgabe, und der Computer lernt die Regeln selbst. Maschinelles Lernen blühte erst in den 90'er Jahren auf, wurde aber schnell zum grössten Teilgebiet der künstlichen Intelligenz.

Beim maschinellem Lernen, lernt die Software im Grunde eine nützlichere Darstellungsweise der Daten bzw. der Eingabe. Anhand dieser anderen Darstellungsweise kann der Computer die Antwort einfach erkennen. Wenn der Computer stufenweise nützlichere Repräsentationen bestimmt, kann er zunehmend komplexe Probleme, in einfacheren Zwischenschritten lösen. Genau das ist *Deep Learning*. Es bezeichnet das Konzept von stufenweisem Lernen und nicht eine Methode selbst. Eine weit verbreitete Methode sind allerdings *tiefe künstliche neuronale Netze*.[vgl. 4]

2.1 Künstliche neuronale Netze

Künstliche neuronale Netze hat man sich, wie der Name schon preisgibt, von der Natur abgesehen. Ähnlich wie in unserem Gehirn gibt es Neuronen bzw. Knoten und dazwischenliegende Verbindungen. Die Verbindungen haben ein Gewicht w. Künstliche Neurone Netze haben sich aber mittlerweile so stark weiterentwickelt, dass sie

nebst der ursprünglichen Idee, nichts mehr mit der biologischen Variante gemeinsam haben.

Der Wert eines Knotens ist eine Funktion der Summe aller seiner eingehenden Verbindungen. Diese Funktion wird Aktivierungsfunktion σ genannt. Die Aktivierungsfunktion ist wichtig, damit das Netzwerk auch nicht lineare Repräsentationen lernen kann. Eine bekannte Aktivierungsfunktion ist zum Beispiel die *RELU* Funktion. Die RELU Funktion unterdrückt negative Werte, bzw. sie ist null für Werte kleiner als null. [19]

$$\sigma(x) = relu(x) = \max(0, x)$$

Der Wert einer eingehenden Verbindung errechnet sich aus dem Produkt des Gewichts w und dem Wert des ausgehenden Knotens. Wenn man alles zusammensetzt ergibt sich für den Wert irgendeinen Knotens o mit Vorgänger Knoten h diese Formel:

$$o = \sigma\Big(\sum_i h_i \cdot w_i\Big)$$

Mit dieser Formel propagieren sich die Werte der Anfangsknoten durch das ganze Netzwerk. Um das Prinzip anschaulicher zu machen, wird ein Beispiel-Durchlauf vorgeführt. Die verwendeten Gewichte sind willkürlich.

Die Aufgabe des vorgestellten Netzwerks könnte zum Beispiel sein, zwischen Hund und Katze zu unterscheiden. Die Eingaben x_1 und x_2 würden dann gewisse Merkmale des Tieres beschreiben. 1 würde heissen das Tier besitzt das Attribut und 0 das Gegenteil. Die Ausgabe o wäre dann die Wahrscheinlichkeit, dass das Tier eine Katze ist.

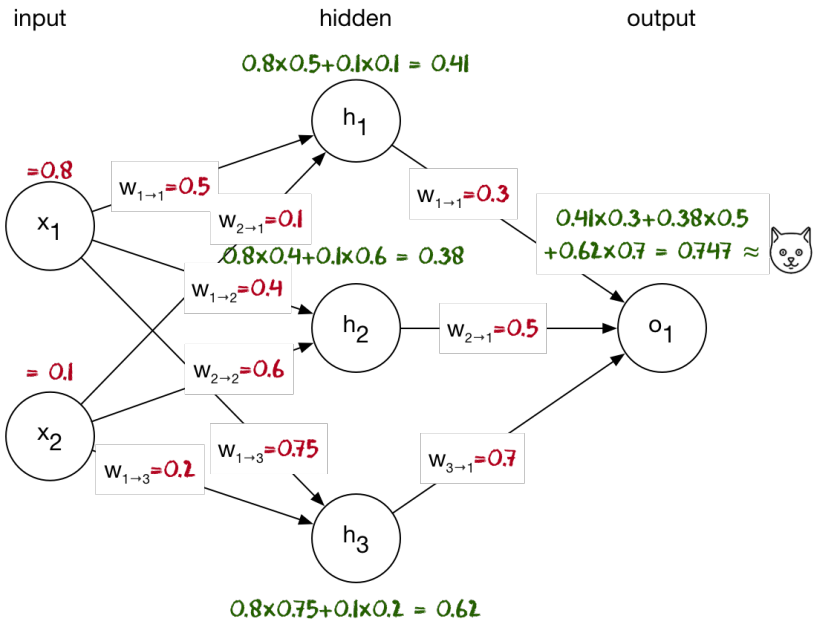

Abbildung 2: Grafische Darstellung eines künstlichen neuronalen Netzwerks anhand eines Beispiels

Die mathematischen Operationen in einem neuronalen Netzwerk lassen sich alle als Matrizen-Operationen berechnen. Mit Matrizen kann der Computer auf einer

Grafikkarte und mit einem BLAS (*Basic linear algebra system*) sehr effizient rechnen [19] .

Was bis jetzt berechnet wurde nennt man den *Vorwärtspass*. Aus einer Eingabe wurde die Ausgabe berechnet. Daran war aber noch nichts intelligent. Erst jetzt können die Parameter der Funktion, die Gewichte, aus diesem Beispiel lernen. Um diese zu verbessern braucht es eine *Verlust Funktion* die uns angibt, wie weit die Ausgabe vom korrekten Ziel entfernt ist. Eine mögliche Verlust Funktion ist die absolute Differenz zwischen dem Ziel und der Ausgabe. Wenn man in unserem Beispiel davon ausgeht, dass die Eingabe wirklich zu einer Katze gehört, ergäbe sich:

$$\text{Verlust}(output, target) = |target - output| = |1.0 - 0.747| = 0.253$$

Um den Verlustwert zu minimieren, passt das Netzwerk die Gewichte schrittweise an. Diesen Teil übernimmt der *Optimierer*. Ein einfacher Optimierer ist zum Beispiel das Gradientenverfahren [8].

