University of Arkansas, Fayetteville

ScholarWorks@UARK

Computer Science and Computer Engineering Undergraduate Honors Theses

Computer Science and Computer Engineering

5-2020

Speech Processing in Computer Vision Applications

Nicholas Waterworth
University of Arkansas, Fayetteville

Follow this and additional works at: https://scholarworks.uark.edu/csceuht

 Part of the Artificial Intelligence and Robotics Commons, Graphics and Human Computer Interfaces Commons, Service Learning Commons, and the Software Engineering Commons

Citation

Waterworth, N. (2020). Speech Processing in Computer Vision Applications. *Computer Science and Computer Engineering Undergraduate Honors Theses* Retrieved from https://scholarworks.uark.edu/csceuht/84

This Thesis is brought to you for free and open access by the Computer Science and Computer Engineering at ScholarWorks@UARK. It has been accepted for inclusion in Computer Science and Computer Engineering Undergraduate Honors Theses by an authorized administrator of ScholarWorks@UARK. For more information, please contact scholar@uark.edu, uarepos@uark.edu.

Speech Processing in Computer Vision Applications

An Undergraduate Honors College Thesis
in the
Department of Computer Science
College of Engineering
University of Arkansas
Fayetteville, AR

by

Nicholas J Waterworth
njwaterw@uark.edu

April, 2020
University of Arkansas

This thesis is approved.

Thesis Advisor:

_____

Khoa Luu, Ph.D

Thesis Committee:

_____

John Gauch, Ph.D

_____

Qinghua Li, Ph.D

Abstract

Deep learning has been recently proven to be a viable asset in determining features in the field of Speech Analysis. Deep learning methods like Convolutional Neural Networks facilitate the expansion of specific feature information in waveforms, allowing networks to create more feature dense representations of data. Our work attempts to address the problem of re-creating a face given a speaker's voice and speaker identification using deep learning methods.

In this work, we first review the fundamental background in speech processing and its related applications. Then we introduce novel deep learning-based methods to speech feature analysis. Finally, we will present our deep learning approaches to speaker identification and speech to face synthesis. The presented method can convert a speaker audio sample to an image of their predicted face. This framework is composed of several chained together networks, each with an essential step in the conversion process. These include Audio embedding, encoding, and face generation networks, respectively.

Our experiments show that certain features can map to the face and that with a speaker's voice, DNNs can create their face and that a GUI could be used in conjunction to display a speaker recognition network's data.
Keywords: *Speaker Recognition, Speech to Face, MFCC, Convolution Neural Networks*

Contents

1 Introduction

Audio data is dense and contains untapped information that is useful for analytic. Speech processing is a well researched and implemented topic, which has a significant foothold in most modern technologies. Many consumer-based products in the last two years have achieved just the start of contemporary speech processing like Home assistants, voice to text, and auto-captions generators. However, other types of speech processing, like audio User-Authentication and Speaker Diarization, are just beginning to reach a point of accuracy to become Consumer-friendly and practically accurate. Speaker Diarization is the segmentation and clustering of speaker samples within a single audio recording. When paired with image processing, these techniques become more suitable for consumer use. Thereby Speech processing still has more room to grow and more facets to research.

1.1 Speech Applications

In the research field of Speech Analysis, there are currently three unique problems. These are Speaker Recognition, Speech Recognition and Speaker Synthesis, respectively. Although each problem has unique solutions, they all use similar methods to interpret speaker signals and extract feature-related information.

Speaker Recognition is a class comparison problem to be able to identify a speaker from the characteristics in a given spoken phrase. The work in Speaker Recognition revolves around creating unique representations of speech samples and training networks to compare with extracted features. Within Speaker recognition, there are speaker identification and speaker verification. Verification is taking a spoken phrase and comparing it to whom they claim to be. Identification is taking in an unknown speaker and identifying who they are in a data set.

Speech Recognition handles listening to a given speaker and interpreting what words are being said and turning them into their textual representation. These methods are already integrated heavily in modern life with technology like home assistants and Speech to Text. Voice profiling is to infer physiological attributes based on speech audio signal alone. This field of research has much literature behind it, each displaying possible links with lots of complexity. Aspects like gender, race, age, and facial features all have effects on the human voice. Although not all face-related factors affect the voice, thereby making it a challenge to recreate a speaker's faces. Humans also can mentally match and recall voices to faces given pictures of that person's face. Many academic papers survey these

features as associating faces to their voice counterpart.

Speaker Synthesis attempts to address the problem of recreating a specific speaker's voice expressions and character, given features highlighting their vocal characteristics. One limiting factor in this field is the length and coverage of an audio clip. Ideally, a clip should cover a sentence to get the maximum amount of inflections to derive a person's face.

1.2 Fourier Transform

In the natural world, sound is the change of air pressure caused by vibrations around us which we then receive and perceive. These changes in pressure are what get picked up by microphones and are the changes we monitor when we talk about frequency. The Fourier transform is a linear transformation that decomposes an input signal into the constituent frequencies that make up the original signal. It also provides the magnitude of each frequency within the signal. By using this transform, one can decompose any signal into a sum of sinusoidal waves.

$$F\{g(t)\} = G(f) = \int_{-\infty}^{\infty} g(t)e^{-2\pi ift}dt \tag{1.1}$$

In the Equation (1.1), f is the frequency and t is time past. In the field of Signal analysis, analysts pass signal data through Fourier Transforms to compress data, isolate noise identify frequency patterns. Fourier transforms exploit that, simple waveforms like sine and cosine waves, compose every complex waveform. From the Fourier transform, we can also get the inverse Fourier transform, which is a re-creation of the original signal only from the Fourier transform graph. The inverse is especially useful since it creates signals that have been modified on a frequency basis back to In the next section, we will cover MFCC's where the Fourier transform plays a vital part in decomposing human speech signals. Figure 1.1 is an example of post Fourier transformed audio spectrogram visualized.

1.3 Spectrograms

A Spectrogram is a visual representation of the distribution of magnitude into frequencies, as found in a power spectrum, over time. With this method, patterns that are represented only with numbers create a human/image processing-friendly visual. Figure 1.1 is an example of a spectrogram visualized.

A possible negative of relying on spectrograms is that recreating the original signal cannot be done with spectrogram data alone. When using spectrograms, information

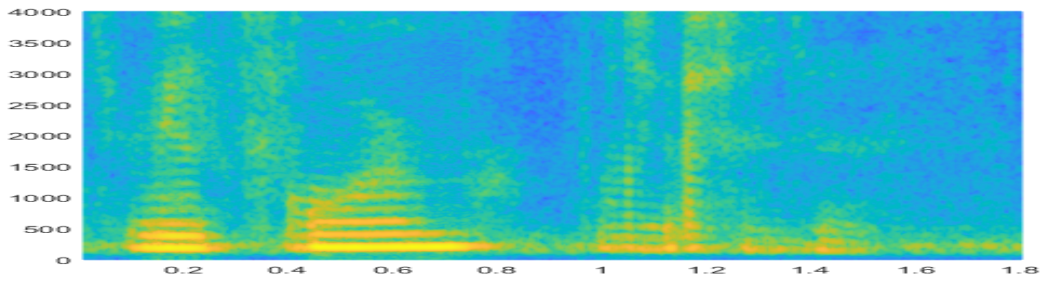

Figure 1.1: Spectrogram of a Fourier Transform [6]

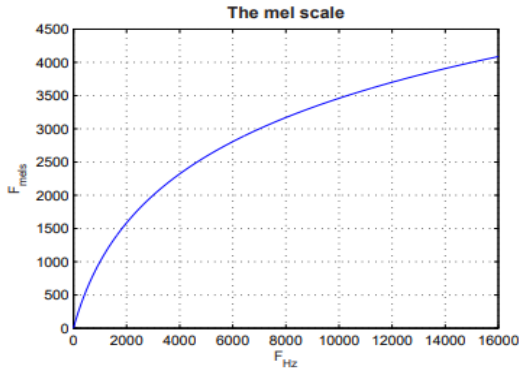

Figure 1.2: The Mel Scale [17]

