SYNTHETIC SPEECH DETECTION USING DEEP NEURAL NETWORKS

RICARDO REIMAO

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE AND ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

MAY 2019

Abstract

With the advancements in deep learning and other techniques, synthetic speech is getting closer to a natural sounding voice. Some of the state-of-art technologies achieve such a high level of naturalness that even humans have difficulties distinguishing real speech from computer generated speech. Moreover, these technologies allow a person to train a speech synthesizer with a target voice, creating a model that is able to reproduce someone's voice with high fidelity.

With this research, we thoroughly analyze how synthetic speech is generated and propose deep learning methodologies to detect such synthesized utterances. We first collected a significant amount of real and synthetic utterances to create the Fake or Real (FoR) dataset. Then, we analyzed the performance of the latest deep learning models in the classification of such utterances. Our proposed model achieves 99.86% accuracy in synthetic speech detection, which is a significant improvement from a human performance (65.7%).

Table of Contents

List of Figures

1 Introduction

1.1 Synthetic Speech Detection Overview

Synthetic speech refers to any utterance generated by a computer. With the advancements in deep learning and other techniques, synthetic speech is getting closer to a natural sounding voice. Some of the state-of-art technologies achieve such a high level of naturalness that even humans have difficulties distinguishing real speech from computer generated speech. Moreover, these technologies allow a person to train a speech synthesizer with a target voice, creating a model that is able to reproduce someone's voice with high fidelity. Such technologies can have negative consequences, since one could maliciously impersonate someone's voice. An example would be training a model with the voice of a famous person and then using this model to generate an utterance with malicious content to defame the person publicly. This kind of impersonation can be seen in several videos online[1], where both image and speech were synthesized to generate a fake video. With this

[1] https://www.youtube.com/watch?v=cQ54GDm1eL0

research, we thoroughly analyze how synthetic speech is generated and propose methodologies to detect such synthesized utterances.

1.1.1 Synthetic Speech Generation Systems

The first step in creating a system that is able to detect synthetic speech is to thoroughly understand how the synthetic speech generators work. The first studies in generating speech with a computer date back to the late 80's [1]. However, it took decades for computer-generated speech to sound more natural. The traditional approach for computer-generated synthesized speech depends on using *Hidden Markov Models (HMMs)* and *Gaussian Mixture Models (GMMs)* to learn specific speech features and replicate them. Although those methods are able to produce clear and understandable speech, they lack naturalness. Recent studies explore the use of *Deep Neural Networks (DNNs)* for synthetic speech generation. DNN-based systems are able to ingest a large quantity of speech data and generate a model that can be used to generate utterances, even for unseen words. DNN-based systems usually generate more natural utterances than the HMM/GMM ones, and for this reason are being used in the majority of the modern *Text-To-Speech (TTS)* applications, such as Amazon Polly and Microsoft TTS.

1.1.2 Synthetic Speech Detection Systems

As any emerging technology, modern TTS systems can be used with malicious intent. One could use DNN-based TTS systems to create a speech model for a target person. With this model, the malicious actor could perform a series of spoofing attacks, including impersonation and/or bypassing automated speaker verification systems. As an attempt to minimize the likelihood of such attacks, researchers have been studying methods to detect synthesized speech. With the increasing concern on the malicious use of such technologies, researchers across the globe created a challenge, called ASVSpoof[2], in which they release a dataset of real and spoofed voices so the community can come up with ways to identify the computer-generated utterances. Several papers have been published showing methods to detect spoofed utterances. The majority of the proposed solutions are based on extraction of frequency features utilizing HMM and GMM models.

Although this challenge is an important milestone in the synthetic speech detection community, the dataset did not contain the most recent state-of-the-art TTS technologies. Also, as in the literature there are already methodologies that achieves high accuracy on the ASVSpoof dataset, we identified a need for a new dataset containing the latest TTS solutions to reflect our current speech synthesis scenario. Therefore, as the synthetic speech generation systems become more

[2] http://www.asvspoof.org/

complex, it is important to explore more complex solutions (such as Deep Neural Networks) for the synthetic speech detection task.

1.2 The Fake or Real Dataset

The first step toward training a Synthetic Speech Detection system is to collect a significant amount of computer-generated speech as well as real human speech. For our research we created a dataset, called *Fake or Real (FoR)*, which contains more than 84,000 synthetic utterances as well as more than 111,000 real utterances. Although previous researchers also generated datasets containing real and synthetic utterances [57, 54], in this research we focus on the latest speech synthesis technologies using neural network architectures. We also focus not only on open-source systems, but also on commercial tools that can be used to generate synthetic speech. The FoR dataset was openly released to the public under the GNU GPLv3 license. The dataset is published in four versions: *for-original, for-norm, for-2seconds* and*, for-rerecorded.* The first version, named *for-original* contains the files as collected from the speech sources, without any modification. The second version, called *for-norm*, contains the same files, but balanced in terms of gender and class and normalized in terms of sample rate, volume and number of channels. The third one, named *for-2seconds* is based on the second one, but with the files truncated at 2 seconds. The last version, named *for-rerecorded*, is a rerecorded version of the for-2second dataset, to simulate a scenario where an attacker sends an utterance through a voice channel (i.e. a phone call or a voice message).

1.2.1 Collecting Synthetic Utterances

Before starting to collect synthetic audio, extensive research was conducted to identify the latest methodologies in speech synthesis, being Text-To-Speech (TTS) systems or Voice Conversion (VC) systems. In this research, several open source and commercial systems were identified, such as:

- DeepVoice 3 [35]
- Neural Voice Cloning [4]
- Baidu TTS[3]
- Microsoft Azure TTS[4]
- Amazon AWS Polly[5]
- Google Cloud TTS with Wavenet[6]

[3] https://www.home-assistant.io/components/tts.baidu/

[4] https://azure.microsoft.com/en-ca/services/cognitive-services/text-to-speech/

[5] https://aws.amazon.com/polly/

[6] https://cloud.google.com/text-to-speech/

With the synthetic speech generators identified, the next step was to identify phrases that will be used as input for the TTS systems. To achieve a good spread of grammatical phrase formations, a dataset of English phrases was used[7]. This phrase dataset contains more than 152,000 English phrases as well as their translation to the French language. As in this study we focus on English utterances, the French part of the dataset was discarded as well as English phrases with more than 30 words. This results in a phrase dataset with more than 105,000 phrases in a wide range of grammatical structures. The list of phrases was then randomly divided into 40 phrase-buckets that was then distributed across the TTS systems.

After identifying the TTS systems and generating a list of English phrases, the next step was to run each TTS system with a set of phrases to obtain the corresponding generated synthetic speech. The utterance extraction process varies according to each TTS system and will be described in detail in the next sections, but in general, the open source TTS systems were ran locally while the commercial tools were used through HTTP APIs. With that done, we obtained a collection of synthetic utterances that will be later pre-processed and used in the training process.

[7] https://www.kaggle.com/percevalw/englishfrench-translations

1.2.2 Collecting Real Utterances

Collecting real utterances is a complex task: we need to ensure a variety of recording methods, a variety of speaker genders, a variety of speaker ages, a variety of accents and even a variety of microphones used for recording. This variety is required to avoid a situation where an algorithm learns patterns in the training audio instead of learning the real differences between a synthetic and a real utterance, and end up performing badly in unseen data. An extensive research was performed to identify speech datasets that could be used in this research. We identified and collected utterances from a series of open source speech datasets as well as other sources of real speech, such as TED Talks and Youtube videos.

After collecting the audio from the data sources, all long utterances (over 10 seconds) were split into 10-second maximum utterances. To keep the naturalness of the speech files, the split was done using the SoX[8] audio processing tool, which is able to detect silences in the audio and truncate the utterance in between phrases. With that done, we then have the final for-original dataset, which was later used as base for the next dataset versions.

[8] http://sox.sourceforge.net/

1.2.3 Audio Normalization

With both synthetic and real data collected and the for-original dataset finalized, the next step was to pre-process the data so it can be used by the machine learning algorithms. The pre-processing steps are described below and were performed in the following order:

1. Filetype Conversion: As the files were collected from several different data sources, the first pre-processing step is to convert all the files to the same filetype. As the WAV format is the most common format in machine learning and digital audio processing, all the files were converted to the WAV filetype.

2. Volume Normalization: As each speech source has their own volume settings, it is important to normalize the volume of all utterances to eliminate the possibility of volume becoming a distinguishing factor. All the utterances, both synthetic and real, were normalized to 0dB.

3. Sample-Rate Normalization: The majority of the TTS systems generate audio at 16kHz sample rate, while the majority of the real audio was recorded at 48kHz sample rate. To decrease training time, all the audio files were downsampled to 16kHz. Considering that the human speech typically ranges from 300Hz to 5000Hz, the downsampling to 16kHz should not cause major quality loss in the audio, since a 16kHz sample rate allows frequencies up to

8kHz.

4. Channel Mixing: As the majority of the TTS systems generate audio in a single channel (mono) and the majority of the real audio had two channels (stereo), all two-channel files were converted to a single channel using channel mixing, which is, combining two audio tracks into a mono track by scaling each track by 0.5 and adding the signals to result in a single track.

5. Silence Removal: In early experiments it was noted that synthetic utterances had around 0.5s of silence in the beginning and end of each utterance, while real utterances had a more random silence pattern. To remove any silence bias, we removed the silence from the beginning and end of each utterance.

6. Gender Balancing: The synthetic utterances were predominantly from female voices, while real audio was mainly from male speakers. To avoid any gender bias during training and classification, the dataset was balanced using downsampling. This resulted in a dataset with even distribution between genders.

7. Class Balancing: After gender balancing, the dataset contained more real utterances than synthetic utterances. To have a class balanced dataset, the dataset was downsampled to ensure a 50/50 distribution between synthetic and real utterances.

With these steps completed, we then have a pre-processed dataset, named *for-norm*, ready to be used in the experiments.

1.2.4 Length Normalization

In early experiments it was noted that synthetic utterances were considerably shorter than real utterances. As this can be a bias factor in the dataset, all the utterances with more than 2 seconds were truncated, while utterances with less than 2 seconds were discarded. This resulted in a dataset, named *for-2seconds*, containing only utterances 2-second long.

1.2.5 Speech Rerecording

To simulate a real world scenario, where an attacker sends a synthetic utterance through a voice channel (i.e. phone call, voice message, etc.), we rerecorded the for-2second dataset. The idea is playing the utterances using one device, and recording them with another (using a non-professional microphone). This smoothes the frequency spectrum (specially in high frequencies) and add room reverberation to synthetic utterances, making a more realistic scenario.

This rerecording process originated the fourth and last version of the dataset, the *for-rerecorded*.

1.2.6 Dataset Division

As is common practice in machine-learning research, the dataset was divided into training, validation and testing:

- Training: Contains 77.73% of the dataset, utilized to train the machine learning models. Gender and class balanced.

- Validation: Contains 15.58% of the dataset, utilized to validate the accuracy of the machine learning models. Gender and class balanced. The validation utterances are unseen during the training phase.

- Generalization Testing: Contains 6.68% of the dataset. Contains only synthetic voices from one unseen algorithm (Google TTS Wavenet) and unseen real voices. Gender and class balanced. It is utilized to test if the trained model can generalize and detect unseen TTS algorithms and unseen real voices.

With the dataset pre-processed and divided, it is ready to be used as training input for our synthetic speech detection classifiers.

1.3 Synthetic Speech Detection Classifier

The goal of a Synthetic Speech Detection Classifier (SSDC) is to accurately classify an utterance as real or synthesized. Although previous research has achieved very good results in the past [54, 58], they did not use in their dataset the latest state-of-art machine-learning TTS systems, such as DeepVoice 3 [35] and Google Wavenet [9]. We use five different approaches for extracting features from the audio: Fast Fourier Transform (FFT), Short-Term-Fourier-Transformation (STFT), Mel-Spectrogram, Mel-Frequency Cepstral Coefficients (MFCC), and Constant-Q Transform (CQT). Those features are then analyzed with 4 traditional machine learning techniques with global frequency analysis (also referred as "frequency based analysis") and 9 deep learning architectures. In this research we study the 65 possible combinations of feature extraction and analysis against two of the dataset versions: *for-2seconds* and *for-rerecorded*.

1.3.1 Frequency Analysis with Traditional Machine Learning

The Frequency Analysis approach consists of analyzing the frequency patterns of an utterance without considering timing features. As an example, using an Average Short-Term-Fourier-Transformation (A-STFT) we can quickly obtain information about the amplitude of the audio in each frequency range. This can be

[9] https://cloud.google.com/text-to-speech/

useful in simple classification tasks, such as classifying if the speaker is male or female, since male voices exhibit higher amplitude in lower frequencies, while women tend to have higher pitched voices resulting in higher amplitude in higher frequencies. Even though the timing features and original content of the audio are lost using this technique, previous research has shown that this approach can be useful in synthetic speech detection [54]. However, synthetic speech generators have evolved and the traditional approaches may not be as efficient as with previous TTS algorithms. In this research, we explore five different approaches for extracting the Global Frequency Features: Fast Fourier Transform, Average Short-Term-Fourier-Transformation, Average Mel-Spectrogram, MFCC, and Average Constant-Q Transform.

After extracting the frequency features for each audio file, we use the data in a variety of machine learning techniques to analyze the performance of each algorithm. With that, we analyze which machine learning algorithm performs better and create a baseline for comparison against the Deep Neural Network approach. The tests were ran using the Weka Project[10] and include the following machine learning techniques: Naive Bayes, Support Vector Machines, Decision Tree, and Random Forests.

[10] https://www.cs.waikato.ac.nz/ml/weka/

1.3.2 Deep Neural Networks

Although Frequency Analysis provides reasonable classification results, it discards temporal structures in an utterance, which may be useful for detecting synthetic speech. Previous research has shown that the variations of frequencies over time do increase performance in synthetic speech detection [56] and for this reason we need a solution that considers not only the frequency features, but also the timing information. An approach for processing both temporal and frequency structures is using audio representations (i.e. a spectrogram) with Deep Neural Networks (DNNs). The DNNs are good feature extractors and are able to learn both frequency and temporal structures given an audio representation of frequency over time of an utterance. The general idea of this approach is extracting the audio representations (STFT, MFCC, Mel-Spectrogram, CQT) from each utterance in the dataset and using those representations as input to a variety of DNN architectures. By doing this, we can compare which architectures perform better in the specific synthetic speech detection problem. For this study, the following DNNs were selected:

- 4-Layer Fully Connected Neural Network
- 2-Layer Convolutional Neural Network
- 3-Layer Convolutional Neural Network

- VGG16 and VGG19 [43]

- InceptionV3 [47]

- ResNet [45]

- MobileNet [20]

- XceptionNet [9]

1.4 Research Contribution

With this research, we provide to the research community a framework for the study of synthetic speech detection. This framework includes:

- A dataset for speech synthesis studies containing more than 87,000 synthetic utterances as well as more than 111,000 real utterances, which can be used for training classifiers as well as other frequency analysis tasks;

- A thorough analysis of the performance of deep neural networks for synthetic speech detection, showing the DNN architectures' performance for this specific problem.

1.5 Thesis Outline

This thesis is divided into five chapters and three appendix sections. In this first chapter, we presented an overview of our research and the expected contributions to the community. In Chapter 2, we present an extensive literature research in synthetic speech detection and related areas. In Chapter 3, we present the details of the FoR Dataset and its versions. In Chapter 4, we present the main experiments related to our research and a discussion about their results. In Chapter 5, we present a conclusion chapter, with the key findings of our research and potential future work. Extra details about the dataset pre-processing can be found in Appendix A. Additional experiments related to the frequency analysis approach can be found in Appendix B, while additional experiments related to deep learning techniques can be found in Appendix C.

2 Background

In this chapter we summarize the ideas and discuss previous research in the main areas touched by this research, including: Neural network architectures; Digital audio and machine learning; Synthetic speech synthesis; and Computer-generated media detection. This chapter is organized in four sections, one for each relevant area. Each section contains an overview of the area and subsections describing the relevant work on the topic. We describe how each area evolved and how they are important in the synthetic speech detection task.

2.1 Neural Networks

The idea behind neural networks goes back to 1943, when researchers created a computational model called threshold logic [36]. The model consists of a collection of connected units that perform logic tasks. Each unit, also called neuron, is composed by an input, an activation function and an output. The neurons can be interconnected using their inputs/outputs and a weight factor. This forms a computational network, also called, artificial neural network. The Figure 2.1 shows an example of a basic 2-layer fully connected neural network, containing two inputs and one output. It is possible to note that every node of the network is a function of the previous layer multiplied by weights. Each "column" of the neural network is called a layer. As the number of layer increases, the computational cost of the neural network increases.

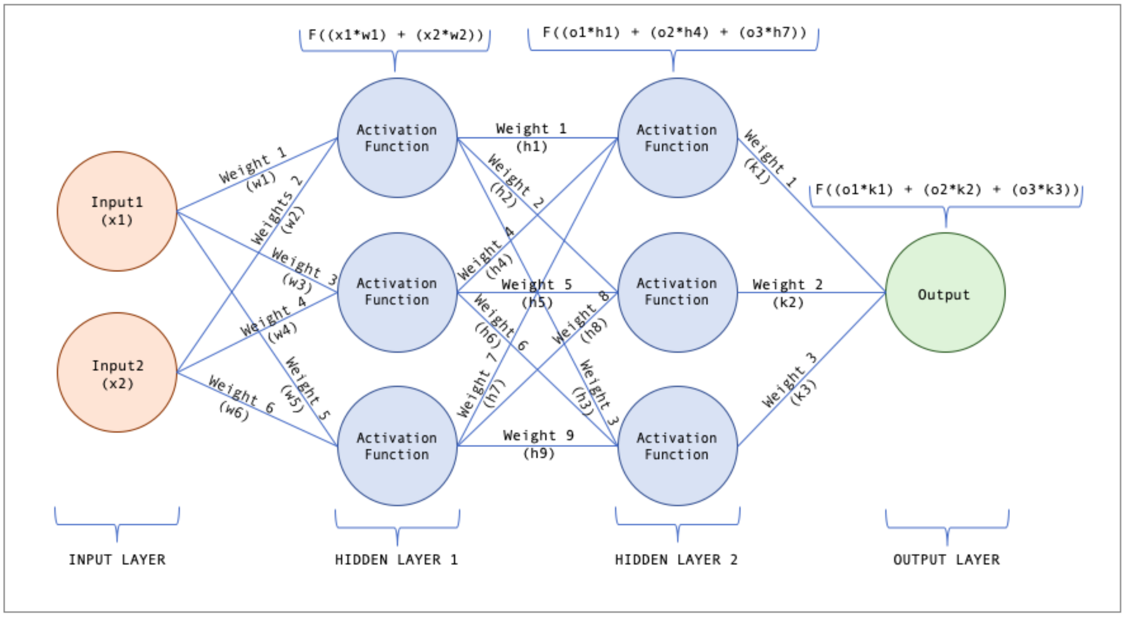


Figure 2.1: Example of a fully-connected neural network

The real potential of neural networks was only explored later in 1975, when researchers from Harvard University published their back-propagation algorithm [53]. This algorithm enabled neural networks to efficiently learn by adjusting the weights in each node, making it possible to train complex neural network models using supervised data. The back-propagation algorithm is still in use today, decades after its creation.

Since the publication of the back-propagation algorithm, scientists started to use neural networks in more complex tasks, such as multi-variable inputs and image processing. Such tasks contain a large amount of input parameters and require a large amount of layers and neurons. This complexity creates issues such as the vanishing gradient problem, in which due to the back-propagation process, the numbers on the neural network shrink exponentially. The vanishing gradient problem can be minimized with the use of rectifiers, such as the ReLU [17]. Another important achievement that improved image processing was the introduction of the max-pooling technique [22], which minimized the impact of shift variance and deformation in image recognition.

2.2 Main Neural Networks Architectures

For this thesis, several publications were analyzed to understand which architectures are relevant for the synthetic speech detection domain. The following subsections describe the relevant architectures for this research.

2.2.1 Convolutional Neural Networks (CNNs)

When first published in the 80's, the idea of *Convolutional Neural Networks (CNNs)* was a groundbreaking finding [15]. More than a decade later, researchers were able to use this architecture to ingest a multi-dimensional input (eg. an image) and learn dimensional/positional relations between the pixels [26], which enable the neural networks to recognize shapes and patterns. The main idea of a convolutional layer is breaking an image into small squares and comparing each square to learnable filters through the use of a convolution operator. Stacking convolutional layers with classic fully-connected layers creates a powerful architecture that is able to identify patterns in an image, from shapes (in the lowest layers) to complex objects (in the highest layers). These architectures can be leveraged for other domains, such as audio and speech processing, hence the reason for studying these image processing methodologies.

One of the main publications related to CNNs is an article from 2012 by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton regarding the use of convolutional neural networks for image classification [23]. In their publication, the authors discuss the use of an eight-layer convolutional network over the ImageNet dataset [38] to detect and classify objects in pictures. The research incorporates several novelties in CNNs, such as: the use of ReLU non-linearity functions (instead of traditional binary and sigmoid functions); the use of dropout layers in order to minimize overfitting; and the use of data augmentation techniques for decreasing overfitting and increasing accuracy. The results of this CNN were impressive at the time of the publication. It achieved 15.3% of top5 error in the ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) competition, which is an error significantly lower than any other competitors at that time. Due to its novelty and impressive results, this paper is considered one of the most influential publications in the field.

One of the biggest challenges when working with CNNs and images is to understand what the neural network is learning and how the model is predicting that an input belongs to a specific class. To help in this challenge, researchers from MIT published in 2016 a methodology to create class activation maps (CAMs) that shows which areas of an image are more relevant for the classification process [63]. The idea behind CAMs is to use the global average pooling layers proposed in [27] on the convolutional feature maps to identify regions of the image that are active during the classification process. The trained CAM model can be used to not only understand what the model is learning, but also to perform image localization. The authors achieved 37.1% top-5 error on the image localization challenge of the ILSVRV2014 dataset, which is close to the best results published regarding this challenge (34.2% [24]). An implementation of this work can be found on github[11].

Following the groundbreaking research of Krizhevsky et al. on CNNs for image classification, Matthew D. Zeiler and Rob Fergus published a paper in which they analyze methods to improve the work done previously [60]. The authors argue that the architecture presented in the previous work was based on trial and error, which means that there is a need for properly understanding the methodology so improvements can be made. The authors use a modified version of DeconvNet [31], which is a reverse engineering technique for convolutional neural networks. With this modified DeconvNet, they are able to visualize each layer of the network in the format of an image, which allows them to observe which features are extracted in each layer, helping them to identify which layers are relevant for the problem and which are not. The DeconvNet methodology works by implementing the same components of the original CNN, but in the reverse order, allowing each layer to be translated to the pixel space (image). With the improved model, the authors reach a record in image classification performance: 14.8% of top5 error on the ImageNet dataset, which was the best result at the time the paper was published. Moreover, to prove the efficacy of the new methodology, the authors use the improved CNN model in other image datasets, also obtaining outstanding results compared to previous work.

[11] https://github.com/jacobgil/keras-cam

2.2.2 Long-Short-Term-Memory (LSTM) Networks

Convolutional neural networks are good at identifying dimensional/positional correlation in the input, however they lack in understanding temporal correlation (e.g. understanding a sequence of events). They also have the constrain that the input/output size must be fixed, which can be a problem for audio/speech processing since the duration of a speech segment can vary.