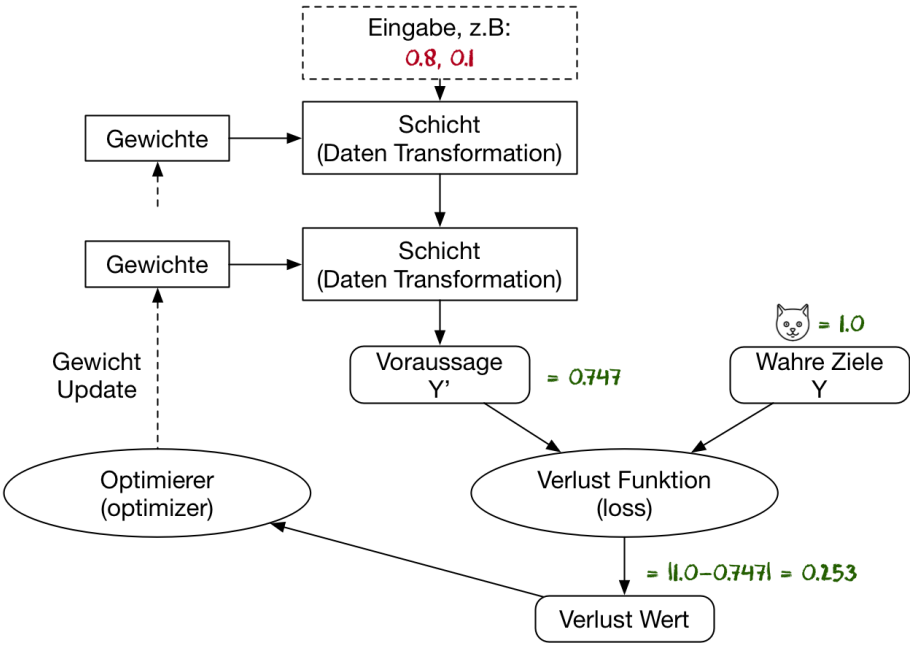

Abbildung 3: Grafische Darstellung des Lern-Prozesses anhand eines Beispiels

Da bei diesem Typ von neuronalen Netzwerken alle Knoten miteinander verbunden sind, wird es oft *Dense Neural Network* genannt. Es ist bewiesen dass solche neuronalen Netze jede Funktion abbilden können[16, Kap. 4].

2.2 Convolutional Neural Networks

Convolutional Neural Networks sind eine sehr weit verbreitete Methode im Feld von *Computer Vision*. Der fundamentale Unterschied zwischen dem oben besprochenen *dense network* und einem *CNN* ist, dass ein *CNN* lokale Muster erkennen kann, wo hingegen das vorherige Netzwerk nur globale Muster erkennen konnte. Das bedeutet, dass ein Muster, das an einer bestimmten Stelle angetroffen wird, an jeder anderen Stelle ebenfalls erkannt wird. [4]

Um das zu erlauben, teilen gewisse Verbindungen das gleiche Gewicht. In Abbildung 4 (Oberer Teil) sind das die gleichfarbigen Verbindungen. Weniger Gewichte führen zusätzlich dazu, dass das Netzwerk schneller lernen kann.

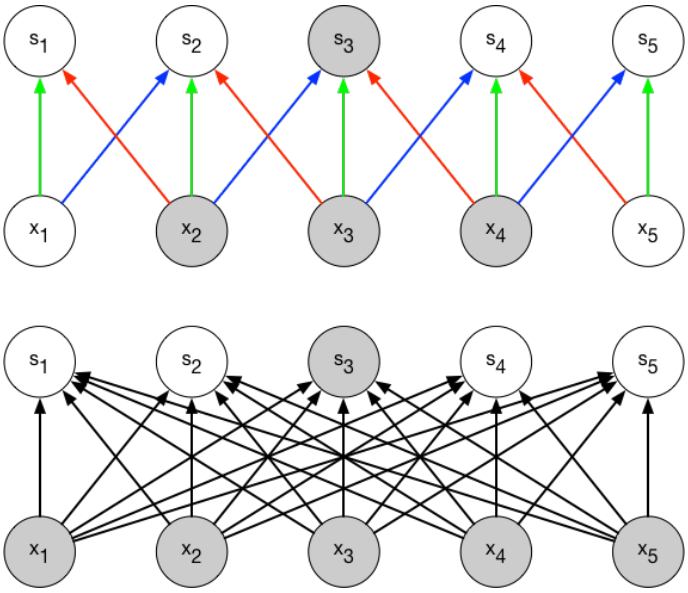

Abbildung 4: (*Oben*)1D Convolution mit *kernel* der Grösse 3. s_3 wird durch 3 inputs beeinflusst. (*Unten*) *Dense Network*. s_3 wird durch alle inputs beeinflusst.[7]

Ein weiterer Vorteil von *Convolutional Neural Networks* ist, dass sie eine räumliche Hyrarchie von Mustern erlernen können. Wenn die Eingabe das Bild einer Katze ist, wird zum Beispiel die erste Schicht unterschiedliche Kanten erkennen, die zweite Schicht dann einzelne Merkmale (z.b Augen), und so weiter.

Damit das gilt, muss aber der analysierte Bereich eines Knotens, von Schicht zu Schicht grösser werden. Deshalb wird meisten nach jedem *Convolution Layer* ein *Pooling Layer* gesetzt. Das *Pooling Layer* fasst mehrere Datenpunkte zusammen um dem nächsten Netzwerk eine grösseren Analysebereich zu verschaffen. Ein oft verwendetes Pooling-Verfahren ist *Max-Pooling*: Angrenzende Knoten werden zusammengefasst durch ihr Maximum.

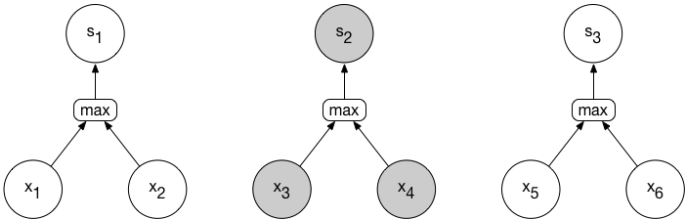

Abbildung 5: Abbildung eines 1D Max-Pooling layer. s_2 ist $\max(x3, x4)$

2.3 Recurrent Neural Networks

Eine gemeinsame Eigenschaft von allen *Dense Neural Networks* und CNN's ist, dass sie keinen Speicher haben. Bei jedem Vorwärtspass berechnet das Netzwerk alles von neuem ohne Erinnerungen an vorherige Durchlaufe. Dieses Verhalten ist das absolute Gegenteil vom menschlichem Denkprozess. Wenn wir einen Satz lesen, durchgehen wir ihn Wort nach Wort und merken uns den vorherigen Kontext.