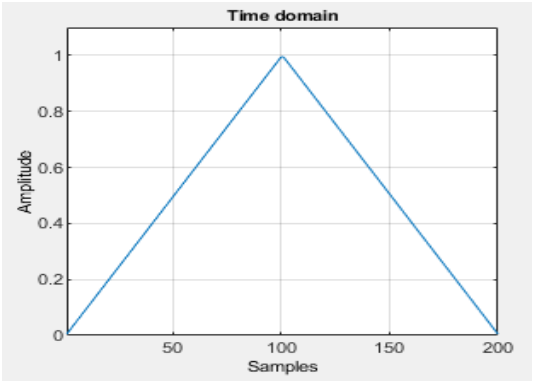

Figure 1.3: A visualization of a triangular window function[7]

about the phase of the signal is lost. Using spectrograms allows for the use of methods like Convolutional neural networks and other image-based learning methods since the problem becomes image-classification.

1.4 MFCC

A Mel-frequency cepstrum is a representation of the short term power spectrum of a sound. Mel-Frequency cepstral coefficients (MFCCs) are coefficients that makeup Mel-frequency cepstral coefficients. To derive an MFCC, we first start by taking the Fourier transform of our input signal. After this step, frequencies are simplified and separated for later mapping and processing. From here, map the powers of the spectrum unto the Mel Scale to get the Mel-frequencies. The Mel scale is a function of the difference between human perceived frequency difference and the actual measured difference of a frequency. Figure 1.2 depicts the Mel-Scale visually.

Initially, researchers calculated this scale using experiments with human listeners. However, today many versions of the Mel Scaling function exist, the most popular and widely used is O'Shaughnessy's method as in Equation (1.2).

$$m = 2595 log_{10}(1 + \frac{f}{700}) \tag{1.2}$$

After calculating the signal Mel-frequencies, the algorithm applies a triangular windowing function. Window functions are filtering functions that amplitude filters a specific sample window in the signal input, acting primarily as filters for spectrum tapering the values provided to the frequency range defined in the function. A triangular window is explicitly a windowing function that forms the shape of a triangle over N frequency samples like below. Triangular window functions usually follow the form of Equation (1.3), which is of the visual form of Figure 1.3.

$$w[n] = 1 - |\frac{n - \frac{n}{2}}{\frac{L}{2}}|, \quad 0 \leq n \leq N \tag{1.3}$$

In Equation (1.3), L is usually equal to $N + 1$ such that the filter will visually take on the shape of an equilateral triangle.

With the triangular window function filtered Mel-frequencies, we apply the logarithmic function to the powers of the Mel-frequencies. Then using the log powers calculated before, we calculate the discrete cosine transform of the Mel-log powers. Finally, the MFCCs are the amplitudes of the resulting from the spectrum calculated. Generally, MFCCs perform poorly in high noise environments, so using normalization and other de-noising techniques enables the lowering of interference. Some have even tried creating their networks to handle and remove specific noise from a given signal. Due to the human perceptual engineered nature of the MFCC conversion graph, there is no guarantee that the representations created are optimal for all speech-related tasks.

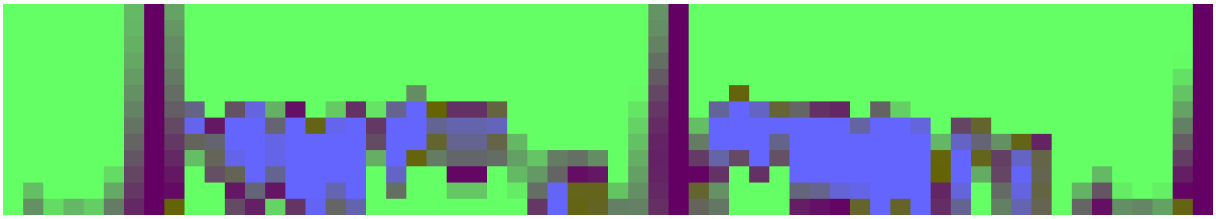

Figure 1.4: An MFCC representation of the phrase "Hello" spoken twice [5]

In Figure 1.4, 13 real-time software extracts MFCC features for each sample across the time domain. Each sample forms one column i.e., along the horizontal axis, while each row represents a unique MFCC feature. Audio input is taken from the Audio Context using navigator.mediaDevices.getUserMedia(...) function, which passes on the Stream object, which creates an audio source from where Meyda.createMeydaAnalyzer(...) is used to extract audio MFCC features. If the RMS value of the received audio signal is more significant than a threshold, MFCC features are depicted in the diagram using Live-Audio-MFCC [5].

1.5 Neural Networks

A neural network is the conglomerate of nodes that form a network of connections to one another. A Neural Network tries to emulate the brain structure of the natural world with how animals have neurons and connections that let them make decisions. Each node has adjustable weights that change over time based on its training, with each node having activation functions. An activation function is usually a differentiable function that defines the output of each respective node for a given input by a gradient descent algorithm. Usually, these come in the form of Sigmoid, Softmax, TanH, as depicted in Figure 1.5.

The Sigmoid activation function can produce a value between 0 and 1; it is excellent at predicting probability but suffers problems such as vanishing gradient, non-zero centered output, slow convergence. TanH is a value between -1 and 1; it usually is used over Sigmoid because it only suffers from vanishing gradient. ReLU has a six times improvement rate of convergence from the TanH function since it is a simple calculation, and it does not suffer from any problems that TanH does. ReLU's most significant issue is that during training, that node weight can change to a point where it will never be able to activate again. Leaky ReLU solves this problem by creating a small slope such that a node will never die after there is a change in its weight value. A perceptron is a single node including its connections, activation function and output as depicted in Figure 1.6.

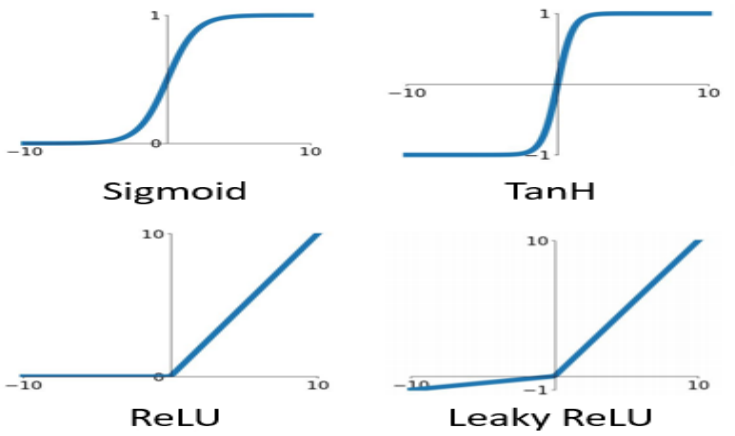

Figure 1.5: Graphic Representations of Common Activation Functions [13]

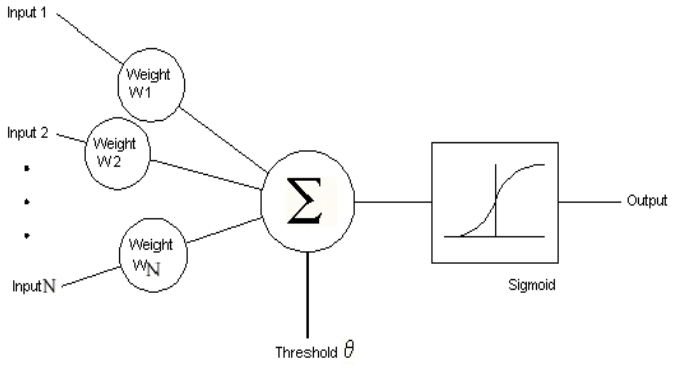

Figure 1.6: Basic Node Structure diagram [9]

Usually, Neural networks either fit into the camps of supervised or unsupervised learning. Supervised learning deals with having both training and testing data that are labeled with correct outputs, so the model learns from the training data set what is right. Unsupervised learning is where the training algorithm gives the neural network only the input data and not the correct output. For Speaker analysis research, all of this work proposed is Supervised Learning. Figure 1.7 depicts what a simple fully connected network could look like.