The idea of *Long-Short-Term-Memory (LSTM) Networks* is based on the concept of Recurrent Neural Networks (RNNs), in which the output of one layer is used as input to the same layer [19]. This provides the architecture with some kind of "memory" and allows it to understand correlation in sequences. In this architecture, instead of using the classical neurons, it uses a memory cell, composed by several components and operations to provide long term memory. The major drawback of LSTM networks is that as the gap between instances (eg. words in a phrase) increase, the learnability of the correlation decreases, and for this reason, several modified versions of LSTMs were proposed.

In 2014, Ilya Sutskever, Oriol Vinyals and Quoc V. Le published a paper introducing an end-to-end methodology for sequence-to-sequence learning using LSTM networks. Previously, all the work related to sequence learning was based on inputs and outputs of fixed lengths, which restrict their use for several real-world problems, such as natural language translation. The authors suggest the use of two Long Term Short Memory (LSTM) networks to remove the fixed input/output constraint: The first LSTM maps the input sequence into a vector of fixed dimensionality, while the second LSTM decodes the target sequence from this fixed-dimensionality vector. This multi-layer approach solves the input fixed-size problem, allowing the inputs and outputs to have any size. One interesting side finding of this work is that inverting the order of the input (eg. inverting the order of the words on the input phrase) increases the performance of the methodology. Although the authors do not provide a theoretical explanation for this fact, it is believed that inverting the order of words reduces the distance between them during the training phase (where the output is given after the input), improving the performance in long inputs. The performance achieved in this work (BLEU[12] score: 34.8) is very similar to the performance of the state-of-art methods (BLEU score: 33.3). As this architecture was new at the time, the authors argued that it could be fine-tuned and potentially surpass the state-of-art performance.

Although the LSTM architecture is capable of learning short term structure dependencies, the model accuracy quickly decreases if the length of the input (number of words in a phrase) increases. The main reason for this fact is that the encoder in a traditional LSTM network only has a limited-length vector to represent the

[12] Bilingual Evaluation Understudy, more information can be found at https://www.aclweb.org/anthology/P02-1040.pdf

whole phrase, meaning that in long phrases information may be lost in the process.

To improve the performance of recurrent neural networks for long inputs, researchers published a paper about an alignment mechanism that is able to identify which parts of the input are important for predicting the output [6]. The main difference between the regular encoder-decoder architecture to the alignment mechanism is that the first one tries to encode the whole input into a fixed-length vector, while the proposed methodology learns which inputs have higher impact on the inference and uses them during decoding. The authors apply this idea to the machine translation problem and also argue that in this specific problem, the "future input" (next words in the phrase) are also relevant in the inference process. To solve this future dependency, the authors propose a bi-directional *Recurrent Neural Network (RNN)*: one part of the RNN analyses the phrase in the normal order and the other part analyzes the words in reversed order. This allows the network to have visibility of past and future words during inference. The results of this proposed methodology show a higher performance (especially with regard to long input phrases) compared to previous machine-learning-based translation solutions. Moreover, the translation accuracy using this methodology is close to the accuracy of the state-of-the-art translation mechanisms using non-machine-learning phrase-translation. This attention mechanism is used by several key publications and real world applications, such as the Google Translator and the Baidu Deep Voice TTS system, which will be discussed later in this chapter.

Following the work on sequence-to-sequence neural networks, researchers published a paper where they combine previous sequence-to-sequence techniques to approach the speech-to-text problem [8]. The authors propose a methodology composed by a listener and a speller that is capable of processing audio input and outputting a sequence of words contained in the audio. The listener component is a pyramidal Bidirectional Long Short Term Memory (pBLSTM) neural network. Basically, the pBLSTM inputs audio through filter-bank-spectra frames and reduces its dimensionality using a pyramidal approach in BLSTM. The listener outputs a vector that represents the input audio. The speller component (also called "Attend and Spell") works as a decoder by adopting an attention-based LSTM approach. At each step, the transducer outputs a probability distribution over the next character based on the input audio and the previous characters seen. The key contribution of this publication is an end-to-end solution for listening-to-spelling problem. However, there are other interesting side findings that are relevant for our synthetic speech detection research. As an example, the authors describe a technique for data augmentation for audio databases that consists in adding noises (room simulators, background noise, reverberations, etc) to the original audio. Those techniques will be used in our research, since it helps to increase the resilience of the network and avoid overfitting over the selected training data.

2.2.3 Generative Adversarial Networks (GANs)

Researchers from the University of Montreal published a paper in 2014 to introduce a new approach for distribution learning using machine-learning algorithms called *Generative Adversarial Networks (GANs)* [18]. Their idea is to use two opposing (adversarial) neural networks: the generator, which generates an output (e.g. image); and the discriminator, which is basically a classifier network.

The key here is that each network helps to train the other: the generator creates an image which is then evaluated by the discriminator. If the image is not "good enough", the weights of the generator are updated. To better understand the concept, the authors use an analogy of a counterfeiter (generative model) trying to produce synthetic currency that the police (discriminator model) will not flag as synthetic. So the counterfeiters will learn ways to improve their currency generator to not get caught.

Mathematically, this can be translated into a min-max game, in which one part (generator) wants to minimize the chances of a mistake, while the other (discriminator) wants to maximize the amount of synthetic utterances flagged. Although the authors argue that this architecture can be used with any machine learning technique, they also argue that using neural networks provides an advantage due to the fact that the whole scheme can be trained using the same back-propagation process. This is done by training the discriminator for n interactions (using data from the dataset and from the generator), then, training the generator (using the discriminator as ground truth), then, training the discriminator again for another n interactions, and so on.

The authors do not present any concrete comparison numbers between the results of GANs with previous approaches, however, they present images generated by the proposed approach and argue that the results are comparable to the existing methodologies. Moreover, the authors claim that their results can be more appealing since the generated images are sharp, which contrasts the other methods (such as encoder/decoder networks) that generate blurred images.

2.2.4 State-of-Art Neural Networks

With the advances in hardware, machine learning researchers are able to experiment with more complex models. As machine learning techniques are basically a sequence of matrix multiplications, the latest GPUs (which are optimized for matrix multiplications) are now able to solve in hours computations that used to take weeks. This allowed the research community to increase the complexity of the models, by increasing the number of layers and/or by adding more complex layers in their traditional neural networks. This increase in the model complexity translates into higher accuracy in complex problems.

In late 2014, researchers from Google published a paper introducing a new architecture called Inception, which not only explores deeper neural networks, but also wider [46]. The idea is to use the traditional convolutional layers in parallel to create a module called *Inception Module*. These modules (with parallel convolutional layers) are then stacked up to form a deep neural network. This approach not only achieved state-of-the-art results in image classification at the time of publication, but also contains significantly fewer parameters than predecessors, showing that with a proper architecture it is possible to achieve better results with less computation efforts. The core of the Inception network is based on the inception module, which is a parallel aggregation of convolutions with various convolution sizes (1x1, 3x3, 5x5 and 7x7). These convolutions are then concatenated by a filter concatenation, which merges the results of the parallel layers and is used as input for the next inception module. It is important to note that all the parameters (including filters) are learned during the backpropagation process, improving the results achieved. The authors implemented a neural network, called GoogLe Net, based on the Inception architecture. This neural network achieved state-of-the-art results at the ILSVRC14 competition, which is a well-known image-processing event. The implemented network contained 22 layers: 3 traditional convolutional layers, 18 inception modules and one fully connected layer. The implementation contains three soft-max layers at different stages, which are then averaged to generate the final output. The Inception architecture is later improved in Inception v2/v3 [48], and recently in Inception v4/ResNet [45]. In these newer architectures, the researchers experiment with variations in the inception module (such as the use of residual networks) and, again, achieve state-of-the-art results by the time the paper was published.

The Inception architecture also motivated the research of similar architectures that explore the idea of Network-in-Network, or in other words, using stacked computational modules. The *Xception Net* is inspired by the Inception Net, however, instead of using inception modules, the researchers use "Extreme Inception" modules, which are composed by depthwise separable convolutions [9]. The use of depthwise separable convolutions reduces significantly the computational cost of the network, allowing the network to grow as deep as 36 convolutional layers (plus fully connected and softmax layers) and to learn more complex features of the input. The researchers of the Xception Net use the ImageNet dataset [38] to compare the performance of the Xception architecture with the predecessor Inception architecture. The results shows a small improvement in accuracy but a considerable improvement in training performance.

Later in 2017, researchers also from Google proposed a new architecture type, called Transformer, which is based only in attention mechanisms [52]. This architecture does not use the typical convolutional or recurrent layers. Instead, it uses only attention-based layers. In their publication, the researchers apply this novel neural-network architecture for machine translation and achieve state-of-the-art BLEU scores. Moreover, the training time of the Transformer network is up to a hundred times faster than previous state-of-the-art models. Considering the translation problem, the idea behind the Transformer architecture is to learn the dependencies between the words independent of how far the words are and generate the output based on these dependencies. Also, in the Transformer network, the output word is based not only on the input phrase, but also on the previous output words, improving the context-awareness of the model. One of the main differences to previous LSTM networks with attention is that the Transformer architecture is based on multi-head attention components, as opposed to the single attention head used in LSTM networks. This multi-head attention component allows the network to learn from information in different representation subspaces and different positions, meaning that the heads are able to learn things like coreferences and correlations at the same time. To test the flexibility of this model, the researchers applied the same architecture in different problems, such as constituency parsing and phrase generation. In both problems, the proposed model outperformed the state-of-the-art methodologies by a significant margin. That publication opens the door for future research using Transformer networks for other areas, such as our speech processing research. Another key contribution for the research community is that the authors also created a library, called tensor2tensor, that allows the easy creation and training of such networks.

2.3 Digital Audio and Machine Learning

With the increase in the research of machine learning methods, the academic community started to experiment with the use of machine learning techniques for digital audio processing. The improvements in hardware allowed neural networks to become more versatile and ingest more complex inputs, such as audio data. This allowed a variety of interesting applications, from audio processing to audio synthesis. In this section, we discuss publications in three areas of digital audio that are directly related to our research: Audio synthesis; Audio processing; and Speech Classification.

2.3.1 Audio Synthesis

The first interesting machine learning application for digital audio processing is audio synthesis. With the use of neural networks, researchers were able to both input and output audio from a model. It is important to note that the evaluation of synthesized audio is subjective, since audio and music perception are dependent on the listener.

In late 2016 a ground-breaking paper called "Wavenet: A generative model for raw audio" was published by researchers from Google UK [32]. In their research, the authors show that using a modified version of CNNs allows dependencies on past inputs to be learned. This architecture, called *Wavenet*, can be applied to several audio-related problems such as *Text-To-Speech (TTS)*, music generation and speech recognition. Prior to Wavenet, the majority of the research on machine learning for audio processing were using *Recurrent Neural Networks (RNNs)* due to its ability of learning sequences. Before the publication, CNNs were adopted mostly for image recognition/classification, due to their ability to recognize patterns using convolutional filters. To be able to learn sequences using CNNs, the authors used "dilated causal convolutional layers", which expand the receptive field by skipping the output of a normal convolution by few steps. In this way, the output of a higher-layer is based on a large number of inputs in the lower layers. This architecture is more efficient than the previously used RNNs, meaning that the training time is lower and the prediction accuracy is higher. In the paper, this architecture was applied to several audio processing problems. The most interesting results were on text-to-speech generation, in which this methodology was able to generate high-quality speech audio that surpassed any previous TTS methodology. In fact, the score given by a survey test shows that the WaveNet-generated speech is very close to natural speech. This paper is important to show the possibility of the use of convolutional neural networks for audio processing, which sparked the interest of other researchers to improve this architecture. The only downside of this publication is the amount of data and time required to generate the output: the dataset was composed by more than 20 hours of training data and it required around 90 minutes to synthesize one second of audio. This is improved in future publications, such as DeepVoice 3[35], which are capable of generating speech in real time.

The publication of the WaveNet paper inspired further research in the audio processing area. For instance, the use of a WaveNet-like architecture to synthesize musical notes [13]. The key achievement in this research is the development of an architecture based on Wavenet that is able to learn long-term structures without external conditioning, meaning that this improved architecture performs well on long scale signals, which was a problem for the original Wavenet implementation. This is achieved by using a deeper convolutional neural network (30 layers) and a variety of specific values for strides and kernels. Another contribution that this paper brings is a large scale dataset containing musical notes. This dataset contains more than 300,000 musical notes across more than a thousand instruments. It is important to note that the majority of these audio snippets were generated by computer software, which doesn't contain the human aspect when playing an instrument.

2.3.2 Audio Processing

The second interesting application of machine learning into the digital audio domain is on audio processing. In the last decade several papers were published showing how neural networks can be applied in audio processing problems that are too complex to be solved by traditional procedural applications.

As example, in 2015, two researchers from University of Alberta submitted their research regarding using *Deep Belief Networks (DBNs)* to transcribe guitar audio [7]. The main idea proposed by the authors is to use two DBNs to estimate the pitch and note given an audio frame. With those values estimated for each frame, a MIDI file is generated with a representation of the original audio. With the MIDI generated, the authors then use a non machine-learning technique to translate the MIDI into common western music notation and guitar tablature.

2.3.3 Speech Processing

The third interesting application of machine learning into the digital audio domain is the use of neural networks for Speech Processing, which includes speaker isolation, emotion recognition and synthetic speech synthesis (which will be discussed in the next section).

The speaker isolation problem, also known as the "cocktail party" problem, aims to mimic the human capability of isolating and paying attention to one speaker in an environment with noise and/or several other speakers. Previous research only accomplished speaker isolation when the same audio was recorded by two microphones in different positions. However, in April 2018, researchers from Google proposed a methodology for speaker isolation using a combination of audio and video [14]. The intuition is that it is easier to identify the source of an audio when the solution can "see" the face of the speaker, due to mouth movements and face expressions. The first challenge for the researchers was to have a dataset containing high quality audio and video from single speakers. For that, the authors used video snippets from high-quality one-speaker sources, such as lectures, TED talks and tutorial clips. Those videos were then broken down into small clips (3-10 seconds long) and filtered to remove the clips where the face of the speaker is not visible. With that, the authors build a dataset with more than 2000 hours of video segments of single speakers containing high-quality audio and video, and with a clear picture of the face of the speaker. This dataset, named AvSpeech Dataset, is available to the public and it is one of the main contributions of the research. With the dataset ready, the authors then implemented a neural network that has as inputs the audio and video (containing multiple speakers) and outputs isolated audio segments for each speaker. This neural network contains two input processing modules: the convolutional dilated network for audio processing and the convolutional neural network for image processing. After the initial processing, both image and audio features are merged using an audio-visual fusion layer. This layer is then input into a bidirectional LSTM layer and three fully connected layers. The neural network then outputs audio spectrograms for each face, which can then be converted to waveforms. The results of this solution are posted on the Google Research blog[13] and are impressive: Just by inputting a video and selecting the face of one of the speakers, the system outputs the audio for the person. The audio quality has room for improvements, however the voice is clear and it is possible to understand the words spoken by each speaker. More than just publishing the dataset, the authors also released the source code of the solution, which is written in TensorFlow, opening a door for the community to research and improve the proposed technology.

Other significant area on speech processing is the speech emotion recognition (SER). Several papers were published introducing deep-learning based methodologies to classify the sentiments in an utterance [49, 44].

[13] https://arstechnica.com/gadgets/2018/04/google-works-out-a-fascinating-slightly-scary-way-for-ai-to-isolate-voices-in-a-crowd/

2.4 Speech Synthesis

One of the key aspects for synthetic speech detection research is to understand how synthetic speech is generated. In this section, we describe how academic research evolved such that the generation of synthetic speech was possible. We start this section by presenting traditional non-machine learning techniques for speech synthesis. Then, we discuss the first relevant publications that introduced machine-learning-based Text-To-Speech solutions. Last, we discuss the state of art of Speech-to-Speech synthesis.

2.4.1 Traditional Speech Synthesis

Synthetic speech refers to the creation of speech by a machine, which can be a mechanical equipment (such as a replication of the vocal tract) or software that is capable of generating speech. In this research we focus on the latter, that is, computer software that is able to replicate human speech. This software is divided into two categories: Text-to-Speech (TTS) systems and Voice Conversion (VC) systems. TTS systems are able to synthesize human-like speech based on words or phonemes, while VC systems convert the voice of an utterance into another voice (keeping the same word content).

One of the first speech synthesis techniques published is through speech concatenation [10]. The idea is to concatenate several pre-recorded speech segments to form a new phrase. This technique produces an utterance with a natural-sounding voice but with several audio glitches due to the lack of speech flow and differences in voice variations during speech. However, even with poor speech naturalness, this speech synthesis technique is adopted in several applications, such as broadcast messages [25].

Another speech synthesis technique is through formant synthesis. The idea is using acoustic models with tuned parameters (such as voice, fundamental frequency and noise) to synthesize speech [50]. These systems do not use human speech samples as input, resulting in a robotic-sounding speech which can be useful if a human-like voice is not required.

Articulatory speech synthesis is a technique that models the human vocal tract and the vocal biomechanics to produce speech. Although this technique was not adopted in large commercial systems, research on this topic is still active [28].

HMM-based speech synthesis algorithms are based on *Hidden Markov Models (HMM)*, which are capable of modulating speech properties, such as frequency spectrum, fundamental frequency and duration. Several implementations of such models can be found online[14] and in published articles [61], however the naturalness of the generated speech is low compared to the latest deep learning methodologies.

[14] http://hts.sp.nitech.ac.jp/

With the increase of the use of deep neural networks for media synthesis, the scientific community started to investigate ways of generating speech using neural networks. Such techniques are discussed in Section 2.4.3.

2.4.2 Style Transfer

The concept of style transfer was originally conceived for images. It consists of extracting the style from one painting (e.g. brushstrokes, colours, etc.) and applying it to the content (e.g. buildings, a mountain, a river, etc.) of a second image. This image style transfer concept is interesting for synthetic speech because in speech we also have the idea of the content (the spoken words) and voice style (pitch, intonation, etc.).

One of the first major publications on image style transfer was in 2016 by two German authors. At that time, three researchers from University of Tuebingen (Germany) published an interesting paper about the use of CNNs for image style transfer [16]. The idea is to create artistic images with high perceptual quality by extracting the artistic style from one image (ie. painting) and applying it to the content of a target photograph. For that, first, the authors formalized two concepts regarding image analysis with CNNs: Image Content and Image Style. The first one, as the name suggests, is the content of the image, or in other words, what is in the image (i.e. buildings, animals, objects, etc). Image Style is how the content is represented (i.e. the colour of the building, the rotation of the object, textures, etc.). As Convolutional Neural Networks are capable of extracting the content from an image (i.e. for image classification), the authors had the idea of using a CNN to extract content and the style of images. The authors use a VGG16 neural network architecture [43] with weights pre-trained in the ImageNet database as base for their work. For the content extraction, the authors ran the original photograph through the VGG16 neural network and extracted the feature vector from the last fully-connected layer, since it is the layer that contains the high-fidelity content representation. For the style extraction, the authors run the style image (i.e. a Van Gogh painting) through the same VGG16 network. However, to extract the style, the authors extracted the difference between the original image and the content-representation of the image in each layer of the neural network. The difference between the content-representation in each layer and the original image is the style representation. The authors extracted the style representation for each layer of the CNN and summed them up to generate the final style representation. For the image generation process, the authors define the loss function as a combination of style loss and content loss. This loss is then minimized by a modified version of the gradient descent algorithm: Normally, in a typical neural network implementation, we aim to minimize the target loss function by varying the weights of the neural network. However, in this paper, the authors minimize the loss by keeping the weights fixed (trained for image classification) and varying the pixels of the target image. In this way, the final image is constructed by the gradient descent, which after few thousand of iterations outputs the final image. The paper presents interesting results, where the style of famous paintings was transferred to regular photos. The generated images present a high perceptual quality, meaning that they look produced by real artists.

In 2018, a paper was published discussing the use of style transfer in the audio domain, including music and speech [42]. The authors of this paper argue that using only a log-magnitude STFT to represent the audio is not sufficient to represent rhythmic and harmonic nature of audio. Therefore, they proposed an improved methodology using the log-magnitude STFT with other techniques, such as Mel spectrograms or *Constant Q Transforms (CQT)*. The idea is that merging the original log-magnitude STFT with the other techniques will provide a better representation of the rhythm and harmonic style. The authors also argue that adopting a key-invariant content representation may improve the final results. Considering that the audio domain has specific characteristics that are not present on image processing (such as the fact that audio is a sequence), the authors also suggest to change the neural network architecture by using SELU activation units instead of RELU, as originally proposed. The results presented by the paper showcase better musical quality than the original audio-style transfer previously proposed in the literature, however, the generated audio is still far from a proper audio-style transfer. In some of the results it is hard to relate the generated sound with the original content, which leaves room for improvement in the audio style transfer domain.

2.4.3 Deep Learning Text-To-Speech Synthesis

The majority of the machine-learning based Text-To-Speech systems use neural networks to learn from utterances how to produce an audio signal (speech) given a sequence of characters (phrase). In the early days, most of those systems were being developed by private companies and their architectures were proprietary. However, researchers from the Baidu Labs published in early 2017 a paper about their project called *Deep Voice*, which aims to create a production-level end-to-end solution for Text-To-Speech (TTS) [5]. The key points of Deep Voice is that it does not require any specialist knowledge during the training/inference process and that the solution is able to generate audio in real-time. Deep Voice breaks down the TTS problem into five models:

- Grapheme-to-phoneme model: Responsible for converting written text (eg. English, Chinese, etc) into phonemes. This is helpful as in the majority of the languages, the same sequence of characters may produce different sounds based on the surrounding characters. For example, in the words "Frost" (Phoneme: frôst) and "Roast" (Phoneme: rōst), the sequence "ro" have totally different sounds, resulting in different phonemes in each word.

- Segmentation model: Utilized only during the training phase. It is responsible for finding the start and ending of phonemes in an audio signal. Once the start and end is found, the audio is isolated so the neural network is able to learn the sound for the exact phoneme.

- Phoneme duration model: Responsible for determining the duration of a phoneme. This is helpful to generate natural-sounding audio, since the same phoneme can have different duration depending on the word, or even on the context of the phrase. As an example, the phoneme "mo" in the words "model" and "more" has different duration: in the first one, it is a short sound, while in the second it's a longer sound.

- Fundamental frequency model: Responsible for predicting whether a phoneme is voiced. If it is, predicts the fundamental frequency throughout the phoneme's duration. This is also very important for generating natural-sounding audio, since in the majority of the languages there are phonemes in words that are not pronounced. As an example, in the word "island", the "s" is silent and should not be pronounced.

- Audio synthesis model: Responsible for generating audio based on the phoneme, duration and fundamental frequency. This model is heavily based on the

WaveNet project, however it implements several improvements that allow the audio generation to be done in real-time.

The majority of the previous works on machine learning based TTS only implemented one or two of those models and relied in specialist-engineered solutions for the other parts of the TTS. However, the problem with relying on specialists is that it may take weeks to fine tune the parameters and find a natural-sounding voice. In the Deep Voice project, all five components are implemented using neural network models, which requires minimal specialist effort. According to the publication, these machine-learning models have similar or better performance than the traditional non-machine learning methods. Another key contribution of the paper is the fact that the authors share a good amount of detail on how this architecture can be implemented to maximize the performance over CPU and/or GPU. The authors compare the execution using several parallelization methods and propose an implementation with impressive results. The authors claim that the inference in DeepVoice is 400 times faster than WaveNet.