Recurrent Neural Networks (RNN) bilden diesen Prozess vereinfacht nach. Sie besitzen eine interne wiederkehrende Schleife die dem Netzwerk Informationen aus

dem vorherigen Durchlauf bereitstellt (Siehe Abbildung 6). Die RNN Zelle berechnet dann die nächste Ausgabe sowohl aus der neuen Eingabe, wie auch mit den Erinnerungen der letzten Ausgabe. [4]

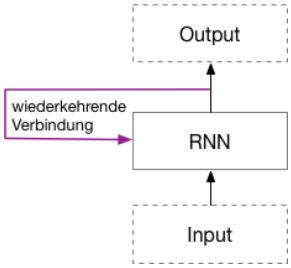

Abbildung 6: RNN mit Schlaufe

Der Vorgang lässt sich grafisch über die Zeit aufgerollt darstellen (Abbildung 7). In dieser Darstellung fällt auf, dass das Netzwerk theoretisch für jeden Schritt eine Ausgabe besitzt. Die zwischenliegenden Ausgaben sind vor allem wichtig, wenn man eine weitere Schicht an das Netzwerk anhängen will. Sonst behält man meist nur die letze Ausgabe, da diese indirekt Informationen über alle anderen beinhaltet.

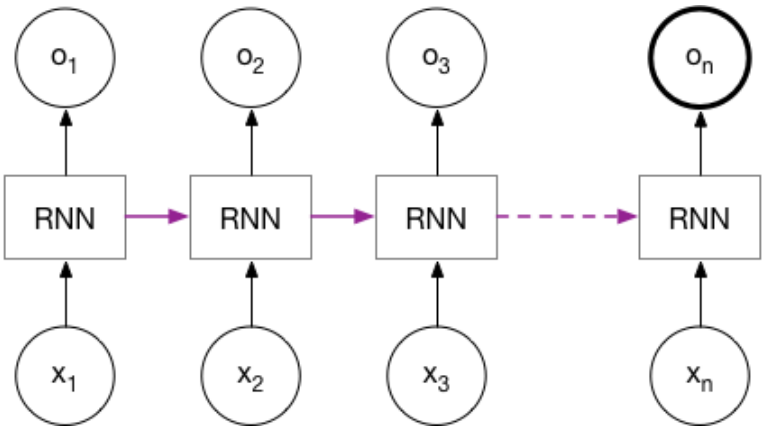

Abbildung 7: RNN aufgerollt über die Zeit

Das ganze Prinzip ergibt jedoch nur Sinn, wenn frühere Eingaben tatsächlich einen Einfluss auf spätere Ausgaben haben. Eine praktische Anwendung ist das Verarbeiten von zeitlichen Sequenzen wie Wetterdaten und Sprache. Die einzelnen Lernbeispiele werden zeitlich zerteilt und stückweise dem Netzwerk gefüttert. Eine fortgeschrittene Implementation von RNN Zellen ist unter anderem die *Long short-tem memory* (LSTM) Zelle [10]. Durch das Einführen von unterschiedlichen wiederkehrenden Verbindungen wird verhindert, dass ältere Signale langsam verschwinden, bzw. vergessen werden [4].

3 Daten

3.1 Datenquellen

Es gibt keine frei zugänglichen umfassenden Datensets für Sprachidentifikation. Datensets wie das *NIST Language Recognition Evaluation*[17] sind nur unter teuren Gebühren zugänglich. Wie verwandte Arbeiten empfehlen [23], wird darum ein eigenes Datenset zusammengestellt. Es werden gleichmässig Daten zu den Sprachen Deutsch, Englisch, und Französisch gesammelt. Die Daten stammen vom *Voxforge*[24] Datenset, von *Youtube*[26] und von *Librivox*[13].

Voxforge ist ein open-source Datenset für Spracherkennung. Es besteht aus vielen kurzen (1-10s), von Benutzern hochgeladenen, Audiodateien. Die englische Sprache dominiert mit rund 120h Audio das Datenset. Über alle drei Sprachen verteilt sind 190 Stunden Ton verfügbar. Die Audioqualität variiert je nach Benutzer.

Die Sprache ist langsam und deutlich verständlich. Sie hört sich eher künstlich an im Vergleich zu einem natürlichem Gespräch. Die Anzahl unterschiedlicher Sprecher ist gering.

Youtube dient als Quelle für abwechslungsreiche Sprache. Es werden populären Nachrichtenkanäle wie CNN, ARD, etc. verwendet (Siehe Tabelle 1). Die Aufnahmen werden oft von diversen Hintergrundgeräuschen begleitet und die Variation der Sprecher gross, da oft fremde Gäste eingeladen werden. Kehrseite ist, dass nicht garantiert werden kann, dass alle Aufnahmen tatsächlich die richtige Sprache beinhalten. Sendepausen und fremdsprachige Interviews kommen vereinzelt vor.

Sprache	Kanäle
Französisch	France24, FranceInfo
Deutsch	ARD, ZDF
Englisch	CNN, BBC

Tabelle 1: Youtube Kanäle

Librivox ist ein öffentlich abrufbares Hörbuch Datenset. Anstatt selbst Hörbücher zu selektieren, wird eine vorgefertigte Selektion verwendet [18]. Verfügbar sind sieben Stunden Aufnahmen mit 90 verschiedenen Sprechern.

