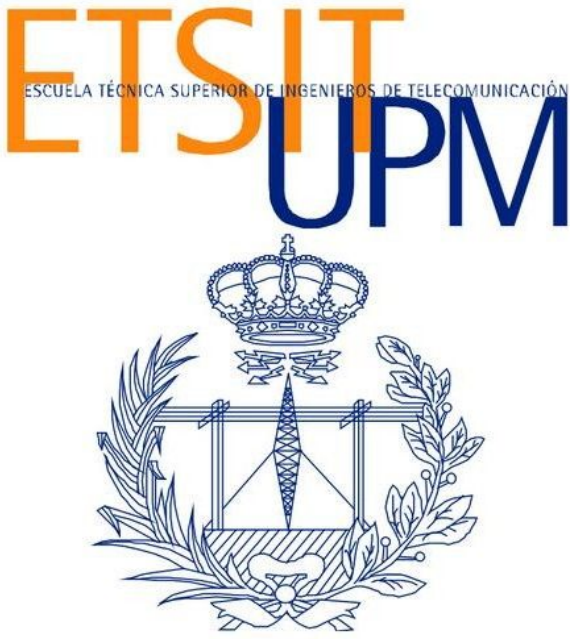
Deep Learning for Acoustic Signal Processing

Final Course Report

[Link to slides with audio]



Sergio Pérez Morillo

07/05/2020

Table of Contents

1. Literature Review

In this section, I present a short overview of the different subfields that encompass the ample field of acoustics. Then, I go through some examples of deep learning solutions that have been applied recently in some of the subareas I am most interested in.

1.1. Acoustic Application Fields

According to the Institute of Acoustics (IOA) [1], acoustics is the interdisciplinary science that deals with the study of all mechanical waves in gases, liquids, and solids including vibration, sound, ultrasound, and infrasound. Many taxonomies are found in the field but perhaps the most common is Lindsay's wheel of acoustics created by R. Bruce Lindsey (Figure 1) in 1964 [2].

The wheel defines the scope of this field from four distinct subfields (Earth Sciences, Life Sciences, Engineering, and Arts). It is also divided into an outer and inner circle to distinguish different disciplines and subfields. The outer circle represents disciplines that commonly define careers in acoustics. On the other hand, the inner circle describes subfields that the previous disciplines might lead to.

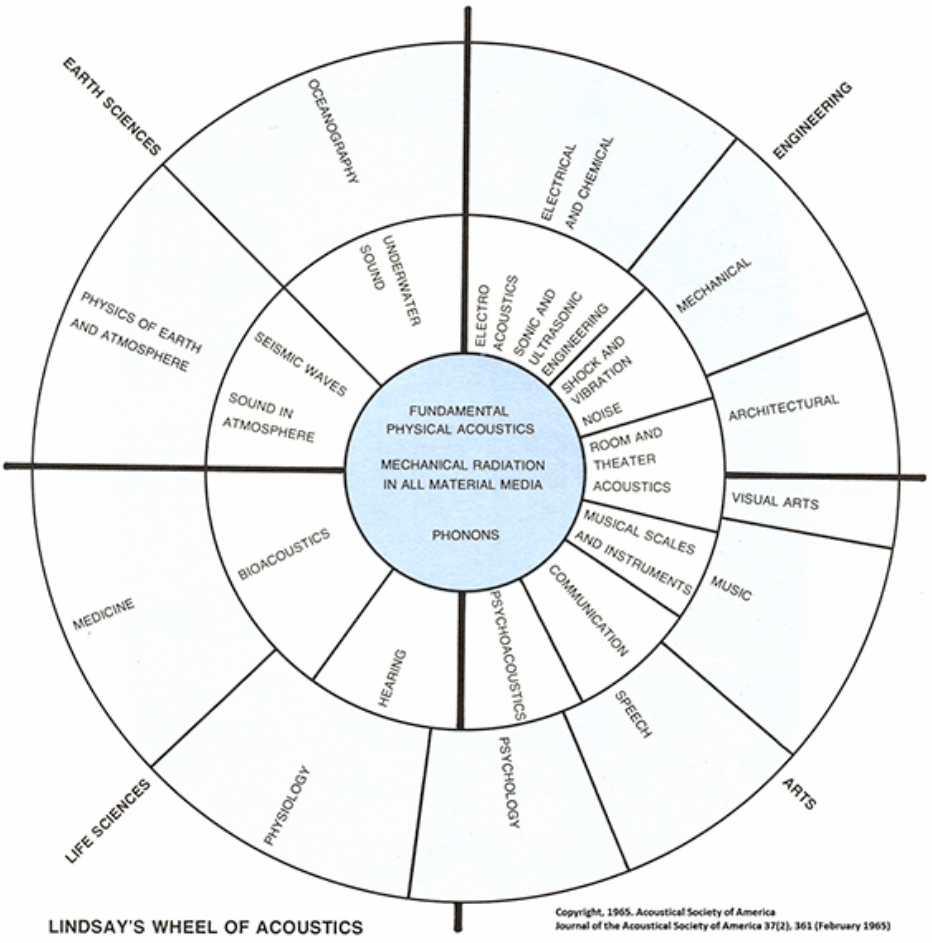


Figure 1 - Lindsay's wheel of acoustics [2].

As observed in Figure 1, the field of acoustics features plenty of different subfields. Bear in mind that they usually overlap. Some of the most important ones [3] that belong to this taxonomy are:

Archaeoacoustics

It is the discipline of acoustics related to archeology. For instance, professionals study the acoustics of caves that belong to prehistoric times, with natural sounds such as whistling. In particular, they are interested in the relationship between acoustics and ancient rituals. It is a fairly new discipline

Aeroacoustics

It is the discipline of acoustics related to sounds from air movement. A common example is a natural movement through fluid air caused by turbulence. The study of this sounds helps to mitigate the noise of aircraft. It is also used for studying wind musical instruments.

Acoustic signal processing

It is the discipline of acoustics related to the electronics procedure applied to acoustic signals. There are plenty of examples such as hearing-aid systems, echo cancellation, and active noise control.

Architectural acoustics

It is the discipline of acoustics related to sounds of buildings, so to speak. Professionals study aspects like vibrations in the infrastructure, privacy in buildings' apartments, or the quality of music.

Bioacoustics

It is the discipline of acoustics related to the hearing of animal calls and sounds of their habitats. For instance, there are examples of models that classify fish species using just their calls.

Electroacoustics

It is the discipline of acoustics related to the procedures involved in the recording, manipulation, and reproduction of audio. A common example is the mobile phone industry. A rather odd example is the acoustics of virtual environments.

Environmental noise and soundscapes

It is the discipline of acoustics related to sounds generated by different fields such as communications (railways, road traffic) or industry (aircraft and factories). Its scope is focused on improving the quality of environments by reducing vibrations or noise to increase tranquility.

Musical acoustics

It is the discipline of acoustics related to the sounds and physics of instruments. It is also related to the neuroscience impulses caused by these instruments.

Speech

It is the discipline of acoustics related to speech recognition and speech synthesis via computer capabilities. It is related to other fields such as linguistics, physics, psychology, and physiology.

Ultrasonics

It is the discipline of acoustics related to frequencies higher than the human hearing spectrum. There are many and exciting examples such as SONAR, medical ultrasonography, or material characterization.

Underwater acoustics

It is the discipline of acoustics related to human-generated and natural sounds found underwater. It features a broad range of applications from whale communications systems to sonic weapons.

Vibration and dynamics

It is the discipline of acoustics related to vibration and interaction of mechanical systems and their surroundings. Some applications are vibration reduction in theatres, vibration control to prevent earthquakes potential damage, and measurement of infrastructure sound moves in buildings.

1.2. Deep Learning in Acoustics

1.2.1. Audio Generation

Out of this wide variety of topics what interested me the most was audio, speech, and music. In particular, I wanted to study how one of the newest and more hyped up topics in Deep Learning, Generative Adversarial Networks (GANs), is being utilized in these areas. Here is what I found:

WaveGAN

Chris Donahue et al. proposed the application of GANs to the unsupervised synthesis of raw-waveform audio **[4]**. Their idea is to leverage the potential of GANs shown in image generation to apply it to audio. They suggest that the generation of images with GANs are successful in keeping both global and local coherence that in audio would translate to structures with different timescales needed for proper synthesis.

The implementation is based on DCGAN, a popular GAN implementation for images (shown in Figure 2). They were able to produce intelligible words with unlabeled data from a small speech dataset. Aside from this, they generated also some sounds of other domains such as pianos or bird vocalizations. The length of the synthesized audio was around one second and sampled at 16kHz.

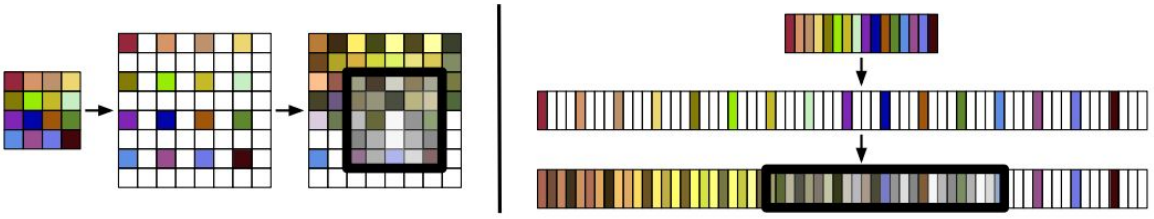


Figure 2 - Transposed convolution in the generators' first layers of DCGAN and WaveGAN [4].

GANSynth

Jesse Engel et al. proposed another GAN architecture for audio synthesis **[5]**. They suggest that it is a complicated task because human perception is sensitive to fine-scale waveform coherence and global structures. Some autoregressive architectures such as WaveNets capture the former at the expense of the latter and slower performance. On the other hand, GANs capture better the latter but have yet to show if they can do the same for the former.

The authors claim that they have accomplished this goal (generation of high-fidelity local coherent audio with GANs) by modeling instantaneous frequencies and logarithmic magnitudes. Using the NSynth dataset and after thorough empirical investigation, they have built models that outperform WaveNets in both automated and human evaluation metrics. Figure 5 shows for roughly similar models how the instantaneous frequency (IF) GAN is much more coherent than the waveform and phase GANs that feature more irregularities. This can be perceived in the strong colors of the harmonics in the bottom images.

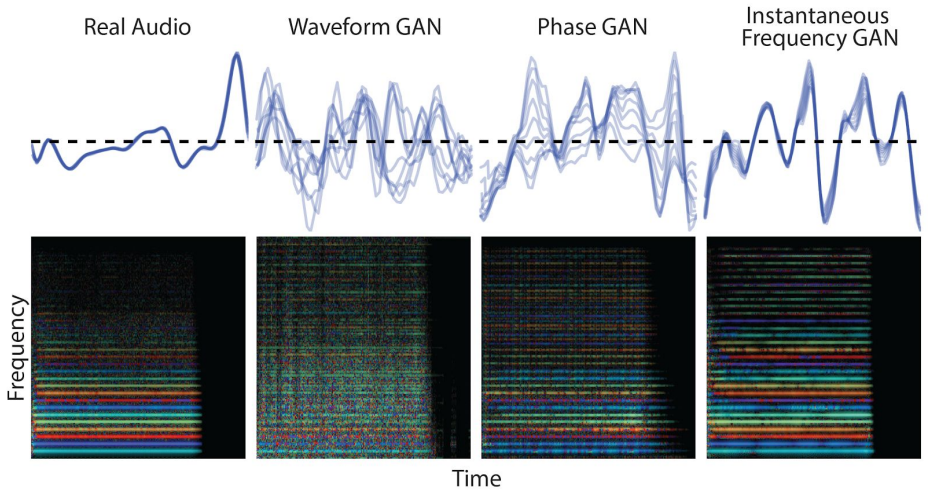


Figure 4 - Results of different parametrized GANs (waveform, phase, and IF) using similar models [5].

Differentiable Digital Signal Processing

Not every piece of work in Deep-Learning-based generated audio uses GANs. From the same first author as before, we have Differentiable Digital Signal Processing (DDSP) **[6]**, an architecture based on simpler neural network models (feedforward autoencoders) that seek to make regular digital signal processing utilized in communications, audio or medical technology differentiable.

Leveraging these well-known components, the authors achieve similar results as WaveNets or GanSynths with smaller (instead of parametrized whole audio samples they only parametrized DSPs' parameters) and more interpretable models (as a consequence of the former explanation). Figure 5 depicts how they devised it.

They add two synthesizers, one that subtracts white noise by filtering it with time-varying filters and another that adds sinusoids with different frequencies. Then, this is passed through a reverberation module to get the final output. The loss function is a combination of the target audio with spectrograms with different frame sizes. Theses enable backpropagation and gradient descent methods.

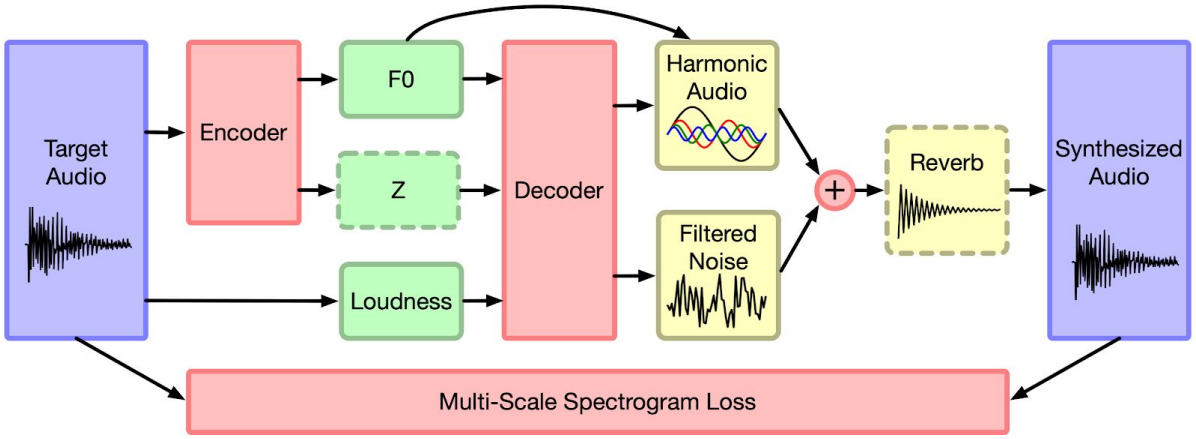


Figure 5 - Differentiable Digital Signal Processing architecture [6].

1.2.2. Speaker Recognition

Another topic I am interested in is speaker recognition. Unlike speech recognition, the former seeks to assign audio samples to speakers without looking at their actual words. Usually, the state-of-the-art models were done with i-vectors before an exciting bunch of new ideas based on Deep Learning were brought to the table. Here are some of them:

SincNet

As in the DDSP solution, Mirco Ravanelli et al. uses some sort of *a priori* in their solution **[7]** to reduce the number of parameters that large Deep Learning architectures usually need, this time for building a speaker recognition system. In this case, the *a priori* is a bank of filters that are based on sinc functions in which only high and low frequencies are parametrized (band-pass filters), thus reducing the number of adjusted parameters during backpropagation.

Doing so, they claim that the convergence was faster as these filters feature more meaningful information than CNN-like filters while improving their performance with smaller networks. Figure 6 shows the architecture (left) and why these filters have more relevant information (clearly seen in the first pair of filters). It is also worth noting that these filters avoid overfitting to datasets' nuances more than regular CNN filters.