Few months after the publication of the Deep Voice paper, researchers from the same company published Deep Voice 2, which is an expansion of the first proposed methodology [3]. In the second paper, the authors propose the following improvements to the original publication: Multi-speaker support, segmentation of modules and increase in training data. Although the overall architectures of both methodologies are very similar, the authors tune the system to achieve better performance and provide the above mentioned new features. The key difference between Deep Voice 1 and Deep Voice 2 is the support for multi-speaker datasets. This can be achieved by using a low-dimensional embedding of each speaker in each module of the Deep Voice 2 architecture, so during training and inference the network is able to assimilate the features of each specific speaker. Another improvement from Deep Voice 1 on the new proposed methodology is the use of a much larger dataset, with more than 250 hours of speech. The results presented by Deep Voice 2 were fairly impressive at the time of the publication. The audio quality surpassed the previous TTS methodologies and accuracy numbers shows that the proposed model is approaching the ground truth. However there is still a statistically significant difference between a human speaker and a computer-generated voice.

After the success of Deep Voice 1 and Deep Voice 2, researchers from the same company published a third paper regarding their continued work, which they called Deep Voice 3 [35]. In this new paper, the researchers proposed a new architecture totally different from the one used in Deep Voice1 and Deep Voice2. With this new architecture, the system is able to train faster, allowing them to scale up to more than 800 hours of training data containing more than 2400 voices. The new architecture is based on an encoder-decoder scheme, or in other words, a fully-convolutional sequence-to-sequence (character to spectrogram) model. The encoder, which is a full-convolutional encoder with convolution blocks, is responsible for receiving the written words and transforming them into an internal learned representation. The decoder, which is a fully-convolutional causal decoder with an attention mechanism, is responsible for decoding this learned representation into a low-dimensional audio representation (mel-scale spectrograms). These audio representations are sent to converters so they can be transformed into audio using traditional (Griffin-Lim, WORLD) or machine-learning (WaveNet) methods. The results were impressive at the time of publication. The Deep Voice 3 training time outperformed other TTS systems showing to be 10 times faster than the latest technologies (Tacotron). The authors claim that the system is able to process more than 10 million TTS requests a day with just one server with a single GPU. Moreover, Deep Voice 3 converges with significantly fewer interactions than other TTS systems. Also, with the improvements in the attention-based convolution blocks, the authors were able to reduce the common errors presented by attention-based networks, meaning that the output phrase had less mistakes (repetitions, mispronunciations, etc.) than previous systems. The naturalness levels of the audio generated by DeepVoice 3 is also leading the rank in comparison with previous researches, showing that even with a fast algorithm it is possible to achieve human-like voice audio.

2.4.4 Speech-To-Speech synthesis

With the evolution of machine-learning based TTS systems, the academic community started to research ways of creating a speech-to-speech synthesis model. The idea to have as input an utterance spoken by person A, and having as output an utterance with the same content (words) but in the voice of person B. Although there are legitimate uses for this technology, such as the creation of personalized voices for commercial applications, this could be used for malicious reasons, i.e. impersonating someone.

Early in 2018, researchers from Baidu Labs started to investigate ways of performing voice cloning [4]. Moreover, their objective was to generate an unseen speaker voice using just few samples. The overall idea is to train an encoder-decoder model capable of listening to someone's voice (decoding) and reproducing the same words but in someone else's voice (encoding). Multi-speaker generative models have already been explored in the past (as seen on Deep Voice and other projects), however the key of this project is an audio-to-audio system capable of cloning a voice with just a few examples (in contrast to previous work that required more than 2 hours of recordings to train a model). The researchers use an encoder-decoder architecture similar to the one used on the project Deep Voice 3: an attention-based convolution deep neural network. However, instead of text-to-speech (as in the Deep Voice 3 project), they use a speech-to-speech model that has audio as input and output. The researchers propose two approaches for cloning voices with few samples: speaker adaptation and speaker encoding. The first one is basically fine-tuning a trained multi-speaker model. This fine tuning can be done through modifying the speaker embedding or through re-training the generative model with the few samples provided. The other approach, speaker encoding, consists of re-training the encoding model from scratch to directly infer a speaker embedding from the cloning audio, which is used in the generative model. The authors compare the results from both approaches and show that the speaker adaptation produces better similarity (between original audio and generated audio) and better naturalness of speech. However, speaker embedding requires less time and uses less memory during the inference process, meaning that it may be more suitable in cases where resources such as CPU and memory are a bottleneck. The authors also published[15] audio generated by the proposed cloning method and the results show a cloned voice very similar to the original voice.

2.4.5 Commercial Speech Synthesis Tools

Even before the first papers published about machine-learning speech synthesis, several companies were already developing in-house TTS systems for their own use.

[15] https://audiodemos.github.io/

Although those companies do not release their source code nor their methodology, one can use APIs to access the TTS systems and generate synthetic speech. It is important to note that some of the commercial systems can generate speech audio that sounds more realistic than the open source counterparts discussed so far. For this reason, we extend our research to include some of the main commercial systems that can be accessed by anyone to generate synthetic speech. Also, some of the systems have the capability of being re-trained with speech samples, meaning that using those commercial systems we could generate synthetic speech to impersonate a target victim.

Google utilizes a combination of machine learning techniques (such as WaveNet) and traditional speech synthesis methods to provide a TTS service[16] that can be used by third-part applications through APIs. This service not only accepts text as input, but also SSML[17] (Speech Synthesis Markup Language), which enables the user to describe how a phrase should be spoken, including the duration of pauses and the intonation in specific words.

Baidu, the largest Chinese internet-related services and products provider, has their own TTS system[18] that can be accessed publicly. Although Baidu released to the public few research papers detailing the use of machine learning for TTS

[16] https://cloud.google.com/text-to-speech/

[17] https://www.w3.org/TR/speech-synthesis11/

[18] https://cloud.baidu.com/product/speech/tts

(Deep Voice 1, 2 and 3), their commercial system (which is sold as a service) has higher performance than the open source codes released on the internet. For this reason, in this research we utilized not only the open source Deep Voice code, but also audio generated by the commercial version of the system.

Amazon provides among its services the Amazon Polly[19]. Amazon Polly is a text-to-speech service in which a user pays a subscription and is able to synthesize speech given a phrase. The service supports SSML and contains voices in more than 20 languages, from both male and female speakers.

Microsoft recently released a new product called Microsoft Text To Speech[20]. Given an input phrase, this product is able to generate utterances in more than 16 different voices. Moreover, Microsoft TTS allows any person to upload voice samples (in conjunction with the dataset of spoken phrases) so their machine learning model can be re-trained to output a desired voice. This allows any person to create a model of their voice that can be saved and used later to generate utterances.

2.5 Synthetic Media Detection

Considering the advancements in synthetic media generation, the academic community started to study methods to detect computer generated media. Those

[19] https://aws.amazon.com/polly/

[20] https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech/

techniques are mostly based in machine-learning algorithms, which are able to detect patterns on the synthesized audio. Although synthetic video and audio can be useful in commercial applications, the main concern of the research community is that such technologies can be used for malicious reasons. For example, a malicious actor could generate a synthetic video of a person to denigrate the image of the victim[21]. In this section, we analyze efforts in detecting synthetic content in two media outlets: images and audio.

2.5.1 Fake Images Detection

With the advancements of image editing and image generation using machine learning, several malicious actors are using this technology to create public misinformation. The concern around false content on the internet is increasing and will worsen in the future due to advancements in technology.

Based on this concern, researchers from University Frederico II (Italy) explored the use of machine learning techniques to detect fake images [29]. In this publication, the authors train a neural network to detect GAN-generated images posted in social networks. The proposed methodology presents interesting results, achieving an average of 89.55% accuracy across a large dataset containing images generated by the most famous image-to-image translation neural networks. The methodology

[21] https://www.youtube.com/watch?v=cQ54GDm1eL0

presented by the authors is fairly intuitive. First, the authors identify the most famous image-to-image translation neural networks. Then, they create thousands of images with those neural networks, which are then put into a dataset in conjunction with real images. This dataset is used to train a variety of neural network architectures, so then the authors can analyze which architecture presents better results. After finding the architecture with best results, the authors test it against a testing dataset, containing both uncompressed and compressed images. During the experiment, the researchers found out that more complex neural network architectures, such as InceptionNet v3 and XceptionNet often produce the best accuracy for both uncompressed and compressed data. The authors observed that after compression (using the same image compression method that Twitter uses), several characteristics of machine-learning-generated images are lost, meaning that the overall accuracy is decreased when the image is compressed. However, as mentioned, more complex neural networks still perform well even with compressed data. The research is key to the community because it brings an automated way of recognizing GAN-generated images. This methodology opens the possibility of later being integrated into the social network algorithms to flag to the user if an image in their timeline is fake. The results presented show that compression may be an issue when classifying the images. However, this issue can be minimized with the use of very deep neural networks, such as XceptionNet.

Following the research in detecting GAN-generated images in social networks, several other authors created machine-learning-based methodologies for identifying fake content. However, as with any complex machine learning methodology, to improve the results it is necessary to have a large dataset containing fake images for training. Based on this need, researchers from Germany and Italy created a dataset containing more than half a million edited images using state-of-the-art image generation technology [39]. With this database, the authors also trained neural networks to recognize fake content in images and videos. These neural networks achieved better performance than humans, showing how the technology can help in this problem. The dataset, called FaceForensics, is based on more than 1000 videos, mainly from the YouTube platform. Those videos were split into short clips, which were then analyzed by a face-detection technology. The clips that didn't contain a face were discarded, while the ones containing a human face were kept. With those clips, the authors then used two state-of-the-art technologies to forge the video clips: Source-to-Target Reenactment, which is when a source face is applied to a target face; and Self-Reenactment, which is when the original face is kept intact and only the facial expressions changes (which can be used for lip-syncing). The results of both forgeries were then combined into the FaceForensics dataset. With the dataset in hand, the authors performed some experiments with it. First, the authors created a classifier that is capable of identifying if a video frame was forged or not. This was done by training several neural network architectures with forged and non-forged images. The authors also consider factors such as compression during testing, since most of the fake videos are uploaded into social media platforms, which compress the media. After the experiments, the authors conclude that compression reduces the accuracy of the majority of architectures. However, XceptionNet performs well in both compressed and non-compressed media. The second experiment that the authors perform with the dataset is the forgery segmentation task. In this case, the researchers are interested in identifying which regions (or pixels) of an image were forged. Although this task is more challenging and presented poorer results than the classification task, XceptionNet still was the best performing architecture. The third experiment performed by the authors was the refinement task, which is using the classification neural network model to improve the current image-generation algorithms. Since the model is capable of identifying the fake frames, this can be used to improve the generated frames. The results for this task are interesting: although the images were perceptually improved (which made humans more prone to mistakes when classifying between fake/non-fake), for the XceptionNet, there was almost no impact. This shows that the perceptual image (what a person sees) is not directly impacting what the XceptionNet had learned, which means that the proposed classifier was able to learn features that humans do not notice.

2.5.2 Synthetic Speech Datasets

With the increase in Automatic Speaker Verification (ASV) solutions, the research community started to develop interest in ways of detecting spoofing attacks against those systems. A replay spoofing attack consists in recording someones' voice and replaying it in an attempt to fool the ASV system and gain access to a system. Motivated by this threat, researchers from across the globe created a dataset containing real voices and spoofed voices [21]. With this dataset in mind, the researchers created a challenge, called ASVSpoof Challenge, so the research community could study and propose methodologies that solved the ASV replay spoofing attack. One of the most cited versions of this dataset is the ASVSpoof2015 dataset, which contains not only spoofed utterances, but also computed-generated speech. The synthetic utterances were generated using traditional text-to-speech systems and voice-conversion systems, which lack in naturalness of speech. The ASVSpoof2015 dataset does not include the latest deep-learning-based synthetic speech systems. The key contribution of this challenge is that it sparked parallel research across the globe in such attacks. More than 49 papers were submitted proposing a solution for the ASVspoof Challenge, some of them achieving error rates (ERR) as low as 6.73%.

In a paper published in early 2016, researchers discuss methodologies for identifying spoofing attacks using automated solutions [55]. In their study, they analyze the effectiveness of 5 text-to-speech systems (TTS) as well as 8 voice-conversion (VC) systems against three automatic speaker-verification (ASV) systems. Their conclusion is that the ASV systems are vulnerable to those spoofing attacks. However adding their proposed spoofing detection system can lower the false-acceptance rates to less than 1%. The main difference between the study in [55] and previous studies is that the authors compare a variety of spoofing systems against a variety of ASV systems, making the study more broad and generic. Also, they focus their research on speech synthesis and voice conversion spoof attacks, in contrast to previous authors that focused on speech replay and impersonation. Although this study considers a large variety of attacks, none of the 13 spoofing methodologies uses the current state-of-art technologies, such as deep neural networks and Wavenet. Instead, the spoofing attacks used in this research mainly uses HMM and GMM based algorithms. Nonetheless, the study brings several interesting findings to the research community. Apart from the key contribution of developing an anti-spoofing system, the authors also published to the research community a dataset, called Spoof and Anti-Spoof (SAS) dataset, that include thousands of utterances from real speakers and synthetic speech. In addition to the main contributions, several interesting side-findings were presented in the publication. The authors observed that the performance of a speech-synthesis algorithm is higher in male voices. The authors speculate that it is harder to model a female voice due to the higher variability of female speech. Also, the authors note that using higher sample rates increases the performance of spoofing attacks. To prove the efficiency of their proposed anti-spoofing methodology, the authors compare the performance of their system to human performance. For that, the researchers recruited 100 native English listeners for a variety of tasks, including the classification of an utterance into synthetic or real. Although the human performance was fairly good (with an average error rate of 7%), the proposed system outperformed humans achieving less than 1% error rate. However, it is important to note that the TTS and VC systems utilized in this research are outdated compared to the current state-of-art speech-synthesis systems.

2.5.3 Synthetic Speech Detection

In mid 2017, a paper was published proposing a set of short-term spectral features that can drastically improve the accuracy in the synthetic speech detection [33]. The authors provide a thorough analysis of the differences between synthetic speech and real speech. The researchers start by extracting features from the audio using several different methods, such as: Mel-Frequency Cepstral Coefficients (MFCC), Modified Group Delay Functions (MGDF) and Cosine-Normalized Phase features. With those representations, the authors compare general features from both synthetic and real speech and identify interesting patterns, such as the fact that lower frequencies (<1kHz) and high frequencies (>7kHz) are the most useful frequencies for discrimination between synthetic and real. Based on those findings, the authors suggest improvements to previous models. For example, since high frequencies are important for classification and since MFCCs lose precision in higher frequencies, the authors propose using inverted MFCCs (IMFCCs), so that higher frequencies are in the beginning of the spectrum and have better representation. The combination of this and other improvements (such as the use of dynamic coefficients and frequency warping) resulted in 100% accuracy in almost all synthetic speech generation algorithms in the ASVSpeech2015 dataset. However, it is important to note that the dataset did not contain the latest state-of-the-art speech synthesis algorithms.

A good portion of the speech synthesis detection studies focuses on extracting frequency information and using this information to train a classifier. This kind of approach usually assumes frame-by-frame independence and does not learn long-term temporal information. However, a study published in 2013 shows that having temporal data increases the performance of synthetic speech classifiers [56]. In the publication, researchers use modulation features deviated from the magnitude/phase spectrum in conjunction with the traditional frequency analysis (such as MFCCs) to achieve better classification results. According to the published results, using temporal features with the traditional MFCC analysis results in 7.17% error (down from 10.98% using only MFCCs). Similarly, using MGDCC with temporal features results in a 0.89% error (down from 1.25% using only MGDCC). This shows that for the speech synthesis classification problem the variation of the frequencies over time contains valuable information for the identification of synthetic speech.

2.5.4 Synthetic Speech Detection Using Deep Neural Networks

With the increase in the popularity of Deep Neural Networks (DNN) solutions, a paper was published in late 2017 regarding the use of DNNs for speech spoofing detection [58]. The main idea is to use DNNs to extract dynamic acoustic features and classify an utterance as real or spoofed. The research shows that this proposed methodology overperforms the traditional static feature analysis with GMMs classifiers. Previous studies [40, 59] show that dynamic acoustic features (such as dynamic filter banks, dynamic MFCCs and dynamic linear prediction cepstral coefficients) are better candidates for spoofing detection than traditional static features (such as magnitude-based features and cosine normalized phase features). Based on that and the fact that DNNs are well known for their capabilities of extracting dynamic features, the researchers decided to implement a 5-layer deep neural network in conjunction with 5 different dynamic filter-bank-based scoring methods to perform classification on the AVSpeech2015 dataset. Although this dataset does not cover the latest deep-learning based TTS systems (such as DeepVoice3), it is a good starting point to train a DNN to detect spoofed speech. The results of the experiment show that DNNs with dynamic features present better performance than the previous methodologies using static features and GMM models. Also, as a side finding, the researchers state that it was observed during the dynamic-feature extraction that higher frequency contributes more for classification than lower frequencies. Another interesting finding is that using a human log-likelihood scoring method presents better performance for spoofing detection than using a traditional log-likelihood ratio scoring method.

Researchers from Japan published a study where they used the ASV system developed in [55] to improve speech synthesis [41]. Inspired by adversarial neural networks, the idea is to use the output of the ASV classifier in the loss function during the speech generator training. This approach increased considerably the naturalness of speech according to a user study performed by the researchers. In their publication, the authors compared real speech against synthesized speech and noted that the machine-generated audio is prone to over-smoothing, which leads to a less natural utterance. To minimize this effect, during the training phase of the 5-layer deep neural network, the authors then use the ASV system output as part of the loss that is minimized during training. This leads to a synthesizer that is able to not only generate speech, but also speech that is close to real speech. As objective evaluation, the trained model is then used against the original ASV algorithm. In this case, a significant increase in the spoofing success is observed when compared to a model trained without the adversarial loss. The authors performed human evaluation and concluded that the proposed method is preferred by the listeners in 60% of the cases, meaning that it brings a slight improvement in the audible quality of the utterance.

In mid-2017, researchers published a paper regarding their investigation of deep-learning frameworks for speaker verification anti-spoofing [62]. In their research, the authors propose the use of CNNs in conjunction to RNNs to identify synthetic speech. Using as baseline the ASVSpoof2015 dataset, the proposed methodology presents the state-of-the-art performance for an end-to-end single system. In the research, the authors explore three different input types (i.e. STFT Spectrograms, TEO-CB-Auto-Env features and Minimum Variance Distortionless Response (PMVDR) features) as well as four deep neural networks structures (i.e. 4-layer DNN, CNN, RNN and CNN+RNN). The experiments show that although using TEO-CB-Auto-Env features and PMVDR features are effective for synthetic speech classification, the best results are found by using STFT Spectrograms as input with a classifier that uses CNN and RNN. The idea of using both architectures in one system is that CNNs are good feature extractors while RNNs are good at recognizing long-term dependencies in the time domain. Using both together (in a single system that can be trained all together) provides high accuracy even in unseen spoofed voices. As side finding, the researchers also show that the size of the utterance matters in classification: in small utterances (2.5s or less), the accuracy is considerably reduced. The optimum audio length hovers around 4 seconds. Also, the researchers show that the performance of spectrograms with CNNs (without the RNN part) is also very satisfactory, having the best performance in the majority of the 10 synthesized voices.

Xiaohai Tian and Xiong Xiao published a paper regarding their work on spoofed speech detection using temporal convolutional networks [51]. Their idea is to use a single convolutional neural network to classify an utterance instead of using handcrafted feature extractors with traditional machine learning approaches. The proposed architecture is tested against the ASVspoof2015 dataset and shows a relevant improvement, especially in unseen spoofing attacks and in temporal-based speech synthesizers. First, the researchers extract temporal and spectral features by generating spectrograms from the input audio. Then, the spectrograms are used as input on the proposed CNN. The CNN contains only three layers: a convolution layer, a max-pooling layer and a feed forward layer. One of the key points of the project is using temporal convolutional neural networks, or in other words, CNNs that have the convolutional filters covering the whole frequency range and few frames of audio. For example, if the spectrogram contains 256 frequency bins, then the height of the convolutional filter is also 256. Also, in their research, the convolutional filter length was 11, meaning that 11 frames of the audio were considered in each convolution. In their experiments, the researchers analyze the performance of the proposed temporal CNN against unseen spoofing attacks and show that the CNN performs better than the traditional approaches.

In late 2017 a paper was published exploring the use of CNNs for end-to-end speech spoofing detection [30]. Although the proposed approach is not new and uses the outdated ASVspoof2015 dataset, the authors present an interesting analysis of what features are being learnt by the model. In this analysis, the authors show that the proposed architecture is mainly learning discriminative information from the lower and higher frequencies, which matches with previous studies that used manual feature extraction with traditional machine-learning algorithms. This shows that the DNN is able to extract frequency features right from the raw audio and that the DNN learns from the same spectrum regions than the traditional approach, with a similar or higher accuracy.

Following the research trend in using DNNs for end-to-end speaker spoofing detection, a paper was published in late 2018 regarding the use of raw audio as input in DNNs for the speech spoofing classification problem [12]. In their work, the authors investigate a variety of DNNs, CNNs and RNNs with raw audio as input and propose a combined model (CNN+LSTM) that provides the highest accuracy on the ASVspoof2015 dataset at the time of publication. The authors start the discussion showing the performance of traditional approaches to the ASV problem, including the use of MFCC and CQCC with traditional machine learning algorithms. Then, the authors show the individual performances of some deep learning approaches, such as DNNs, CNNs and RNNs. The individual performance is relatively good but does not surpass the traditional approach for the problem. The authors argue that CNNs are good at extracting frequency features and that RNNs are good at extracting timing features, so it is a good idea to combine both approaches in one single model, in which we have a CNN (to extract frequency features) connected to a RNN (to extract timing features), ending in a fully connected layer and a classification layer. This approach results in the highest accuracy for the ASVspoof2015 dataset at the time of publication. Apart from the proposed architecture, the authors also discuss a series of interesting side findings. For example, they discuss the importance in the length of the sampled audio and conclude that, for end-to-end RNN architectures, it is better to use small frame sizes and large sequence lengths. They also conclude that the amount of training data is crucial for avoiding overfitting, and that the ASVspoof2015 dataset may not contain enough data points.

2.6 Background Analysis

From the literature reviewed in our research, several needs were identified:

- Need for a larger dataset: As identified in [12], the main synthetic speech datasets do not contain enough data to train complex deep learning algorithms. This creates a need for a larger dataset that contains enough data to train the latest DNNs.
- Need to address latest speech synthesizers: The most utilized dataset for synthetic speech classification is the ASVSpoof2015 dataset, which was released more than four years ago. In this meantime, several high-end speech synthesizers were released, such as the DeepVoice 3 [35], Microsoft TTS and Amazon Polly. This creates a need for a dataset containing the latest TTS systems as well as real utterances.
- Need to understand the human performance against latest speech synthesizers: To our knowledge, the last major human study on synthetic speech perception was published in 2016 [55]. Since then, several high-end TTS systems were released, creating a need for a new user study.
- Need to propose an updated model for synthetic speech detection: The majority of the papers published in synthetic speech detection focus on an outdated

dataset (ASVSpeech2015). With the creation of an updated dataset it is possible to train and analyze the performance of the latest DNN models on the synthetic speech detection problem.

Our research aims to contribute to the research community by introducing a large dataset, presented in Chapter 3, containing utterances from real humans and the latest speech synthesizers. This dataset can be used by researchers to train complex machine learning algorithms that classify the veracity of an utterance. We utilize the introduced dataset to perform a user study, presented in Section 4.2, to better understand the human perception of synthetic speech. In Chapter 4 we utilize the created dataset to train some of the latest neural network models to demonstrate the learnability of the problem and propose a model capable of distinguishing between real and synthetic utterances.