3.2 Daten Auswahl

Die Modelle werden grundsätzlich mit den Daten von Voxforge und Youtube trainiert und ausgewertet. Es werden zwei unterschiedliche Datenquellen verwendet um die *Stichprobenverzerrung* zu minimieren. Stichprobenverzerrung bedeutet, dass das Datenset nicht repräsentativ für alle Sprachaufnahmen ist. Bei Voxforge ist zum Beispiel die Gefahr, dass das Modell sich überanpasst an die geringe Anzahl Sprecher. Anstatt die Sprache zu erkennen, könnte das Modell das Mikrofon des Sprechers identifizieren und jedem Sprecher eine Sprache zuordnen. Die Leistung des Modells für neue Sprecher wäre dann nicht besser als ein Zufallsgenerator.
Um die Stichprobenverzerrung zu messen werden die Modelle zusätzlich auf dem

Netz	Voxforge	Youtube	Librivox
Trainingset	56h	56h	-
Validationset	7h	7h	-
Testset	7h	7h	2h

Tabelle 2: Daten Verteilung

Librivox Datenset ausgewertet. Die Daten von Librivox teilen keine systematischen "Fehler"wie zum Beispiel Sprecher mit den Trainingsdaten. Das Librivox-Testset besteht aus insgesamt 2 Stunden Aufnahmen.
Von Youtube und Voxforge werden gemeinsam 139 Stunden Audiodaten heruntergeladen, was 100'000 5s Aufnahmen entspricht. Die Datenmenge wird bewusst klein gehalten, weil grössere Datenmengen ressourcenaufwändiger und ineffizienter wären. Die einzelnen Sprachen und Quellen sind zu gleichen Teilen repräsentiert.
Die Daten werden weiter in 80% Trainingsdaten, 10% Testdaten und 10% *Validationset* gespalten. Das Validationset wird verwendet um während dem Training zu beobachten, wie das Modell auf neue Daten reagiert. Die *Hyperparamter* (Parameter die das Netzwerk nicht selber lernen kann, z.B die Anzahl Knoten) werden manuell so angepasst dass das Netzwerk möglichst gut auf dem Validationset abschneidet. Tabelle 2 gibt eine Übersicht über die Verteilung der Daten.

3.3 Preprocessing

Sprache besteht aus Wörtern und Wörter sind grundsätzlich eine Abfolge von Lauten. Verschiedene Sprachen unterscheiden sich an den verschiedenen Abfolgen von Lauten, manchmal sogar an den verwendeten Lauten selbst. Die kleinste relevante Einheit für Spracherkennung, sowohl für den Menschen wie die Maschine, ist also ein Laut.

Wenn der Computer mit dem Mikrofon aufnimmt, misst er kleinste Druckunterschiede, bzw. Schallwellen. Eine unkomprimierte Audiodatei zeigt den Schalldruck über die Länge der Aufnahme, siehe Abbildung 8 *(oben)*. Die einzelne Schallwelle ist für den Menschen nicht erkennbar, deshalb ist sie auch für die Sprache von keiner Bedeutung. Erst mehrere Schallwellen, bzw. die daraus folgende Frequenz lässt sich als Laut hören.

Das Verfahren um aus einer Schallwelle die unterschiedlichen Frequenzen zu bestimmen heisst *Fourier-Transformation* [21]. Falls dem Netzwerk als Eingabe rohe Schallwellen gefüttert werden, muss es dieses Verfahren erlernen, um dann aus den Lauten die Sprache erkennen zu können. Allerdings ist die Fourrier-Transformation zu erlernen ein zusätzlicher Aufwand und fordert den Computer darum mehr. Um dem Algorithmus die Aufgabe zu erleichtern, kann man ihm darum als Eingabe die berechneten Laute anstatt der rohen Schallwelle geben.

Die Prozedur dem Algorithmus bereits vorgerechnete Werte zu füttern, heisst *Preprocessing* und ist weit verbreitet im Feld von *Machine learning*. Das Vorrechnen ist unter dem Namen *Feature Engineering* bekannt. *Features*, also Merkmale z.b Laute werden aus den Rohen Daten extrahiert. [vgl. 4]

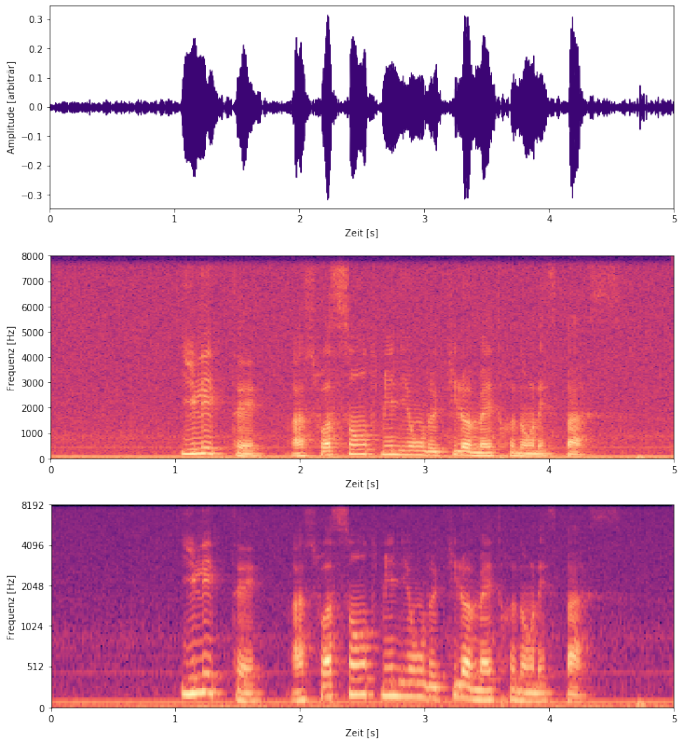

Abbildung 8: Audio-Preprocessing: *(oben)* Rohe Schallwelle, *(mitte)* Dezibel-Spectrogramm, *(unten)* Mel Dezibel-Spectrogramm

3.3.1 Spektrogramme

Spektrogramme sind grafische Darstellungen eines Hörsignals nach der Anwendung der Fourrier-Transformation[21, 'Spectrograms'], ersichtlich in Abbildung 8 *(Mitte)*. Frequenzen über 10kHz können abgeschnitten werden, da menschliche Sprache grösst Teils darunter abläuft[22]. Im Spektrogramm lassen sich von Auge Muster erkennen und unterscheiden. Der visuelle Charakter der Spektrogramme erlaubt ausserdem das verwenden von *Convolutional Neural Networks*.

3.3.2 Mel Filtering

Spektrogramme haben relativ viele Datenpunkte und sind deshalb recht aufwändig zu verarbeiten. Ein weiter *preprocessing* Schritt wird deshalb oft angewendet: *Mel Filtering*[25]. Die Frequenzen werden dabei in grössere Eimer gepackt. Unter 1kHz sind die Eimer linear verteilt und darüber logarithmisch, siehe Abbildung 8 *(unten)*. Das Modell entspricht unserer Hörfähigkeit, die recht präzise unter 1kHz arbeitet, höhere Frequenzen aber schlecht unterscheiden kann[22].