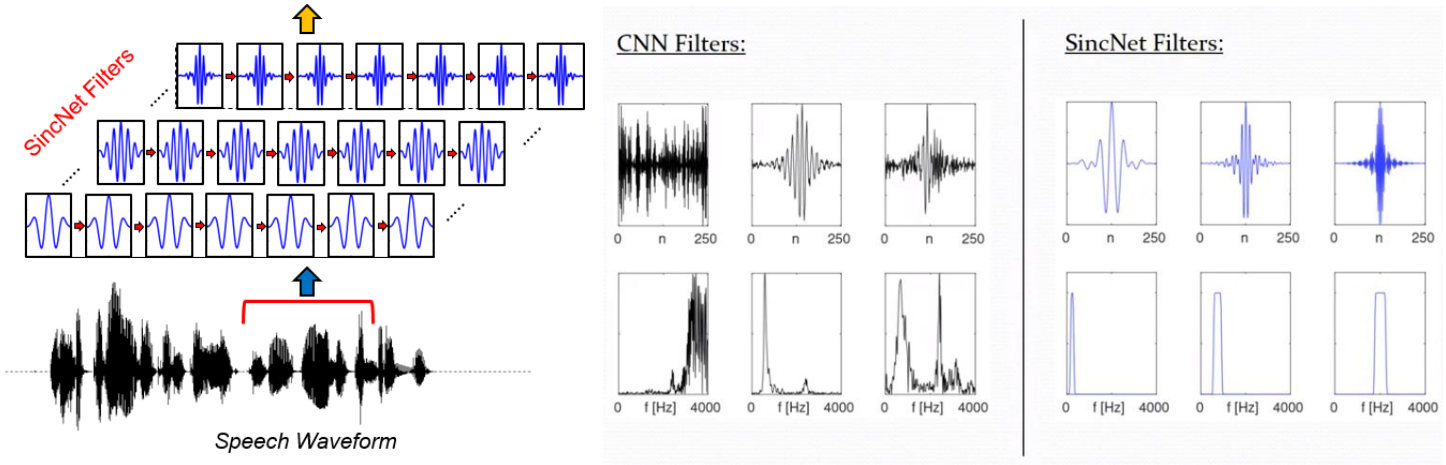


Figure 6 - SincNet's architecture (left) and regular CNN filters vs. SincNet's filters (right) [7].

Embeddings: d-vectors and x-vectors

Recent works in speaker recognition have centered around the idea of extracting features of audio samples using Deep Neural Networks (DNN). In short, this is performed by training a DNN and extracting the embedding (output) of one of its intermediate layers. This technique is utilized in many other Deep Learning fields. The most prominent of those related to audio tasks are d-vectors and x-vectors.

The former was introduced first. Ehsan Variani et al. propose a frame-level DNN that classifies speakers [14]. Figure 7 shows its architecture. The d-vectors are extracted from the last hidden layer during speaker enrollment. These are vectors are obtained from averaging frames. Each utterance outputs its d-vector and then it is compared to the enrolled speaker model in order to make a decision.

Their work was compared to i-vectors on a small task, another well-known technique used in audio feature extraction, though not based on Deep Learning. The results were superior as for instance the DNN architecture showed a more favorable behavior against noise injection. It outperformed i-vectors by roughly 20% in equal error rate for different inputs (raw and noisy).

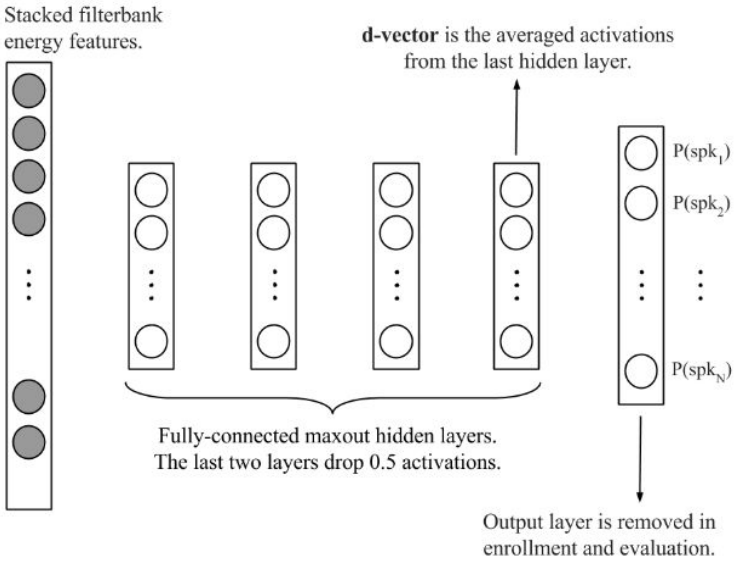


Figure 7 - DNN model from which the d-vectors are extracted [14].

On the other hand, the most novel approach to feature extraction is a variant of d-vectors, x-vectors. David Snyder et al. propose a Time Delay Neural Network or TDNN to extract the vector. A TDNN tries to model context at each layer of the network and classify patterns with shift-invariance, so it does not need explicit segmentation.

Figure 8 shows a simplified version of the architecture. The input features have 24 from applying a filter bank with a frame-length of 25ms. The frame-level part consists of five layers, each comprising a bigger time window. Then, the TDNN output of these frames from each utterance is averaged in the stat pooling layer to be fed in two consecutive fully-connected layers and the final softmax layer. The x-vectors are extracted from the last hidden layer, the second fully connected layer.

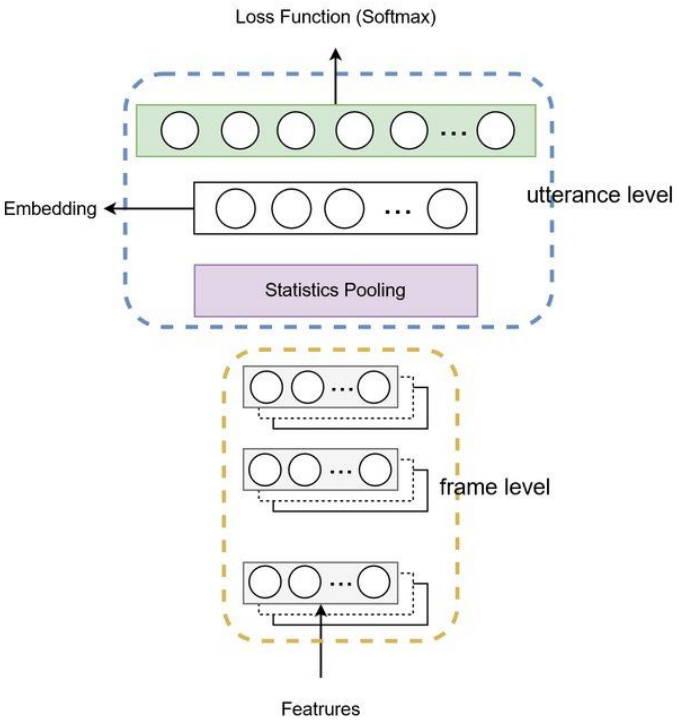


Figure 8 - TDNN model from which the x-vectors are extracted [16].

2. Project Description

In this section, I describe the idea behind this project, how I envision it. First, I set the goal I would like to achieve. Then, I give a brief explanation of the topic I based this project on. And finally, I illustrate the methodology that I follow to develop this project and in which the following sections are based on.

2.1. The Goal

After thoroughly analyzing not only theoretical resources but code implementations of state-of-the-art solutions such as WaveGan, DDSP, SincNet, etc. I come to realize that trying to replicate these works without previous expertise on the topic is quite complicated if I wanted to accomplish meaningful results and asses them correctly.

By doing so, I would have gotten stuck with just running experiments already devised by the author and any proposed change would have been difficult to asses. Instead, I decided to set the bar a little low, and use more general (yet, advanced) deep learning tools such as siamese networks or transfer learning (audio embeddings) to address this topic. Bear in mind, I have not discarded the first approach, just move it to the final part of the project if there is any time left to avoid getting stuck.

On the other hand, I also wanted to work with a novel, fun, appealing, small-to-medium size dataset instead of a large and overused state-of-the-art one. My reason was to speed up processing, training, and testing the data to try many different things and configurations, and maybe implement some parts in several frameworks. However, this also has drawbacks such as overfitting due to lack of data (though, it can be mitigated with data augmentation) and lack of resources and feedback from other projects that used the same data sources.

2.2. Speech Accent Archive

After evaluating, several speaker recognition options for the dataset **[8]**, I chose the Speech Accent Archive provided by Steven H. Weinberger and George Washington University and uploaded by Rachael Tatman on Kaggle **[9]**. It contains audio samples of English speech from speakers of 200 different native languages from 177 different countries. This enables the analysis of which native language has the closest accent to native English speakers and asses if it is possible to classify using a speaker recognition system. It is a small-to-medium size dataset containing 2138 audio samples that last roughly 19 to 35 seconds. All audio sample contains the exact same narrated speech. The text is the following:

“Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station.”

Another reason for choosing this topic is rather personal. I have been studying English since I was three years old. One thing that has always struck me during this time learning the language is pronunciation, fluency, and rhythm. Spanish speakers like to label themself as terrible English speakers, yet they manage to be competent at other skills such as listening or reading. So, this

seems to be an excellent opportunity to find out how good each native language speaker actually is. Though it is not a big sample size to draw strong conclusions on each native language accents, it is still fun to find which accents are close to each other.

Another good reason is that it includes other types of information about the speaker like age and sex. Hence, I can build many different classifiers to evaluate which human features are easier to categorize. It also helps to build more feasible classifiers. Native language classification might look a great and innovative idea but with a small sample size of the same speech, there is a real and strong possibility that it does not yield good results and thus, meaningful insights. So, the opportunity of building a sex classifier that has been proven successful in other works helps to assess whether bad results in native language classification come from the lack of quality data or a faulty implementation.

2.3. Methodology

As in previous projects in the subjects: *Predictive and Descriptive Learning* and *Machine Learning Laboratory* **[10][11]**, I have based my methodology on one provided by Aurélien Géron in his book *"Hands-On Machine Learning with Scikit-Learn & Tensorflow"* **[12]**. However, this time I have condensed a little bit of the different sections to be more concise and avoid redundancy and smaller details of previous works.

Project Framing & Literature Review

These two parts have already been developed in this section and the previous one. This new understanding of the topic helps to develop intuition about problems that arise in the following sections. It will ultimately impact the quality of the results and speed up the learning process in this area.

Exploratory Data Analysis

This section combines both the data description and the exploratory data analysis (EDA). It also briefly mention some minor data wrangling. An EDA helps to acquire new insights. I will use distributions, correlations, and some visualizations of audio features.

Data Preprocessing

Audio can be processed in many different ways prior to being fed into a Deep Learning model. In this section, I used techniques such as MFCC, and Deep Learning Audio Embeddings. I also take a look into different techniques of data augmentation as a lack of data might be a central issue. This usually is a key step in modeling an audio problem and I expect to learn many new concepts from here.

Deep Learning Modeling and Results

Once the data is prepared, the implementation of different Deep Learning models is carried out. I focus mainly on Convolutional Neural Networks and Siamese Neural Networks as I think they fit best in this problem. Other architectures are also considered. Aside from these, other aspects of

Deep Learning are also addressed here such as data splitting (quite relevant in Siamese Networks), and hyperparameter-tuning.

I also summarize the final results yielded by the different Deep Learning architectures and draw some conclusions from them. I will also play a bit with embedding 2D and 3D representations that will help to evaluate part of the goal of this project, analyzing how English accents are related between native speakers of different languages. Not only that, but I use it also in other speaker features such as age and sex.

3. Exploratory Data Analysis

In this section, I take a look at the dataset for the first glance of its data. First, I described the information provided about the speakers and then about the audio files. Bear in mind that I skip the part about wrangling the data of the former as it is not really relevant. There were some minor changes but it would have made this document unnecessarily longer that it should have.

3.1. Description

Apart from audio files, the dataset also features information about the characteristics of each speaker. From those, I created other features based on them that were useful to further obtain insights. The feature *filename* is omitted as is only for connecting the speaker's information and their corresponding audio file. Here is the full list:

Age & Age Range

Description: The age of the speakers (featured) and range of the ages as a categorical feature (implemented).

Comments: The age of speakers as seen in Figure 6 is not balanced across a normal human lifespan (~80 years). There are more samples from adult speakers (namely young adults) than the rest of the spectrum. This might be a problem modeling age as the most distinctive voices (kids, adults, and elderly) are really unbalanced.

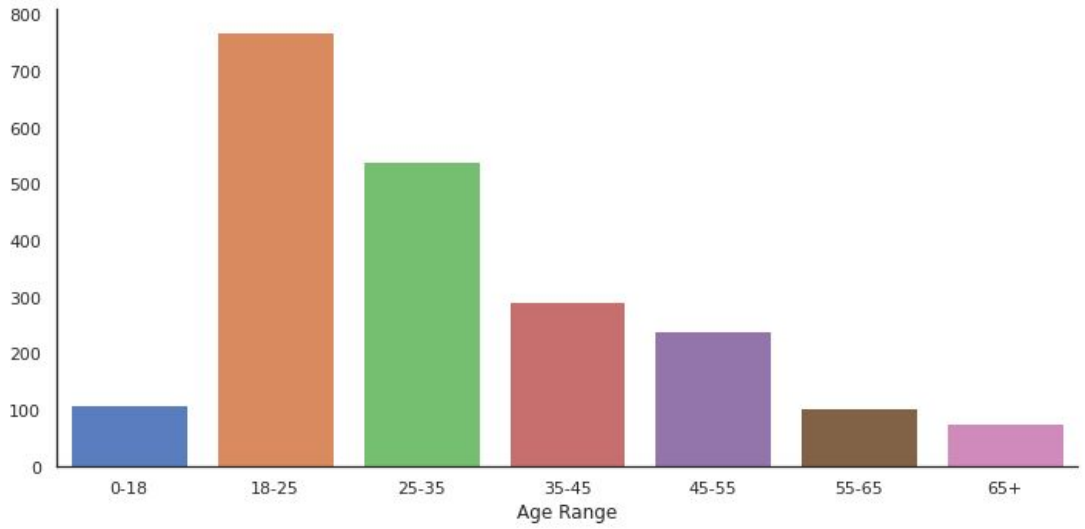


Figure 9 - Age distribution of speakers from age_range feature implemented from age feature.

Sex

Description: The sex of the speakers (featured).

Comments: This time, the feature is actually balanced. This ultimately helps to model better a sex classifier. Moreover, sex has already been proved as a feasible feature to model. So I expect a rather easier implementation of a model that yields meaningful results.

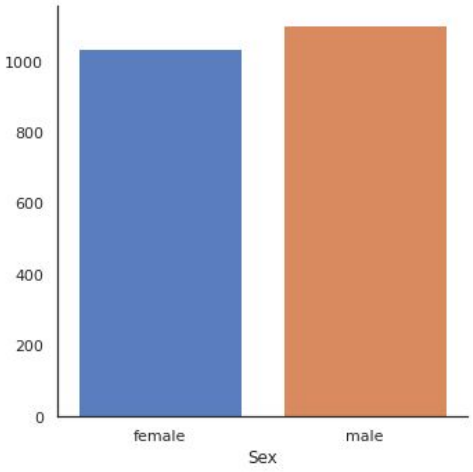


Figure 10 - Sex distribution of speakers from the sex feature found in the dataset.

Native Language

Description: The native language of each speaker, featuring both native and non-native English speakers (featured).