3 FoR Dataset

An important component for the development of a Synthetic Speech Detection system is a dataset containing both synthetic speech and real speech. Although in the past several such datasets were published [21, 54, 58], the vast majority of them do not contain utterances from the latest deep-learning-based speech synthesis algorithms. Another limitation of previous publications is that the number of utterances was not enough for training complex neural network models [12]. Moreover, the majority of the published datasets focus on the detection of spoofed utterances for automatic speaker verification systems. In this research, we introduce the *Fake or Real (FoR)* dataset, which is composed by more than 84,000 synthetic utterances as well as more than 111,000 real utterances (from a large variety of individuals).

The main difference between the FoR dataset and previous works is that our dataset contains utterances from the state-of-the-art speech synthesis algorithms, which are utterances with naturalness similar to real human speech. Also, our dataset contains a large number of data points and, according to our experiments, it is enough to train complex models, such as InceptionV3, without overfitting.

The FoR dataset is under GNU GPLv3 license and is publicly available to the community[22].

In this chapter, we present how the data was collected in Section 3.1 and Section 3.2. Then, we present the various dataset versions in Section 3.3. Last, we discuss the dataset division in Section 3.4.

3.1 Collecting Synthetic Speech

The first half of the dataset is the synthetic utterances. As previously discussed, the use of deep learning for speech generation has increased in the past few years. With that in mind, an extensive research was conducted to identify the latest methodologies in speech synthesis, both Text-To-Speech (TTS) systems and Voice Conversion (VC) systems. For this research, several open source and commercial systems were identified, such as:

- DeepVoice 3 [35]: An open-source system developed by Baidu that is capable of learning speech synthesis from a dataset of phrases and utterances. This system is considered a groundbreaking study since it is an end-to-end TTS system that is able to train without any human tuning.

[22] http://bil.eecs.yorku.ca

- Neural Voice Cloning [4]: An end-to-end solution for voice cloning in which the input is real speech and the output is synthesized speech in a different voice uttering the same sentence.

- Baidu TTS[23]: Developed by the same team that published DeepVoice 3, Baidu TTS is the commercial version of their publication.

- Microsoft Azure TTS[24]: A cloud solution for TTS that is based on deep-learning techniques with supervision and tuning of speech specialists. This combination of machine learning with human supervision resulted in synthesized speech very close to real speech.

- Amazon AWS Polly[25]: Similarly to Microsoft TTS, Amazon AWS Polly combines the deep-learning techniques with human supervision to provide a high-quality TTS system.

- Google Cloud TTS[26]: Although Google already had developed a TTS system in the past, they recently published a new cloud TTS service that uses deep learning and traditional techniques to generate speech. Moreover, in one of its versions, the Google Cloud TTS incorporates the Wavenet architecture into

[23] https://www.home-assistant.io/components/tts.baidu/

[24] https://azure.microsoft.com/en-ca/services/cognitive-services/text-to-speech/

[25] https://aws.amazon.com/polly/

[26] https://cloud.google.com/text-to-speech/

their TTS system, achieving very high naturalness in the generated speech.

With the scope of TTS systems defined, the next step is to generate a list of phrases that will be used to generate utterances.

3.1.1 Phrases

All the TTS algorithms have as input a phrase and as output audio containing the generated utterance. One of the main concerns when creating a dataset is to have a high variety of data points to ensure that the underlying distribution is well represented in the dataset. With that in mind, it is important to choose a high variety of phrases to be used as input in the TTS systems.

In our research, we utilized a phrase dataset[27] that is commonly used in natural language translation. This dataset is open to the public and contains over 150,000 English phrases and their French translation. Since our work focuses on the English language, the French part of the dataset was discarded, leaving us with a dataset of English phrases with a high variety of grammatical structures (passive/active phrases, simple/complex phrases, short/long phrases, affirmative/question phrases, etc.). The 150,000 phrases were filtered to remove duplicate phrases as well as phrases surpassing 30 words, resulting in a final phrase dataset containing 105,000 phrases.

[27] https://www.kaggle.com/percevalw/englishfrench-translations/kernels

The resulting phrase dataset was then randomly divided into 40 phrase buckets, containing 2645 phrases each. Each phrase bucket was used by only one TTS voice. This ensures that there are no repeated utterances in the dataset, minimizing the risk of the model learning specific words/phrases instead of a generalized model to differentiate between real and synthetic speech. With the phrase-buckets ready, we then proceed to extract utterances from each TTS system.

3.1.2 DeepVoice 3

The DeepVoice 3 system is an end-to-end TTS solution developed by the Baidu Labs[35]. This model is capable of generating an audio representation (spectrogram) given a phrase as input. This representation can then be transformed into an utterance using a spectrogram to audio function or a Wavenet architecture.

For this experiment we obtained an implementation of the DeepVoice 3 system[28] which was then trained using the utterances from the LJSpeech speech dataset (see Section 3.2.1). The reason for using LJSpeech for both generating synthetic utterances as well as for the real utterances part of the dataset is to ensure that the model is learning real characteristics of synthetic and real speech, since using the same voice for both classes minimizes the chances of the model classifying based on voice properties (pitch, intensity, etc.).

[28] https://github.com/r9y9/deepvoice3_pytorch

After the model was trained on the LJSpeech dataset, one phrase-bucket was used to generate utterances, resulting in a total of 2645 speech files.

3.1.3 Google TTS

The original Google TTS is one of the most known text-to-speech systems. It is present in several Google products, such as Google Translate[29] and Google Home. Although this system does not utilize the latest deep learning techniques, it is a popular TTS system thus it is included in our research.

As Google TTS is a proprietary system, it is not possible to get access to its source code. However, it is possible to use API calls to extract audio from the system. By creating a script that reads a list of phrases, sends API requests and saves the returning result, we were able to generate 2645 utterances from this system.

3.1.4 Google Cloud TTS

With the increase in the popularity on cloud TTS services, Google created its own cloud TTS service[30]. This service uses the latest deep learning techniques in conjunction to manual tuning to provide a cloud TTS service. The resulting

[29] https://translate.google.com/

[30] https://cloud.google.com/text-to-speech/

utterances have a high naturalness, approaching human-like speech. Moreover, the system also accepts a SSML[31] (Speech Synthesis Markup Language) file as input, meaning that one can define several speech variables, such as emphasis and break times.

To extract utterances from the cloud API, first it was required to create a Google Cloud account. Then, it was required to create a project key so the API can be accessed. After having the keys created, it was required to install the Google SDK and create a script that reads a file containing phrases and retrieves the utterances from the Google Cloud. With that done, 5290 utterances were extracted in two different voices (2645 utterances per voice).

3.1.5 Google Wavenet TTS

Similarly to the Google Cloud TTS, Google released a premium version of its TTS system. This premium version, called Google Cloud Text-to-Speech with Wavenet[32], uses a mixed model using deep learning techniques in conjunction to a Wavenet model to generate the utterances. The system also accepts as input SSML files, meaning that precise speech can be generated. This improved model is able to generate utterances with very high naturalness, where the synthesized speech is

[31] https://www.w3.org/TR/speech-synthesis11/

[32] https://cloud.google.com/text-to-speech/

almost indistinguishable from real speech.

The process of utterance extraction on the Google Cloud TTS Wavenet is very similar to the non-premium Google Cloud TTS: First it is necessary to obtain API keys, then install the Google SDK and finally use a script to iterate over a text file and obtain the utterances through the API. With that done, 5290 utterances were extracted in two different voices (2645 utterances per voice).

3.1.6 Microsoft TTS

Similar to Amazon and Google, Microsoft recently released its text-to-speech solution. Called Microsoft Azure Text-To-Speech[33], this service is capable of generating utterances from input phrases. The interesting part about Microsoft TTS is that it provides 16 voices just for the English language (in a variety of accents). Moreover, the Microsoft TTS system allows any person to upload samples of their voice so the model can learn and reproduce their voice. This allows a much more customized experience for customers, since one can have their own voice being spoken in a system.

The process to extract utterances from the Microsoft TTS system is fairly simple: First, you create an API key to access the service. Then, using an HTTP request, one can send a phrase to the system that answers with an MP3 file. To au-

[33] https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech/

tomate the collection, a script was created to read an input file (containing phrases) and submit POST requests to the Microsoft TTS server, which then returns an MP3 file. A total of 42,320 utterances were synthesized from the Microsoft TTS system, 2645 utterances for each one of the 16 voices.

3.1.7 Amazon Polly

Amazon Polly[34] is one of the most known TTS systems. Similar to other commercial solutions, this service allows the synthesis of utterances using the latest deep learning techniques. At the moment this research was conducted, 8 English voices were available in a variety of accents (American English, British English, Australian English and Indian English). One of the main advantages of Amazon Polly is that it is able to synthesize natural speech with high pronunciation accuracy (including abbreviations, acronym expansions, date/time interpretations, and even homograph disambiguation).

To synthesize utterances using Amazon Polly first it is necessary to create an Amazon AWS account. With the account created, it is possible to generate an API key that enables the access to the Polly TTS system. A script was created to read phrases from a text file and interact with the Polly API to retrieve the utterances. A total of 21160 utterances were extracted from this system, 2645 utterances for

[34] https://aws.amazon.com/polly/

each one of the 8 speakers.

3.1.8 Baidu TTS

Although Baidu published research papers detailing the use of machine learning for TTS (DeepVoice 1 [5], DeepVoice 2 [3] and DeepVoice 3[35]), they also developed a commercial system (which is sold as a service) that has higher performance than the open source codes released on the internet. This service, called Baidu Cloud TTS[35] is able to generate utterances given text as input. Although the system is mainly trained for the Chinese language, the service is also offered in English (however, with lower naturalness compared to the Chinese voice).

Baidu offers an interface through their speech synthesis app through which it is possible to submit phrases and obtain audio files. To automate the process, a script was created to generate HTTP requests to the app and receive the resulting MP3 file. As there is only one English voice available, 2645 utterances were extracted from this system.

3.2 Collecting Real Speech

The second half of the dataset consists of real utterances, i.e. speech recordings from humans. The process of collecting real utterances is a complex task since we need

[35] https://cloud.baidu.com/product/speech/tts

to ensure that the collection methods are not impacting on the training results. For example, it is necessary to ensure that the utterances are recorded using a variety of microphones, otherwise the machine learning algorithm may learn to classify based on features specific to one recording device, instead of learning the real differences between a synthetic utterance and a real utterance. Similarly, for the same reasons, it is required to have a large variety of voices (from all genders) as well as a good variety of accents.

The first step was to identify potential sources of real utterances. Two main source categories were identified: Open source datasets, which provide a large amount of pre-processed speech; and Internet recordings, in which we extract speech from online content, such as Youtube.

3.2.1 Open Source Datasets

Open source speech datasets are a quick way of obtaining real speech. Those datasets usually are already pre-processed and provide a clean recording. For this study, the following open source datasets were selected:

- Artict Dataset[36]: This dataset contains 1132 utterances spoken by 7 professional voice actors, resulting in a total of 7924 utterances. This dataset was chosen because it contains a good variety of accents as well as having utter-

[36] http://festvox.org/cmu_arctic/

ances from all genders. Also, as we have the same utterances being spoken by 7 different speakers, it increases the chances of the classifier learning a more generalized model for real/synthetic classification.

- LJSpeech Dataset[37]: The LJSpeech dataset contains 13,100 utterances from one female speaker. This dataset is a well known real-speech dataset used in several TTS publications, such as DeepVoice 3[35], and for this reason it was chosen for this research. Also, this dataset was used to train the DeepVoice 3 model, meaning that we have synthetic and real utterances from the same voice, increasing the chances of the classifier learning a more generalized model for real/synthetic classification.

- VoxForge Dataset[38]: VoxForge is an open source real-speech dataset in which any person can record and submit utterances to the project. This creates a dataset with a large variety of voices, recording devices and even audio quality. At the time of the collection, this dataset contained more than 86,000 utterances, from more than 1,200 persons using a large variety of recording devices. This dataset was chosen due to its large amount of different voices as well as the large variety of recording devices, which increases the chances of the classifier learning a more generalized model.

[37] https://keithito.com/LJ-Speech-Dataset/

[38] http://www.voxforge.org

3.2.2 Internet Recordings

Social media platforms, such as Youtube, can be an excellent source for speech data: they provide a high variability of voices as well as recording devices. However, audio from such sources usually contains a large amount of background noise and/or background music. To minimize the chances of background noise and poor recording quality, we selected a variety of educational videos as source of speech. Educational videos (such as TED talks, online courses and tutorials) are good candidates because they typically are recorded in a silent environment (using a high quality recording device) and typically contain only one speaker.

For this research, 140 videos (speakers) were selected. From those videos, the full audio was extracted and the SoX[39] tool was utilized to segment the audio where a silence of 2 seconds or more was detected. The purpose of segmenting the audio based on silence is to avoid broken utterances, where audio is cut while someone is speaking. This process resulted in a total of 3720 utterances from 140 speakers.

3.3 Dataset Versions

Since the dataset is used in a variety of machine learning models, it is important to pre-process the utterances in a way that eliminates bias. Based on the pre-

[39] http://sox.sourceforge.net/

processing applied, we identified and generated four different versions of the dataset for our study. In this section we describe each one of them.

3.3.1 Original Dataset (for-original)

The original dataset, named *for-original*, contains the files as collected from the speech sources, without any modification or class/gender balancing. A total of 195,541 utterances are present in this dataset version. The gender and class distributions can be found in Figure 3.1.

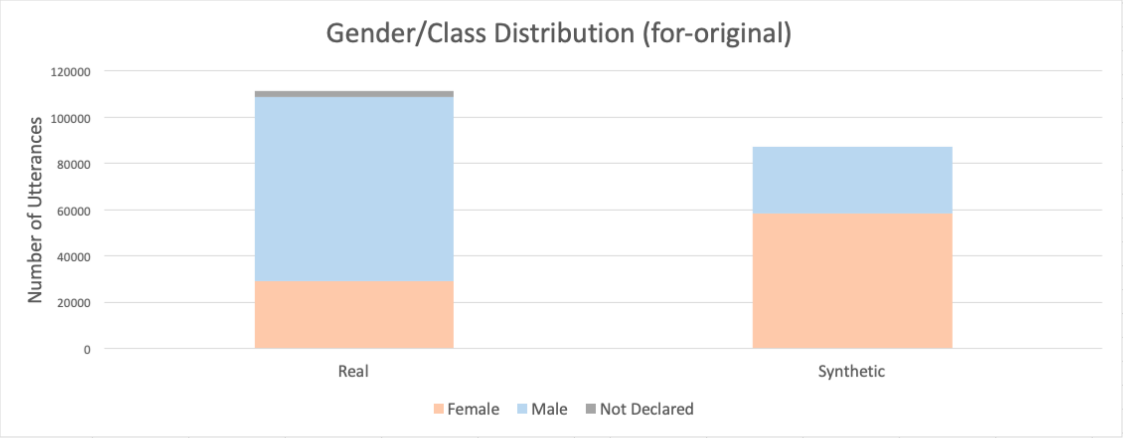


Figure 3.1: Gender Distribution - FoR Original

Although the data in this dataset version is heavily unbalanced (in terms of gender distribution and class distribution), it is being published so the research community can use the raw data with their own pre-processing techniques.

3.3.2 Normalized Dataset (for-norm)

As the dataset is composed by utterances from several audio sources, it is essential to normalize the data to eliminate any bias due to data collection. The normalized dataset, named *for-norm* contains the same files as the original dataset but with the audio converted to WAV, normalized to 0dBFS, downsampled to 16kHz sample rate, converted to mono and with silences removed from beginning and end of the utterances. The details of each of the normalization steps can be found on .

As this dataset version is used in some of the experiments, we also balanced the data to achieve even distribution between genders and classes (synthetic/real). The resulting distributions can be seen in Figure 3.2, where it is possible to note a more even picture in terms of class and gender. Due to the balancing process (which includes downsampling), the resulting dataset version contains a total of 69,400 utterances.

3.3.3 2-Second Dataset (for-2seconds)

After initial analysis on the original dataset, it was noted that the synthetic audio was considerably shorter than the real audio. While the synthetic audio was on average 2.35 seconds long (with a standard deviation of 0.83), the real utterances were on average 5.05 seconds long (with a standard deviation of 1.95). Figure

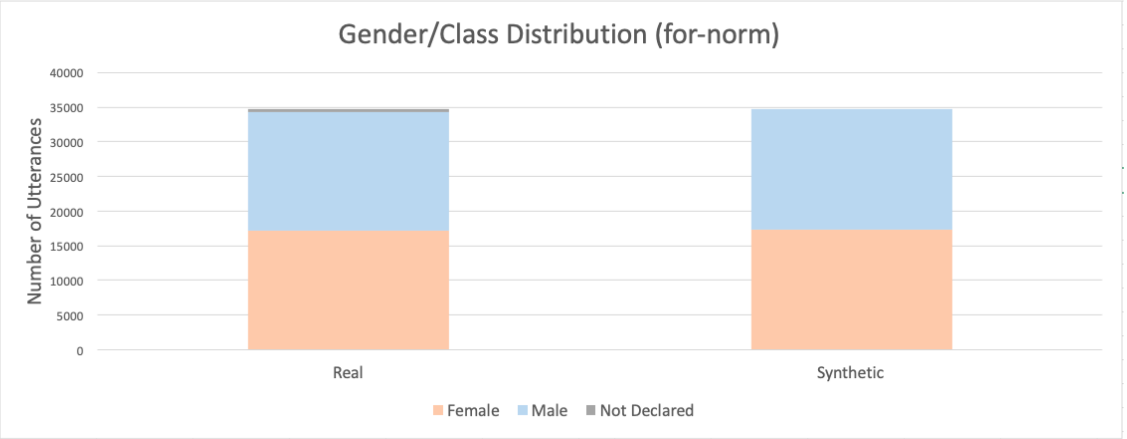


Figure 3.2: Gender Distribution - FoR Normalized

3.3 shows the audio length distribution for both real and synthetic audio. This significant length difference may affect the classification, since the neural network may learn to distinguish between real and synthetic based on the length of the audio.

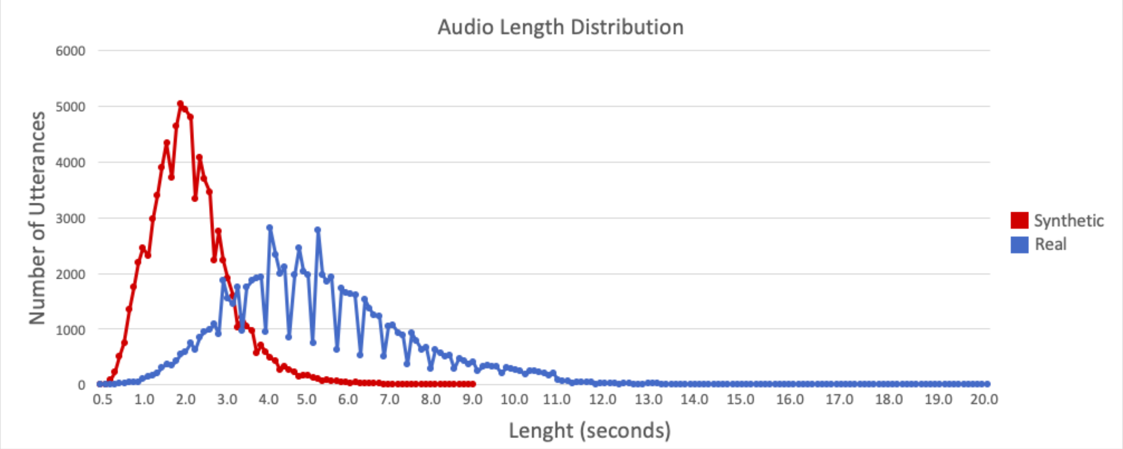


Figure 3.3: Audio Length Distribution

To solve this, we modify the normalized version of the dataset (for-norm) as follows. We start by discarding all files shorter than 2 seconds. Then, we truncate the remaining files at the two-second mark. The last step is to re-balance the dataset to achieve the same distribution as the for-norm version of the dataset. The resulting version of the dataset, named *for-2seconds*, contains a total of 17,870 utterances and it is the main version to be used in our experiments.

3.3.4 Re-recorded Dataset (for-rerecorded)

To simulate a real-world synthetic speech attack, it was decided to test the resiliency of the model against audio re-recording. The idea is that in a real-world scenario, a malicious person may generate/play the synthetic speech with one device (e.g. a computer) and record it using another device (e.g. smartphone). This is an example where the attacker is trying to impersonate someone via a communication channel (e.g. a phone call or a voice message).

To simulate this scenario, we played the utterances from the for-2seconds dataset using a regular computer speaker and recorded them using a non-professional microphone, simulating a casual attacker. The resulting version of the dataset, referred to as *for-rerecorded*, contains re-recorded utterances that simulate a real world attack. As the utterances were already recorded at 16kHz sample rate and the volume was constant during recording, there is no need for downsampling nor volume normalization.

To get a better understanding on which frequencies were most affected by the re-recording process, we used the Chi-Square technique (described in Section 4.3.1) to identify which frequencies are more distinct between original and re-recorded audio. The results of this analysis can be seen in Figure 3.4, in which it is possible to observe that higher frequencies in synthetic speech were more affected by re-recording. This reduces considerably the differences on high frequencies between real and synthetic speech.

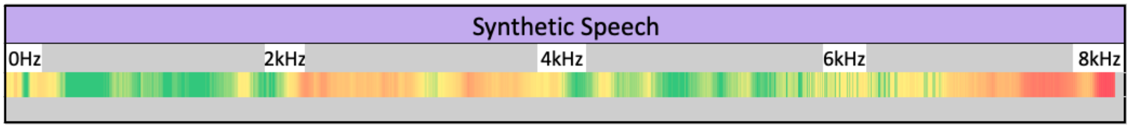


Figure 3.4: Chi-Squared Frequency Change Map - Synthetic Speech

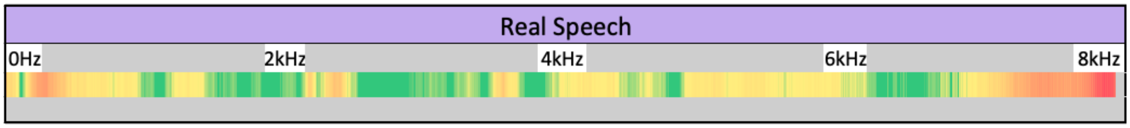


Figure 3.5: Chi-Squared Frequency Change Map - Real Speech

The same analysis was made for the differences between real speech from the original dataset and re-recorded dataset. The results are fairly similar, the higher frequencies are the most affected by the re-recording process. However, it was noted that the differences in the frequency bins are not as high as in synthetic audio, meaning that the re-recording process has less affect on real utterances. Figure 3.5 shows the most impacted frequencies in real speech.

3.4 Dataset Division

After the dataset is pre-processed, it is ready to be used to train the classifiers. As is common practice in machine-learning research, all the dataset versions were divided into:

- Training: Contains 77.73% of the dataset, utilized to train the machine learning models. Gender and class balanced.
- Validation: Contains 15.58% of the dataset, utilized to validate the accuracy of the machine learning models. Gender and class balanced. The validation utterances are unseen during the training phase.
- Generalization Testing: Contains 6.68% of the dataset. Contains only synthetic voices from one unseen algorithm (Google TTS Wavenet) and unseen real voices. Gender and class balanced. It is utilized to test if the trained model can generalize and detect unseen TTS algorithms and unseen real voices.

With the dataset versions created, processed and divided, they are ready to be used in our experiments.

4 Experiments

With the dataset completed, the next step consists in performing experiments to compare ways of detecting synthetic audio. We start this chapter by presenting the tools used in this research. Then, we present the performance of humans in the synthetic speech detection task. In Section 4.3 we present a series of experiments to analyze the accuracy of several detection methods against the for-2second dataset. Also, we present an analysis of a real-world attack scenario using the re-recorded dataset (for-rerecorded). For both dataset versions we also present the performance of the proposed models on a totally unseen TTS algorithm.