3.3.3 Implementation

Die Oben genannten Transformationen können vor dem Training direkt auf die Daten angewendet und abgespeichert werden. In diesem Fall hätte man die rohen Daten löschen können. Allerdings sollte in dieser Arbeit sowohl an den rohen Daten wie auch an der Transformation flexibel experimentiert werden können, weswegen die Daten unbedingt beibehalten werden mussten.

Anstatt die Transformationen selber zu implementieren wird ein Framework verwendet. In diesem Fall bietet sich an *kapre*[3] an. Mit Kapre lassen sich währende dem Training die Daten in Echtzeit verarbeiten. Das Training dauert dabei aber 20% länger. Konkret verhält sich *kapre* wie eine Schicht vor dem eigentlichen Neuronalen Netzwerk.

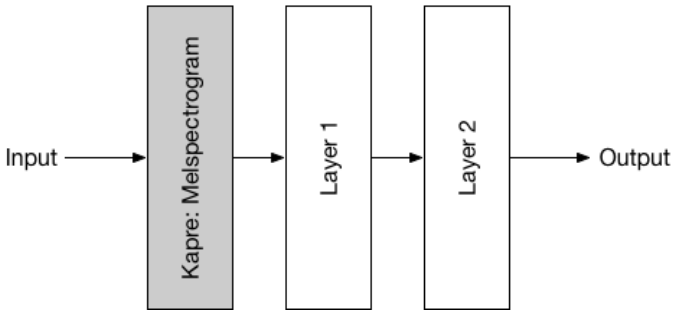

Abbildung 9: *Kapre*[3] als Schicht

4 Modelle

Im folgenden Teil werden die entwickelten Deep Learning Modelle vorgestellt. Für jedes Modell ist der grundlegende Ablauf derselbe (siehe Abbildung 10). Die Aufnahme wird vorab auf die Länge 5s zugeschnitten. Anschliessend wird ein Spektrogramm generiert. Die Werte des Spektrogramms werden in die Dezibell Skala umgerechnet und dann normiert. Die Grösse des Spektrogramms unterscheidet sich zwischen den Modellen. Schliesslich berechnet ein Neuronales Netzwerk aus dem Spektogramm eine Vorhersage für die drei Sprachen.
Insgesamt werden drei grundsätzlich verschiedene Neuronalen Netzwerk Architekturen vorgestellt. Die ersten zwei Modelle sind Convolutional Neural Networks während das dritte Modell eine Kombination zwischen CNN und Recurrent Neural Network ist. Jedes Modell endet mit einer Ausgabeschicht der Grösse drei (für die drei Sprachen) und der Aktivierungsfunktion *softmax*:

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=0}^{n} \exp(x_j)}$$

Die Funktion eignet sich generell für Klassifizierung, denn sie bildet die Werte auf eine Wahrscheinlichkeitsverteilung über die berechneten Ausgabeklassen ab [7, S. 180-184]. Für alle anderen Schichten wird immer die Aktivierungsfunktion *relu* verwendet.
Die Parameter der Modelle, wie zum Beispiel die Grösse des Spektrogramms wurden empirisch bestimmt. Allerdings ist die Anzahl Experimente beschränkt, da jeder Durchlauf zeitaufwändig und nicht parallelisierbar ist.

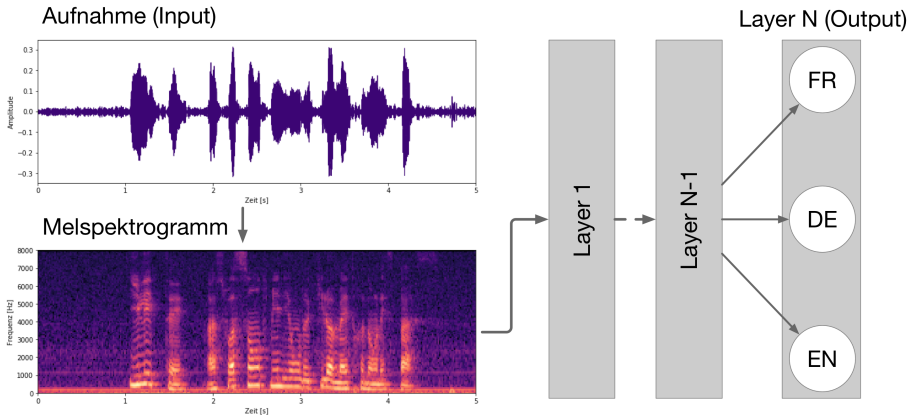

Abbildung 10: Grundlegender Aufbau aller Modelle

4.1 CNN

Das CNN Netzwerk akzeptiert Spektrogramme der Grösse 28x313, wobei 28 verschiedene Mel-Frequenzeimer berechnet werden an 313 Zeitpunkten. Das Netz besteht aus drei Convolution-MaxPooling Blocks und zwei Dense Schichten (Abbildung 11). Die Convolution-Grösse ist immer 3x3 und MaxPooling geschieht im Bereich 2x2. Die Anzahl Convolution's (Kanäle) nimmt von 64 auf 128 zu.
Die *Fully Connected* Schicht besteht aus 512 Knoten mit 30% *Dropout*. Bei Dropout wird ein zufälliger gewählter Anteil der Eingabe der Schicht mit 0 ersetzt. Das

Netz lernt dann, ohne diese Verbindungen auszukommen und somit mehrere Verbindungen einzelnen starken Verbindungen vorzuziehen. Die Eigenschaft hat einen positiven Einfluss auf die Robustheit des Netzes gegenüber neuen Daten [4, Kap. 4.4.3].
Die Architektur entspricht massgeblich der vorgestellten Architektur in [23] plus Dropout.