Comments: Figure 8 shows the top ten native languages found in the dataset but there are actually 200 different languages, a bigger number than previously anticipated. After the first 20 languages, no other language feature more than 20 audio samples. This might be a problem, as the dataset is heavily biased to native speakers and to a lesser extent to Spanish and Arabic speakers. I expect a harder time modeling this feature than modeling the previous one.

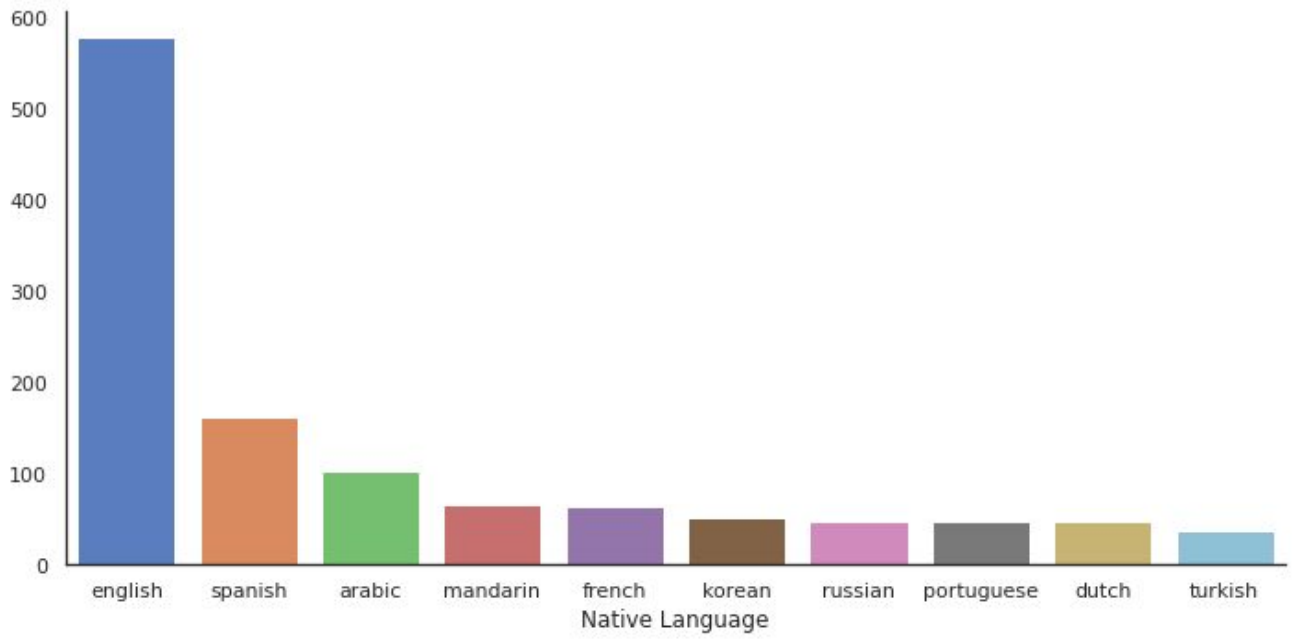


Figure 11 - Native language distribution of speakers of the top ten most featured languages.

Native English Speaker

Description: Whether the speaker is a native English speaker or not (implemented).

Comments: To mitigate the problem of the previous feature, I devised a new one based only on whether the speaker is a native or non-native English speaker. Although the feature is still unbalanced, the information is more concentrated and still shed some light on the relation of different native and non-native speakers.

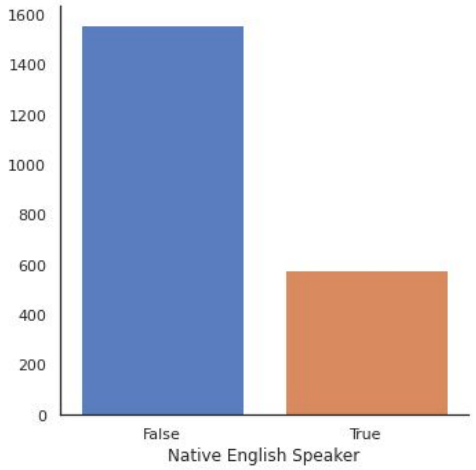


Figure 12 - Native English distribution implemented using the feature native language.

Home Country

Description: Home country of the speaker (featured).

Comments: Alternatively to modeling native language, the home country of the speakers can be modeled. After wrangling the data, there are 176 different countries. By looking at them instead of the native language, we can also model accent from different countries with the same language, e.g.: Spain and Mexico, or the USA and UK. Though, we lost some unity doing so. For instance, Figure 11 shows that Spanish speakers have diluted so much that they do not even feature the top ten.

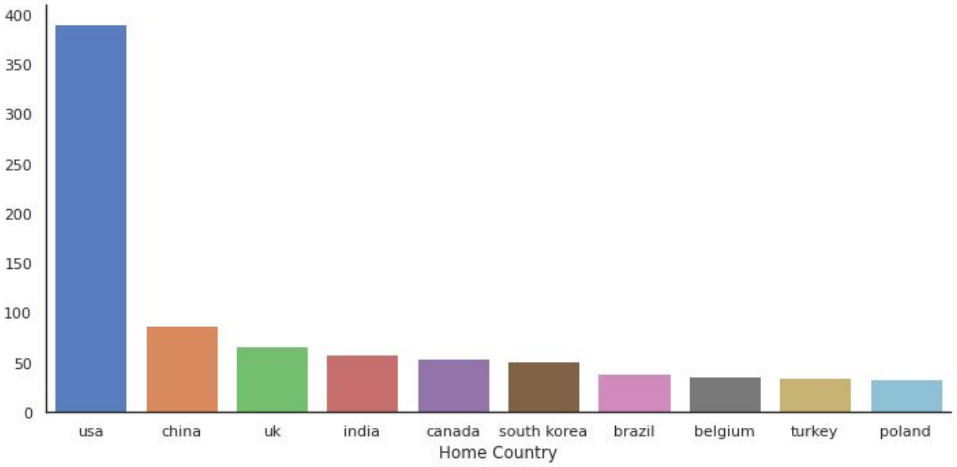


Figure 13 - Home country distribution of the top ten most featured countries.

Birthplace

Description: Birthplace of the speaker (featured).

Comments: The last feature is their birthplace. There more than 1.200 different places so it is modeling looks, at first glance, not feasible without using a siamese neural network, for instance. It has some advantages though. There is not a birthplace that dominates, rather a more even distribution of values. It is worth noting that there is not a native English place among the most featured cities, as seen in Figure 11. The modeling of native English countries using their birthplace is also a possibility.

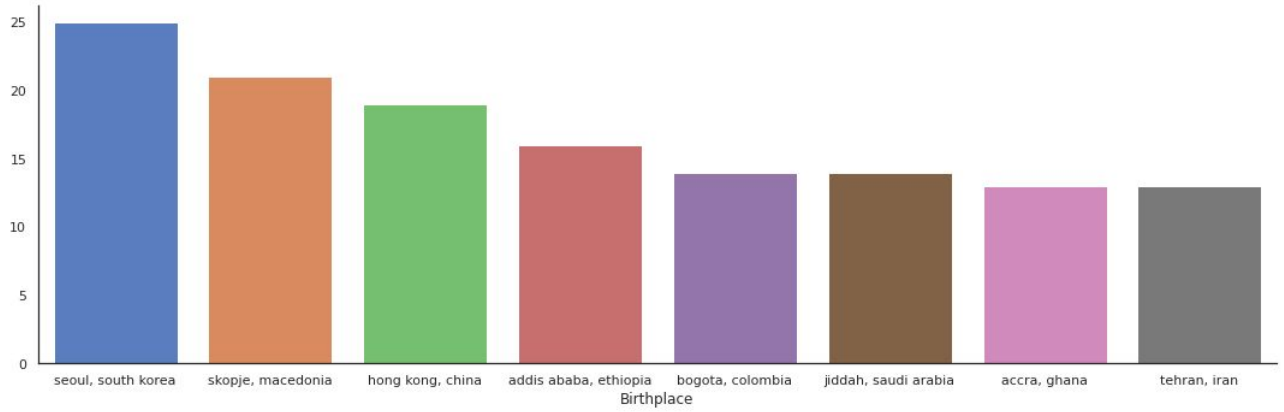


Figure 14 - Birthplace distribution of the top eight most featured cities.

3.2. Findings

After analyzing the information about the speakers independently, I show some graphs of correlations between them to check if there are possible biases, that could worsen the results of the model (e.g. if all native English speakers are female, there would be bias and hence the results might not be reproducible).

Age distribution of most featured native languages

Figure 12 shows the age distribution of the top-seven most featured native languages in a violin plot. Most of them have a different shape and range. While it is not flagrant this might induce some bias to models. The good thing is that the top-three languages (English, Spanish, and Arabic) have more or less a similar shape and range.

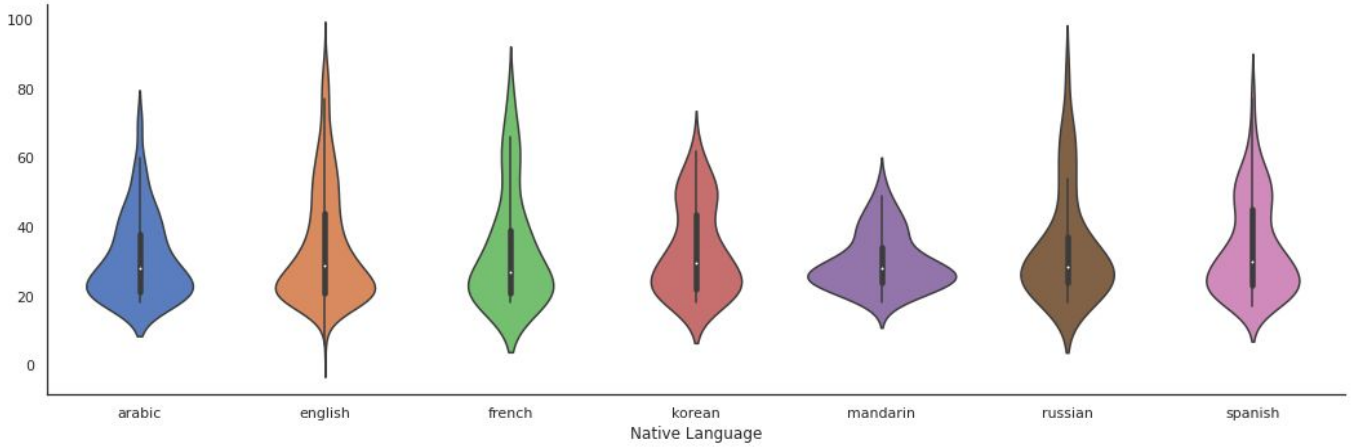


Figure 15 - Age distribution of the top-seven featured countries.

Sex distribution of most featured native languages

Figure 13 depicts the example stated in the introduction of this subsection. It shows that the most featured countries have even sex distribution with no clear bias which is what I looking for in this graph.

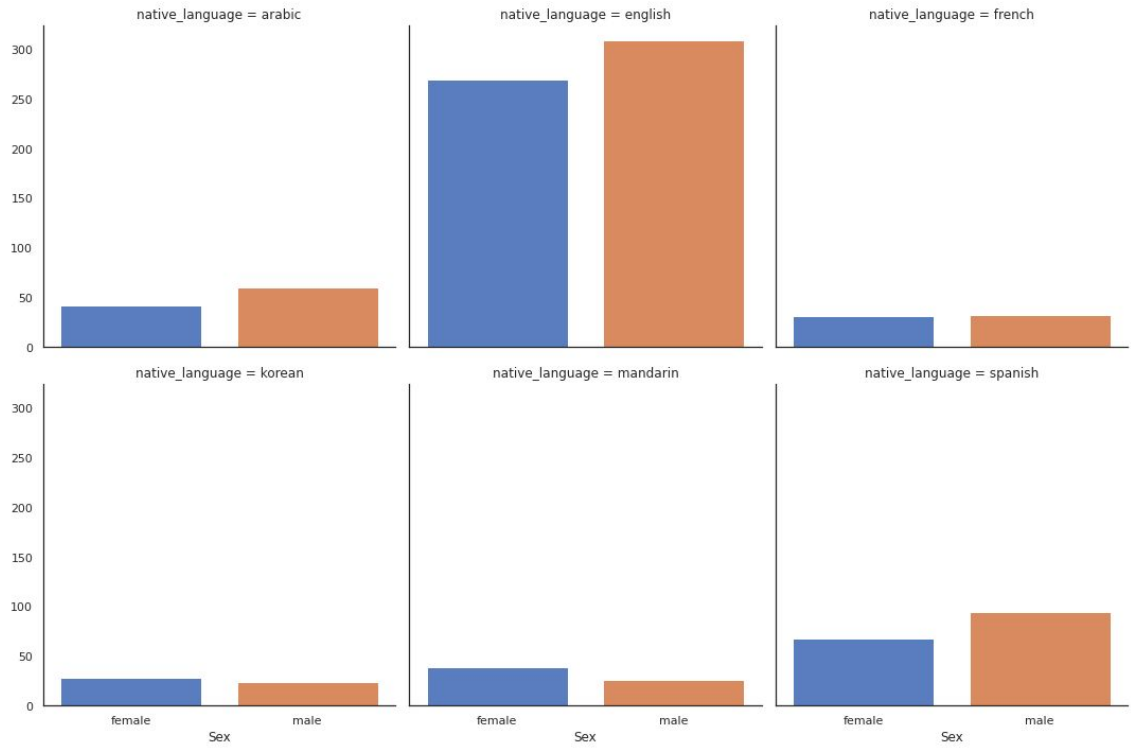


Figure 16 - Sex distribution of the top-six featured countries.

Birthplace of native English speakers

As mentioned in the previous section, it might be interesting to model the birthplace of English speaking countries to assess how close are accents from each place to the rest. Even though English is the predominant language in the dataset, its birthplaces are diluted (hough evenly distributed) with more than 82 places with at least two samples but less than 20 with more than three. It ultimately hinders its possible modeling.

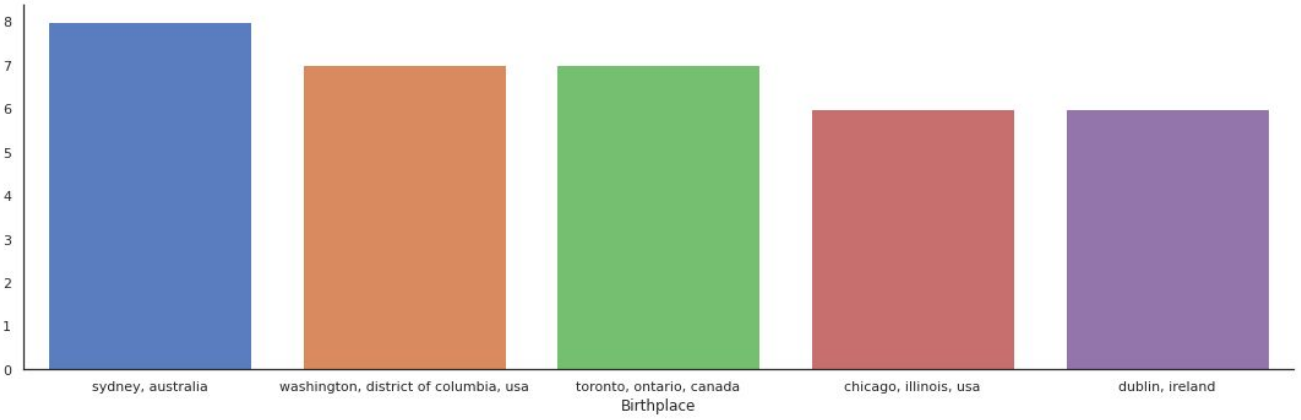


Figure 17 - Distribution of the top-five most featured native English birthplaces.