4.1 Tools

The main tool utilized for audio processing is the SoX[40] tool, which is a cross-platform command line utility for audio processing. The tool not only is capable of converting audio, but also applying effects, filters and generating spectrograms.

[40] http://sox.sourceforge.net/

Since it is a command line tool, it can be easily scripted to automate pre-processing in large datasets.

The Weka[41] tool is a software for data mining analysis well-known in the academic community. This tool provides a GUI interface for several data mining algorithms, making it easy to run experiments and compare results. This tool was utilized in the non deep learning experiments, such as the frequency analysis and audio feature analysis.

The deep learning experiments were performed using the TensorFlow[42] Python library, which is an open source library for high performance numerical computation. The advantages of using Tensorflow is the easy implementation of neural network models as well as the increased performance due to support for GPU processing.

To simplify the implementation of the most common deep learning models, the Keras[43] library was chosen. This library runs on top of the TensorFlow library and provides a high level API for neural networks. Keras allows easy and fast prototyping through its modularity. The majority of the well-known deep learning architectures are already implemented in it and ready to use.

[41] https://www.cs.waikato.ac.nz/ml/weka/

[42] https://www.tensorflow.org/

[43] https://keras.io/

4.2 Synthetic Speech Detection by Humans

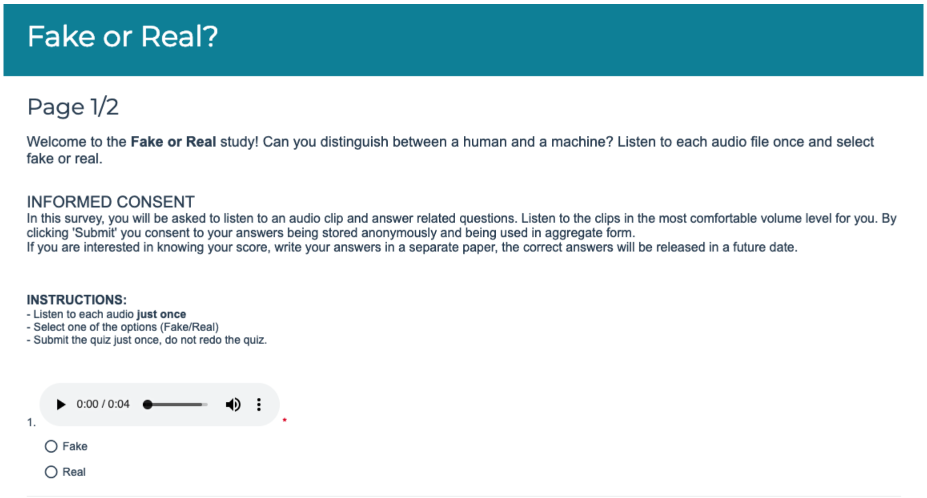
As one of our goals is to present a synthetic speech detection technique that performs better than humans, the first step is to analyze the human performance in this task. With the latest advancements in synthetic speech, the naturalness of computer-generated speech is getting closer to human speech. To evaluate the human perception of synthetic speech, we conducted a survey with 29 participants. Similar to a Turing test, the idea is playing utterances and asking people to judge if the speech was generated by a computer or not.

4.2.1 Preparation

Ten synthetic utterances and ten real utterances were selected from the original dataset (for-original). For the synthetic speech, we randomly selected one utterance from each speech source (eg. one from DeepVoice3, one from Amazon Polly, etc.). For real utterances, two audio files were randomly sampled from each real-audio source, resulting in 10 real utterances.

The online survey, named "Fake or Real?"[44], is seen in Figure 4.1, were it is possible to see the survey instructions and the first questions. This survey contains 20 questions, one for each utterance. Each question contains only two options: "Fake" (for synthetic speech) and "Real" (for human speech).

[44] https://ca.surveygizmo.com/s3/50043593/Fake-or-Real



Fake or Real?

Page 1/2

Welcome to the **Fake or Real** study! Can you distinguish between a human and a machine? Listen to each audio file once and select fake or real.

INFORMED CONSENT
In this survey, you will be asked to listen to an audio clip and answer related questions. Listen to the clips in the most comfortable volume level for you. By clicking 'Submit' you consent to your answers being stored anonymously and being used in aggregate form.
If you are interested in knowing your score, write your answers in a separate paper, the correct answers will be released in a future date.

INSTRUCTIONS:
- Listen to each audio **just once**
- Select one of the options (Fake/Real)
- Submit the quiz just once, do not redo the quiz.

1. ▶ 0:00 / 0:04 *

○ Fake

○ Real

Figure 4.1: “Fake or Real?” Survey

4.2.2 The Survey

The participants of the survey were asked to listen to each audio just once (to simulate a real environment where you hear the speech only once) and select the options “Fake” or “Real” according to their guess. To ensure privacy, no personal information was collected. The survey was answered by a total of 29 participants over a period of three weeks.

4.2.3 Results

After the period of three weeks, the results were gathered, compiled and analyzed. The average overall human accuracy is 64.83%, 60.34% for synthetic speech and 69.31% for real speech. Figure 4.2 shows the accuracy per question as well as the overall average accuracy, in which it is possible to note that some synthetic speech algorithms are detected easier than others. For example, question 2 (Traditional Google TTS voice) had 0% human error, while question 4 (Microsoft TTS) had 48.28% error.

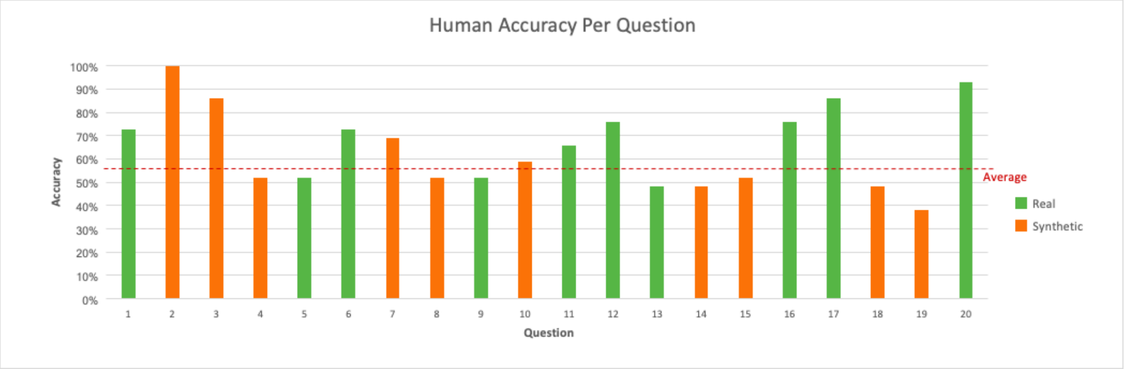


Figure 4.2: Survey Results - Accuracy Per Question

To better understand the variance in human accuracy, Figure 4.3 shows the accuracy distribution for this survey.

The numbers on this survey show that, on average, humans would miss 1 out of 3 synthetic utterances. If considering only high-performance algorithms (such as Microsoft TTS and Amazon Polly), humans mistake synthetic for real about half the time. This shows that the human perception of synthetic speech is vulnerable to the latest speech synthesizers and that an automated synthetic speech detector is needed.

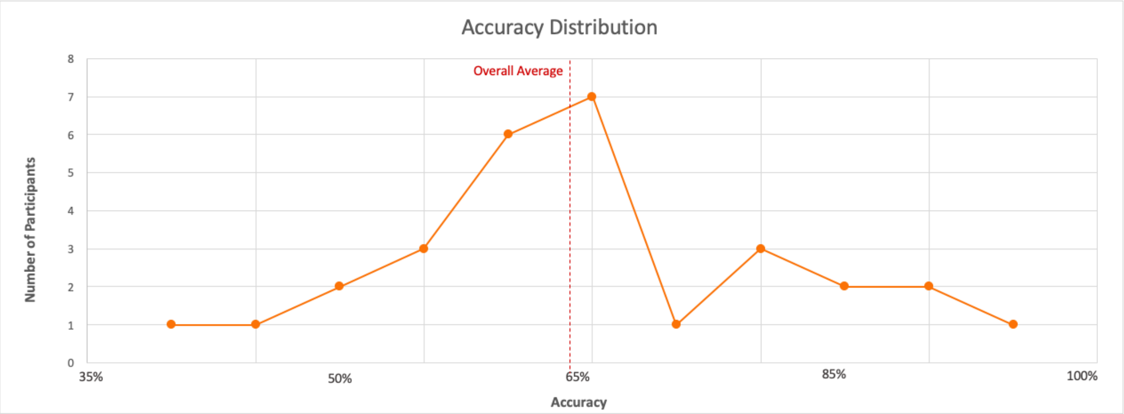


Figure 4.3: Survey Results - Accuracy Distribution

4.3 Synthetic Speech Detection

With the human performance evaluated, the next step is to propose and analyze the performance of a variety of synthetic speech detection methodologies. For this experiment we use the for-2seconds version of the dataset to understand the performance of both frequency analysis and deep learning approaches.

4.3.1 Frequency Analysis

The frequency analysis experiments consist in using traditional machine learning techniques to classify utterances on the FoR dataset. This creates a baseline to compare the traditional machine learning methods with the deep learning approaches. These experiments are based on frequency analysis, i.e. the extraction of a frequency representation and classification using machine learning techniques.

The process for this experiment consists in extracting an audio representation (such as an STFT matrix) for each audio file, averaging the representation over time to obtain a frequency-activation vector, inputing this vector into Weka with the appropriated classes (synthetic/real) and comparing the results of the main algorithms. Even though during the averaging process the temporal information is lost, this technique is still valid for several audio classification problems and is useful since the full audio representation is too complex for the traditional machine learning approaches. The following audio representations were chosen for this experiment:

- Fast Fourier Transform (FFT): It is a computation of the discrete Fourier transform to convert a signal in its original domain (in this case, time) to a representation in the frequency domain.

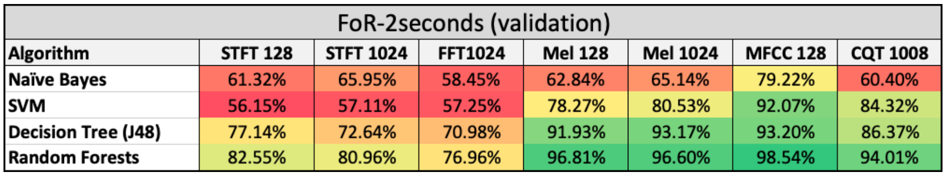
- Short-time Fourier Transform (STFT): It is a type of Fourier transform that calculates the frequency content of local sections of a signal. In our case, transforms an audio signal into a matrix representing the audio magnitude in terms of frequency and time.

- Mel-Spectrograms: Similar to STFT, it is a transformation that calculates the frequency content of local sections of the signal, but in this case, in a non-linear mel-scale frequency.

- Mel-frequency Cepstral Coefficients (MFCC): The coefficients that collectively make up an MFC, derived from a cepstral representation of the audio clip.

- Constant-Q Transform (CQT): Similar to a Fourier transform, it transforms a data series into a frequency-domain representation using a series of logarithmically spaced filters.

The audio files were processed using the Librosa[45] audio processing library, which was used to generate audio representations in the following formats: STFT (128 frequency bins), STFT (1024 frequency bins), FFT (1024 frequency bins), Mel-Spectrograms (128 frequency bins), Mel-Spectrograms (1024 frequency bins), MFCC (128 coefficients) and Constant-Q Transform (1008 frequency bins). The matrix-shaped audio representations (STFT, Mel-Spectrograms, MFCC and CQT) were then averaged on a horizontal axis, meaning, the frequency features were averaged over time. This results in one vector for each utterance.



FoR-2seconds (validation)

Algorithm	STFT 128	STFT 1024	FFT1024	Mel 128	Mel 1024	MFCC 128	CQT 1008
Naïve Bayes	61.32%	65.95%	58.45%	62.84%	65.14%	79.22%	60.40%
SVM	56.15%	57.11%	57.25%	78.27%	80.53%	92.07%	84.32%
Decision Tree (J48)	77.14%	72.64%	70.98%	91.93%	93.17%	93.20%	86.37%
Random Forests	82.55%	80.96%	76.96%	96.81%	96.60%	98.54%	94.01%

Figure 4.4: Frequency Analysis Accuracy - FoR 2 Seconds

[45] https://librosa.github.io/

Figure 4.4 shows the results of the frequency analysis experiments, in which it is possible to observe that using the MFCC audio representation with the Random Forests method achieves up to 98.54% accuracy on the validation dataset. This shows that even though frequency analysis may not be the best method, using just frequency information it is possible to achieve high accuracy.

To better understand the classification accuracy and the differences in the frequency spectrum, we decided to investigate which frequency ranges are more important for the classification task. We used two attribute ranking methods, Chi-Square[11] and Information Gain[37], to generate a *Frequency Classification Activation Map (FCAM)*.

Using the STFT (1024 bins) audio representation (due to its frequency bin linearity) we utilized the Weka tool to calculate the values of Chi-Square and Information gain for each of the frequency bins. To help with the visualization of the results, the frequencies were ordered (from 0Hz to 8kHz) and a colour was attributed to it: red meaning high-importance for classification, green meaning low importance for classification. The resulting frequency classification activation map can be seen in Figure 4.5.

The results of this experiment show that high frequencies (above 7.2kHz) are the most important frequencies to distinguish real speech from synthetic speech in its original form. Additional experiments to understand the audio distinctions

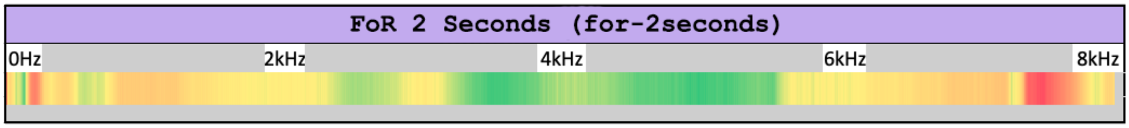


Figure 4.5: Frequency Classification Activation Map - FoR 2 Seconds

between real and synthetic speech can be found in Appendix B.

4.3.2 Deep Learning

In this subsection we detail the experiments conducted involving deep learning techniques for synthetic speech detection. Following previous literature, we translate the audio classification problem into an image classification problem by using visual audio representations (i.e. spectrograms). This conversion is useful since the majority of the deep learning models available are designed for image classification. Figure 4.6 shows an example of a STFT spectrogram for synthetic speech and real speech.

The deep learning experiments consist in extracting audio features (STFT, Mel-Spectrograms, MFCC and CQT) from the FoR dataset (for-2seconds) and converting it to an image. This process was done using a custom script and the Librosa library. The resulting images were then used to train each one of the nine selected architectures for a maximum of 50 epochs (early stop if accuracy improvements were not seen in the last 10 iterations). The selected architectures are:

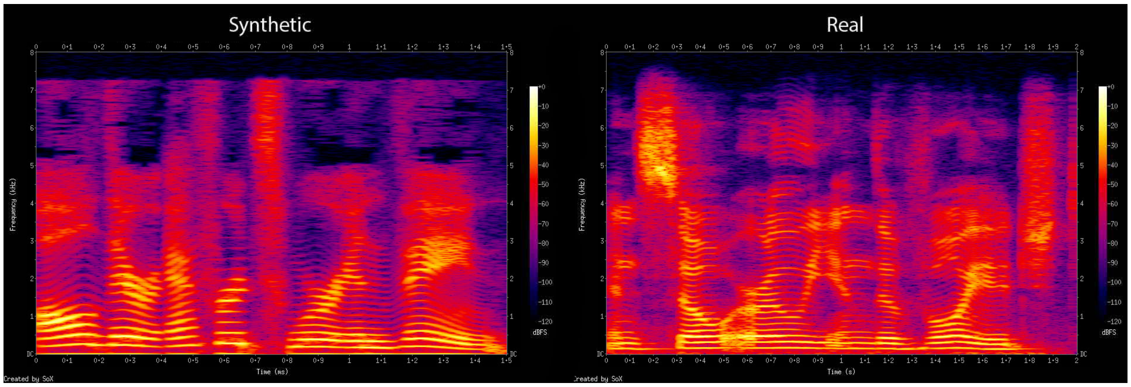
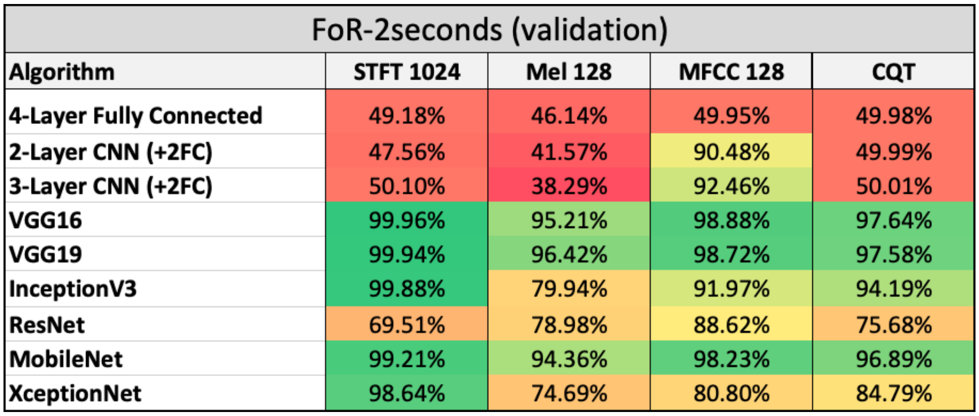


Figure 4.6: Example spectrograms

- 4 Layer Fully Connected
- 2-Layer CNN with extra 2 fully connected layers
- 3-Layer CNN with extra 2 fully connected layers
- VGG16[43] using the ImageNet weights and re-training only the last 5 layers
- VGG19[43] using the ImageNet weights and re-training only the last 5 layers
- InceptionV3[48] using the ImageNet weights and re-training only the last 2 inception blocks (249 layers)
- ResNet[45] using the ImageNet weights and re-training all the layers
- MobileNet[20] using the ImageNet weights and re-training all the layers
- XceptionNet[9] using the ImageNet weights and only re-training the last top layer

The results can be seen in Figure 4.7, where it is possible to observe that the VGG16 and VGG19 models with STFT audio representations presented the highest validation accuracy (99.96% and 99.94% respectively). As a side finding, we noted that simpler models (such as 4-Fully connected layer) were not able to learn the problem at all. The MobileNet, which is one of the top performers, is considerably faster to train/classify than the other top performer algorithms, meaning that it could be a good candidate if time is a constrain.



FoR-2seconds (validation)

Algorithm	STFT 1024	Mel 128	MFCC 128	CQT
4-Layer Fully Connected	49.18%	46.14%	49.95%	49.98%
2-Layer CNN (+2FC)	47.56%	41.57%	90.48%	49.99%
3-Layer CNN (+2FC)	50.10%	38.29%	92.46%	50.01%
VGG16	99.96%	95.21%	98.88%	97.64%
VGG19	99.94%	96.42%	98.72%	97.58%
InceptionV3	99.88%	79.94%	91.97%	94.19%
ResNet	69.51%	78.98%	88.62%	75.68%
MobileNet	99.21%	94.36%	98.23%	96.89%
XceptionNet	98.64%	74.69%	80.80%	84.79%

Figure 4.7: Deep Learning Accuracy - FoR 2 Seconds

One interesting technique in deep learning is the generation of Classification Activation Maps (CAMs). These maps show which areas of an image are more important for the classification task. As an example, Figure 4.8 contains the activation maps for dog images, where it is possible to observe that the eyes and ears are the main classification factors for detecting a dog.

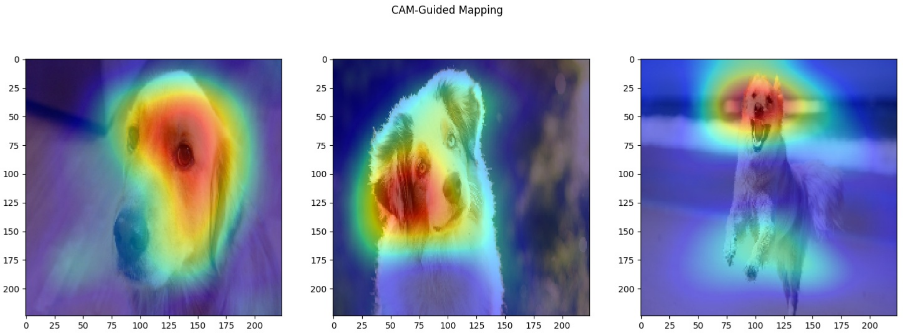


Figure 4.8: Example CAMs for dog classification

Similarly to the example in Figure 4.8, we created activation maps for real speech and synthetic speech. Figure 4.9 shows activation maps for randomly selected real utterances, while Figure 4.10 shows activation maps for randomly selected synthetic utterances. In both images, the X-axis represents the time in milliseconds while the Y-axis represents the spectrogram frequency bins. The CAMs were generated using the VGG19 model and the STFT audio representation.

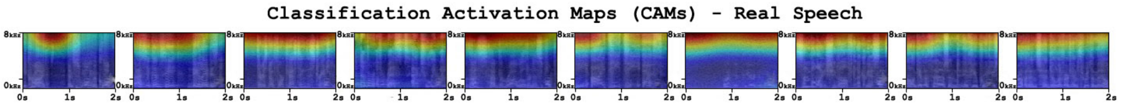


Figure 4.9: Example of CAMs for real utterances

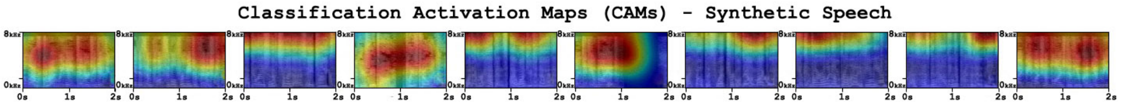


Figure 4.10: Example of CAMs for synthetic utterances

To create a general activation map, we generated the individual CAM for each spectrogram in the dataset and averaged the results into one CAM. The *Average Classification Activation Map (ACAM)* for real and synthetic speech of the for-2seconds dataset (VGG19 model, STFT audio representation) can be seen in Figure 4.11.

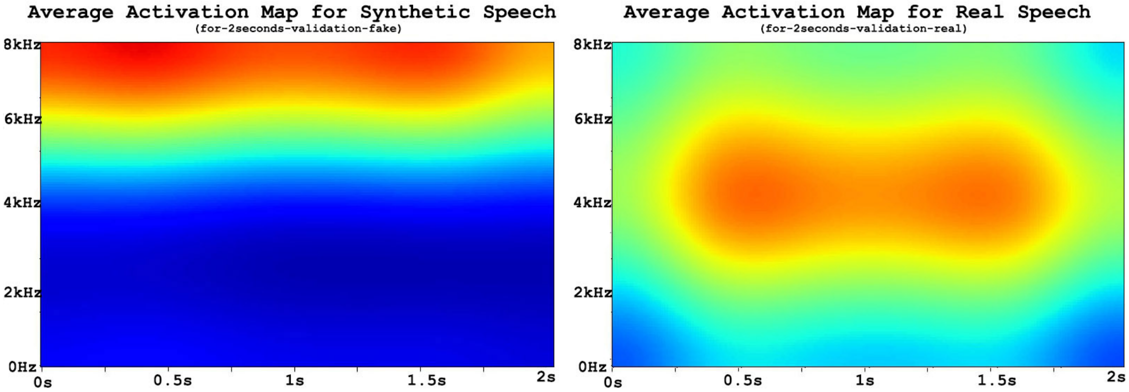


Figure 4.11: Average CAM for Real and Synthetic audio

From both individual CAMs and the average CAM it is possible to note that the higher frequencies are the most critical area for classification, especially in real utterances. This may indicate that synthetic speech generates audio mostly on the frequencies related to speech, while real audio may contain data in higher frequencies due to background noise or recording noises. Also, due to the fact that synthetic speech is mostly generated at 16kHz sample rate, the signal amplitude in frequencies close to 8kHz is low. This matches with the results on the frequency analysis experiments, which showed that high frequencies play an important role for synthetic speech detection.