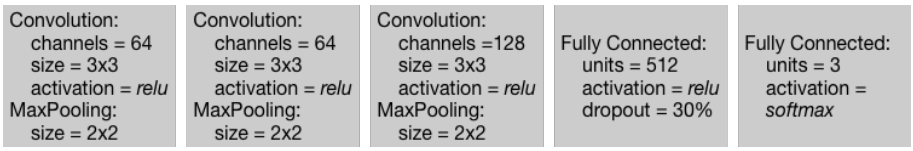

Abbildung 11: CNN Architektur

4.2 MobileNet-CNN

Andere Arbeiten hatten gezeigt, dass grosse CNN Architekturen die auf dem *ImageNet*[6] Datenset (Riesiges Bilddatenset mit tausenden Bidklassen) stark sind, ebenfalls für Audioklassifizierung geignet sind [9]. Um Leistung zu sparen wurden Experimente mit *MobileNet* durchgeführt. MobileNet ist eine bekannte CNN Architektur entwickelt von *Google* [20, (V2)]. Das Modell ist speziell entwickelt für Mobilgeräte, die in der Regel schwächere Hardware als Desktop-Computer besitzen. Das Ziel war ein effizientes Netzwerk für Bilderkennung im grossen Umfang, wie z.B ImageNet. Da das Modell in dieser Arbeit nur zwischen drei Klassen unterscheiden muss, wird die Anzahl Knoten im Netzwerk so weit wie möglich verkleinert, bzw. der Parameter α im Paper wird auf 0.25 gesetzt.
Das Netzwerk besitzt insgesamt 21 Schichten. Das Netzt funktioniert am bessten mit Eingaben der Grösse 224x224, der Bildgrösse in ImageNet. Die Spektrogramme werden darum auf dieses Format skaliert.

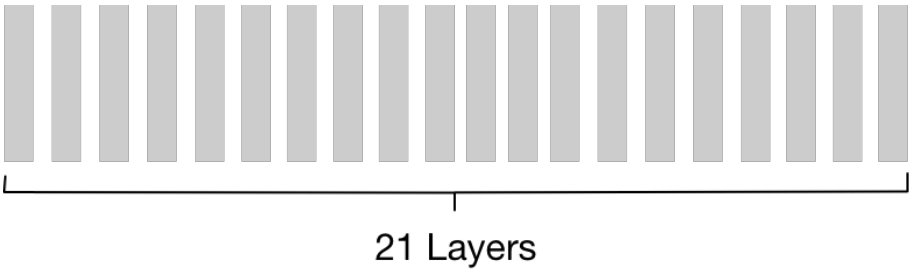

Abbildung 12: MobileNet Architektur

4.3 CRNN

Convolutional Recurrent Neural Networks, eine Mischung aus CNN und RNN, wurden bereits erfolgreich auf Sprachidentifikation angewendet [2][11]. RNN's für Tonaufnahmen zu verwenden ist sinnvoll, denn Audiodaten sind naturgemäss zeitliche Folgen, also ideal für RNN's. RNN's haben aber den Nachteil, dass sie langsam lernen. Darum soll der CNN Teil im Vorfeld aus dem Spektrogramm eine kompaktere

Repräsentation berechnet, z.b eine Abfolge von Phonemen. Da die Lernzeit proportional zur Eingabegrösse ist, lernt der RNN teil mit dieser Eingabe schneller. Es gibt keine Möglichkeit zu verifizieren, ob das CNN wirklich Phoneme berechnet aber die Resultate zeigen, dass das CRNN besser abschneidet als das CNN.
Der CNN Teil besteht aus fünf Blocks von Convolution, MaxPooling und *Batch Normalization* (Siehe Abbildung 13). Die Idee von *Batch Normalization* ist, dass die Ausgabe der vorherigen Schicht so normalisiert wird, dass sie den Durchschnitt 0 und die Varianz 1 besitzt. *Batch Normalization* bringt den vor allem den Vorteil, dass das Modell schneller konvergiert, also schneller bessere Leistungen erbringt. [12]
Die Anzahl Convolutions steigert sich schrittweise von 16 auf 64. Das Convolution-Fenster hat immer die Grösse 4x4 und MaxPooling das Fenster 2x2. Bei den Convolutions wird 0.01 *L2-Regularisation* verwendet:

$$Verlust = Verlust + \sum_i w_i^2$$

Das bedeutet, dass dem Verlustwert zusätzlich das Quadrat der Gewichte addiert wird [7, Kap. 7.1.1]. Übermässig hohe Gewichte werden überproportial gewertet. Die Regularisation hat das gleiche Ziel wie Dropout, das Netz robuster zu gestalten.

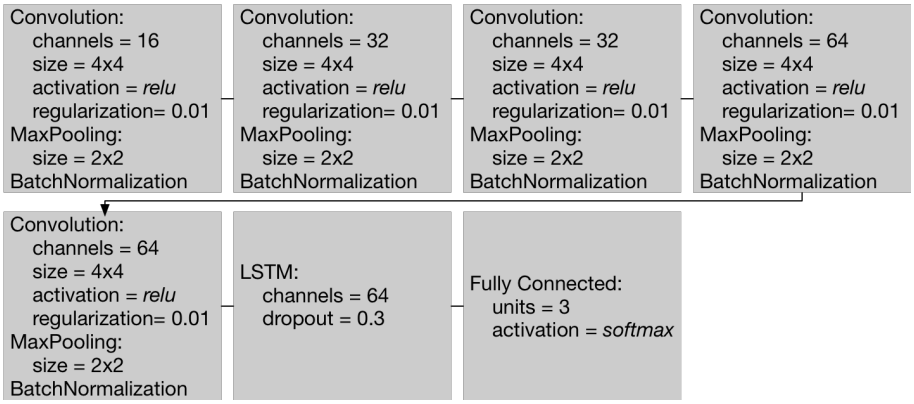

Abbildung 13: CRNN Architektur

5 Auswertung

5.1 Auswertung an Voxforge und YouTube

Netz	Architektur	Accuracy	MAE
CNN	3 Conv	95.1%	0.046
CRNN	4 Conv + LSTM	96.2%	0.031
CNN	28 Conv + Dense	87%	0.03