Native languages of home countries

Even though I mentioned that using countries might help to distinguish accents from different countries such as the USA and UK, the same reasoning is applied to the counterexample. For instance, Figure 16 shows the distribution of the native languages of Spain. So, while the feature country segregates accent from countries as opposed to the feature native language, it also aggregates native languages as in this example. So, in reality, there is a trade-off.

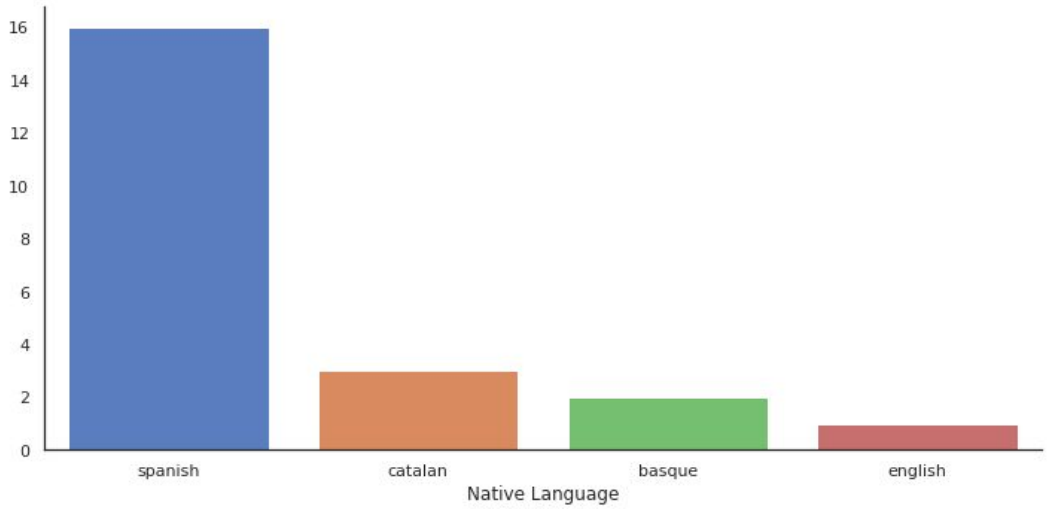


Figure 18 - Distribution of native languages of Spain.

3.3. Audio-based Analysis

Apart from the information about the speakers, there are also the actual audio files. The total count of audio files is 2.138 and their duration is roughly within 20 seconds and 35 seconds depending on the pace and fluency of the speaker.

Most of the techniques used in this subsection will be described in more detail in subsection 4.1 Data Preparation as they are utilized to extract audio features for later feeding the models. Nevertheless, we will take a brief look at some of them.

Audio comparison based on sex

Perhaps the easiest way to find differences in this dataset is by comparing two speakers from different sex. To add a degree of difficulty and make this comparison more reproducible, I select two speakers with similar age and birthplace, so their more distinctive feature is their sex. The audio files corresponding to the speakers are *english76.mp3* and *english483.mp3*. They are an 18 years-old female and a 19 years-old speaker respectively that have the same birthplace: Boston, Massachusetts, USA. The duration is also almost similar, 18 seconds and 19 seconds respectively.

First I look at the raw audio file. I use a sample rate of 16.000 Hz which is later used in generating audio embeddings from their wav file version. Figure 17 shows the complete audio in the time domain. Here, some similar patterns are appreciated in both speakers but more distinctive perhaps might be the amplitude of the male speaker around the six-second mark.

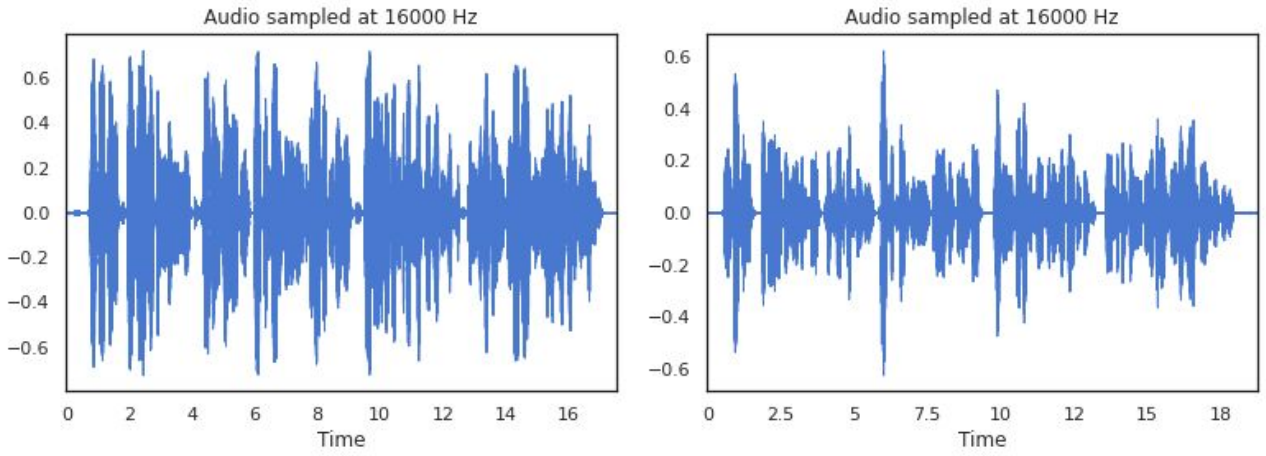


Figure 19 - Raw audio files from the female and male speakers respectively.

Then, I have plotted the Log Mel-Spectrogram. I have utilized 132 components for printing the first five seconds. Looking at the top part of both graphs, there are higher values for the highest frequencies for the female speaker which correlates to biological studies that demonstrate that female voices reach higher frequencies than male voices.

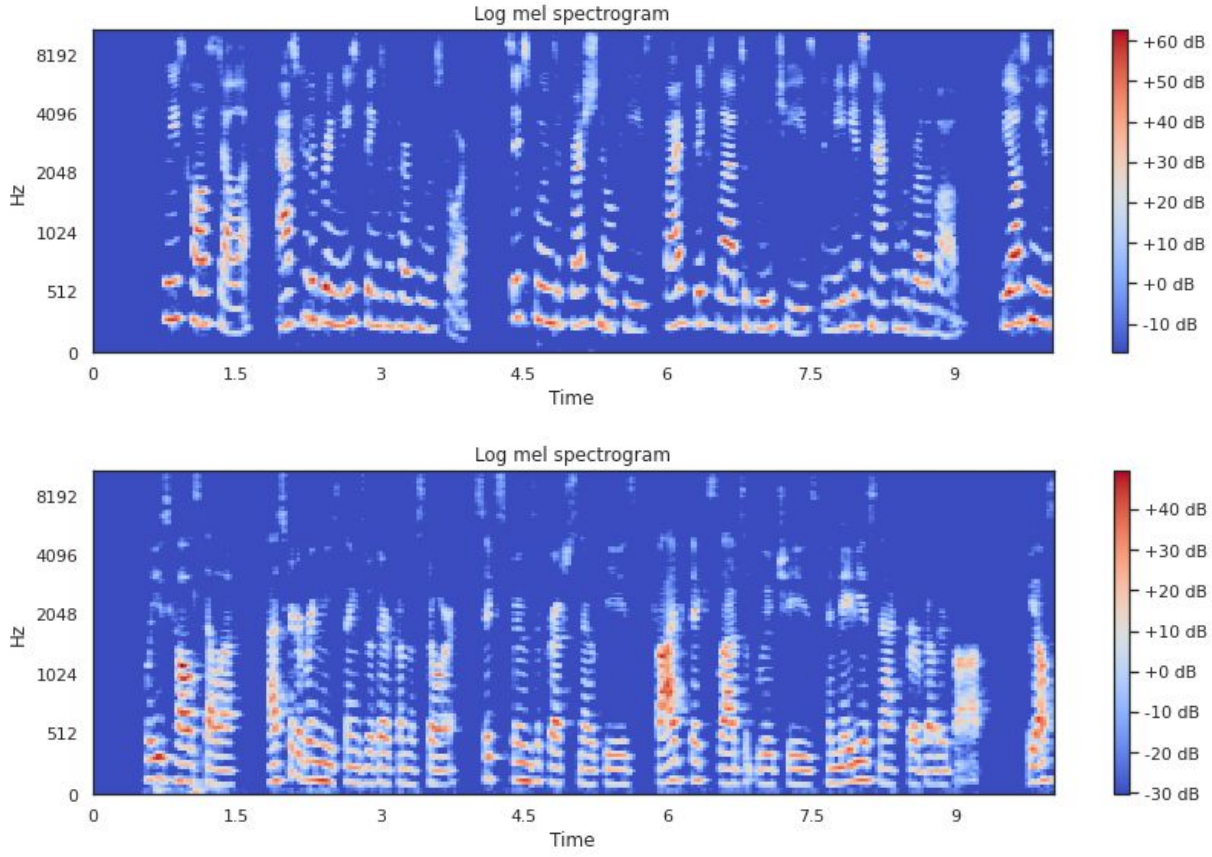


Figure 20 - Log Mel-spectrograms from the female and male speakers respectively.

Now, I look at Mel-frequency Cepstral Coefficients or MFCC. I use only just five components and a short period of time (five seconds) to see better the differences between them. On the first component, the patterns are more distinctive (looking at the blue parts) while the last component is more similar between both.

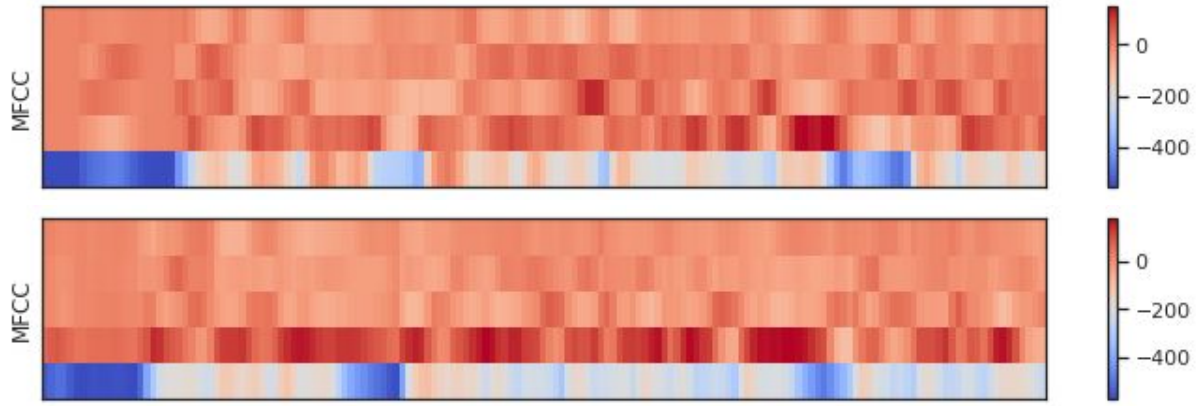


Figure 21 - MFCC plot using five components from the female and male speakers respectively.

Finally, I use a chroma plot of these audios. Even though it is utilized in music mostly, I thought that it might be interesting to see the results apply to speech recognition. It is only applied to the first five seconds of the audio. Visually it looks like the most distinctive plot. The female speaker has clearly lower values across the spectrum. We will see later on if it is also useful in deep learning modeling.

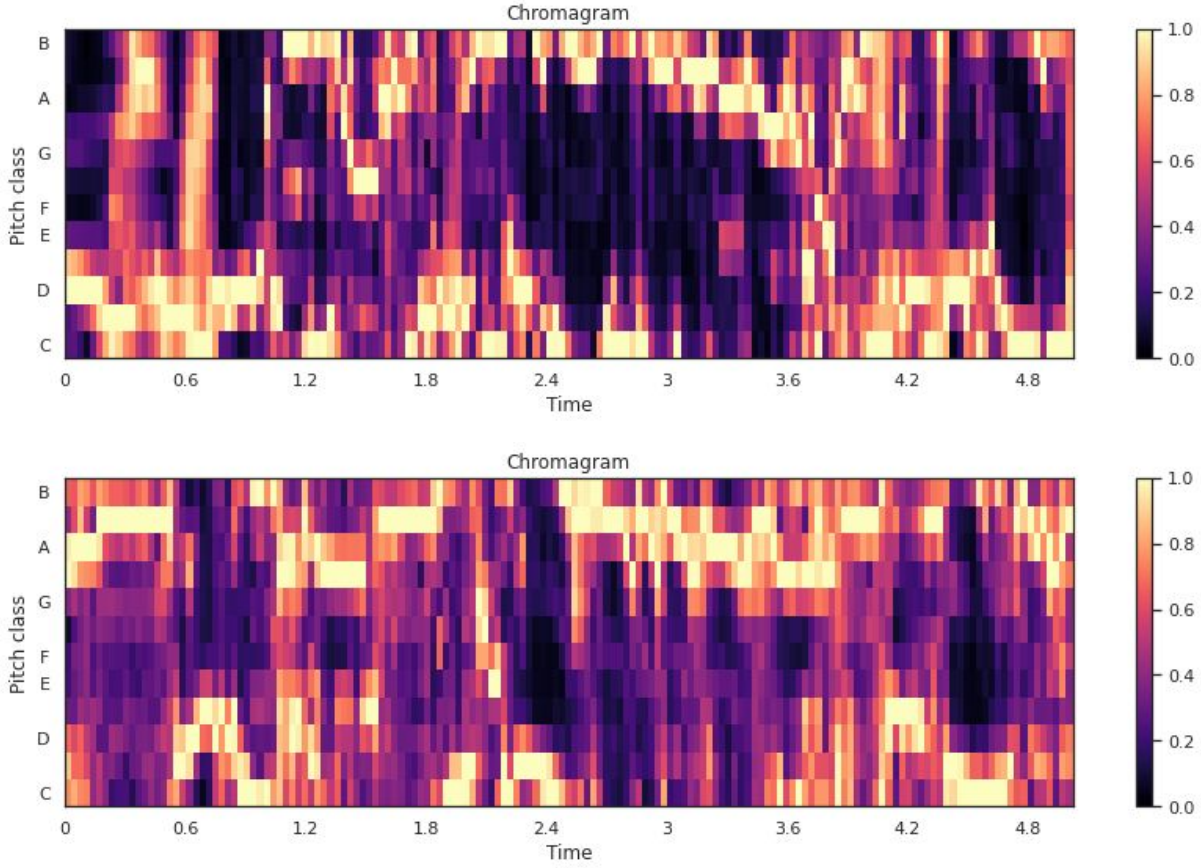


Figure 22 - Chronogram from the female and male speakers respectively.

4. Data Preprocessing

In this section, I describe how the data was processed before feeding them to deep learning models. First, I comment on different techniques to extract audio features, that were previously mentioned in section 3. Then, I explain some methods to augment data in audio applications.

4.1. Feature Extraction

There are deep learning models such as Wavenets, that use raw audio as inputs. While this is a viable solution, it also very computationally expensive. For extracting audio features in a more efficient way, there are other techniques that help to mitigate this problem.