Dropout is a function that applies to a layer during training, given a random probability p is whether or not a node is kept in the layer or removed. This technique reduces over-fitting during the training of a neural network and avoids co-adapting with input data. Dropout also creates more robust features within the network by facilitating the generalization of the data.

A fully connected layer is simply a layer where each node has a connection to all

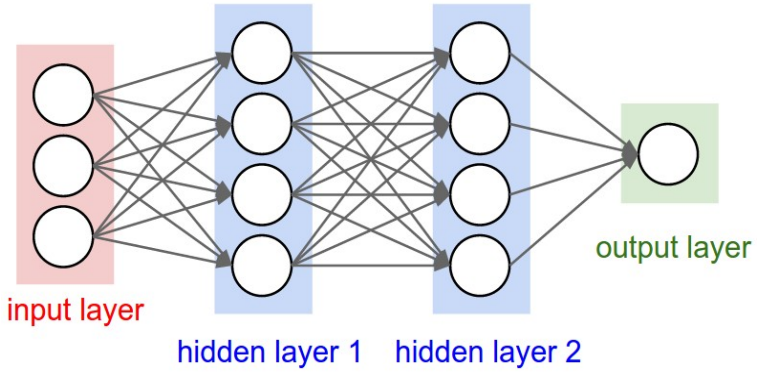

Figure 1.7: Fully Connected Network Structure [14]

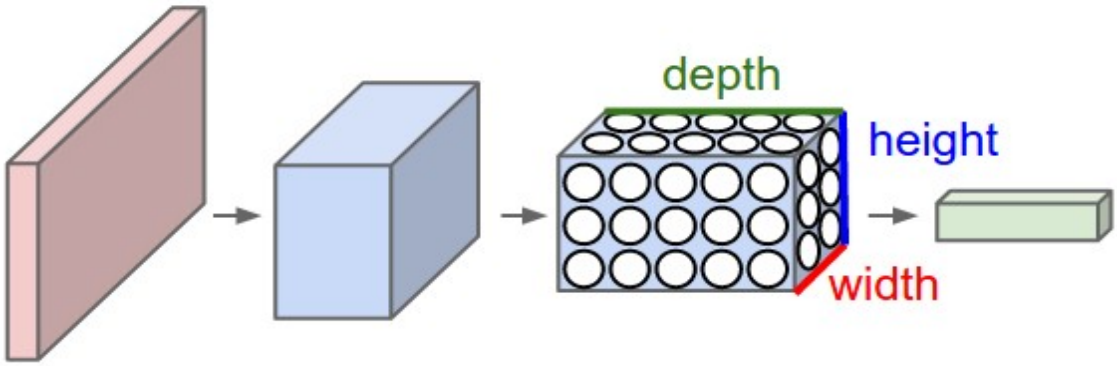

Figure 1.8: Convolutional Network Structure [14]

nodes of the previous layer. A fully connected layer allows for all features to incorporate with one another to each node, which increases performance, but it is computationally expensive on a large scale.

Loss functions are functions that are used to asses the performance of the output of a supervised network to the real answer and then calculate a score. Beyond a straight loss function, there is also back-propagation where the algorithm computes a gradient for the loss function to update the weights and minimize loss. From this gradient, the neural network adapts it is weights through its optimization function according to how it performed to try and hit a better optimum in the model space. Usually, software trains a model in batches for a specific number of epochs of iteration. Each iteration tests the outcome of the model, on different data than what the network uses in training, then changing weights and iterating again. These are all features that make up a basic neural network, only using these methods results in a Feed-Forward Network.

1.6 Convolutional Neural Networks

Deep learning is the broad name for a group of Neural networks that use a high density of fully connected layers to refine raw input data into high-level feature data that perform operations such as classification on output features. A convolutional neural network is a form of deep learning, which specializes in handling feature extraction and processing by refining data through many layers of convolution and pooling. A visual representation of a CNN can be viewed in Figure 1.8 Regular neural networks may work well on problems like basic classification and linear regression, and it does not scale well with complex data. For example, a classic neural network usually handles a one-dimensional vector of data, and an RGB image data is composed of 3 matrices of different colors, which would cause exponentially more weights for a single node. Thereby, traditional neural networks are ineffective in problems with complex multi-dimensional data like an image. What makes a convolutional neural network different than the traditional network is the series of extraction convolution and pooling layers that occur before the data passes into a standard fully connected network. A Convolutional Neural net can capture spatial and temporal data by maintaining the structure of an image instead of flattening it into a one-dimensional vector, like what is in a Feed-Forward network. In order for a Neural network to be a convolutional network, one layer or more needs to have a convolutional layer that performs the convolution operation.

The Convolutional layer performs a sliding dot product or multiplication operation between two matrices, a convolution operator, where one matrix is the set of learn-able parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is a smaller (x and y plane) and deeper (z plane) matrix, which acts as a filter for each sub-section of the input matrix and returns a smaller matrix of convolved features that represent the region it encoded. A kernel will parse an input image in its forward pass phase by moving the defined length of its stride to calculate the respective convolved values. The convolution operation is depicted in Figure 1.9.

Within a Convolutional neural network, there are pooling layers that are responsible for reducing the spacial size of network output. Pooling facilitates the removal of variance between images by phasing out small details of each layer and removing noise. Pooling layers can use Max or Average pooling. Max pooling performs better than average pooling because Max pooling reduces noise problems within the input images. Examples of max and mean pooling can be viewed in Figure 1.10. After several operations of convoluting and pooling, the resulting matrix flattens in where it transforms into a one-dimensional vector that passes through a fully connected network that follows the functionality of a

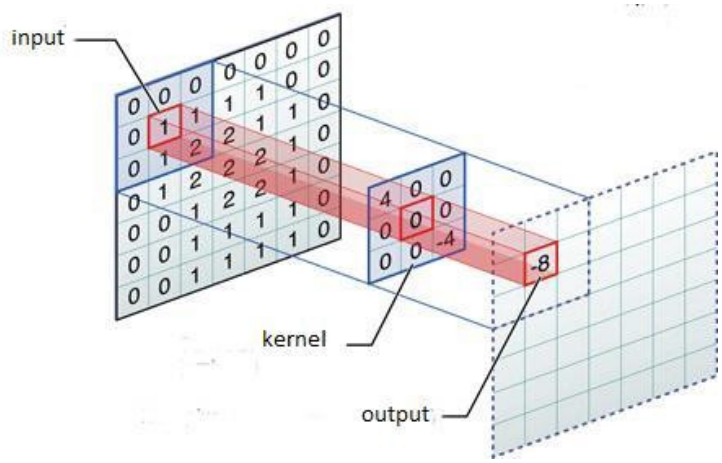

Figure 1.9: Convolution Operation Example [14]

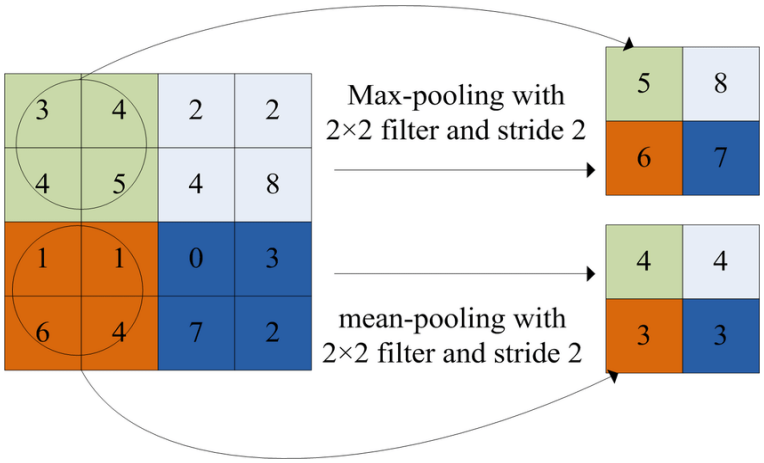

Figure 1.10: Pooling Operation Example [11]

traditional feed-forward neural network and performs classification on the given input. Fully connected layers of nodes usually follow convolution layers to perform classification or feature output operations.