Tabelle 3: Voxforge und YouTube Richtigkeit

5.2 Auswertung an Librivox

Netz	Trainingsdauer
CNN	21 min
CRNN	68 min
CNN	2h

Tabelle 4: Trainingsdauer

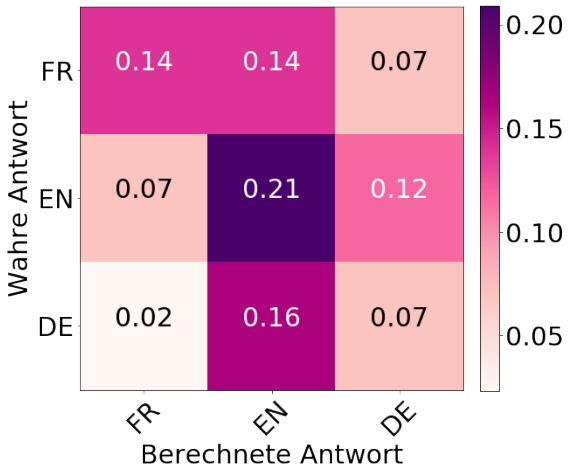

Abbildung 14: Normierte Wahrheitsmatrix

6 Diskussion

Im Rahmen dieser Arbeit wurde ein System zum klassifizieren von Sprachaufnahmen implementiert. Das Programm unterscheidet zwischen Französisch, Deutsch, und Englisch. Mit Deep Learning hat das Programm selbst gelernt die Sprache von kurzen Aufnahmen zu erkennen. Anstatt rohe Schallwellen zu verarbeiten wurden in Echtzeit berechnete Spektrogramme verwendet. Die Spektrogramme konnten mit Bilderkennungsmethoden erfolgreich eingeordnet werden. Ein hybrides Modell aus CNN und RNN besass die höchste Genauigkeit. Die Trainingsdaten wurden wegen fehlenden maßgeschneiderten Datensets aus verschiednen Quellen selbst kompiliert. Es wurde mit Daten von Voxforge und Youtube trainiert um mit Daten aus der Librivox Hörbuchdatenbank getestet zu werden. Die hohe Korrektheit aller Modelle bestätigt, dass tiefe neuronale Netze eine angemessene Methode für Sprachidentifikation sind.

Der Vergleich der Resultate mit anderen Arbeiten ist nur mit Vorsicht möglich. Der wichtigste Unterschied ist, dass andere Daten verwendet werden und dementsprechend nur andere Resultate entstehen können. Glücklicherweise sind Arbeiten zu Automatischer Sprachidentifikation im Internet jedoch frei erhältlich, denn das Problem ist in der Informatik recht jung. Erst in den 70'er Jahren wurde man für das weiterleiten von Telefonanrufen auf das Thema aufmerksam. Eine lange Zeit galt, dass die bessten Systeme diejenigen waren, die die fortgeschrittensten sprachlichen Merkmale berechneten. Der Nachteil an diesen Systemen ist, dass die Erweiterung auf andere Sprachen mit einem grossem Aufwand verbunden ist. [14]
Erst mit dem Aufkommen von Deep Learning als konkurrenzfähige Methode im letzen Jahrzehnt kamen wieder Systeme mit weniger Preprocessing auf [4]. 2009 erreichte Grégoire Montavon mit Deep Learning für 3 Sprachen auf dem Voxforge Datenset eine Genauigkeit von 80.1% [15]. Das Voxforge Datenset hat sich seither verändert. Es kann angenommen werden, dass die hier vorgestellten Modelle, sein System deutlich übertreffen, da ähnliche Arbeiten dass getestet haben [23]. 2016 wurde mit einem sehr ähnlichen Modell wie das CNN Modell dieser Arbeit Genauigkeiten von 93% und 85% für 2 Sprachen respektive 4 Sprachen gemessen [23]. Hrayr et al. [11] stellen im selben Jahr ein CRNN Modell für einen Wettbewerb vor, dass zwischen 176 Sprachen mit 99.67% Genauigkeit differenzieren kann. Ein weiteres CRNN Modell von Bartz et al. [2] erreicht 2017 91% bei der Unterscheidung von vier Sprachen.
Die in dieser Arbeit vorgestellten System leisten vergleichbare Genaugigkeiten wie die genannten Modelle. 95% ist eine höchst zufriedenstellende Genauigkeit unter anbetracht das das Youtube Datenset nicht fehlerfrei ist. Das Datenset wurde bekanntlich nicht manuell gesäubert. Es können dementsprechend Aufnahmen ohne Sprache oder mit fremdsprachigen Interviews enthalten sein. Der Anteil von fehlerhaften Aufnahmen ist jedoch wahrscheinlich klein. Der Fall auf 80% Genauigkeit beim Librivox Datenset ist normal. Das Modell ist selbstverständlich für die Youtube und Voxforge Daten optimiert und nicht für Hörbücher. Die Librivox Daten haben zum Beispiel unterschiedliche Hintergrundgeräusche und Aufnahmeprotokolle. 80% bleibt in jedem Fall ein erfreuliches Resultat. Ein Zufallsgenerator würde im Kontrast nur 33% erreichen. Es kann also aus dem Resultat abgelesen werden, dass das Modell angemessen generalisiert.

Bis jetzt wurde das Modell nur an öffentlich verfügbaren Daten ausgewertet wo klar war welche Sprache gesprochen wurde. In der Realität soll das Modell aber helfen die Sprache von unbekannten Aufnahmen zu erkennen. In anderen Worten soll das Modell in der Praxis angewendet werden. Dafür wurde ein Interface programmiert wo Benutzer sich selbst Aufnehmen und das Modell damit abfragen können. Das Interface ist in Form einer Webseite auf den Schulservern frei verfügbar (Die meisten Mobilgeräte und Computer sind kompatibel): Die Umgebung bei Smart-

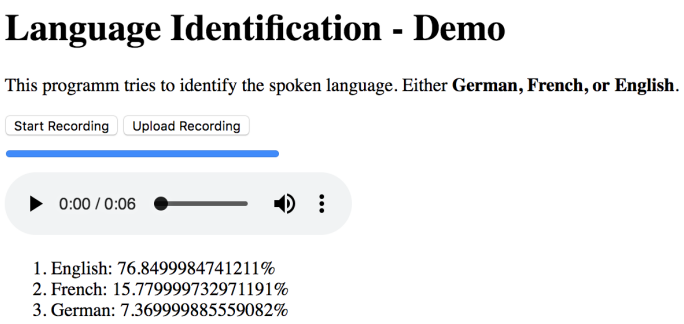

Abbildung 15: Interface Schnappschuss

phone Aufnahmen ist erneut eine andere wie die von Voxforge und YouTube. Die Genauigkeit die dort erreicht wird ist eine weitere Art die Generalisierbarkeit des Modells zu messen. Aus Neugier wurde ein sehr kleines Datenset aus Aufnahmen von Verwandten und Freunden zusammengestellt. Es wurde darauf geachtet, dass die Sprache deutlich verständlich ist. Insgesamt sind es nur 70 ungleichmässig verteilte Aufnahmen. Die Resultate daraus sind wegen der kleinen Anzahl und Varianz kaum signifikant. Das Programm entdeckt in 70% der Fälle die richtige Sprache.