Raw Audio

First of all, I devise a strategy to load and manage the audio files. This is a really important part since the project is developed using Google Colaboratory which is well known for its great GPU speed for a free solution but also for its dead slow I/O speed. So being efficient in the first steps saves a lot of time later on.

The quickest way possible to load the audios files that I found was to upload a zip file with them on Google Drive, then move them to the remote computer file system and unzip them there. This procedure saves a lot of time when trying different ways of processing the audio, without it some parts would have been unfeasible. Then, to load each audio file, I utilized the library SciPy instead of Librosa which on average was more than 20 times faster.

The sampling rate was 16kHz, a common sampling size for the audible spectrum, and also meet the required size of the audio embeddings of the next section. Aside from loading the files, I also use raw audio as the input of neural networks. The problem was that I ran into out of memory errors (OOM) as the tensors were especially large due to the sampling. To solve it, I used just around 6 seconds of each file and try increasing batch sizes starting from one to check how much memory was left.

Common Techniques

As shown in section 3.3, I implemented different common audio feature techniques used in the field. First, I tried Mel-frequency Cepstral Coefficients or MFCC. I used the code implementation provided by Librosa. I set the number of components to 20. As seen in other implementation this number goes from 12 to 40 coefficients. I wanted a number enough to use 2D-CNN layers properly but not large enough to slow down the preprocessing, training, and testing. I also reduce the audio to 13 seconds which gives an image dimension of 20 by 391. MFCC was the fastest technique of the three.

Then, I tried Log-Mel Spectrograms. I again used the code implementation provided by Librosa. I set the number of coefficients to 128, which was used in other works. I would have liked to try different coefficients to see how they affect performance, but some models had more than 20 hyperparameters to set (without counting the neural network architecture) so some of them had to be fixed from the beginning and these were the most “standard”, so to say. This technique was a

little bit slower than the previous one. The shape of the spectrogram was 128 by 391 after reducing the audio to 13 seconds as in the other technique.

Finally, I tried Chroma, although it is used in music applications, I wanted to give it a try just for fun. I again utilized the code implementation developed by Librosa. There weren't any hyperparameter involved in the process. It was the slowest technique by far, around five times slower to be precise. This fact prevented me to try some models' configurations that increased the computational time. Moreover, it did not yield the best results. The final shape was 12 by 391 again after reducing the audio to the same size as the others.

Audio Embeddings

Another novel option to extract audio features is deep learning embeddings. The first layers of a neural network learn the low-level features of the data while the last layers learn the high-level feature. By leveraging this characteristic, we can use enormous and successful deep learning architectures trained beforehand to extract these low-level features (the embedding) that usually are common to all data sources of the same field, this is also called transfer learning.

The embedding that I have used is called VGGish **[13]**, made by the Google AI team. It is a 126-dimensional embedding trained using TensorFlow on the Youtube dataset, a preliminary version of the Youtube-8M dataset that features more than 350,000 hours of video, 2.6 billion audio/visual features for 3,862 different classes. The architecture is based on the well-known VGG-16, hence the name.

The model was trained using wav files with audio resampled to 16 kHz mono, spectrogram computed with a Short-Time Fourier Transform (STFT) with a window size of 25 ms, a window hop of 10 ms, and a Hann window.

Then it computes a mel-spectrogram to a using 64 coefficients and the human frequency range. Then, the logarithm is applied with an offset of 0.01 to avoid problems. The resulting features are framed into non-overlapping examples of 0.96 seconds (though this can be parameterized). Each example covers 64 mel-bands of 10 ms.

The embedding is extracted from the last but one fully connected layer. Although it is a fully connected layer, the size of the embedding vector is actually bidimensional for each audio file. The first dimension of the embedding depends on the hop size and the audio sample size (thus, the sample rate and duration). The second dimension as previously stated is 128, as the number of fully connected neurons.

So far, I have used embeddings of sample rate 16000 Hz, duration five seconds, and a hop size of 100ms. This results in a shape of 41 for the first dimension. I decided to start with a big hop size and a small duration because the extraction would have taken too long in other cases and I would have needed bigger models later on.

However, this might result in poorer results. In the future, I might extract bigger embeddings. An example of the resulting embeddings using the same two speakers as in subsection 3.3 is found in Figure 21.

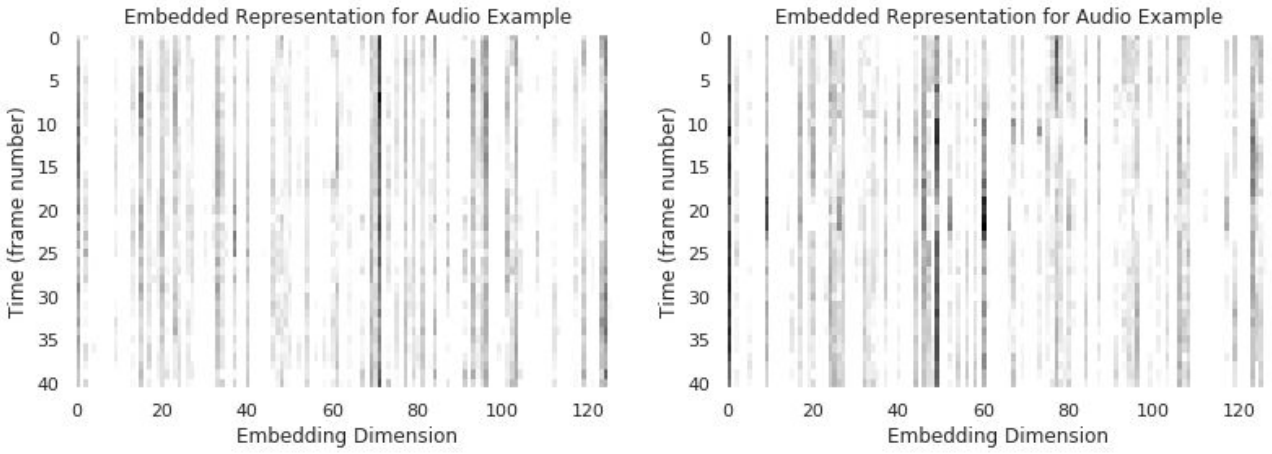


Figure 21 - Audio embedding extracted from VGGish of the female and male speakers respectively.

As I mentioned, an embedding aims to extract low-level features. However, this fact does not mean that these low-level features do not already include relevant information about our own labels. There are many examples in Deep Learning works that use the embedding with no further Deep Learning modeling to plot it by reducing its dimension to two or three with a dimensionality reduction method such as PCA, LDA, or t-SNE. This process helps to gain more insights about the data and asses what kind of labels in our data are more easily to model.

Figure 22 shows the representation of the VGGish embedding reducing its dimension to three using t-SNE. Before doing so, I compute the mean of each dimension of the embedding to get a vector of 128 components as these algorithms need a vector (a one-dimensional array) to work. Another option is to flatten it but it is more computationally expensive.

In the figure, it is clear that the label sex is really well separated into two clusters just by using the embedding. I expect a really easy process modeling the speaker's sex. In fact, it is already almost perfect and further modeling might not be necessary at all and end up overfitting the model. On the other hand, the native language of the speaker does not have any clear clusters and further modeling is needed.

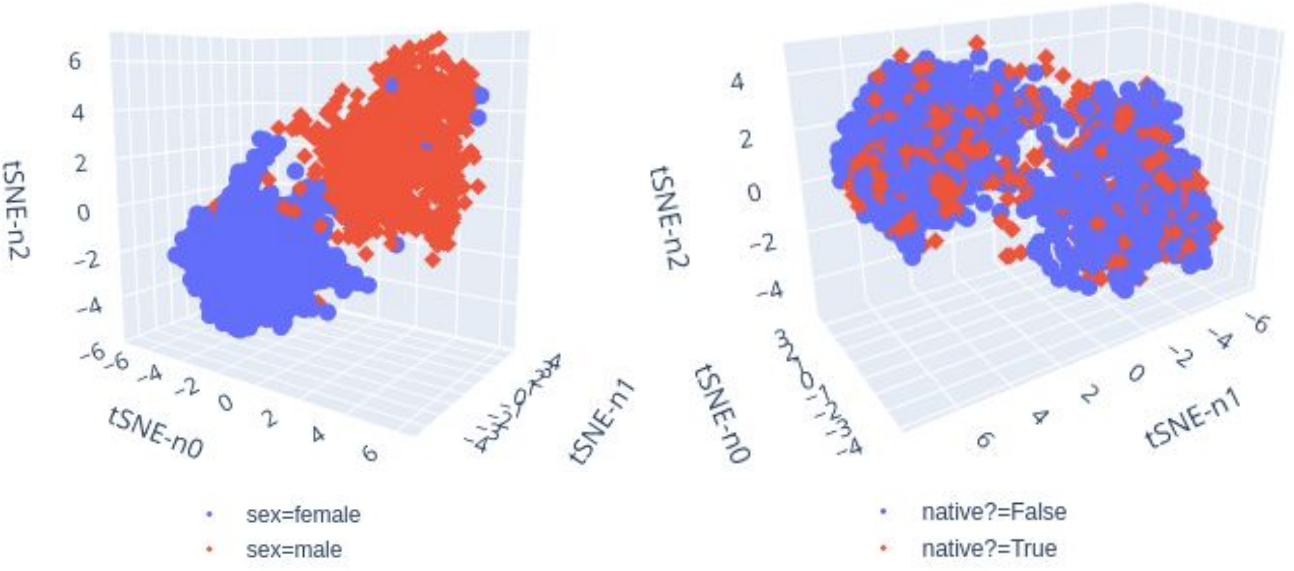


Figure 22 - Embedding representation using t-SNE for sex and native English speaker labels.

4.2. Data Augmentation

One of the main drawbacks of this dataset might be the size. This could translate into overfitting in the models which invalidate the results. To mitigate this, I decided to use techniques for augmenting the data specifically for the audio field. Here are the methods that I implemented.

Noise Injection

The most common data augmentation technique in audio is Noise Injection. By adding white noise to raw audios, the model further improve their robustness again small variations. To do so, I generate a vector of the same size as the cropped audio with the standard distribution N(0,1).

To scale properly the noise of each audio file and avoid making audio with lower amplitudes undistinguishable, I computed the signal to noise ratio and multiply it by a coefficient and the noise signal before adding it to the audio signal. The equation below summarizes this reasoning. Figure 22 shows the effect of these techniques.

$$S_{noisy} = S + \alpha * N * SNR$$

$$SNR = \frac{\Sigma_{n=0}^{N} S_n^2}{\Sigma_{n=0}^{N} N_n^2} \qquad \alpha \subseteq (0, 1)$$

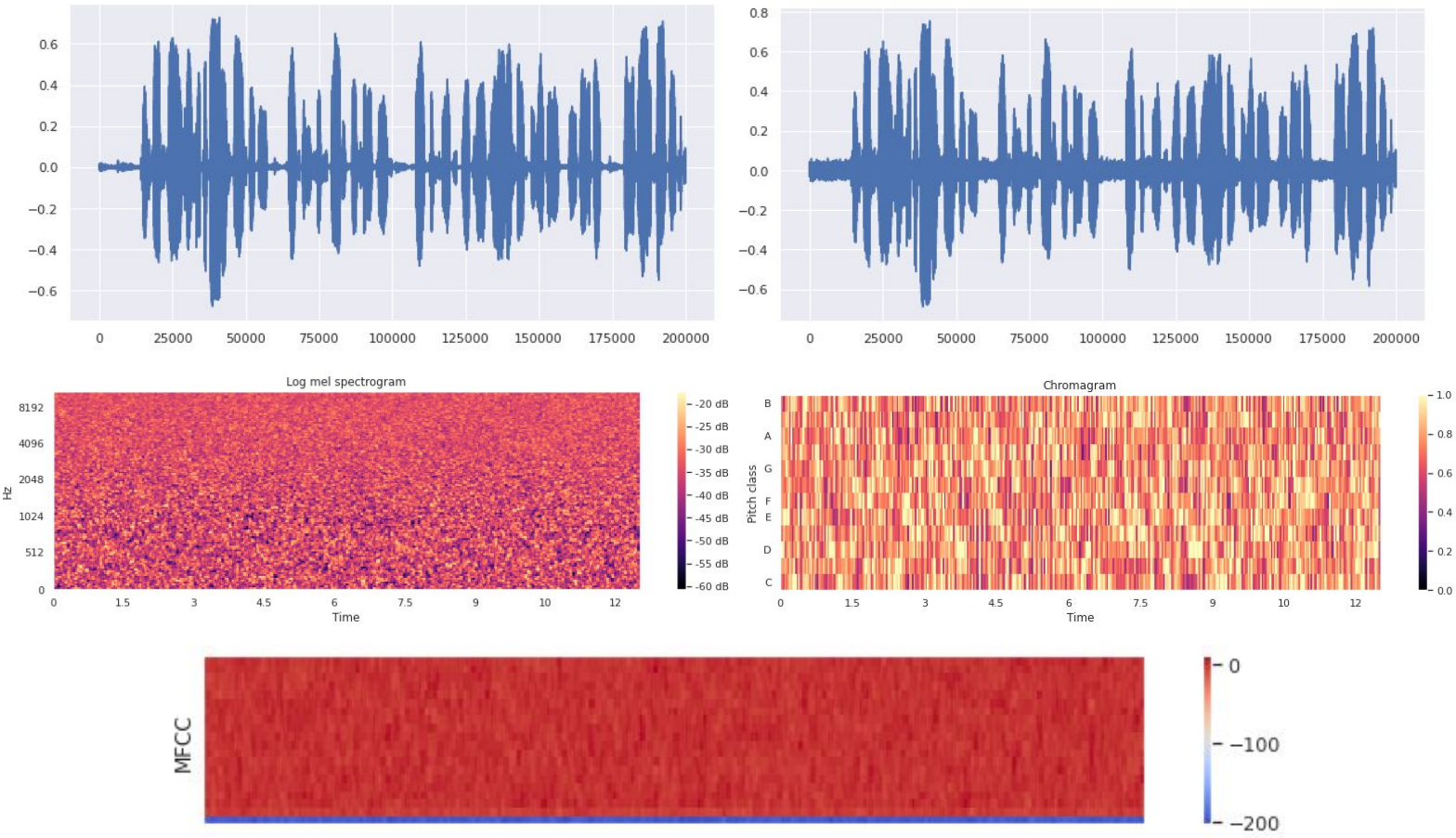


Figure 23 - Raw and noisy audio signal, and noise signals of each feature extraction technique.

Time Shifting

Time shifting is pretty straightforward. It is the application of a time offset to move the signal across the time domain. This technique was used with the audio embeddings described in the previous section. I have extracted different embeddings of five seconds that start at the zero-second mark and are displaced with an offset the 2.5 seconds (i.e. the starting time is 0, 2.5, 5, 7.5, 10, etc.).

Figure 23 shows two embeddings with a different offset from the same audio. It is really interesting to see that both follow the same pattern in every embedding dimension (vertical lines) but each of them has a different intensity (right feature has more intensity or higher values). It was also used in the other common feature extraction techniques, though just offsetting once the start of the audio.