2 Speech Processing and Applications

Speech processing is the parsing of Audio data for useful information related to the speaker or content of what is said. Speech processing has many parts and pieces that hold it together, like signal conversions and the physiological research behind speech profiling. The requirement of empirical connection found in voice profiling is essential to all applications, like speaker identification and generation, to work.

2.1 Voice Profiling

Voice profiling is the field of inferring physiological information about a person only from their voice and vocal characteristics. There exists material that confirms the connection of many characteristics to what is present in the voice. Characteristic features like gender, nationality, age, and various facial features have all been explored for correlation. Some of these more complicated characteristics are inferred by the human ear but never empirically proven. The conglomeration of these features provides connections between the features of the face and features of the voice. The basis of voice profiling facilitates the research of speaker driven areas since all speakers are assumed to have unique voices, that are dependent on their physical being. From Speaker identification to generation, the concept of voice profiling is essential for the understanding of further work.

2.2 Speaker Identification

Most commercially available speaker Identification frameworks use i-vectors to generate speaker embeddings. Some of these frameworks rely on extraction methods like MFCCs or Frequency Banks, which, due to their limited adaptability, could hinder results. Instead of using traditional MFCCs or i-vectors, SincNet utilizes custom made filter layers that utilize Fourier transforms and combine images of each of the signal frequencies to perform a neural network band-pass filtering. The motivation behind it was features like MFCC smooth the speech spectrum, which could cut out narrow information-dense characteristics like pitch and formats. By taking in raw spectrograms or single audio signals, the model can employ more features that hand-crafted methods would filter out. This project also uses Deep learning methods in the form of a CNN into a DNN to better extract and emphasize speaker features.

The i-vector representation is the most commercially available and successful implementation of speaker embedding vectors. The i-vector system is based on Gaussian Mix-

ture Model-Universal Background Models (GMM-UBM) and works as a dimensionality reduction of the GMM super-vectors. Mixture models are a probabilistic way to represent specific vectors belonging to specific sub-population within a higher population. Here the population is composed of multiple speakers while each speaker population is the identities that make up a speaker. Visually it can be viewed as the probability of the feature vectors belonging to one speaker or characteristic of a population. One example of this would be mapping gender or age given a feature vector through a GMM model to get an i-vector.

With the advent of DNNs, many more advances came in the methods of extracting features from audio and performing identification. X-vectors, are one of the many implementations of DNN generated embedding vectors that take in filter-bank features or MFCCs and creates a one-hot encoding output of all the possible classifications. Unlike i-vectors which rely on GMM's, X-vectors use dynamic grouping using multiple layers and convolution layers. Instead of using hand-crafted representations of the audio signals like MFCCs and FBank, SincNet takes in the direct waveform and lets the deep neural network filter the pure audio signal. This research achieved a dynamic network that could handle changing frequency ranges and the ability to extract unique features older methods will miss.

2.3 Speaker-Face Verification

Speaker face verification is the field of research where people do cross-modal matching on face and voice samples. With the onset of CNNs, this field and others were able to make significant advancements in creating solutions for high dimension problems. Matching face to voice data is a relatively new and growing research area, with the use of methods like cross-modal networks. Cross-modal networks take in inputs of different domains, like audio and images, and then have individual layers for each to parse them independently. Once parsed, the data from both domains are concatenated and propagated to the output layer for the combined prediction. Convolution is essential here to reduce the dimensions of the data and make it all compatible with the output layer to work. Matching between voice and face is an N-way classification problem, that generally works as a selection problem. One such paper used an audio sample to decide which face the voice belonged to between a set of N faces with success [18], which could serve in the future as an authentication method.

2.4 Human Face Synthesis

Given an audio sample from a specific speaker, one aims to create their face by using speaker embeddings and mapping them to facial features. Assumptions and connections proved that given a speaker's voice, a model could estimate several features of a face. However, some uncorrelated speaker features like hair and background can not be consistent with output since there is no correlation with voice. Usually, through cross-modal models or feature encoders, faces are generated from voices. From there, A Generative Adversarial Network is generally used to recreate the face from a feature vector. Generative Adversarial networks (GANs) work on a high level by having two models play a game with one another. One is a generator and tries to create the most realistic face as possible. While the other is a classifier and tries to weed out the fake faces from the real ones within the output of the Generator model. Mathematically this game that they are playing represents min-max; the models compete for several iterations until they reach optimal results. The resulting output is a network that can generate close to real outputs and a network that can differentiate fakes from real images well. With recent work, GAN's are proving to be a valuable asset in recreating faces from feature vectors. [19] shows that using GANs in composite with CNNs allows for generations of close faces. Their architecture uses a voice embedding network to create an embedding that is then passed through a Generative model to be then discriminated against as a face or not and classified as a specific person in the GAN's data set.

3 Deep Learning in Speech Processing

Below, the section describes the two project design proposed designs. The first being the embedding network for face generation, followed by the speaker recognition GUI design.

3.1 Embedding Network

When handling speaker-related audio, noise pollution is inevitable. Features present in speaker signals are muddled and hidden by extraneous noises. When listening with the human ear, noise is naturally filtered out since the brain focuses on the specific speaker they are hearing. Within neural networks, the use of convolutional layers emulates filtering and emphasis of features. For the feature extractor to optimally perform, our implementation uses five convolution layers to expand out the data embedded within the MFCC feature vector. Each convolutional layer also has average pooling layers to ensure features are not too extreme and spread noise evenly among the feature vector. Between each convolution, the network uses Batch Normalization functions to increase the learning rate and improve the gradient flow. Each layer of nodes also uses ReLU activation for the properties of quick convergence. By utilizing these methods, the model produces fine-tuned feature embeddings of speaker data.

3.2 Embedding Design Flow

The data flow of the proposed model begins with a given waveform file. First, the waveform is pre-processed to remove extraneous data. To smooth out peaks in the amplitude of the audio data, the model uses normalization while amplitude thresholds splice silence out of the signal. In order to feed audio into a neural network, pre-processing modifies the audio into a consistent format. With the variability of time in audio clips, audio signals need to be partitioned into equal chunks of time then formatted to information not dependent on time. Parsing mass amounts of audio data can also be a slow process if the audio is not formatted. Most modern audio files have extreme sample rates of up 44100 samples per second. However, with a decreased rate of 16,000 samples, essential information can be lost due to the removal of vital sample points from the higher dimension signal. The audio signal can be compressed into a format like spectral features or MFCCs to preserve vital information while reducing processing time. MFCCs are useful for highlighting features in the human frequency range of a signal, by using its custom mapping function. Using MFCCs most closely emulates the human ear, making it ideal

for trying to connect physical features to vocal characteristics. MFCCs thereby are a viable choice in human-related feature extraction of audio-based features. One drawback of this choice is due to an MFCCs static nature; they do not handle outliers not well fit with the Mel scale.