Additional experiments related to deep learning techniques, such the impact of noise in classification accuracy and the use of waveforms for classification, can be found in Appendix C.

4.3.3 Experiment Discussion

Although the deep learning techniques presented the highest accuracy (99.96%), the frequency analysis methodologies also presented good results (98.54%). This indicates that there are clear distinctions on the frequency spectrum of real speech and synthesized speech. Those frequency discrepancies could be observed in both Frequency Classification Activation Map (Figure 4.5) and aCAM (Figure 4.11), which shows that synthetic speech has lower amplitudes in higher frequencies when compared to real speech.

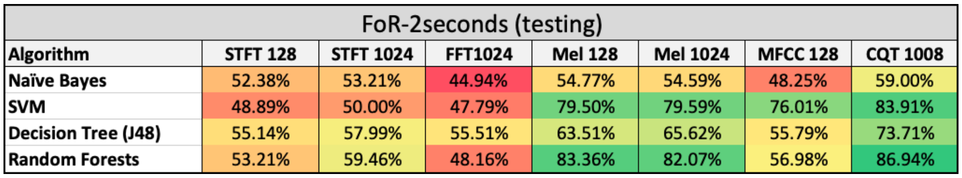
The next experiment (Section 4.4) tests the performance of the proposed methodologies against a totally unseen TTS algorithm. With that, we can evaluate if the frequency discrepancies are present in the new Google TTS Wavenet algorithm and if just the frequency discrepancies (without a seen voice/algorithm during training) are enough to detect synthetic speech.

4.4 Unseen Algorithm - Synthetic Speech Detection

To evaluate the generalization ability of the proposed models and to evaluate if just frequency discrepancies are enough to detect synthetic speech, we test the performance of the models trained in Section 4.3.1 and Section 4.3.2 against the testing part of the dataset (see Section 3.4), which is a totally unseen TTS algorithm (Google TTS Wavenet, which was not included in the training/validation dataset). This experiment also simulates how the models would react if an attacker creates a new TTS system.

4.4.1 Frequency Analysis

The first step is to observe the performance of frequency analysis methodologies against the unseen algorithm. Following the same process as the previous experiments, we generate the audio representations for the testing dataset and use the previously trained Weka models to classify the utterances of the unseen algorithm.



FoR-2seconds (testing)							
Algorithm	STFT 128	STFT 1024	FFT1024	Mel 128	Mel 1024	MFCC 128	CQT 1008
Naïve Bayes	52.38%	53.21%	44.94%	54.77%	54.59%	48.25%	59.00%
SVM	48.89%	50.00%	47.79%	79.50%	79.59%	76.01%	83.91%
Decision Tree (J48)	55.14%	57.99%	55.51%	63.51%	65.62%	55.79%	73.71%
Random Forests	53.21%	59.46%	48.16%	83.36%	82.07%	56.98%	86.94%

Figure 4.12: Frequency Analysis Accuracy - Unseen Algorithm

Figure 4.12 shows the result of this analysis. One interesting point to note is that the top performer audio representation (MFCC) from the last experiment (Section 4.3.1) had its accuracy drastically reduced. This may indicate that for unseen algorithms, MFCC may not be the best representation. However, the CQT audio representation presented good performance (94.01%) in the first experiment (Section 4.3.1) and now presents the best performance (86.94%).

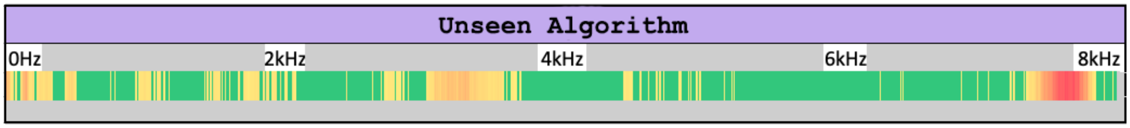
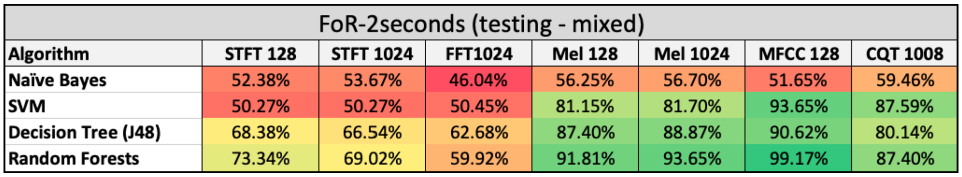


Figure 4.13: Frequency Activation Map - Unseen Algorithm

The frequency classification activation map (using Chi-Square and STFT) was generated for the testing dataset and can be seen in Figure 4.13. It is possible to note that the frequency classification regions are similar to the ones in the original experiment (Section 4.3.1), meaning that even though it is a new algorithm, the biggest differences between synthetic audio and real audio are still on the high frequencies.



FoR-2seconds (testing - mixed)

Algorithm	STFT 128	STFT 1024	FFT1024	Mel 128	Mel 1024	MFCC 128	CQT 1008
Naïve Bayes	52.38%	53.67%	46.04%	56.25%	56.70%	51.65%	59.46%
SVM	50.27%	50.27%	50.45%	81.15%	81.70%	93.65%	87.59%
Decision Tree (J48)	68.38%	66.54%	62.68%	87.40%	88.87%	90.62%	80.14%
Random Forests	73.34%	69.02%	59.92%	91.81%	93.65%	99.17%	87.40%

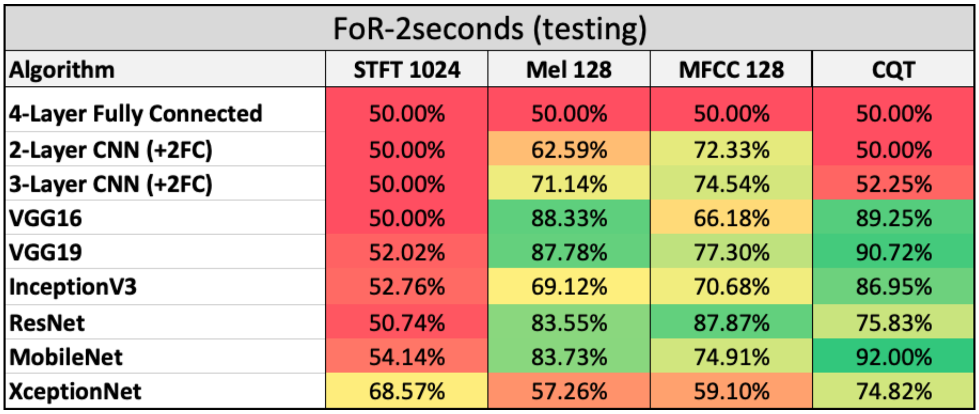
Figure 4.14: Frequency Analysis Accuracy - Learning a New Algorithm

To investigate if the models would be able to learn a new algorithm, we added

200 utterances (approximately 7 minutes) of the testing dataset into the training dataset and re-evaluated the testing dataset. The results can be seen in Figure 4.14, which shows that the models were able to adapt to the new TTS algorithm and deliver high accuracy (99.17%).

4.4.2 Deep Learning

To verify if deep learning algorithms perform better than frequency analysis on unseen algorithms, we used the same deep learning analysis process as the previous experiments on the testing part of the dataset. We used the models trained in Section 4.3.2 to classify the Google Wavenet TTS utterances.



FoR-2seconds (testing)

Algorithm	STFT 1024	Mel 128	MFCC 128	CQT
4-Layer Fully Connected	50.00%	50.00%	50.00%	50.00%
2-Layer CNN (+2FC)	50.00%	62.59%	72.33%	50.00%
3-Layer CNN (+2FC)	50.00%	71.14%	74.54%	52.25%
VGG16	50.00%	88.33%	66.18%	89.25%
VGG19	52.02%	87.78%	77.30%	90.72%
InceptionV3	52.76%	69.12%	70.68%	86.95%
ResNet	50.74%	83.55%	87.87%	75.83%
MobileNet	54.14%	83.73%	74.91%	92.00%
XceptionNet	68.57%	57.26%	59.10%	74.82%

Figure 4.15: Deep Learning Accuracy - Unseen Algorithm

Figure 4.15 shows the results of this experiment. Similarly to the Frequency Analysis methodology, the accuracy also decreased but stayed significantly high (92.00% using CQT and MobileNet). One interesting observation is that STFT, which in the previous experiment (Section 4.3.2) was the top performer, is now the audio representation with lowest accuracy. This may indicate that in unseen cases, CQT audio representation is the best option to be adopted.

To better understand the decline in the accuracy, we generated the ACAM for the synthetic part of the testing dataset. The ACAM was generated using CQT audio representation and the VGG19 model (second highest performance since MobileNet currently does not support reverse engineering). The results can be seen in Figure 4.16, which shows that high frequencies were still the main reason for the classification. This aligns with the results previously presented (Section 4.4.1), which shows that the biggest difference between Google Wavenet TTS and real utterances is on the high frequencies.

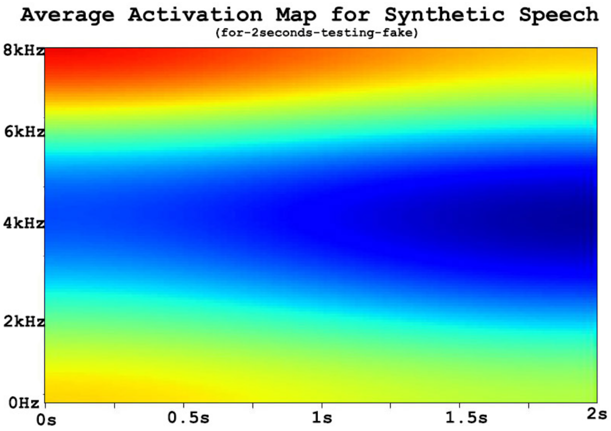
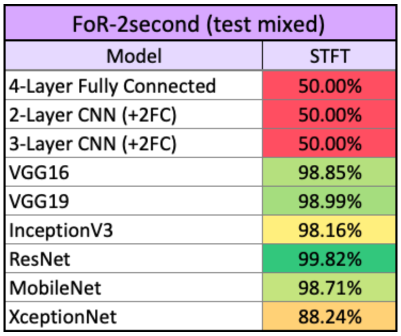


Figure 4.16: Averaged Class Activation Maps (ACAMs) for Unseen Synthetic Utterances

To investigate if the models would be able to learn a new algorithm, we added 200 utterances (approximately 7 minutes) of the testing dataset into the training dataset. Then, we re-trained the models (with STFT audio representation) and checked the testing accuracy. The results are shown in Figure 4.17, where it is possible to observe a considerable improvement in the testing accuracy(99.82%). This shows that although the proposed deep learning models had a significant decrease in performance with a fully unseen algorithm, only a small amount of data is required from the new TTS system to regain the original performance on the deep learning models.



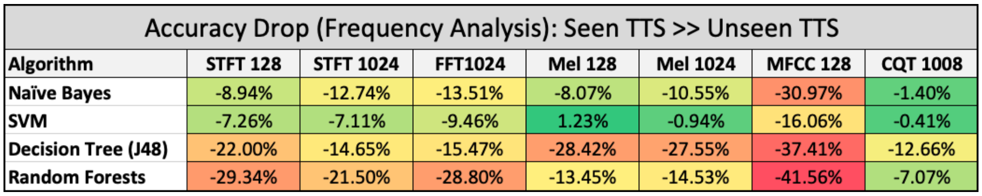
FoR-2second (test mixed)	
Model	STFT
4-Layer Fully Connected	50.00%
2-Layer CNN (+2FC)	50.00%
3-Layer CNN (+2FC)	50.00%
VGG16	98.85%
VGG19	98.99%
InceptionV3	98.16%
ResNet	99.82%
MobileNet	98.71%
XceptionNet	88.24%

Figure 4.17: Deep Learning Accuracy - Mixed Dataset

4.4.3 Experiment Discussion

In this experiment it is possible to observe that the behaviour of deep learning models was very similar to the frequency analysis methodologies: The accuracy drops with an unseen TTS algorithm but it is regained if just few utterances are added to the training data. However, throughout the whole experiment, the accuracy presented by the top performing deep learning methodology is higher than the one presented by the frequency analysis methodologies.

To better visualize the accuracy changes between the experiments in Section 4.3 and the unseen-TTS experiments, we present in the Figure 4.18 the accuracy drop for frequency-based methodologies, and in Figure 4.19 the accuracy drop for the deep learning approaches. It is interesting to note that some audio representations (such as MFCC for frequency analysis and STFT for deep learning) had a significant decrease in accuracy when compared to the experiments in Section 4.3. This may indicate that there is no general best solution for audio representation, instead, each case (seen or unseen TTS algorithms) has its own properties and requires different audio representations.

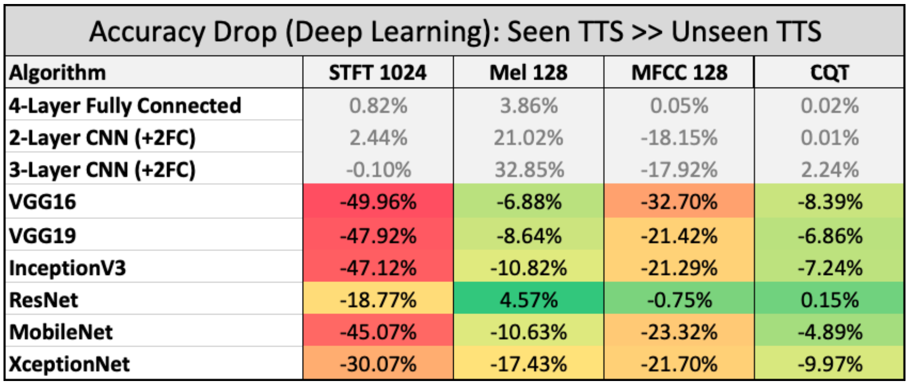


Accuracy Drop (Frequency Analysis): Seen TTS >> Unseen TTS

Algorithm	STFT 128	STFT 1024	FFT1024	Mel 128	Mel 1024	MFCC 128	CQT 1008
Naïve Bayes	-8.94%	-12.74%	-13.51%	-8.07%	-10.55%	-30.97%	-1.40%
SVM	-7.26%	-7.11%	-9.46%	1.23%	-0.94%	-16.06%	-0.41%
Decision Tree (J48)	-22.00%	-14.65%	-15.47%	-28.42%	-27.55%	-37.41%	-12.66%
Random Forests	-29.34%	-21.50%	-28.80%	-13.45%	-14.53%	-41.56%	-7.07%

Figure 4.18: Accuracy Drop - Unseen Algorithm - Frequency Analysis

In the next experiment, we test if the proposed methodologies are able to deliver good performance in a dataset where the magnitudes in the frequency bins are more uniform. As seen in Section 3.3.4, the re-recorded dataset presents lower frequency



Accuracy Drop (Deep Learning): Seen TTS >> Unseen TTS				
Algorithm	STFT 1024	Mel 128	MFCC 128	CQT
4-Layer Fully Connected	0.82%	3.86%	0.05%	0.02%
2-Layer CNN (+2FC)	2.44%	21.02%	-18.15%	0.01%
3-Layer CNN (+2FC)	-0.10%	32.85%	-17.92%	2.24%
VGG16	-49.96%	-6.88%	-32.70%	-8.39%
VGG19	-47.92%	-8.64%	-21.42%	-6.86%
InceptionV3	-47.12%	-10.82%	-21.29%	-7.24%
ResNet	-18.77%	4.57%	-0.75%	0.15%
MobileNet	-45.07%	-10.63%	-23.32%	-4.89%
XceptionNet	-30.07%	-17.43%	-21.70%	-9.97%

Figure 4.19: Accuracy Drop - Unseen Algorithm - Deep Learning

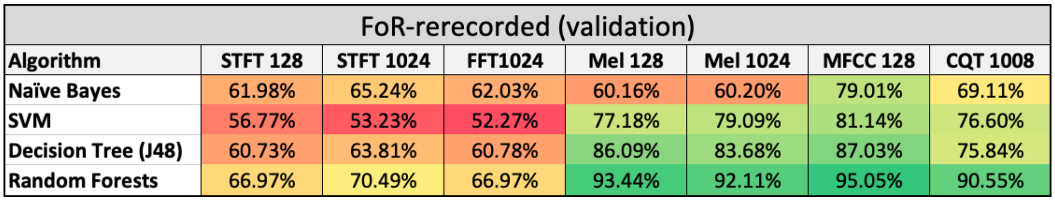
discrepancies between synthetic/real speech, especially in higher frequencies.

4.5 Rerecorded Synthetic Speech Detection

To evaluate the efficiency of the aforementioned detection approaches in a real-world scenario where an attacker plays a synthetic utterance through a voice channel, we apply them to the re-recorded dataset (for-rerecorded). This experiment is also important to test the accuracy of our models in a dataset where the differences in the high frequencies are reduced due to the rerecording process.

4.5.1 Frequency Analysis

Using the same process as in Section 4.3.1, we generated the frequency analysis results for the for-rerecorded dataset. The results from Weka are presented in Figure 4.20, which shows that the highest performance (95.05%) is with the MFCC audio representation and Random Forests, the same as the for-2second results (Section 4.3.1). It is also possible to note a small drop in accuracy when compared to the for-2seconds result (from 98.54% to 95.05%). The presented numbers indicate that even though the frequencies were more uniform, the algorithms were still able to identify the reduced frequency differences.



FoR-rerecorded (validation)

Algorithm	STFT 128	STFT 1024	FFT1024	Mel 128	Mel 1024	MFCC 128	CQT 1008
Naïve Bayes	61.98%	65.24%	62.03%	60.16%	60.20%	79.01%	69.11%
SVM	56.77%	53.23%	52.27%	77.18%	79.09%	81.14%	76.60%
Decision Tree (J48)	60.73%	63.81%	60.78%	86.09%	83.68%	87.03%	75.84%
Random Forests	66.97%	70.49%	66.97%	93.44%	92.11%	95.05%	90.55%

Figure 4.20: Frequency Analysis Accuracy - FoR Rerecorded

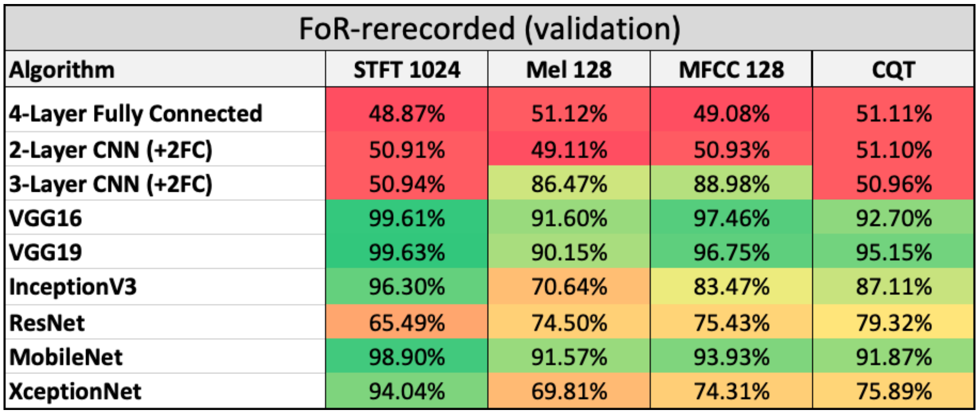
To better understand the decline in the accuracy, a binary classification experiment was conducted with original synthetic utterances and re-recorded synthetic utterances. Then, we used traditional frequency analysis techniques, such as Naive Bayes and Random Forests, to perform the classification. The result was a classification accuracy of 99.86% (using random forests), showing that there are significant differences between original and re-recorded audio.

The main hypothesis for the decline in the accuracy is related to the fact that, as seen in Section 3.3.4, the re-recording process reduces the frequency discrepancies between synthetic and real speech, especially in high frequencies.

4.5.2 Deep Learning

Using the same process presented in Section 4.3.2, we analyzed the impact of speech re-recording for deep learning models. The idea is to test if deep neural networks are able to distinguish real and synthetic utterances in re-recorded audio.

Using the re-recorded dataset presented in Section 4.5 (for-rerecorded), we trained and evaluated the deep learning models selected for this research. Figure 4.21 shows the result of this analysis, where it is possible to observe that the highest validation accuracy (VGG19 and STFT, 99.63%) is similar to the highest accuracy on the for-2second dataset (VGG16 and STFT, 99.96%). This shows that the re-recording process had little-to-no impact on the performance of the deep learning methodologies.



FoR-rerecorded (validation)

Algorithm	STFT 1024	Mel 128	MFCC 128	CQT
4-Layer Fully Connected	48.87%	51.12%	49.08%	51.11%
2-Layer CNN (+2FC)	50.91%	49.11%	50.93%	51.10%
3-Layer CNN (+2FC)	50.94%	86.47%	88.98%	50.96%
VGG16	99.61%	91.60%	97.46%	92.70%
VGG19	99.63%	90.15%	96.75%	95.15%
InceptionV3	96.30%	70.64%	83.47%	87.11%
ResNet	65.49%	74.50%	75.43%	79.32%
MobileNet	98.90%	91.57%	93.93%	91.87%
XceptionNet	94.04%	69.81%	74.31%	75.89%

Figure 4.21: Deep Learning Accuracy - FoR Rerecorded

To better understand what is being learned by the model, the ACAMs were generated for both synthetic and real audio (using STFT and VGG19), which can be seen in Figure 4.22. The interesting point is that the model now presents a smoother classification area, showing that the discrepancies in the higher frequencies are not as drastic as in the for-2second dataset.

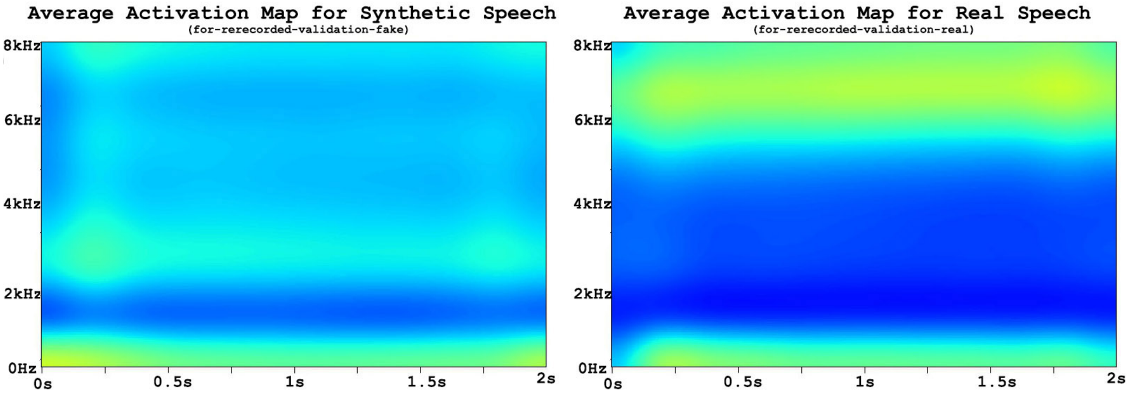
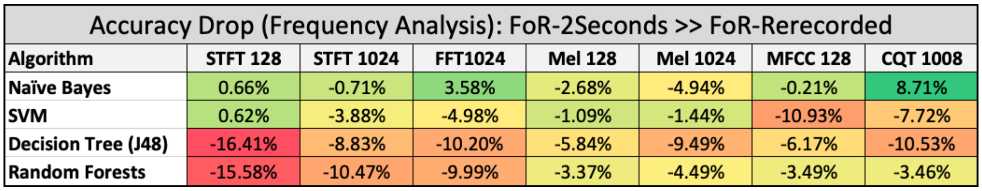


Figure 4.22: Averaged Class Activation Maps (ACAMs) on Re-recorded Dataset

4.5.3 Experiment Discussion

This experiment shows the advantages of using deep learning techniques over frequency based ones. The rerecording process had little-to-no impact on the accuracy of deep learning classification techniques, while the frequency analysis approaches had a considerable decrease in accuracy. To observe the accuracy changes from the original experiments (Section 4.3), we created the Figure 4.23, which shows the accuracy drop between the original frequency analysis experiment and the rerecorded frequency analysis experiment. It is possible to note a significant decrease in accuracy, specially for the STFT audio representation.