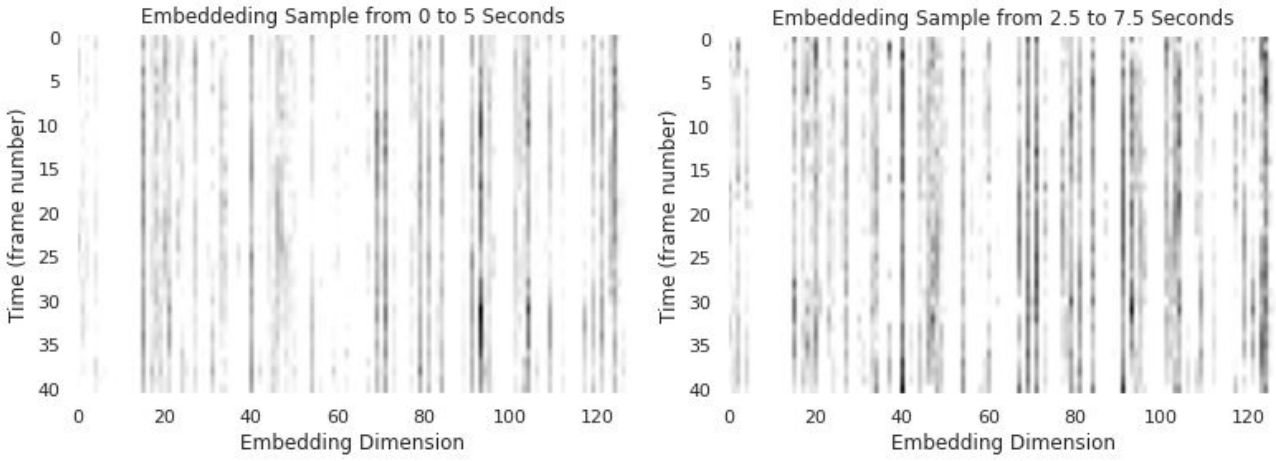
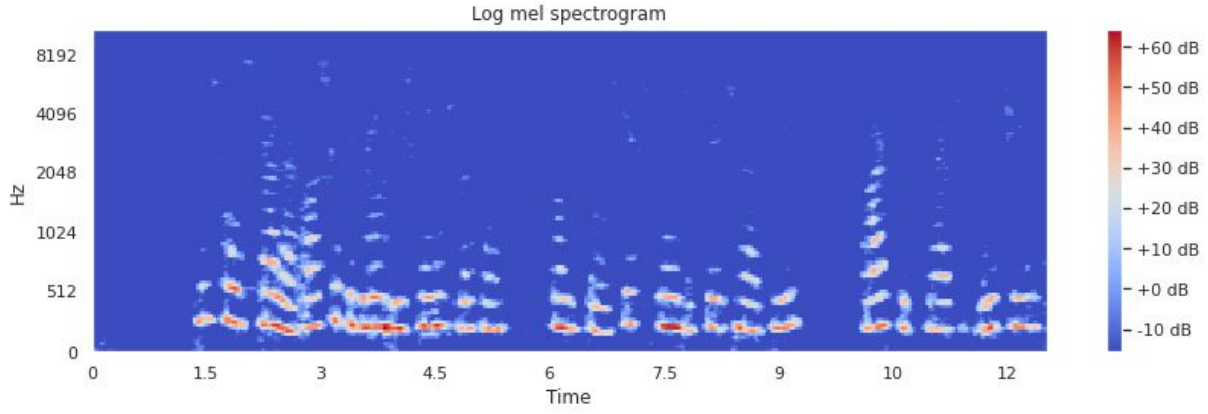


Figure 24 - Embedded representation of the same audio and duration but different offset.

Speed Variations

The last audio augmentation technique is speed variations. As the name implies, it consists in reproducing the audio at a faster or slower pace. This effect should still make the signal audible for a human, as with the noisy audio signals.

To keep dimensions fixed, I pad the audio with zeros when the audio is shortened. Figure 24 shows the effect of applying this technique a factor of 33% both for increasing and decreasing the speed on the feature extraction method log mel-spectrogram.



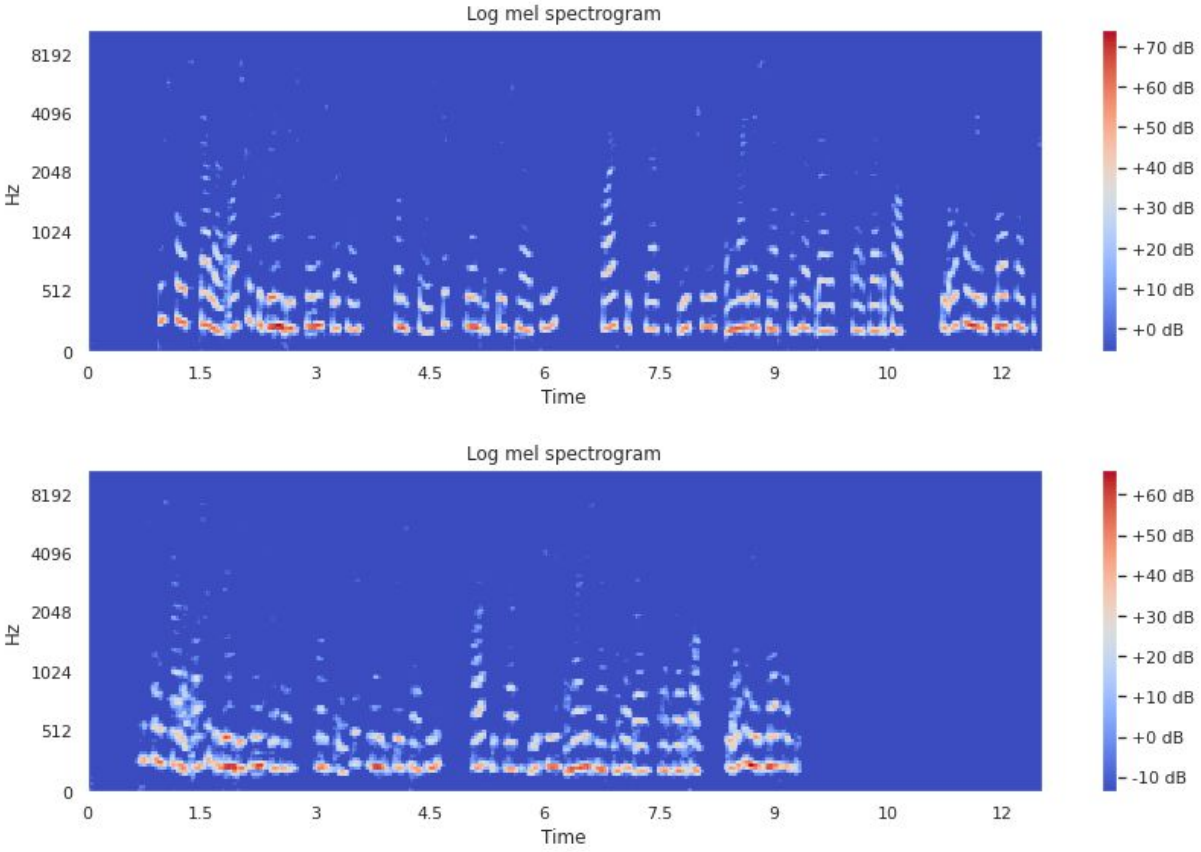


Figure 25 - Spectrograms from the same audio file with a speed rate of 66%, 100%, and 133% respectively.

5. Deep Learning Modeling and Results

In this section, I explain how I have modeled the dataset using different Deep Learning strategies, the various pre-processing techniques explained in the above section and the many labels explained in section 3. In all Deep Learning models implemented, I used consistent seeding by setting all possible seeds to obtain reproducible results.

First, I utilize simple Convolutional Neural Networks which are easier to implement and train, albeit worse handling unbalanced datasets. To alleviate this problem, I utilize more complex strategies, namely Siamese Neural Network models. I divide each subsection into a brief explanation of the design choices of each model and then into the different labels I wanted to model.

5.1. Convolutional Neural Networks

5.1.1. Design and Implementation

For the CNN model, I wanted to keep it simple. Its applications are two-fold. First, I use it to test whether the implementation of each part is properly coded before using it in Siamese models which are more difficult to debug. Second, I use it to assess how different design options might affect performance as this model is faster to train than Siamese models. The CNN models have an unavoidable problem, dealing with too many labels to classify that are hugely unbalanced.

Moving on to hyperparameters, I used random search manually to set them as there were many of them. The starting value of each hyperparameter was set from past experience in other works. Here is a list of them by subgroups:

Data Processing

These hyperparameters were mentioned in other sections. The data comes from the audio files and it is processed and used as already explained in five ways: raw samples, mfcc features, log mel-spectrograms, chromagrams, and VGGish embeddings. Mfcc and spectrograms yield the best results, although the latter was slower to train as it needed a bigger network. Raw samples were really slow and consumed a lot of resources, yet their results weren't bad. Finally, the chronogram and VGGish embeddings yield good results also but worse than the two better models. VGGish embeddings were the quickest model to converge as I was expected from an embedding.

On the other hand, I also used scaling to process the data. I implemented the most well-known methods, normalization, and standardization. It is worth noting that it did not have that much effect on the models as I use other means to normalize the data such as batch normalization and gradient clipping. I think it also has to do with the nature of audio features as they have a similar range, unlike some tabular datasets, that have many different ranges.

Data Splitting

For data splitting, I used the common train, validation, and test splits. I keep low ratios for the validation and test splits around 20% and 10% respectively. Splits are stratified. Another important fact for modeling native languages was how many languages to model. I play with this number by

slicing the languages by a minimum number of samples, being two the minimum samples (one for training an another for testing).

Neural Network Architecture

The neural network architecture depends on the preprocessing used before. For raw audio, I used a deep Conv1D architecture based on the one provided in this project **[17]**. It has four 1D-VGG16-like blocks with big kernel and max pooling window size in the first of them to try to lower the number of parameters as the value of the second dimension (the number of frames per audio) is really high.

As it is a big network I tried to constrain it to avoid overfitting using batch normalization, spatial dropout (around 10-25%). Finally, to reduce even more the parameters, I used just a 16-neuron fully connected layer as the last hidden layer before the softmax output layer.

For the rest of the input methods, I used rather small Conv2D architectures. The mfcc features and VGGish embeddings required just two VGG16-like blocks, chroma needed three and log-mel spectrogram four. I do not include batch normalization, although I used regular dropout. All of them were followed by a 64-neuron fully connected layer and a softmax output layer.

Neural Network Training

Finally, the training hyperparameters. The batch size was fixed to 128 for all models but the raw audio input which was set as explained in section 4 to 32. Epochs were set to 20 to slow models and 150 to fast models. Class weights were used when modeling unbalanced labels and are proportional to the number of samples of each class (i.e.: class 1 samples: 100, class 1 weight: 1; class 2 samples: 300, class 2, weight: 0.33).

The loss function used is categorical cross-entropy. The optimizer used is Adam and it was the main contributor to the performance of the model. For bigger and constrained models, I used lower learning rates (0.00001), which again were key to the success of the models. For smaller and unconstrained models, it was a little less relevant, the value was set to 0.001.

Neural Network Testing

The testing of the models is based on just a few metrics to avoid unnecessary complexity and embedding representations to better assess whether it generalizes well or not. The metrics are accuracy, weighted f1-score, and macro f1-score. These three provide different insights to assess the model. Then, the embedding is extracted from the last layer of the model and is reduced to three dimensions by using mainly PCA.

I used as the baseline model a naive model that classifies all samples to the dominant class. Finally, there isn't a cross-validation strategy as it would take to much time to compute all models, but I ensured that the process of splitting data is completely random each time. Bear in mind all training times are expressed in minutes.

5.1.2. Speaker's Sex Modeling Results

The first feature I wanted to model was the sex of the speaker, as I thought it would be easier, and indeed it was. This time, I only used two input modes (ways of preprocessing the audio inputs) to

model this feature as the results were near perfect. In the next features, I use the rest of the input modes since they are harder to model.

Looking at table 1, the VGGish embedding was the quickest and best way of fitting the data, as it was expected from the previous study in section 4 about this type of embedding. The model needed less than 5 epochs to converge. On the other hand, MFCC yields also great results, but it took more time to converge. In figure 26, the embedding space shows a clear separation between the two classes with just a couple of outliers. Overall, sex is a fairly easy feature to model using audio data.

Input mode	Parameters	Training time	Accuracy	F1-weighted	F1-macro
Baseline	-	-	52%	35%	34%
VGGish	409,058	0:10	99%	99%	99%
MFCC	401,586	1:15	97%	97%	97%

Table 1 - Sex CNN modeling results.

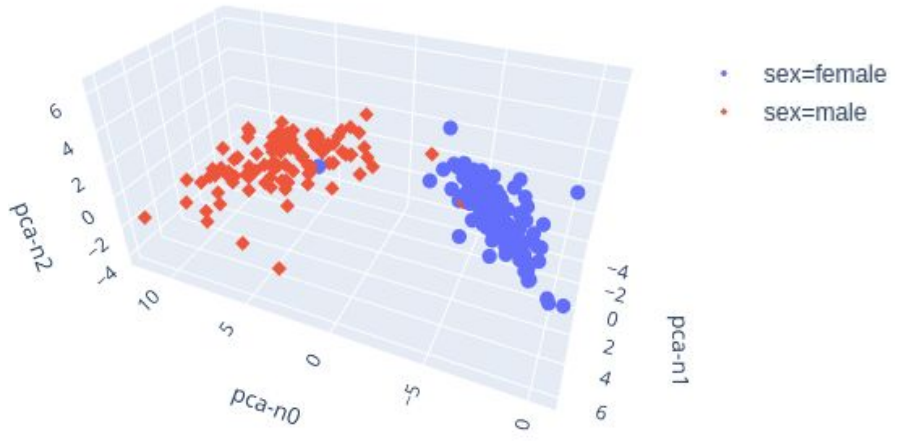


Figure 26 - Test set space of the best CNN sex model's last hidden layer using PCA.

5.1.3. Speaker's Age Modeling Results

The age of the speakers was harder to model than the previous feature. Even so, the results were acceptable for the classification of two age ranges that were balanced, those are from 0 to 28 years and from 28 to the maximum age. More than two numbers of ranges were tested and yielded better results than the baseline. However, the accuracy dropped sharply each time I increased the number of ranges.

Table 2 shows the results for the range described. Again as in the previous feature, VGGish embeddings were the best and quickest way to process the data. It is well separated from the rest of the group. Moving on, figure 27 sheds some light on how the model behaved. It forms a three-head cluster so to say. The range from 28 to the maximum has most of its data in one of the "heads", while the other range is more scattered. While the results of this feature are not bad, there is definitely room for improvement.

Input mode	Parameters	Training time	Accuracy	F1-weighted	F1-macro
Baseline	-	-	50%	33%	33%
Raw Audio	6,416,722	4:35	55%	55%	55%
MFCC	401,586	0:21	62%	60%	60%
Mel-Spec.	485,218	5:34	62%	62%	62%
Chroma	213,830	0:35	58%	56%	56%
VGGish	409,058	0:05	68%	67%	67%

Table 2 - Age CNN modeling results.