From a given MFCC embedding vector, the embedding network can fine-tune and emphasize specific features within an MFCC based on training success to create the optimal feature vector for mapping human voices. The embedding network has many layers, as defined above, where it convolves the embedding into longer vectors to draw out more information from the original MFCCs maximum values. From there, the last output layer reduces back down to the original input feature size and is the input of the voice embedding to the face embedding network.

3.3 Feature Mapping Network

Face feature vectors can be created from Audio feature vectors using an Encoding network. A fully connected feed-forward network is used as the encoder since fully connect layers allow for all features to affect the output of each node. Using fully connected layers enables features of the network to be more discriminant within the nodes. The input layer of the network uses the size of the audio features, while the output layer uses the size of the face feature vector. The structure of the hidden layers follows the node count of [128, 256, 512]. The network implements a growing cone structure to increase the complexity of each feature's interaction for better voice/face feature connections. This network utilizes Leaky ReLU for fast convergence and to prevent a node from dying during activation in the network during training. Once encoded, the network passes the embedding to the Vec2Face framework for image generation. For training, the network uses Adam optimization since it handles large amounts of features well, is computationally efficient and converges quickly. For our loss function we used the combination of angular loss and soft-max in Equation (3.1).

$$angular_loss * 10 + soft - max_loss * 5 \tag{3.1}$$

Equation (3.1) makes features more discriminative after each iteration. The modification of the nodes weights using this method also improves features representation.

3.4 Vec2Face Framework

Using the Voice to Face network outlined earlier, The feature vector of voice data is passed through the linear layers of the network and expanded into a face feature vector based on

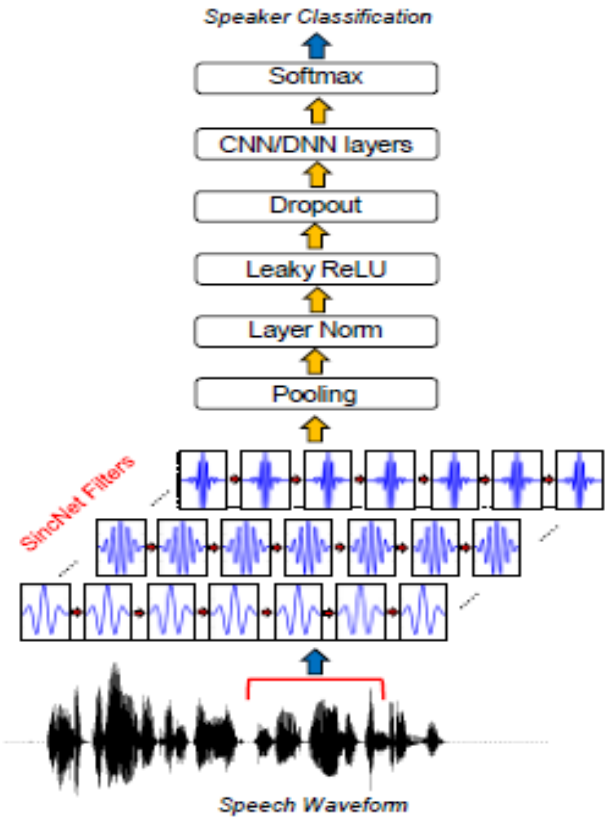

Figure 3.1: SincNet Structure Diagram from [10]

the trained weights on what to emphasize in the voice feature network.

3.5 SincNet Framework

The SincNet framework, as defined in Figure 3.1, performs identification by using a DNN architecture on a pure audio waveform. By removing hand-crafted audio features, the model was able to handle outlier cases and out-perform current methods. The speech signal is windowed to handle the dynamic nature of time and then fed into the first convolution layer. This convolution layer works differently than traditional convolutional layers by using a predefined convolution function to act as a rectangular band-pass filter for the input audio signal. Here the network learns weights for each of these filtering layers to find an optimal filter for the range of the human voice. SincNet also utilized the Mel Scale to jump-start the learn-able filter to an already performing level.

SincNet utilizes Hamming windows to further high-frequency selectivity and avoid harsh changes in the filter function gradient. After the first filtering convolution layer in

SincNet, the standard CNN layer operations like pooling, normalization, and others can be applied again. Any order of layers can be used after this output to use a soft-max classifier then to decide the identity for a speaker. Our model stuck with the example of a traditional CNN followed by a fully connected network and then a soft-max classification layer. The advantages of the fully connected nodes allowed for the features to be discriminant among one other and have an effect on the output of each node.

3.6 Speaker Recognition Application

A simple live speaker identifier can be created by listening to live audio by utilizing the SincNet framework defined above in Figure 3.1. The software can tell when a user speaks using silence detection. Using this, we can record the spoken words to a waveform to use with SincNet. SincNet handles raw audio data, which is beneficial on the implementation side since the engineer does not have to deal with extracting embeddings and can feed in a processed audio waveform straight into the network. From this, the network returns a class value to identify which speaker a clip will belong to in the speech. Using this, we can display the appropriate face to the end-user, displaying the image of the speaker the network predicts. One the User removes the window, the software can wait for live audio once again and repeat the process. The purpose of the software is to help in the visualization of mapping audio data to the facial counterpart and display one of the possible applications of speaker identification.

4 Unveil the Face Behind a Speech via Gromov - Wasserstein Loss

4.1 Introduction

Audio and visual signals are the most common modalities used by humans to identify other humans and sense their emotional states. Features extracted from these two signals are often highly correlated, allowing us to imagine the visual appearance of a person just by listening to their voice, or build some expectations about the tone or pitch of their voice just by looking at a picture of the speaker. When it comes to image generation, however, this multi-modal correlation is still under-explored.

Previous works' limitations. In order to synthesize general-looking faces of an individual given his/her speeche signals, prior methods usually adopt an encoder-decoder framework that takes the audio as an input and outputs synthesized faces presented for that subject. In order to guarantee the biometrics reservation property, these methods can be divided into two groups, i.e. early and late ID constrained approaches. In the former approach, the ID reservation constraint is employed at the early stage of the synthesis process and put more effort to make the audio embedding as similar as possible as the face embedding obtained from the ground-truth faces. Then, a pretrained face generator is adopted to synthesized the desired faces. Meanwhile, in the latter approach, the constraint is emphasized on the ID similarity between synthesized and ground truth faces.

In either cases, as speech (audio) and face (visual) signals come from different domains, their feature spaces usually embed different information and variations according to audio and visual nature, respectively. Moreover, these spaces are incomparable in nature. Therefore, learning a direct mapping to align audio feature representation to the absolute location of facial features with high accuracy is challenging. Moreover, these designs only transfer the absolute location of facial feature to audio embedding space and neglect the topology, i.e. relationship between one sample with other samples within a class and other classes. This prevents these two distributions to be effectively synchronized and reduce the robustness of the reconstruction process.