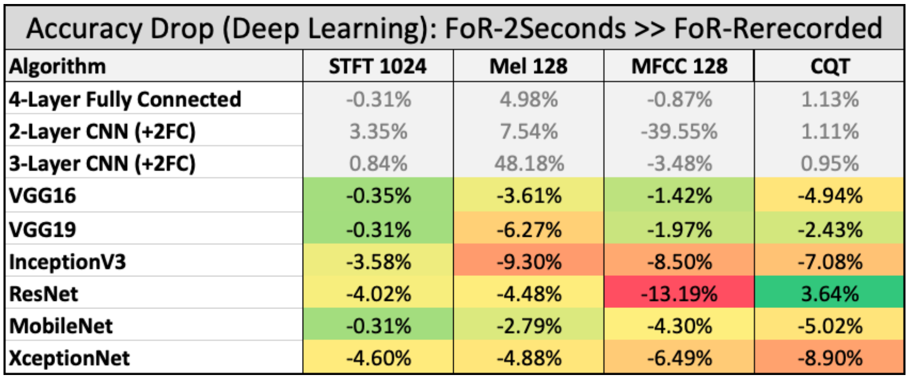
Similarly, we calculated the accuracy drop for Deep Learning, shown in Figure 4.24. It is possible to observe an overall lower impact on the accuracy when compared to frequency analysis. It is interesting to note that, while for frequency analysis the STFT was the most affected audio representation, for deep learning it had almost no impact on the accuracy (especially on the VGG19/VGG16 architectures, which were the best performing ones).



Accuracy Drop (Frequency Analysis): FoR-2Seconds >> FoR-Rerecorded

Algorithm	STFT 128	STFT 1024	FFT1024	Mel 128	Mel 1024	MFCC 128	CQT 1008
Naïve Bayes	0.66%	-0.71%	3.58%	-2.68%	-4.94%	-0.21%	8.71%
SVM	0.62%	-3.88%	-4.98%	-1.09%	-1.44%	-10.93%	-7.72%
Decision Tree (J48)	-16.41%	-8.83%	-10.20%	-5.84%	-9.49%	-6.17%	-10.53%
Random Forests	-15.58%	-10.47%	-9.99%	-3.37%	-4.49%	-3.49%	-3.46%

Figure 4.23: Accuracy Drop - Rerecorded Dataset - Frequency Analysis



Accuracy Drop (Deep Learning): FoR-2Seconds >> FoR-Rerecorded

Algorithm	STFT 1024	Mel 128	MFCC 128	CQT
4-Layer Fully Connected	-0.31%	4.98%	-0.87%	1.13%
2-Layer CNN (+2FC)	3.35%	7.54%	-39.55%	1.11%
3-Layer CNN (+2FC)	0.84%	48.18%	-3.48%	0.95%
VGG16	-0.35%	-3.61%	-1.42%	-4.94%
VGG19	-0.31%	-6.27%	-1.97%	-2.43%
InceptionV3	-3.58%	-9.30%	-8.50%	-7.08%
ResNet	-4.02%	-4.48%	-13.19%	3.64%
MobileNet	-0.31%	-2.79%	-4.30%	-5.02%
XceptionNet	-4.60%	-4.88%	-6.49%	-8.90%

Figure 4.24: Accuracy Drop - Rerecorded Dataset - Deep Learning

Since in the rerecorded dataset the frequency spectrum is more uniform between real and synthetic (see Section 3.3.4), the frequency based approaches were more impacted by the rerecording process, while the deep learning based methods were almost not impacted by the rerecording process and delivered an accuracy similar to the original for-2second dataset.

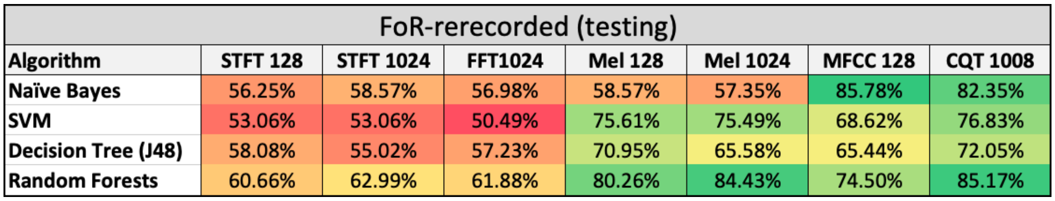
The final and most challenging test is to analyze the performance of the proposed methodologies against a totally unseen TTS algorithm (Google TTS Wavenet) in a rerecorded scenario. This experiment is presented in the next section.

4.6 Unseen Algorithm - Rerecorded Synthetic Speech Detection

To evaluate the generalization capabilities (through an unseen TTS algorithm) in a frequency-uniform dataset (through rerecording) we analyzed the performance of the proposed methodologies against the rerecorded version of the testing dataset (for-rerecorded - testing). This simulates a scenario where an attacker possesses a new TTS algorithm and sends the utterances through a voice-channel (i.e. a voice message or a call).

4.6.1 Frequency Analysis

As in the previous experiments, we start by analyzing the performance of the frequency based methodologies against the testing part of the rerecorded dataset (for-rerecorded). Following the same methodology, we generated the comparison between audio representations and classification models. The comparison can be seen in Figure 4.25.



FoR-rerecorded (testing)							
Algorithm	**STFT 128**	**STFT 1024**	**FFT1024**	**Mel 128**	**Mel 1024**	**MFCC 128**	**CQT 1008**
Naïve Bayes	56.25%	58.57%	56.98%	58.57%	57.35%	85.78%	82.35%
SVM	53.06%	53.06%	50.49%	75.61%	75.49%	68.62%	76.83%
Decision Tree (J48)	58.08%	55.02%	57.23%	70.95%	65.58%	65.44%	72.05%
Random Forests	60.66%	62.99%	61.88%	80.26%	84.43%	74.50%	85.17%

Figure 4.25: Frequency Analysis Accuracy - Unseen FoR Rerecorded

As seen in Figure 4.25, the highest accuracy (85.78%, CQT and Random Forests) is considerably lower than the original for-2second dataset (98.54%, MFCC and Random Forests). This shows that the frequency analysis method is significantly impacted by the rerecording of an unseen TTS algorithm. It is interesting to note that the shift in the best audio representation, from MFCC to CQT, is similar to the unseen TTS experiments (Section 4.4.1). This confirms our theory that CQT may be the best audio representation if it is an unseen algorithm (independent from it being rerecorded or not).

To better understand which frequencies are more relevant for the classification process, we generated the Frequency Classification Activation Map (using STFT) for the unseen utterances of the re-recorded dataset. The FCAM can be seen in Figure 4.26.

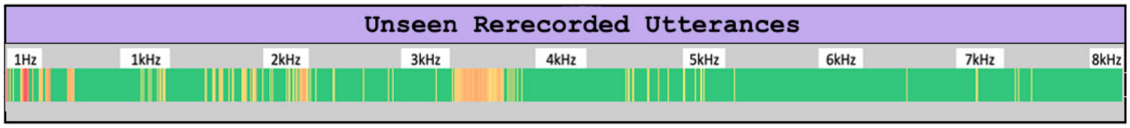


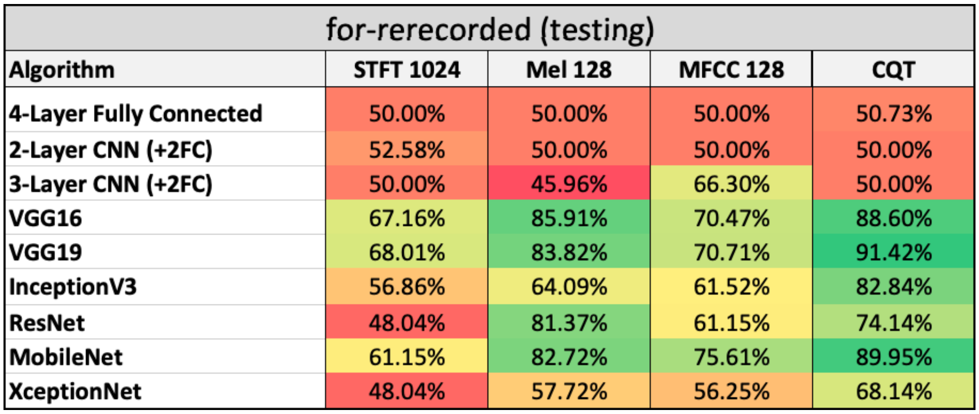
Figure 4.26: Frequency Classification Activation Map - Unseen FoR Rerecorded

The FCAM shows a shift in the high-relevance classification area (red and orange areas of the FCAM) when compared to the FCAM presented for the for-2second dataset (in Section 4.3.1). In this experiment, the high frequencies were not the main factor for classification, potentially due to the fact that the rerecording process reduces the discrepancies on high frequencies. Interestingly, the main classification area was shifted to low frequencies (around 140Hz), which may justify the decrease in the accuracy.

4.6.2 Deep Learning

Similarly to previous experiments, we took the target dataset (for-rerecorded - testing), converted to a variety of audio representations and generated a comparison between audio representations and deep learning models. The results can be seen in Figure 4.27.

The highest accuracy across this experiment (91.42%, CQT and VGG19) is reduced when compared to the for-2second dataset but is still high (over 90%). This shows that although the deep learning algorithms were affected by the fact



for-rerecorded (testing)				
Algorithm	**STFT 1024**	**Mel 128**	**MFCC 128**	**CQT**
4-Layer Fully Connected	50.00%	50.00%	50.00%	50.73%
2-Layer CNN (+2FC)	52.58%	50.00%	50.00%	50.00%
3-Layer CNN (+2FC)	50.00%	45.96%	66.30%	50.00%
VGG16	67.16%	85.91%	70.47%	88.60%
VGG19	68.01%	83.82%	70.71%	91.42%
InceptionV3	56.86%	64.09%	61.52%	82.84%
ResNet	48.04%	81.37%	61.15%	74.14%
MobileNet	61.15%	82.72%	75.61%	89.95%
XceptionNet	48.04%	57.72%	56.25%	68.14%

Figure 4.27: Deep Learning Accuracy - Unseen FoR Rerecorded

that the utterances were unseen and rerecorded, the best deep learning performer still performed well on the synthetic speech detection task. It is also interesting to note that the shift in the best accuracy for audio representation, from STFT to CQT, is the same shift observed in the non-rerecorded algorithm. This is one more piece of evidence that for unseen TTS, the CQT is the best performing audio representation.

To visualize what is being learned by the model, the ACAM for the VGG19 model (CQT audio representation) was generated and can be seen in Figure 4.28. Similarly to what was observed in the frequency analysis, the main classification area is now the lower frequencies. This aligns with the previous observation that the rerecording process smooths out the high frequencies, which leads to a shift on the classification areas in the ACAM.

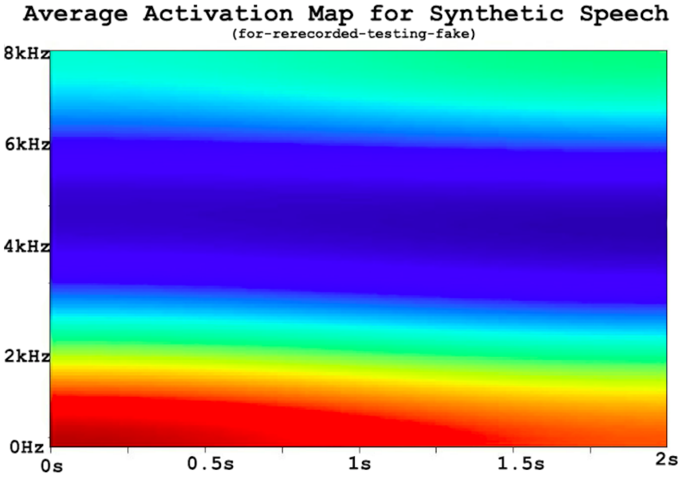
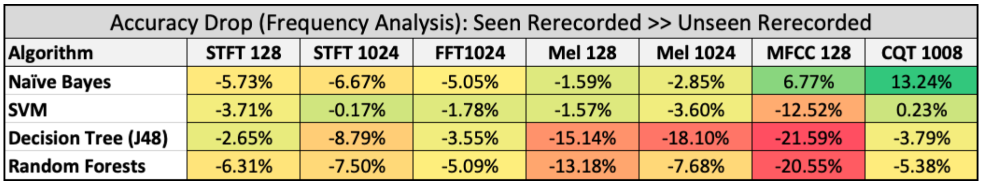


Figure 4.28: Averaged Class Activation Maps (ACAMs) on Unseen Re-recorded Dataset

4.6.3 Experiment Discussion

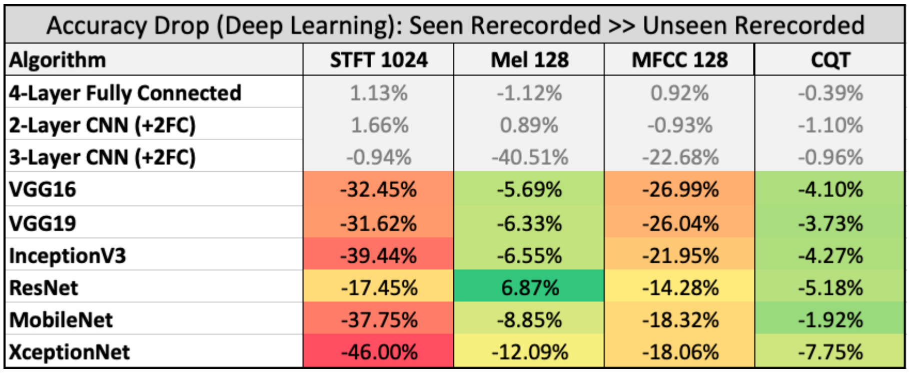
This experiment shows a clear distinction between the performance of frequency analysis and deep learning methodologies. While the best accuracy on the frequency analysis had a decline of 12.76% in accuracy, the best accuracy on deep learning had a decline of only 8.54%. This means that deep learning models are 66.92% (8.54% over 12.76%) more effective than frequency analysis in the detection of unseen synthetic speech in a re-recorded environment.

To better visualize the accuracy drop (from seen for-rerecorded to unseen for-rerecorded) between each audio representation and model, we created Figure 4.29 that shows the accuracy drop for frequency analysis methods and Figure 4.30 that shows the accuracy drop for deep learning methods.



Accuracy Drop (Frequency Analysis): Seen Rerecorded >> Unseen Rerecorded							
Algorithm	**STFT 128**	**STFT 1024**	**FFT1024**	**Mel 128**	**Mel 1024**	**MFCC 128**	**CQT 1008**
Naïve Bayes	-5.73%	-6.67%	-5.05%	-1.59%	-2.85%	6.77%	13.24%
SVM	-3.71%	-0.17%	-1.78%	-1.57%	-3.60%	-12.52%	0.23%
Decision Tree (J48)	-2.65%	-8.79%	-3.55%	-15.14%	-18.10%	-21.59%	-3.79%
Random Forests	-6.31%	-7.50%	-5.09%	-13.18%	-7.68%	-20.55%	-5.38%

Figure 4.29: Accuracy Drop - Unseen Rerecorded - Frequency Analysis



Accuracy Drop (Deep Learning): Seen Rerecorded >> Unseen Rerecorded				
Algorithm	**STFT 1024**	**Mel 128**	**MFCC 128**	**CQT**
4-Layer Fully Connected	1.13%	-1.12%	0.92%	-0.39%
2-Layer CNN (+2FC)	1.66%	0.89%	-0.93%	-1.10%
3-Layer CNN (+2FC)	-0.94%	-40.51%	-22.68%	-0.96%
VGG16	-32.45%	-5.69%	-26.99%	-4.10%
VGG19	-31.62%	-6.33%	-26.04%	-3.73%
InceptionV3	-39.44%	-6.55%	-21.95%	-4.27%
ResNet	-17.45%	6.87%	-14.28%	-5.18%
MobileNet	-37.75%	-8.85%	-18.32%	-1.92%
XceptionNet	-46.00%	-12.09%	-18.06%	-7.75%

Figure 4.30: Accuracy Drop - Unseen Rerecorded - Deep Learning

Similarly to the unseen for-2second experiments (Section 4.4), it is possible to observe a significant drop in certain audio representations: MFCC for frequency analysis and STFT for deep learning. This confirms our theory that there is no generalized best audio representation for synthetic speech detection and each case should adopt its own appropriate audio representation. For deep learning approaches in seen data, the STFT audio representation is recommended, while for unseen data CQT is recommended. For frequency based approaches, MFCC is recommended for seen data while CQT is recommended for unseen data. It is also interesting to note that CQT presented the lowest averaged accuracy drop in all experiments, which may indicate the CQT is the most reliable audio representation if the type of the data (original/rerecorded, seen/unseen) is unknown.

Across all our frequency analysis experiments, it was possible to note that Random Forests presented the best performance in 3 out of 4 experiments (being only 0.61% behind in the unseen rerecorded experiment), meaning that it may be the best frequency based model for synthetic speech detection.

Across all deep learning experiments, the VGG19 model presented the best performance in 2 out of 4 experiments, being behind only 0.06% in the first experiment and 1.28% behind on the second experiment. This shows that VGG19 presents an overall good performance and may be the best classifier model for synthetic speech detection.

5 Conclusion

As synthetic speech generation improves, the need for synthetic speech detection speech increases. With this work we hope to stimulate further research in synthetic speech detection. In this section, we discuss the conclusions from our research, including the overall results from our experiments, thesis contributions and a proposed roadmap for future work.

5.1 Overall Results

To summarize the results from all experiments, we created a diagram, shown in Figure 5.1, presenting the accuracy for all experiments performed. From those results it is possible to observe and compare the impact of the experiments in the overall accuracy.

From all the experiments in our research, it is possible to observe that, in general, the results from the top-performers deep learning based techniques present better accuracy than the top-performers traditional frequency analysis methods.

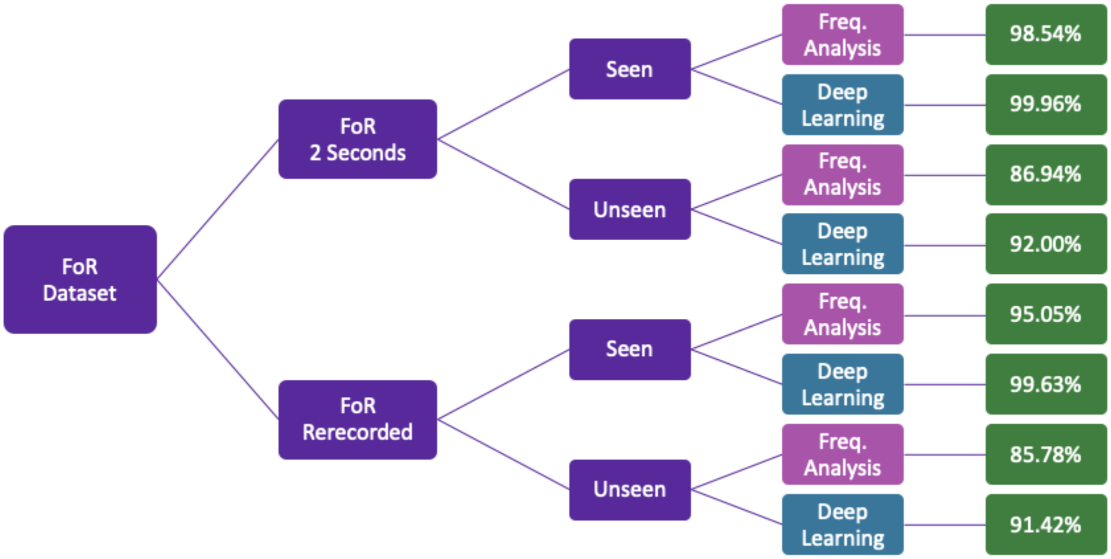


Figure 5.1: Experiments Results Summary

5.2 Thesis Contributions

To summarize the observations from this research, we compiled a list of key findings in this thesis:

- The performance of humans against the latest speech synthesizers: According to our study presented in Section 4.2.2, the human performance on detecting the latest TTS algorithms is low (64.83%). This shows the need for an automated system that is capable of detecting synthetic utterances.

- The frequency discrepancies between real and synthetic speech: From the majority of experiments presented in Chapter 4, it was possible to note that the major differences between real and synthetic speech is on the high frequencies (above 7.2kHz). Those discrepancies were the main classification factor for both frequency based methodologies and deep learning techniques.

- The impact of rerecording in synthetic speech detection: According to our observations, the rerecording process (which simulates an attacker sending an utterance through a voice channel) mainly impacts the higher frequencies of the audio. After the rerecording, the discrepancies between real and synthetic speech in high frequencies were minimized, causing a reduction on accuracy, specially on frequency based methodologies.
- The importance of audio normalization: Since the synthetic utterances and real utterances came from very distinct sources, with distinct audio properties (such as sample rate, number of channels, etc.), we noted that performing pre-processing to normalize the audio is very important. In early experiments, the discrepancies in sample rate and audio channels were directly interfering on the learning process. After normalization, the bias resulting from the audio collection process was minimized.
- The advantages of using deep learning techniques for synthetic speech detection: In all our experiments, the top performing deep learning methodologies presented higher accuracy than frequency based ones. Also, the deep learning

techniques are better at generalizing and adapting to different scenarios, such as detecting rerecorded synthetic utterances and unseen TTS algorithms.

With this publication, we provide to the community a solid framework for synthetic speech analysis and detection. This framework includes:

- A new synthetic speech dataset: The FoR dataset was released to the public with the hope that the community explore the data and find even more interesting aspects about the synthetic speech detection. Containing more than 195,000 utterances, the FoR Dataset includes the latest TTS algorithms and a large variety of real speech.

- A thorough comparison of synthetic speech detection: Our study includes 7 audio representations, as well as 9 deep learning architectures and 4 frequency analysis methods. This large variety of audio representations and models provide a detailed analysis of which algorithms and representations perform better in each synthetic speech detection scenario.

- A series of trained models for synthetic speech detection: Our study produced a total of 72 trained deep learning architectures and 56 trained frequency analysis models. The accuracy of each model is presented in Chapter 4.

5.3 Future Work

There are several additional research topics that could be explored, such as:

- Raw-audio classifiers: The majority of our deep learning research was based on the transformation of audio into spectrograms. However, as seen in papers such as Wavenet[32], it is possible to directly input raw audio into neural networks, without the conversion to spectrograms. This was widely explored for speech synthesis. However, to the best of our knowledge, raw-audio was never used as classifier for the synthetic speech detection problem. This may increase the classification accuracy and reduce the pre-processing time (since spectrograms are not needed).