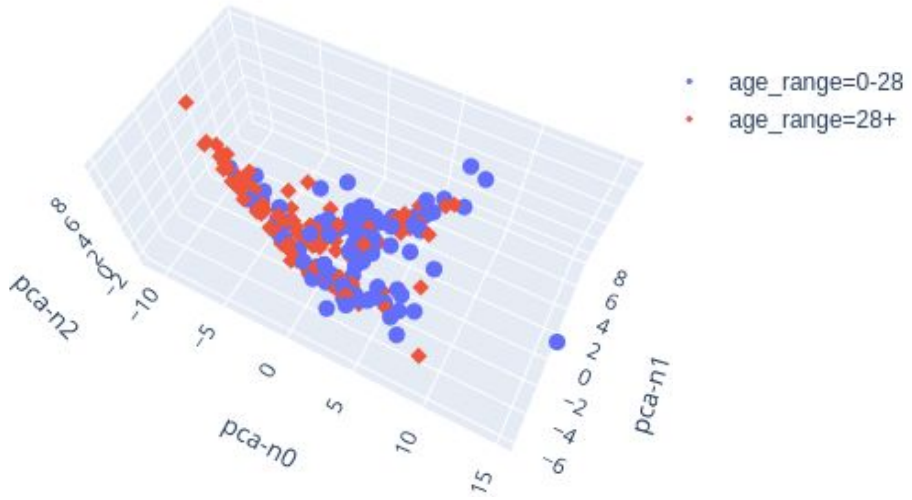


Figure 27 - Test set space of the best CNN age model's last hidden layer using PCA.

5.1.4. Speaker's English Accent Modeling Results

The modeling of 200 native languages featured in the dataset, with most of them having less than five audio samples using a simple CNN was bound to fail from the get-go. So to make this task a little easier to model, I set the minimum number of samples per language to 50. Even though it is still a fairly low number for a Deep Learning model, there are only six languages that meet this condition.

The table below shows the results of this model. It is worth noting that a model fitted to an unbalanced dataset tends to classify all samples as the dominant class. So, this time, it is really important to compare the results to the baseline. Bear in mind that, as explained before, I used class weights to try to balance the classes.

The raw audio model yields poor results, worse than the baseline method. The classic feature extraction techniques give the best results but all of them are slightly better than the baseline. Obviously, these better models generalize better than the baseline but are not able to surpass the accuracy of this one. It needs more data to enhance its performance. On the other hand, the embedding in the figure below shows a big cluster for the dominant class and smaller ones for the other classes but not clear and well-distinguished clusters.

Input mode	Parameters	Training time	Accuracy	F1-weighted	F1-macro
Baseline	-	-	57%	41%	12%
Raw Audio	6,416,722	13:25	33%	36%	19%
MFCC	401,586	0:24	55%	52%	28%
Mel-Spec.	485,218	8:01	57%	53%	28%
Chroma	213,830	1:48	53%	52%	29%
VGGish	409,058	1:32	40%	41%	19%

Table 3 - Speaker's native language CNN modeling results.

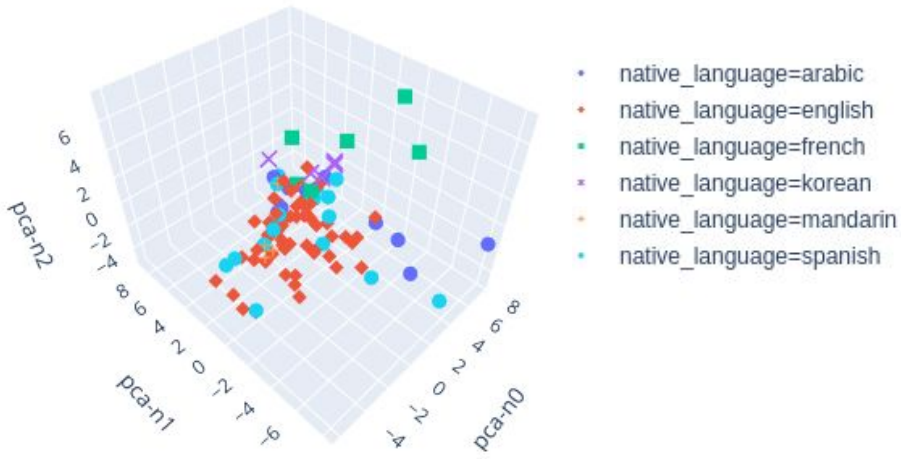


Figure 28 - Test set space of the best CNN native language model's last hidden layer using PCA.

Another approach easier to model is to use the native English Speaker feature explained in section 3. It is again an unbalanced classification task. Again, I balanced the classes using class weights. The table shows better improvements compared to the baseline model than the previous one. Here, it is clear that models using inputs such as mfcc features and log mel-spectrograms are the best models and generalize well, albeit not perfectly. Again, more data is needed to achieve greater results. The embedding shows a big cluster comprised of both classes, but they are well separated with some samples in the center that are inadequately placed.

Input mode	Parameters	Training time	Accuracy	F1-weighted	F1-macro
Baseline	-	-	73%	62%	42%
Raw Audio	6,416,722	13:36	68%	69%	61%
MFCC	401,586	0:56	82%	82%	79%
Mel-Spec.	485,218	5:40	83%	83%	79%
Chroma	213,830	0:58	72%	71%	63%
VGGish	409,058	0:46	77%	77%	70%

Table 4 - Native English Speakers CNN modeling results.

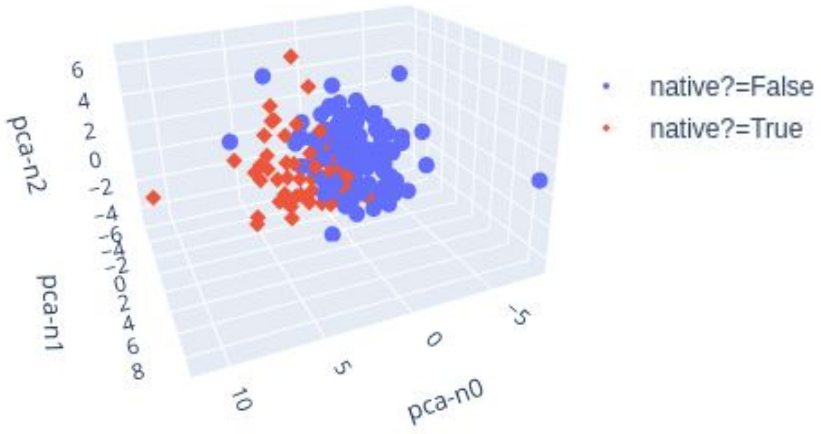


Figure 29 - Test set space of the best CNN native English speaker model's last hidden layer using PCA.

5.2. Siamese Neural Networks

5.2.1. Design and Implementation

Siamese Neural Networks are preferred over simple Convolutional Neural Networks for several reasons. First and foremost, SNNs deal considerably better with extremely unbalanced and small datasets. That is really a key point of the dataset I am dealing with. Another reason is that it also deals better with a high number of classes in classification tasks. Finally, in some cases, SNNs yield better embeddings, so as with CNNs, I explore embedding representations.

Several parts of the implementation are based on this article **[18]**, namely the distance used in the neural network, the training, and some parts of the testing. Other parts were used from the implementation of the SNN used in the Satellite Avoidance Challenge proposed in the subject PRDL and MLLB. Though, that implementation was more atypical and specific for that problem. This time, I wanted to utilize more standard methods. Here are the main decisions of its design:

Data Processing

As in the CNN models, I utilized the processing strategies from section 4. However, thanks to the insights gained with CNNs and some preliminary time tests with them, I decided to reduce the strategies to VGGish embeddings, mfcc features, and log mel-spectrograms as the raw audio was dead slow training time, chroma a slow feature extraction time, and both were the strategies with worst results.

On the other hand, I also used data augmentation described in section 4 for these models. Though the results did not improve that much, it was a non-factor. It might be because I don't do enough data augmentation due to time computing and extracting features. I utilized all kinds of augmentations at the same time for the common feature extraction techniques and just shifting for the VGGish embeddings. For the former ones, I used a noisy coefficient of 0.15, an offset of 20%, and a speed rate of 80%.

Data Splitting

The splitting and cross-validation decisions took for the CNN models are the same here. Aside from that, SNN has other decisions to take on this matter. Normally in other works such as **[17]**,

[18], the test is comprised of just unseen categories. While this is a great way to split the data, it is also a bit short-sighted and not fit for all datasets. In the cited works, they have at least a medium-sized and balanced dataset.

On the other hand, this dataset does not have many samples and they are really unbalanced, so skewed categories might become a problem. Thus, I used another strategy to compare with the aforementioned. In lieu of unseen categories, I used all categories but each of them has at least one unseen sample for the test set. This strategy yields better results but usually, it was when using fewer categories.

Neural Network Architecture

The neural network architecture used is the same as in CNNs. I wanted to test them in those models to later attach them to the Siamese models. However, I needed to add batch normalization to all models along with gradient clipping and low learning rate in the optimizer to make the network learn.

There was a lot of tuning to make it work as the network parameters (biases and weights) did not update correctly with unconstrained networks. For the distance function, I wanted to keep it simple, so I used just the absolute difference between the pairs.

Neural Network Training

The training is performed by batch instead of by epoch. This is done because I generate batches during training time by sampling at random a set of pairs equal to the batch size, always keeping 50% of matches and 50% of mismatches to have balanced classes.

As for the hyperparameters, I used again Adam as optimizer with learning rates around 0.00001 and clipnorm equal to 1, binary cross-entropy as loss function, a batch size equal to 24, and around 6000 iterations (i.e. processed batches) of training time, although it varies for each model.

Neural Network Testing

The testing is completely different compared to CNNs, besides the use of embedding spaces. First, I used as baseline model a random model, which fits great with the following testing strategy. This time, I used a standard method for siamese networks, n-way one-shot learning (note that some works label this method as n-way k-shot learning **[18],** and others as k-way n-shot learning **[17]**, I stick it to the former).

So first during training time, I compute every 100 iterations the 5-way one-shot learning accuracy of a batch of 48 tested samples and support sets (i.e. a 5-way support set is a set of five different samples in which there is only one sample that matches to a sample that is being tested).

Once the training is done, I compute and plot the accuracy of n-way one-shot learning being n a set from 2 to 10. To make the process more reliable, I get the accuracy of 48 batches 15 different times and compute the mean of all of them for each n-way. Bear in mind all training times are expressed in minutes.

5.2.2. Speaker's Sex Modeling Results

Again, as in the CNN models, the VGGish is the best model. Even though the modeling is almost perfect in both VGGish and Log Mel-spectrograms, the former is much quicker. Table 5 might not show this, as the training time there is for 6000 iterations. The real-time that took the VGGish model get to a near-perfect was 32 seconds and 300 iterations, really impressive.

The embedding of the best model in figure 31 has the variance less concentrated than CNN one. While the CNN model had almost its variance in the first PCA component, the siamese model has the information in the second component for the same accuracy and outliers.

On the other hand, the mfcc model drops its performance quicker than the other when in the CNN model was much closer to the VGGish model, this might indicate that siamese performance across larger support set drops considerably fast if the performance of smaller set is not incredibly high. Another important take is that siamese models are much slower compared to CNN models.

Input mode	Parameters	Training time	2-way	5-way	10-way
Baseline	-	-	50.00%	20.00%	10.00%
VGGish	409,432	07:21	98.75%	99.02%	95.00%
MFCC	401,905	05:48	90.55%	79.72%	72.08%
Mel-Spec.	486,689	10:59	96.66%	93.61%	89.16%

Table 5 - Speaker's sex SNN modeling results.

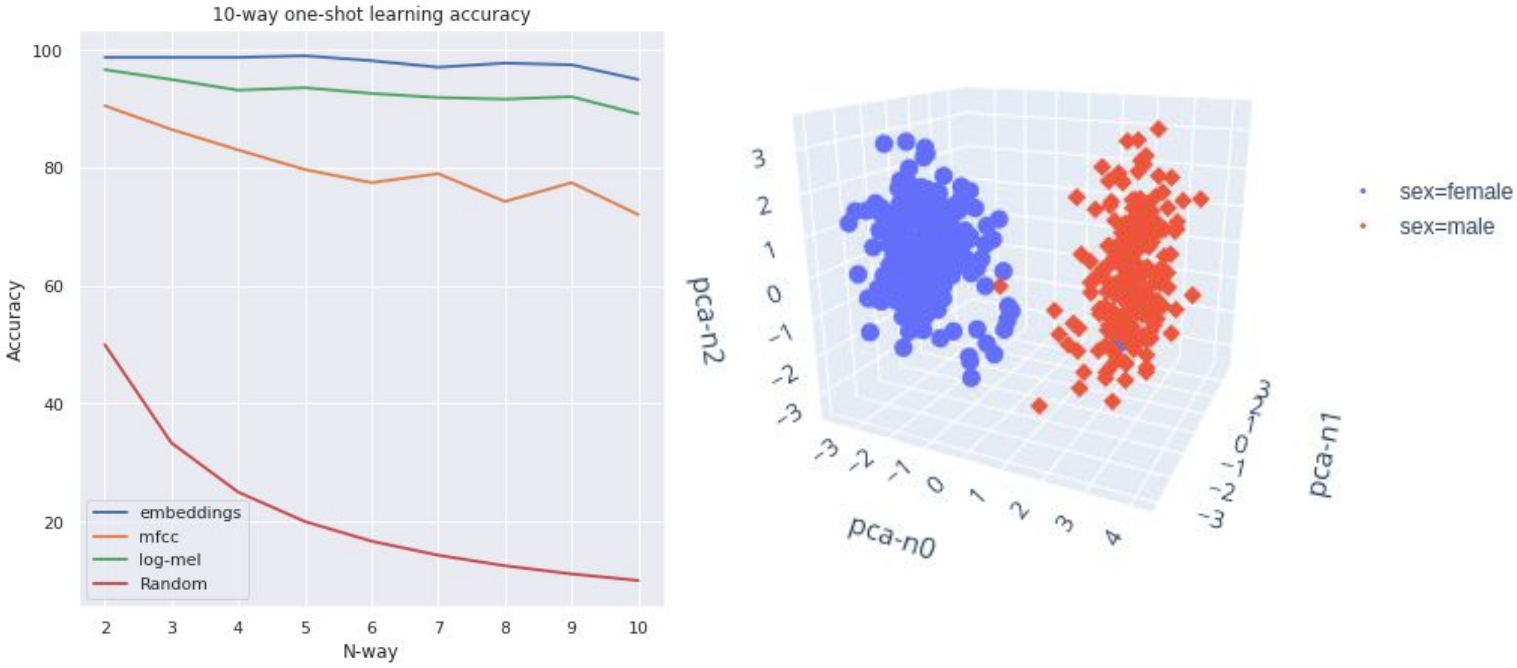


Figure 30&31 - 10-way one-shot learning accuracy of Speaker's sex SNN (30) and test set embedding space of the best SNN sex speaker model's last hidden layer using PCA (31).

5.2.3. Speaker's Age Modeling Results

The speaker's age siamese model turned out a bit disappointing. Bear in mind that I tried to classify more realistic age ranges this time. I defined three classes (young, adults, and seniors). These ranges correspond to 0 to 30 years old, 30 to 60 years old, and 60 to the maximum age in the data. It is worth noting that these classes are unbalanced.

The results were consistent but just slightly better than the random baseline model. Those results were for the mfcc coefficients and log mel-spectrograms. For the VGGish embeddings, the result was much worse. It simply behaved like a random model, so that particular model was not useful at all.

Input mode	Parameters	Training time	2-way	5-way	10-way
Baseline	-	-	50.00%	20.00%	10.00%
VGGish	409,432	02:58	50.55%	18.05%	10.00%
MFCC	401,905	08:44	56.80%	26.38%	13.88%
Mel-Spec.	486,689	28:31	56.50%	24.28%	12.66%

Table 6 - Speaker's age SNN modeling results.