Therefore, in this chapter, we focus on cross-modal visual generation, i.e. the generation of facial images given only speech signals. Two recent approaches have recently popularized this research venue. Jamaludin et al. [4] presents a method for generating a

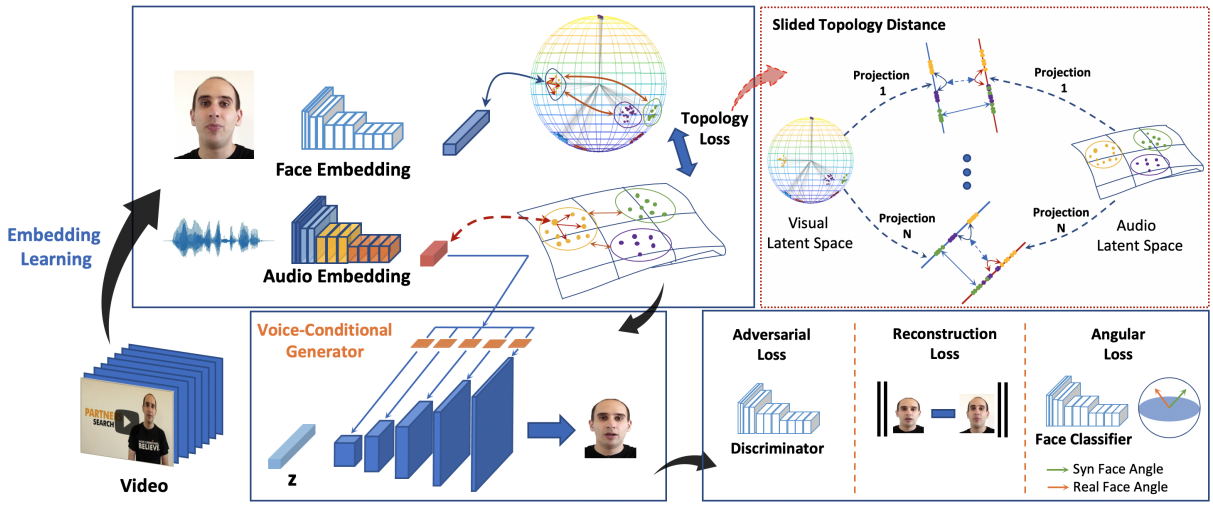

Figure 4.1: Overall network structure

video of a talking face starting from audio features and an image of him/her (identity). Suwajanakorn et al. [15] focus on animating a point-based lip model to later synthesize high quality videos of President Barack Obama. Unlike the aforementioned works, however, we aim to generate the whole face image at pixel level, conditioning only on the raw speech signal, i.e. without the use of any hand-crafted features, and without requiring any previous knowledge, e.g speaker image or face model. For learning such a model, high quality, aligned samples are required. This makes the most commonly used datasets such as Lip Reading in the wild [3], or VoxCeleb [8] unsuitable for our approach, as the position of the speaker, the background, and the quality of the videos and the acoustic signal can vary significantly across different samples.

4.2 Voice to Face Synthesis

In this work, firstly, we present a new *Gromov-Wasserstein loss function* incorporated into the deep convolutional neural networks to effectively model the relationship across incomparable spaces, i.e. the distributions of speech versus facial features (as shown in Fig. 4.1). Here, the relational information between samples, i.e., the topology of the reference data manifold, is preserved. Therefore, the proposed system is not only effectively map the speech features to the correct IDs but also embed other face variations. Secondly, a voice-conditional GAN-based framework is presented to learn a more robust generator to synthesize realistic faces with ID preservation. Thirdly, the experimental results are evaluated with different face attributes, i.e. aging, gender, ethnicity to show the robustness of the proposed framework.

Let $\mathcal{D} = \{\mathbf{a}_i, \mathbf{v}_i, y_i\}_{i=1}^{N}$ be the audio-visual dataset including N triplets of an audio recording $\mathbf{a}_i \in \mathcal{A}$ and face $\mathbf{v}_i \in \mathcal{I}$ of the subject y_i. A function $F^a : \mathcal{A} \mapsto \mathcal{F}^a$ represents

a mapping function from the audio domain $\mathcal{A}$ to its embedding spaces $\mathcal{F}^v$ and $\mathcal{F}^a$. In addition, let $G : \mathcal{F}^a \mapsto \mathcal{I}$

Let $F : \mathcal{I} \mapsto \mathcal{F}$ be a function that maps an input image I from image domain $\mathcal{I}$ to its high-level image embedding feature $F(I)$ in latent domain $\mathcal{F}$. Similarly, a function $V : \mathcal{S} \mapsto \mathcal{V}$ denotes a mapping function from a voice/ audio clip $\mathbf{s}$ to its audio embedding feature $V(\mathbf{s})$.Let G be a function (generator) that reconstructs facial images of an individual given his/her embedding. An voice-to-face problem can be formulated as follows,

$$\bar{I}^* = \arg\min_{G,V} \mathcal{L}\left(ID^v(\mathbf{v}), ID^f([G \circ V](\mathbf{s}))\right) \tag{4.1}$$

where ID^v and ID^f denote the functions mapping a voice recording $\mathbf{s}$ and an image I to their identity, respectively.

Generally, a direct approach is to learn a voice embedding function $V' : \mathcal{S} \mapsto \mathcal{V}'$ such that $\mathcal{V}'$ approximates $\mathcal{F}$, i.e. minimizing the distance between voice and face embeddings $||F(I) - V'(\mathbf{s})||_2^2$. Then a pretrained face generator G can be straightforward adopted to reconstructed the desired faces. However, face and voice signals in intuition come from two different domains with distinct properties that result in two different distributions of their feature embeddings. Moreover, This is particularly the case for features learned with deep learning as they can usually be arbitrarily rotated or permuted. In this context, the W distance with the naive cost $c(x, y) = ||x - y||$ fails at capturing the similarity between the distributions. As a result, not robust according to large variations in both speech and face signals and limit in capturing the topology structures of face embedding space. Particularly, the topology structure includes intra- and inter-class relationship presented in the distributions.

5 Experiment Setup

This section describes the implementations of the proposed methods and the aspects of our experimental setup.

5.1 Speech Database Collections

The data sets used for the experiments were Vox-Celeb1[8] and Vox-Celeb2[2]. Vox-Celeb1 is composed of labeled audio clips of 1,251 unique celebrities. Vox-Celeb2 contains 150,480 labeled audio and video clips for 6,112 celebrities from YouTube videos. VGG-Face2[1] contains face data for each of the speakers mentioned above of Vox-Celeb2. The Vox-Celeb2 data set was used in conjunction with VGG-Face2 to map audio samples to specific face images of a given celebrity. These data sets were collected because of their free academic use of their data and mapping between celebrity voice samples and face samples. It is not the largest data set for speech and faces available, but it contains the required connections needed for the model to work. The one glaring issue with Vox-Celeb is the nationality distribution of speakers where a majority of speakers are American, English, or Germanic/Indian. The gender distribution also leans towards males with 61 percent of utterances, while the female only speakers account for 39 percent. Most utterances in the data set are between 4 and 10 seconds, with there being a maximum of 20-second utterance lengths. Another possible issue is the Aesthetic nature of celebrities, being that they do not represent the rest of society's appearances.

The Timit data is used to set to test the original baseline set by SincNet. Timit specializes in speech recognition tasks, by having speakers say similar sentences. This data set includes exclusively American speakers of different dialects. However, instead of being random segments of speech, each speaker reads phonetically rich sentences to help with features in voice profiling. In this data set, there are a cumulative 630 speakers, each reading ten sentences. The Corpus design was a joint effort among the Massachusetts Institute of Technology (MIT), SRI International (SRI), and Texas Instruments, Inc. (TI).

For the custom lab database, there are five 20 second long recordings for six lab members. This data set was created for interactive testing to see how different models handled data outside of their trained sets. All lab members are male of varying nationality and a close age range. Speakers were given various media to read to try and cover several phrases and achieve an optimal range of features to extract.

5.2 Environment Set-Up

For testing the created frameworks, two machines were used to train and run the models. Lambda was used for training and testing on Vox-Celeb and only used for training for speaker recognition while the Optiplex machine was used exclusively for running and visualizing the speaker recognition tasks.

Lambda is a local campus Server with an Intel(R) Xeon(R) W-2195 CPU @ 2.30GHz CPU, four NVIDIA Quadro RTX 8000 GPUs, and 251 gigabytes of RAM. The Optiplex is an Ubuntu 64 bit computer with a Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz as its CPU, and Radeon HD 8670 / R7 250/350 for its GPU with 24 gigabytes of RAM.