[14] 1. Gut -> Vergleich - iLID 92.9% - crnn 91%, inception 95% 2. What you truly want -> Webserver 3. guru 4. it's bad -> improving 5. data augmentation, more power more data doesn't help, -> better data or model undercuts fine tuning more validation data

6.1 Zuverlässigkeit

6.2 Erweiterung

7 Anhang

7.1 Beispiel-Modell Code

```
# Importieren von Bibliotheken
from keras import models, layers
from kapre.time_frequency import Melspectrogram

# Definieren des Modells
architecture = [
    Melspectrogram(n_dft=512, input_shape=(1, 5 * 16000,),
                   padding='same', sr=16000, n_mels=28,
                   fmin=0.0, fmax=10000, power_melgram=1.0,
                   return_decibel_melgram=False, trainable_fb=
                       False,
                   trainable_kernel=False),
    layers.Conv2D(64, (3, 6), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 6), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 6), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(1024, activation='relu'),
    layers.Dense(3, activation='softmax')
]

model = models.Sequential(architecture)

# Zusammenbauen des Modells
model.compile(optimizer='Rmsprop',
              metrics=['accuracy'],
              loss='categorical_crossentropy')

# Trainieren des Modells
model.fit(train_data, train_labels,
          batch_size=64,
          epochs=9,
          validation_data=(val_data, val_labels))

# OUTPUT
# Train on 7500 samples, validate on 3750 samples
# Epoch 1/9
# 7500/7500 [==============================] - 17s 2ms/step -
    loss: 1.0368 - acc: 0.4888 - val_loss: 1.0159 - val_acc:
    0.5152
# Epoch 2/9
# 7500/7500 [==============================] - 13s 2ms/step -
    loss: 0.8200 - acc: 0.6319 - val_loss: 1.1726 - val_acc:
    0.4509
# ...
```

Index

Literatur

[1] *AlphaGo — Wikipedia, The Free Encyclopedia*. 2018. URL: https://en.wikipedia.org/wiki/AlphaGo (besucht am 19. 10. 2018).

[2] Christian Bartz u. a. "Language Identification Using Deep Convolutional Recurrent Neural Networks". In: *CoRR* abs/1708.04811 (2017). URL: http://arxiv.org/abs/1708.04811.

[3] Keunwoo Choi, Deokjin Joo und Juho Kim. "Kapre: On-GPU Audio Preprocessing Layers for a Quick Implementation of Deep Neural Network Models with Keras". In: *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*. ICML. 2017.

[4] François Chollet. *Deep learning with Python*. Shelter Island, NY: Manning Publications Co., 2018.

[5] *Deep Blue (chess computer)— Wikipedia, The Free Encyclopedia*. 2018. URL: https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer) (besucht am 19. 10. 2018).

[6] J. Deng u. a. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[7] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[8] *Gradient descent — Wikipedia, The Free Encyclopedia*. 2018. URL: https://en.wikipedia.org/wiki/Gradient_descent (besucht am 11. 09. 2018).

[9] Shawn Hershey u. a. "CNN Architectures for Large-Scale Audio Classification". In: *CoRR* abs/1609.09430 (2016). arXiv: 1609.09430. URL: http://arxiv.org/abs/1609.09430.

[10] Sepp Hochreiter und Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), S. 1735–1780. ISSN: 0899-7667.

[11] Hrant Khachatrian Hrayr Harutyunyan. *Combining CNN and RNN for spoken language identification*. 2016. URL: https://yerevann.github.io/2016/06/26/combining-cnn-and-rnn-for-spoken-language-identification/ (besucht am 15. 12. 2018).

[12] Sergey Ioffe und Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

[13] *Librivox*. URL: https://librivox.org/ (besucht am 24. 12. 2018).

[14] Kay M. Marc A. / Berkling. "Automatic Language Identification". In: *MIST-1999* (1999).

[15] Grégoire Montavon. "Deep learning for spoken language identification". In: (Jan. 2009).

[16] Michal Nilson. *Neural Networks and Deep Learning*. Dez. 2017. URL: http://neuralnetworksanddeeplearning.com/index.html (besucht am 21. 07. 2018).

[17] *NIST 2017 Language Recognition Evaluation*. URL: https://www.nist.gov/itl/iad/mig/nist-2017-language-recognition-evaluation (besucht am 24. 07. 2018).

[18] Tomasz Oponowicz. *Spoken language dataset*. URL: https://github.com/tomasz-oponowicz/spoken_language_dataset (besucht am 24. 12. 2018).

[19] Tariq Rashid. *n*. Heidelberg: O'Reilly, 2017.

[20] Mark Sandler u. a. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation". In: *CoRR* abs/1801.04381 (2018). arXiv: 1801.04381. URL: http://arxiv.org/abs/1801.04381.

[21] Julius O. Smith. *Mathematics of the Discrete Fourier Transform (DFT)*. online book, 2007 edition. URL: http://ccrma.stanford.edu/~jos/mdft/ (besucht am 11. 09. 2018).

[22] S. S. Stevens, J. Volkmann und E. B. Newman. "A Scale for the Measurement of the Psychological Magnitude Pitch". In: *The Journal of the Acoustical Society of America* 8.3 (1937), S. 185–190.

[23] Thomas Werkmeister Tom Herold. "Practical Applications of Multimedia Retrieval. Language Identification in Audio Files". In: (2016). URL: https://github.com/twerkmeister/iLID/blob/master/Deep%20Audio%20Paper%20Thomas%20Werkmeister%2C%20Tom%20Herold.pdf.

[24] *Voxforge*. URL: http://www.voxforge.org/ (besucht am 24. 07. 2018).

[25] Hsiao-Wuen Hon Xuedong Huang Alex Acero. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, 2001.

[26] *Youtube*. URL: https://www.youtube.com/ (besucht am 24. 07. 2018).

Abbildungsverzeichnis