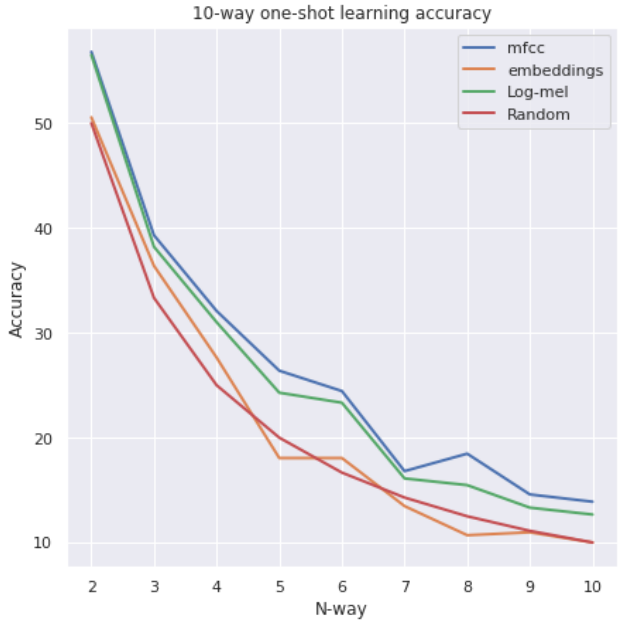


Figure 32 - 10-way one-shot learning accuracy of Speaker's age SNN.

5.2.4. Speaker's English Accent Modeling Results

In the modeling of the speaker's native language, I decided to take an alternative route. After trying several ways of splitting the data as explained in subsection 5.2.1 in an attempt to model as many native languages as possible, I found that the results were underwhelming. Not even once they were better than the baseline model. I need something different.

So instead, I set the input mode to just MFCC as it is quick to train, and the number of classes to the minimum, which is two. In that way, I get positive results, not great, but the model learned

something. After that, I increase the number of languages to see where is the maximum before hitting the baseline model. It was six languages as in the CNN model which the best model yielded the same accuracy as the baseline model. So definitely, no matter which tool is used to model this dataset, it needs much more data than it has for modeling the native language of the speaker.

Input Mode	Nº Languages	Training time	2-way	5-way	10-way
-	Baseline	-	50.00%	20.00%	10.00%
MFCC	Two	02:55	60.27%	31.66%	18.61%
MFCC	Three	02:57	57.08%	24.44%	13.88%
MFCC	Four	02:58	55.55%	24.86%	10.97%
MFCC	Five	05:05	52.63%	21.25%	11.80%
MFCC	Six	03:00	52.22%	18.47%	10.69%

Table 7 - Speaker's native language SNN modeling results.

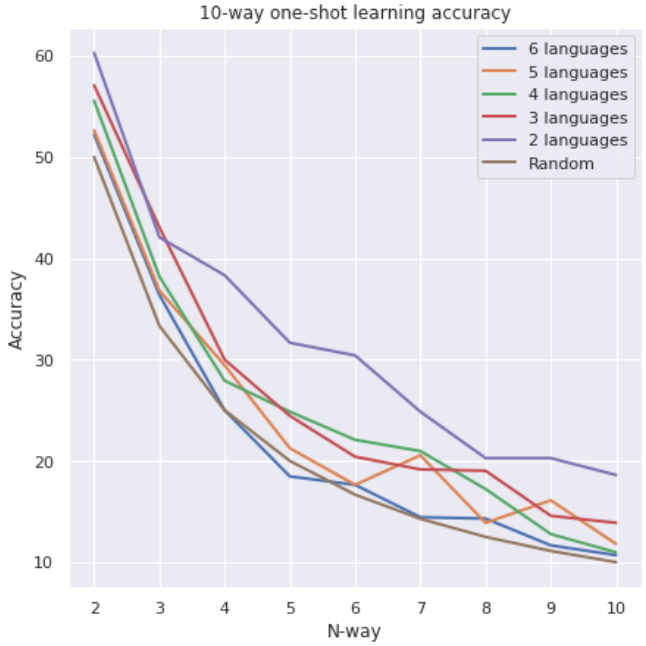


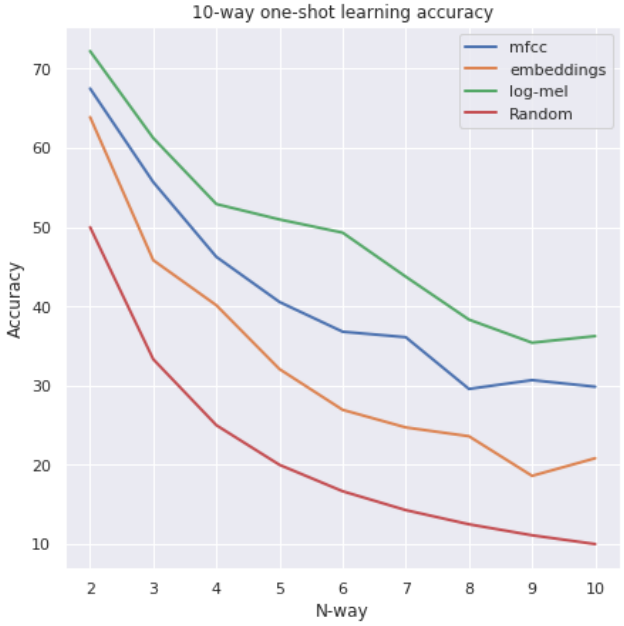
Figure 33 - 10-way one-shot learning accuracy of Speaker's native language SNN.

Finally, I modeled the Native English speaker feature as it proved successful in the CNN model. Here, the siamese model did learn significantly better than the baseline model, around +20% in both larger and smaller support sets for the best model. However, this model is lagging behind sex speaker's models and the gap is larger than their respective CNN models.

This might not be bad, as Siamese models deal better with unbalanced datasets. So although the performance is worse, the models are certainly more robust and might be better suited for a real deployment. On the other hand, the embedding uses a tSNE instead of a PCA. This is because the PCA representation did not show in a meaningful way the data as all variance belonged to the first component. In the image, it is clear that both classes share the same cluster and are distinguished inside it, albeit with multiple outliers.

Input mode	Parameters	Training time	2-way	5-way	10-way
Baseline	-	-	50.00%	20.00%	10.00%
VGGish	409,432	02:52	63.88%	32.08%	20.83%
MFCC	401,905	08:37	67.50%	40.55%	29..86%
Mel-Spec.	486,689	47:44	72.22%	50.97%	36.25%

Table 8 - Native English SNN modeling results.



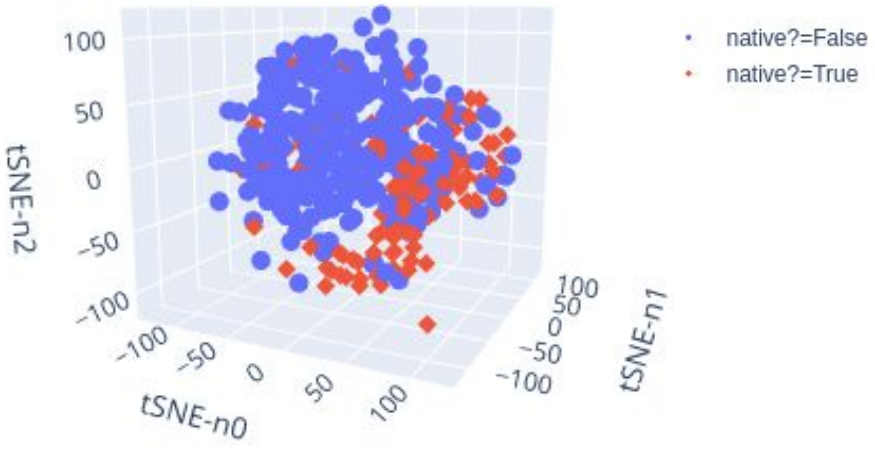


Figure 34&35 - 10-way one-shot learning accuracy of Native English speakers SNN (34) and test set embedding space of the best SNN Native English speakers model's last hidden layer using tSNE (35).

6. Conclusions

The project sought to study how speakers from all around the world can be classified based on features such as sex, age, and most importantly English accent using audio data. The main tools were the speech accent archive dataset and Deep Learning models, namely regular Convolutional Neural Networks, and their Siamese Neural Network variant. The most important results can be summarized in:

1. The **dataset**, while interesting and easy to handle, was too small and extremely unbalanced in some features for obtaining impressive results on all of them. In any case, it was great working with these data and learning more about audio from it.
2. As for features, the **sex of the speaker** was by far the easiest to model. In particular, when the VGGish embedding was applied. On the other hand, the rest of the features were far from easy.
3. The **speaker's age** models obtained some positive results and were notably better than baseline models, but the results were far from perfect and they were achieved using just two classes that were chosen because they were balanced rather than because they were useful from real modeling standpoint.
4. The **English accent** feature was hardly modeled, it yielded slightly better than the baseline models when using fewer categories than the total (6 from 200, those with at least 50 audio samples), so some hope is there for better results as it is not a random model, but it seems impossible without more data samples and balanced classes.
5. On the other hand, there were positive results modeling whether the speaker was a **native English speaker** or not, both in regular CNN and SNN models. Thus, more evidence that the accent can be modeled. Again, I would like to emphasize that without more data the results cannot improve greatly.
6. Concerning the **models**, regular CNNs yielded better results. However, the way regular CNNs handle unbalanced classes (with non-equal class weights during training) makes me suspect that the results will not scale well. On the other hand, siamese models look more robust in the way they handle extremely unbalanced data and the drop in performance does not seem so big. The real drawback of the siamese models was to find architecture that was well suited to this data.
7. The **feature extraction techniques** gave different results. Vggish embeddings were amazing for sex modeling and also yielded the best results for age modeling. The mfcc and log mel-spectrogram were overall great and the best for English accent modeling. Both Vggish embeddings and mfcc were the quickest models. Raw audio and chroma yielded good results, just slightly worse than the other, however, they were really slow.
8. The **data augmentation methods** used did not have much success. They yielded similar results as they less processed counterparts, and the time computing them was extremely high due to Google Colab's speed with I/O operations. Due to this last reason, I could not incorporate more data as I wanted to, which would surely have helped to obtain better results.

Aside from this, the implementation of the project with Keras was quite easy and straightforward thanks to past experience with these tools. However, finding the right hyperparameters training and evaluating for all models and writing this memory was much more time consuming than I thought it

would be. There many other things that I would have loved to try if I had more time. For instance, there are advanced works on siamese models that use triplet and other contrastive losses **[19]**, I could have just copied one of the implementations, but I wanted first to learn about them first. For data augmentation, there are new methods such as Google's SpecAugment **[20]** which looked really promising. As for GANs, OpenAI unveiled last week Jukebox **[21]** which looks really awesome. Finally, I did play around a little bit with this x-vectors implementation in Pytorch **[22]**, but I scrapped the idea of including it because it would have taken too much time that, unfortunately, I do not have right now.

7. Bibliography

[1] Institute of Acoustics. c1974-2019. Silbury Court, 406 Silbury Boulevard, Milton Keynes, MK9 2AF, UK: Institute of Acoustics; **[accessed 2020 Feb 10]**. https://www.ioa.org.uk/

[2] What is Acoustics? c2015-2019. N243 ESC Brigham Young University Provo, UT 84602: Brigham Young University; **[accessed 2020 Feb 10]**.
https://acoustics.byu.edu/content/what-acoustics

[3] Acoustics. c2009-2019. Wikipedia; **[accessed 2020 Feb 10]**.
https://en.wikipedia.org/wiki/Acoustics

[4] Chris Donahue, Julian McAuley, Miller Puckette, "Adversarial Audio Synthesis," ICLR, 2019 **[accessed 2020 Mar 21]**.

[5] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, Adam Roberts, "GANSynth: Adversarial Neural Audio Synthesis," ICLR, 2019 **[accessed 2020 Mar 21]**.

[6] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, Adam Roberts, "DDSP: Differentiable Digital Signal Processing," ICLR, 2020 **[accessed 2020 Mar 21]**.

[7] Mirco Ravanelli, Yoshua Bengio, "Speaker Recognition from raw waveform with SincNet," SLT, 2018 **[accessed 2020 Mar 25]**.

[8] Alex Nguyen, "20 Best Speech Recognition Datasets for Machine Learning," lionbridge.ai, 2019 **[accessed 2020 Apr 09].**
https://lionbridge.ai/datasets/best-speech-recognition-datasets-for-machine-learning/

[9] Steven H. Weinberger, "Speech Accent Archive," George Mason University, 2013 **[accessed 2020 Apr 09].** https://www.kaggle.com/rtatman/speech-accent-archive

[10] Sergio Pérez, "Modeling Obstructive Sleep Apnea using Machine Learning," Github, 2019 **[accessed 2020 Apr 09].**
https://github.com/spmorillo/OSA-ML-Modeling#osa-ml-modeling

[11] Sergio Pérez, "Satellite Collision Avoidance Challenge," Github, 2019 **[accessed 2020 Apr 09].** https://github.com/spmorillo/Satellite-Collision-Avoidance-Challenge

[12] Aurélien Géron, *"Hands-On Machine Learning with Scikit-Learn & Tensorflow,"* O'Reilly, 2017 **[accessed 2020 Apr 09].**

[13] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, Marvin Ritter, "AudioSet: An ontology and human-labeled dataset for audio events," ICASSP, 2017 **[accessed 2020 Apr 09].**
https://github.com/tensorflow/models/tree/master/research/audioset/vggish

[14] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. G-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. ICASSP*, 2014, pp. 4080-4084 **[accessed 2020 May 01]**.

[15] David Snyder, Daniel Garcia-Romero, Daniel Povey and Sanjeev Khudanpur, “Deep Neural Network Embeddings for Text-Independent Speaker Verification,” INTERSPEECH 2017 **[accessed 2020 May 01]**.

[16] Yexin Yang, Shuai Wang, Man Sun, Yanmin Qian, Kai Yu, “Generative Adversarial Networks based X-vector Augmentation for Robust Probabilistic Linear Discriminant Analysis in Speaker Verification,” ISCSLP, 2018. **[accessed 2020 May 01]**.

[17] Oscar Knagg, “Building a Speaker Identification System from Scratch with Deep Learning,” Medium, 2018. **[accessed 2020 May 02]**
https://medium.com/analytics-vidhya/building-a-speaker-identification-system-from-scratch-with-deep-learning-f4c4aa558a56

[18] Harshall Lamba, “One-Shot Learning with Siamese Networks using Keras,” Medium, 2019.
https://towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras-17f34e75bb3d **[accessed 2020 May 02]**

[19] Eric Craeymeersch, “One Shot learning, Siamese networks and Triplet Loss with Keras,” Medium, 2019. **[accessed 2020 May 06]**
https://medium.com/@crimy/one-shot-learning-siamese-networks-and-triplet-loss-with-keras-2885ed022352

[20] Eric Craeymeersch, “State of the Art Audio Data Augmentation with Google Brain’s SpecAugment and Pytorch,” Towards Data Science, 2019. **[accessed 2020 May 06]**
https://towardsdatascience.com/state-of-the-art-audio-data-augmentation-with-google-brains-specaugment-and-pytorch-d3d1a3ce291e

[21] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, Ilya Sutskever, “Jukebox: A Generative Model for Music,” OpenAI (early-stage), 2020. **[accessed 2020 May 06]**
https://towardsdatascience.com/state-of-the-art-audio-data-augmentation-with-google-brains-specaugment-and-pytorch-d3d1a3ce291e

[22] Dannynis, “Pytorch implementation of X-vectors embedding,” GitHub, 2019. **[accessed 2020 May 06]**
https://towardsdatascience.com/state-of-the-art-audio-data-augmentation-with-google-brains-specaugment-and-pytorch-d3d1a3ce291e