When picking what language to use for the software, Python is the best choice for many reasons. Python has many Open-source modules that can be used for functionality such that there is no repeated work using the Pip package installer. Python is also a language that is presented as a data-science language and has many neural network libraries. For our Neural network creation, the open-source Python Module PyTorch was used. The NN module in PyTorch allows for easily defined neural networks that are easy to implement and interpret. PyTorch has predefined convolution layers and operations like pooling, dropout, and a full selection of the activation functions listed above. Even if a layer type needed is not defined, PyTorch has systems to use that lets one create custom layer types. For this project, the network needed no custom layer implementations since all applied layers were already defined. The created software also utilizes Python modules that handle Images, Wave files, and Audio operations when needed.

5.3 Voice to Face Synthesis

The embedding framework worked by given a directory of audio samples in the Wave format, loading in a single sample. The Wave python library is used to first load the Wave file in, which needed to be on a mono channel, with a sample width of 2 and a rate of 16000 Hz. First, the software removes silent clips from the sample object with a specific noise threshold. Then, the Wave object is sent to a custom MFCC library that converts the sample into several windows of MFCC representations. The window length for each sample is set to 400 frames. A 1024 point fast Fourier transform is performed to get the entire window. From there, a Basic Mel filter matrix is created with defined maximum values and minimum values. Then the frequencies are filtered through a triangle window 64 times, and respective inverse matrices are calculated. Next, the log spectrum is calculated each frame in the signal and is returned as an array of MFCC values. Then each Mel-frequency is normalized using mean and variance calculations. For consistency

of performance, if an audio sample is shorter than 10 seconds, it is repeated until a length of 10 seconds is achieved. From here, the MFCC matrix is turned into a PyTorch friendly Numpy array of MFCC values feed into the embedding network. The structure of the convolutional embedding net takes in an input vector of length 64 and returns an output float32 embedding of length 64. The Convolution layers of the embedding network follow the format [256, 384, 576, 864]. Here we are using convolutions to enlarge patterns in the data and increase the significance of any change in the MFCC embedding. After each convolution, the network applies average pooling to maintain noise in the network. Once done, the program saves the embedding in a specific location sent in by the user. The embedding network is run preemptively using a bash script. On the completion of each iteration, the script creates a mirror directory of the input directory of wave files to save each embedding. Then for each speaker folder, the wave files are fed through the pre-trained model used in [19]. Once the script creates all embeddings, we have a mirror directory composed of encodings instead of waveforms ready to be parsed by the encoding network.

The encoding network is created using the Tensorflow library to integrate better with Vec2Face, which had already been defined using Tensorflow. This network was applied using fully connected Tensorflow dense layers with leaky ReLU activation on each node. Voice embeddings and their output face embeddings were used of VoxCeleb2 and their respective VGG-Face2 counterparts to train the network. This network was trained using 200 speakers embeddings and face images for 1,000,000 iterations. The loss function used was a combination of angular loss and soft-max loss. For our optimizer, the network uses Tensorflow's implementation of the Adam optimization algorithm.

5.4 Speaker Recognition

5.4.1 Data Set

The Vox-Celeb1 data set was used together with our Vox-Lab data set for training and testing on Speaker recognition. SincNet was initially trained on the Timit data set to ensure that their baselines were performing the same as ours. From there, a custom Vox-Celeb set up was used, and the framework was trained on speakers from Vox-Celeb1.

5.4.2 Pre-processing

The Vox-Celeb model of SincNet was trained on 200 speakers from Vox-Celeb over 1500 epochs with a batch size of 256 with 800 batches per epoch. Each specific sample loads in with a window length of 200 frames along with a shift of 10 frames for each audio sample

using the soundFile python module. For each sample, random chunks of the audio are taken in by using a random sample start and adding the window length. Here the batch size is just the number of times this is randomly done on the sample.

From there, the network is composed of three more networks following the structure CNN, DNN1, DNN2. The first layer in the architecture is a Convolutional layer that performs time-domain convolutions between the input waveform and some filters. The filters they use have learnable parameters. However, it starts as a rectangular band-pass filter and cuts off learned low and high frequencies. The MFCC could also be pre-loaded into the CNN filtering layer since MFCCs excel at extracting voice-profiling characteristics from the lower frequencies of human speech. Pre-processing utilizes a Hamming window defined as Equation 5.1 to create the windows of each of the frequencies.

$$w[n] = 0.54 - 0.46 * cos(\frac{2\pi n}{L}) \tag{5.1}$$

The first DNN layer has layers of the form [2048,2048,2048], each of which uses Leaky ReLU as its activation function. The second DNN layer functions as the output Layer with our 200 possible classes of speakers defined, with soft-max activation.

5.4.3 Lab Member Experiments

Using the model trained with the VoxCeleb data set, more speakers were appended to the total classes, and the model was retrained. The Lab data set speakers were appended to the total number of speakers, so now our total number of possible classes became 206, and the network was trained for 1000 epochs in small batches of 4.

5.4.4 GUI Implementation

A script was created to continuously record a live mic-line and feed it into the SincNet framework to test the trained framework. This script worked by first loading in the network through PyTorch with its past parameters. The program uses the Python module PyAudio to read from the primary microphone and record audio. For the audio to be compatible, the software uses a sample rate that works with the network along with the same number of channels. By using thresholding values on the amplitude of the signal, the program implements audio detection to capture only high amplitude noises like speech. Once the speaker stops talking, the program passes the waveform into a normalization method. The Wave is normalized by the maximum amplitude of 16384, for each frame of the wave file. Then the silence at the start of the recording and end is trimmed to ensure only speaking segments are passed to SincNet. From here, using the Python Wave

module, a finalized wave file is created to re-load into PyTorch. Then the waveform is passed through the loaded PyTorch network, and the network returns an integer class prediction. For each window chunk in the wave file, the program pools the predictions are and selects the most common prediction for each window. With the Python imaging library, the correct face image is selected and shown to the user. The script was primarily used to test lab-member training and show that it was able to differentiate between trained lab members.

Figure 6.1: Results of face-reconstruction using Reconstructing Faces from Voices [19]. Top left Face is the speaker, All other faces are generations of that speaker.

6 Experimental Results

6.1 Voice to Face

The proposed Voice To Face method has been evaluated on the Vox-Celeb and VGG-Face2 data sets and compared to the baselines set in [19].

First, we tested the Reconstructing faces results on a few of our lab members as a baseline and got the images in Figure 6.1 as a result. From this, we can see that samples of Nick Waterworth achieved older men with some similar facial features such as nose shape and size, and ethnicity. However, when introduced to other lab members of other than American origin, the model struggles at recreating their faces. This model also struggled at determining age of the speaker, since it produced the faces of older men rather than younger.

For a group of given speakers in the Vox-Celeb2 data set, the Generative Adversarial Network was able to generate close images of a celebrity face. However, when feeding in audio embeddings from outside of its data set, the network had trouble recreating the faces of an unknown speaker. Comparing to the work in [19] the face vector embedding framework creates higher resolution images. In the Figure 6.2, the output shows promise

Figure 6.2: Results of face-reconstruction using our Framework. Top left Face is the speaker, All other faces are generations of that speaker.

in determining gender and age from a speaker voice. However facial features such as nose and general face shape perform worse than before. From visual analysis for an unknown speaker, there could be consistent features that stay the same for a given number of runs like nose size and position. Certain facial features may not map to the face, thereby the variability of hair and other trivial face features will appear fluid in the output of the model.

6.2 Speaker Recognition

After training for 500 epochs, the trained model was scoring 16 percent error on its training data, while on its testing data, it scored a 75 percent error(25 percent accuracy). From these scores, Vox-Celeb1 as a data set does not compliment the structure of SincNet since Vox-Celeb has different conditions in the training data that is not present in the testing data. These conditions being different speaking environments, audio coverage, and speech length. The models created over-fits the training data of the set and fails on outside data that the model was not trained to predict. Like the Vox-Celeb trials, the network over-trained on the Lab data and produced a low error rate of 2 percent on the training data along with a higher error rate of 63 percent on testing. When running the lab model, the consistency of predictions was variable at best between lab members. Due to the inconsistency of vocal characteristics in between speakers in the lab data set, the model

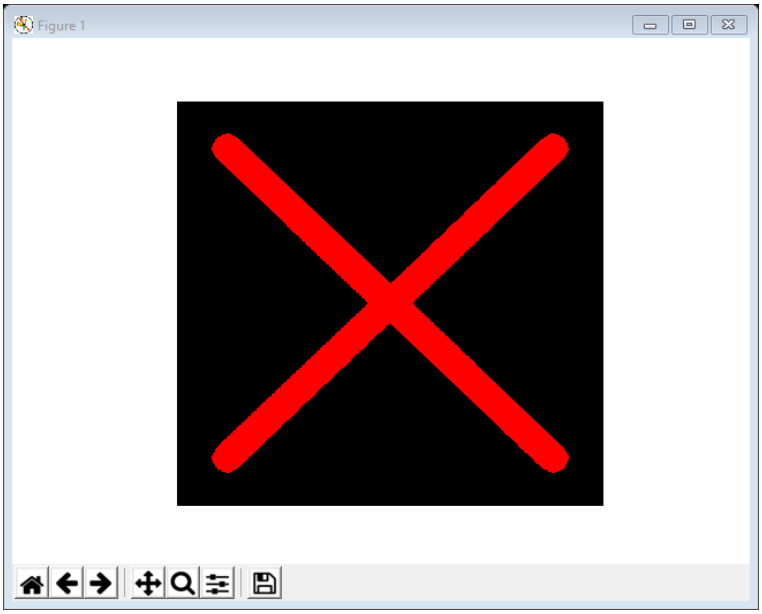

Figure 6.3: Graphic showing lab created graphics, from unknown speakers.

performed poorly on predicting the correct speaker on data not present in the training data set. When testing on Celebrity recordings re-played from the Vox-Celeb training data, the model performed better than on unknown speakers. These tests showed the effect of over-fitting experienced when using these data sets and over-training and over-fitting the data. Figure 6.3 and Figure 6.4 show the programs output based on speaker input.

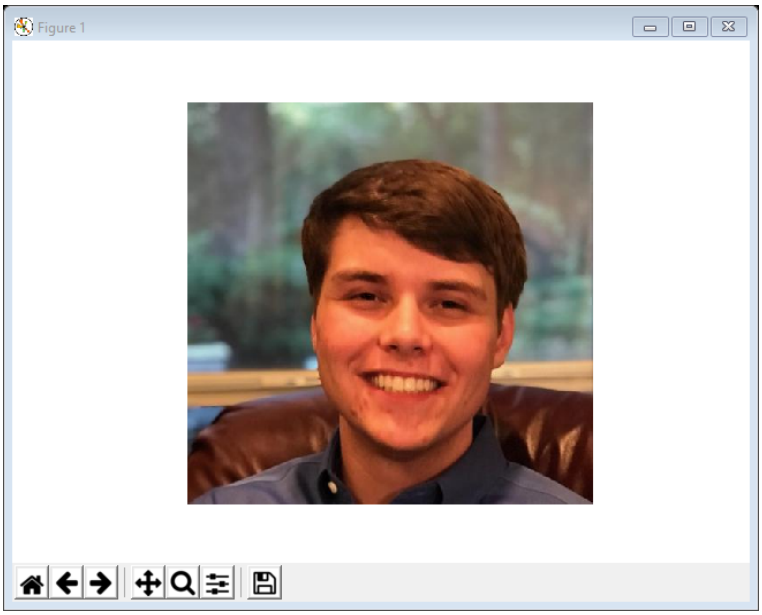

Figure 6.4: Graphic showing lab created graphics from a known speaker, Nick Waterworth.

7 Conclusion

The Voice to face network worked to create a more detailed image thanks to using the Vec2Face generation network, but with random speakers got a mixed result of similar-looking faces that had less facial features that mapped to the original speaker. Voice Independent features also played a big role in face construction and work as noise in the generation process. These features included hair, ear position, and eye color. Future improvements would be the removal of features in speech vectors that do not contribute to the face because they are mostly noise to the output vector. Thereby the output of the network would be more consistent with only mapping features that connect within voice-profiling. Another improvement would be performing training on a higher capacity of faces and speaker samples.

The Speaker recognition project was able to re-implement SincNet on the Vox-Celeb database, but due to aspects of the data set, over-training occurred and created an insufficient network. However, a successful GUI was implement using python libraries to show a speakers' face based on the ID returned by the model. For the speech recognition framework, it would be incorporating in a cross-modal network with the SincNet framework to see if adding face data would create a better network for face to voice matching than traditional features in cross-modal networks. Another improvement could be to add in better voice detection, so loud noises are not registered as a speech sample. In the end, both projects achieved a baseline of what each had set out to achieve as working models and software.

Bibliography

[1] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *International Conference on Automatic Face and Gesture Recognition*, 2018.

[2] J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

[3] Joon Son Chung and Andrew Zisserman. Lip reading in the wild. pages 87–103, 03 2017.

[4] Amir Jamaludin, Joon Son Chung, and Andrew Zisserman. You said that? : Synthesising talking faces from audio. *International Journal of Computer Vision*, 2019.

[5] Live-Audio-MFCC. Live-Audio-MFCC javascript mfcc visualizer software. `https://pulakk.github.io/Live-Audio-MFCC/tutorial`. Accessed: 2020-04-25.

[6] Mathworks. Mathworks spectrogram visualization. `https://www.mathworks.com/help/signal/ref/xspectrogram.html`. Accessed: 2020-04-25.

[7] Mathworks. Mathworks triangle window visualization. `https://www.mathworks.com/help/signal/ref/triang.html`. Accessed: 2020-04-25.

[8] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

[9] University of Alaska Fairbanks. University of Alaska Fairbanks perceptron figure. `https://www.cs.uaf.edu/2007/fall/cs441/proj1notes/schamel/`. Accessed: 2020-04-25.

[10] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet, 2018.

[11] ResearchGate. ResearchGate pool operation figure. `https://www.researchgate.net/figure/Demonstration-of-pooling-operation_fig5_325649211`. Accessed: 2020-04-25.

[12] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333, 2018.

[13] Stanford. Stanford activation function lecture. `http://cs231n.stanford.edu/slides/2019/cs231n2019lecture04.pdf`. Accessed: 2020-04-25.

[14] Stanford. Stanford cnn overview graphs. `https://cs231n.github.io/convolutional-networks/`. Accessed: 2020-04-25.

[15] SUPASORN SUWAJANAKORN, STEVEN M. SEITZ, and IRA KEMELMACHER-SHLIZERMAN. Synthesizing obama: Learning lip sync from audio. *SIGGRAPH*.

[16] Amirsina Torfi, Nasser M Nasrabadi, and Jeremy Dawson. Text-independent speaker verification using 3d convolutional neural networks. *arXiv preprint arXiv:1705.09422*, 2017.

[17] Aalborg University. Aalborg University mfcc worksheet. `http://kom.aau.dk/group/04gr742/pdf/MFCCworksheet.pdf`. Accessed: 2020-04-25.

[18] Yandong Wen, Mahmoud Al Ismail, Weiyang Liu, Bhiksha Raj, and Rita Singh. Disjoint mapping network for cross-modal matching of voices and faces, 2018.

[19] Bhiksha Raj Yandong Wen, Rita Singh. Reconstructing faces from voices. *arXiv preprint arXiv:1905.10604*, 2019.