- Temporal Convolutional Networks: The experiments performed in this work were using regular convolutional networks (squared convolution filters). However, as the spectrogram is a series of frequency-magnitude measurements, one could try to implement a classifier based on temporal convolutional networks[34].

- A generic model for synthetic speech detection: In our work, we created separate models for detecting "original" synthetic speech and rerecorded synthetic speech. One interesting research topic would be creating a unified model that is able to properly detect a synthetic utterance independently if it is re-recorded or not.

- A heterogeneous rerecorded dataset: Our utterance rerecording was performed using only one type of speaker and microphone. An interesting experiment would be using a large variety of recording/playing devices in a large variety of recording rooms. This would create a more heterogeneous rerecorded dataset and would create a more generalized synthetic speech detection model.

- A in-depth study about the accuracy variation across audio representations: As observed in our experiments, some audio representations perform well on a specific experiment and poorly in others. An interesting research topic would be understanding the reason behind this phenomenon and whether audio compression plays a role in this variation.

- A browser plugin: Since we developed models for synthetic speech detection, one could create an application (such as a browser plugin) that is able to detect if synthetic audio is being played in a web page. This would benefit the community by telling people if the audio that they are listening is synthesized or real, reducing the likelihood of successful impersonation attacks.

Bibliography

[1] Jonathan Allen, Sharon M. Hunnicutt, and Dennis Klatt. From text to speech: The mitalk system. *Cambridge University Press*, 1989.

[2] Russell Mason Andy Pearce, Tim Brookes. Audio commons: An ecosystem for creative reuse of audio content, 2019.

[3] Sercan Arik, Gregory Diamos, Andrew Gibiansky, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep Voice 2: Multi-Speaker Neural Text-to-Speech. *arXiv:1705.08947 [cs]*, May 2017. arXiv: 1705.08947.

[4] Sercan O. Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural Voice Cloning with a Few Samples. *arXiv:1802.06006 [cs, eess]*, February 2018. arXiv: 1802.06006.

[5] Sercan O. Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, Shubho Sengupta, and Mohammad Shoeybi. Deep Voice: Real-time Neural Text-to-Speech. *arXiv:1702.07825 [cs]*, February 2017. arXiv: 1702.07825.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. bibtex:attention2.

[7] Gregory Burlet and Abram Hindle. Isolated guitar transcription using a deep belief network. *PeerJ Computer Science*, 3:e109, March 2015. bibtex:transcription1.

[8] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015. bibtex:attention1.

[9] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv:1610.02357 [cs]*, October 2016. arXiv: 1610.02357.

[10] T.K. Chu. *Speech Synthesis by Concatenation of Digital Waveform Fragments*. Thesis (Ph.D.)–University of British Columbia, 1978.

[11] William G. Cochran. The χ^2 test of goodness of fit. *Ann. Math. Statist.*, 23(3):315–345, 09 1952.

[12] Heinrich Dinkel, Yanmin Qian, and Kai Yu. Investigating Raw Wave Deep Neural Networks for End-to-End Speaker Spoofing Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2002–2014, November 2018.

[13] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1068–1077, 2017. bibtex:wavenet2.

[14] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T. Freeman, and Michael Rubinstein. Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation. *arXiv:1804.03619 [cs, eess]*, April 2018. arXiv: 1804.03619.

[15] Kunihiko Fukushima. *Neocognitron: A self-organizing neural network model for a mechanism of pattern recogni- tion unaffected by shift in position*, volume 36 of *4*. Springerr, 1980.

[16] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image Style Transfer Using Convolutional Neural Networks. pages 2414–2423. IEEE, June 2016.

[17] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[20] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]*, April 2017. arXiv: 1704.04861.

[21] Tomi Kinnunen, Md. Sahidullah, Héctor Delgado, Massimiliano Todisco, Nicholas Evans, Junichi Yamagishi, and Kong Aik Lee. The ASVspoof 2017 Challenge: Assessing the Limits of Replay Spoofing Attack Detection. In *Interspeech 2017*, pages 2–6. ISCA, August 2017.

[22] J.F. Kolen and S.C. Kremer. *A Field Guide to Dynamical Recurrent Networks*. Wiley, 2001.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. bibtex:alexnet.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

[25] L F Lamel, J L Gauvain, B Prouts, C Bouhier, and R Boesch. Generation and Synthesis of Broadcast Messages. page 4.

[26] Yann Lecun, L. D. Jackel, Harris A. Eduard, N Bottou, Corinna Cartes, John S. Denker, Harris Drucker, Eduard Sackinger, Patrice Simard, and Vladimir Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. In *Neural Networks: The Statistical Mechanics Perspective*, pages 261–276. World Scientific, 1995.

[27] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *arXiv:1312.4400 [cs]*, December 2013. arXiv: 1312.4400.

[28] Jorge Lucero, Jean Schoentgen, and Mara Behlau. Physics-based synthesis of disordered voices. 08 2013.

[29] Francesco Marra, Diego Gragnaniello, Davide Cozzolino, and Luisa Verdoliva. Detection of GAN-Generated Fake Images over Social Networks. pages 384–389. IEEE, April 2018.

[30] Hannah Muckenhirn, Mathew Magimai-Doss, and Sebastien Marcel. End-to-End convolutional neural network-based voice presentation attack detection. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 335–341, Denver, CO, October 2017. IEEE.

[31] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning Deconvolution Network for Semantic Segmentation. *arXiv:1505.04366 [cs]*, May 2015. arXiv: 1505.04366.

[32] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499 [cs]*, September 2016. arXiv: 1609.03499.

[33] D. Paul, M. Pal, and G. Saha. Spectral Features for Synthetic Speech Detection. *IEEE Journal of Selected Topics in Signal Processing*, 11(4):605–617, June 2017.

[34] Charlotte Pelletier, Geoffrey I. Webb, and Francois Petitjean. Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series. *arXiv:1811.10166 [cs]*, November 2018. arXiv: 1811.10166.

[35] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O. Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning. *arXiv:1710.07654 [cs, eess]*, October 2017. arXiv: 1710.07654.

[36] Warren S. McCullochWalter Pitts. Ia logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[37] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, Mar 1986.

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 12 2015.

[39] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics: A Large-scale Video Dataset for Forgery Detection in Human Faces. *arXiv:1803.09179 [cs]*, March 2018. arXiv: 1803.09179.

[40] Md Sahidullah, Tomi Kinnunen, and Cemal Hanilçi. A comparison of features for synthetic speech detection. 09 2015.

[41] Y. Saito, S. Takamichi, and H. Saruwatari. Training algorithm to deceive Anti-Spoofing Verification for DNN-based speech synthesis. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4900–4904, March 2017.

[42] Youngmoo Kim Shaun Barry. “STYLE” TRANSFER FOR MUSICAL AUDIO USING MULTIPLE TIME-FREQUENCY REPRESENTATIONS, 2018.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[44] Tatinati Sivanagaraja, Mun Kit Ho, Andy W H Khong, and Yubo Wang. End-to-end speech emotion recognition using multi-scale convolution networks. pages 189–192. IEEE, December 2017.

[45] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. page 7.

[46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *arXiv:1409.4842 [cs]*, September 2014. arXiv: 1409.4842.

[47] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[48] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567 [cs]*, December 2015. arXiv: 1512.00567.

[49] I. J. Tashev, Zhong-Qiu Wang, and K. Godin. Speech emotion recognition based on Gaussian Mixture Models and Deep Neural Networks. In *2017 Information Theory and Applications Workshop (ITA)*, pages 1–4, February 2017.

[50] Paul Taylor. *Text-to-Speech Synthesis.* Cambridge University Press, 2009.

[51] Xiaohai Tian, Xiong Xiao, Eng Siong Chng, and Haizhou Li. Spoofing speech detection using temporal convolutional neural network. In *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–6, Jeju, South Korea, December 2016. IEEE.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, June 2017. arXiv: 1706.03762.

[53] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University, 1975.

[54] Z. Wu, P. L. De Leon, C. Demiroglu, A. Khodabakhsh, S. King, Z. Ling, D. Saito, B. Stewart, T. Toda, M. Wester, and J. Yamagishi. Anti-Spoofing for Text-Independent Speaker Verification: An Initial Database, Comparison of Countermeasures, and Human Performance. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):768–783, April 2016.

[55] Z. Wu, P. L. De Leon, C. Demiroglu, A. Khodabakhsh, S. King, Z. Ling, D. Saito, B. Stewart, T. Toda, M. Wester, and J. Yamagishi. Anti-Spoofing for Text-Independent Speaker Verification: An Initial Database, Comparison of Countermeasures, and Human Performance. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):768–783, April 2016.

[56] Z. Wu, X. Xiao, E. S. Chng, and H. Li. Synthetic speech detection using temporal modulation feature. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7234–7238, May 2013.

[57] Zhizheng Wu, Tomi Kinnunen, Nicholas Evans, and Junichi Yamagishi. Automatic speaker verification spoofing and countermeasures challenge (asvspoof 2015) database, 2015.

[58] H. Yu, Z. Tan, Z. Ma, R. Martin, and J. Guo. Spoofing Detection in Automatic Speaker Verification Systems Using DNN Classifiers and Dynamic Acoustic Features. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4633–4644, October 2018.

[59] Hong Yu, Achintya Sarkar, Dennis Alexander Lehmann Thomsen, Zheng-Hua Tan, Zhanyu Ma, and Jun Guo. Effect of multi-condition training and speech enhancement methods on spoofing detection. 07 2016.

[60] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. bibtex:zfnet.

[61] Heiga Zen, Takashi Nose, Junichi Yamagishi, Shinji Sako, Takashi Masuko, Alan W Black, and Keiichi Tokuda. The hmm-based speech synthesis system (hts) version 2.0. In *SSW*, pages 294–299. Citeseer, 2007.

[62] Chunlei Zhang, Chengzhu Yu, and John H. L. Hansen. An Investigation of Deep-Learning Frameworks for Speaker Verification Antispoofing. *IEEE Journal of Selected Topics in Signal Processing*, 11(4):684–694, June 2017.

[63] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

Appendix A: Dataset Pre-Processing

In this appendix chapter, we describe in more details the pre processing steps applied on the for-original dataset to create the subsequent dataset versions.

A.1 Filetype Conversion

The original dataset contains audio in two formats: WAV and MP3. Since WAV is the preferred format for input for the machine learning solutions, all audio files were converted to WAV using the ffmpeg[46] tool. To automate the conversion of all the files, a script was created to convert the whole dataset keeping the same folder structure.

One of the concerns during the experiments was that the format in which the file was recorded could impact on the classification accuracy. To test this hypothesis, we performed an experiment in which we converted the whole dataset to MP3 and then converted everything back to WAV. This process did not affect the final accuracy, which may suggest that, for our synthetic speech detection problem, the MP3 compression does not affect in the classification process.

A.2 Volume Normalization

Normalizing the volume in an audio dataset is a common practice in machine learning research, since inconsistent volume levels can impact on learning and classification. As the files were collected from several data sources, each one with their own volume settings, it is important to normalize the volume of all utterances to eliminate the possibility of volume becoming a distinguishing factor. Using the SoX tool[47], all the audio files were normalized to 0dBFS.

[46] https://www.ffmpeg.org/

[47] http://sox.sourceforge.net/

A.3 Sample Rate Normalization

The sample rate is an important factor when training a machine learning algorithm: all the input audio should be at the same sample rate to ensure the audio is processed correctly. The majority of the audio files had 16kHz as sample rate, but there were also files recorded at 22kHz, 24kHz and 48kHz. Since the human voice frequency spectrum typically ranges from 300Hz up to 5000Hz, using 16kHz as default sample rate provides enough room for the task, since it can accurately represent audio signals up to 8000Hz.

Using the SoX tool[48] in conjunction with a custom script, the whole dataset was downsampled to 16kHz.

A.4 Channel Mixing

All the synthetic speech solutions generate audio in a single channel (mono), while a good fraction of the real utterances were recorded in two channels (stereo). To avoid this becoming a distinguishing factor, all the audio files were converted to mono using the SoX tool[49]. This tool uses a channel mixing technique, which combines two audio tracks into a mono track by scaling each track by 0.5 and adding the signals to result in a single track.

A.5 Silence Removal

During early experiments with our dataset, it was noted that one distinguishing factor between real and synthetic speech was the beginning and the end of each file. After manual analysis, it was noted that synthetic audio always had around 0.2 seconds of silence in the beginning and in the end of the utterance, while in real utterances this silence was longer in most of the cases. This means that one of the classifying factors that a neural network could incorrectly learn is the amount of silence in the beginning and end of the file. As we want the neural network to learn the real differences between synthetic and real audio, it was necessary to remove the silence in the beginning and end of all the files.

The SoX tool has a feature which allows to remove the silence in the beginning of an audio file. To automate the process, a script was created to automate the silence removal in the whole dataset. As SoX does not have a feature to remove the silence in the end of the file, first we had to reverse the audio file (also using the

[48] http://sox.sourceforge.net/

[49] http://sox.sourceforge.net/

SoX tool), then cut the silence from the beginning, then reverse the audio again. As a result of this processing step, all the files have no silence in the beginning nor in the end of the audio.

Appendix B: Frequency Analysis - Additional Experiments

In this appendix chapter, we discuss the additional experiments performed related to frequency analysis. We start by presenting an interesting analysis using data from real and synthetic utterances of the same voice, showing that the speech synthesis process does affect frequency information of the audio. Also, we perform a study to analyze other audio features of real and synthetic audio, such as roughness and brightness.

B.1 Frequency Analysis - Same-voice Classification Analysis

One of the main concerns when performing any kind of machine learning analysis is to ensure that the classifier is learning the real characteristics of the problem instead of any bias in the data. In our research, the main concern is that the classifier may learn features that are not the real differences between real and synthetic audio. To test this hypothesis, we used one of the voices of the real dataset to train one of the synthetic speech generators. This resulted in a temporary dataset where the real voice and the synthesized voice are from the same person, eliminating any voice bias.

The LJSpeech dataset (which is included in our FoR dataset) was selected as source of real voice since it contains clear recordings of a professional speaker. This dataset was used to train a DeepVoice 3 model[50]. The resulting model was able to generate speech with the same voice as the input voice. With this sub-dataset created, the audio representation was extracted (using Librosa STFT 1024 bins) and the data was input into the Weka tool.

Using Random Forests, the accuracy achieved is 95.63%, showing that even though the voices are the same, the synthesizing process does affect the overall frequency spectrum. Upon inspection of the generated audio it is possible to note differences on the low and high frequencies of the synthetic utterances: High fre-

[50] https://github.com/r9y9/deepvoice3_pytorch

quencies are more noticeble in real audio, while low frequencies are more evident in synthetic speech. This finding confirms the observations in the activation map experiment, where it was noted that the main differences are in lower and higher frequencies.

B.2 Timbre Model Analysis

From manual inspection of the real and synthetic utterances it is possible to observe a substantial difference in the audio features, such as brightness and roughness. To test this observation, we extracted measurements for four main timbre models based on the AudioCommons standards [2]:

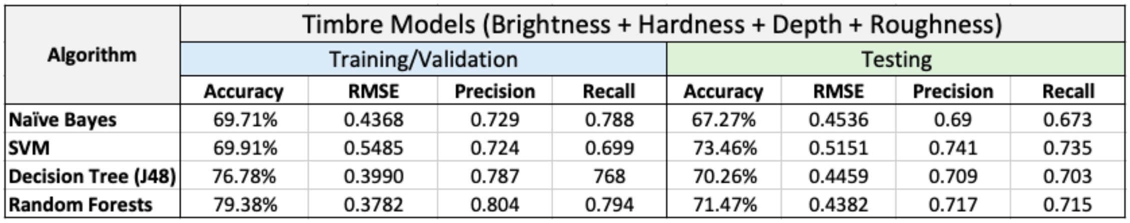
- Brightness: A bright sound is one that is clear/vibrant and/or contains significant high-pitched elements.
- Hardness: A hard sound is one that conveys the sense of having been made by something solid, firm or rigid; or with a great deal of force.
- Depth: A deep sound is one that conveys the sense of having been made far down below the surface of its source.
- Roughness: A rough sound is one that has an uneven or irregular sonic texture.

Those timbre models were extracted using the Audio Commons[51] tool, which is able to generate scores for each of the above mentioned features and much more.

With the timbre models extracted for each utterance in the dataset (unbalanced for-norm), the data was then input into Weka for analysis. First, we evaluated how well would a classifier perform if only those four features were provided. Figure 5.2 shows the results of this experiment. Using Random Forests we achieved 79.38% accuracy in the validation dataset and using SVM 73.46% accuracy in the testing dataset (which contains unseen algorithms). These numbers show that although they are not the best classification attributes, timbre models are statistically different in real utterances and synthetic utterances.

The next step was to understand which of the four audio features was more effective in the differentiation between real and synthetic utterances. To analyze that, both Chi-Square and Information Gain techniques were utilized. As seen in Figure 5.3, the depth of the audio is the main differentiator between real and synthetic audio. The average depth score for synthetic utterances is 43.84 (standard

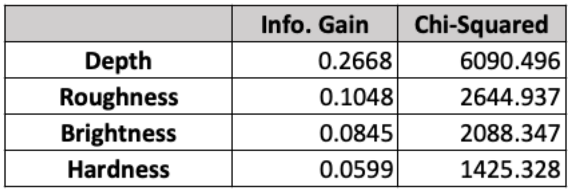
[51] https://www.audiocommons.org/2018/07/15/audio-commons-audio-extractor.html



Algorithm	Timbre Models (Brightness + Hardness + Depth + Roughness)							
	Training/Validation				Testing			
	Accuracy	RMSE	Precision	Recall	Accuracy	RMSE	Precision	Recall
Naïve Bayes	69.71%	0.4368	0.729	0.788	67.27%	0.4536	0.69	0.673
SVM	69.91%	0.5485	0.724	0.699	73.46%	0.5151	0.741	0.735
Decision Tree (J48)	76.78%	0.3990	0.787	768	70.26%	0.4459	0.709	0.703
Random Forests	79.38%	0.3782	0.804	0.794	71.47%	0.4382	0.717	0.715

Figure 5.2: Timbre Model Analysis

deviation 5.62) while for real utterances is 45.11 (standard deviation 9.35), which shows that real speech is, on average, deeper than synthetic speech.



	Info. Gain	Chi-Squared
Depth	0.2668	6090.496
Roughness	0.1048	2644.937
Brightness	0.0845	2088.347
Hardness	0.0599	1425.328

Figure 5.3: Timbre Models Information Gain

B.3 Overall Results

To summarize the experiments of this section, we create a diagram, shown in Figures 5.4, presenting all the additional experiments described in this appendix. From those results it is possible to observe and compare the impact of the experiments in the overall accuracy.

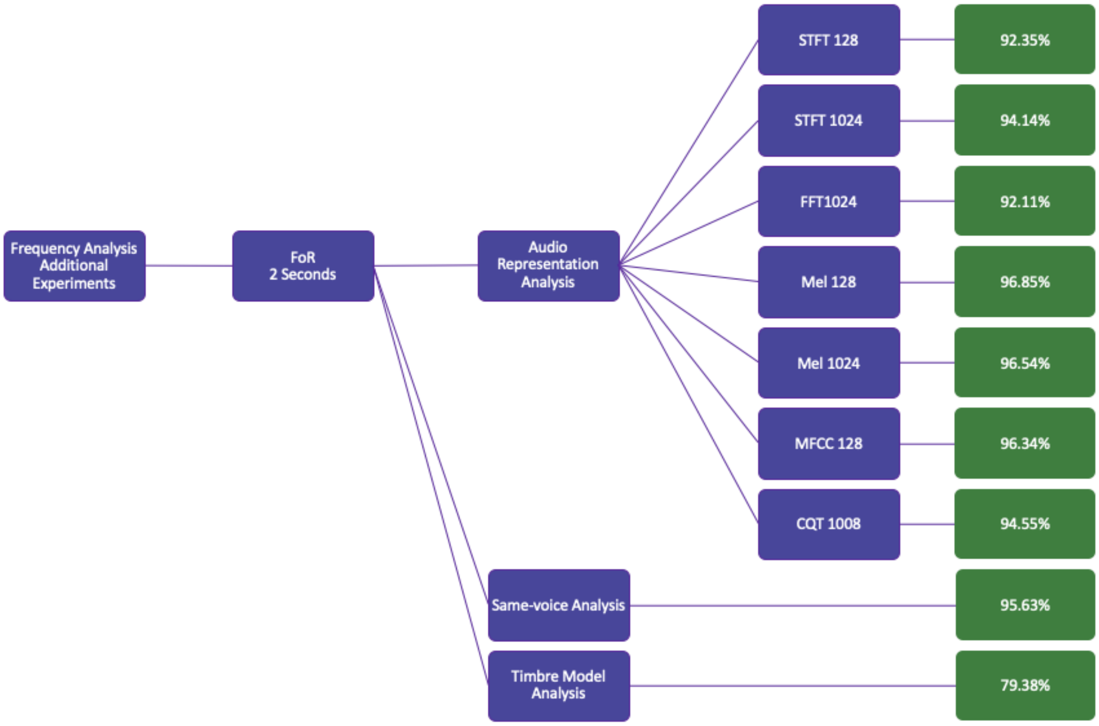


Figure 5.4: Additional Frequency Analysis Experiments Results

Appendix C: Deep Learning - Additional Experiments

In this appendix chapter, we discuss the additional experiments performed related to deep learning methodologies. We explore the possibility of using only waveform images for classification instead of spectrograms. After that, we apply several frequency filters to the audio to observe the impact on the accuracy. Also, to simulate a real world scenario, we investigate the impact of noise on the classifier performance. Last, we investigate the detection of a real-world attacks using the re-recorded dataset.

C.1 Waveform Classification

As seen in the Experiments section, the accuracy using spectrograms is very high. This prompts the idea that simpler audio representations, such as waveform images, may be enough for the classification of real and synthetic utterances. Figure 5.5 shows an example of a waveform from a synthetic utterance as well as a real utterance.

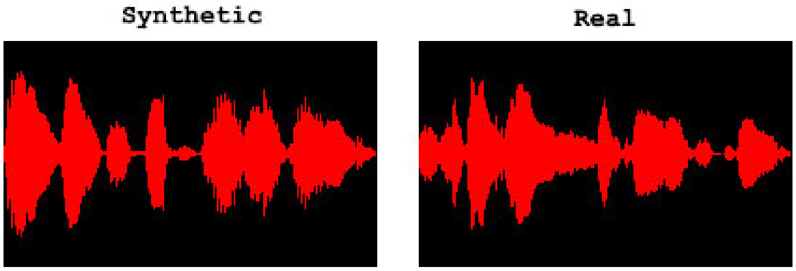


Figure 5.5: Waveform Example

For this experiment, we use the gender-unbalanced 2-second version of the dataset (for-2seconds). The ffmpeg tool[52] was used to convert the audio files into

[52] https://www.ffmpeg.org/

waveforms. With that done, the resulting images were then input into the VGG19 model for training and validation.

Using purely the waveform image with the VGG19 model, we achieved 89.79% accuracy on the validation dataset. This shows that although it is possible to classify using only waveforms, spectrograms provide a much higher accuracy in synthetic speech detection.

C.2 Dynamic Range Compression Analysis

To ensure that the classification is not being made due to volume variations, we analyze the use of dynamic range compression (DRC) in the dataset. DRC reduces the volume of loud sounds and amplifies the quiet sounds to reduce the dynamic range of the audio. An example of dynamic range compression can be seen in Figure 5.6, where a high amount of compression is applied to an utterance.

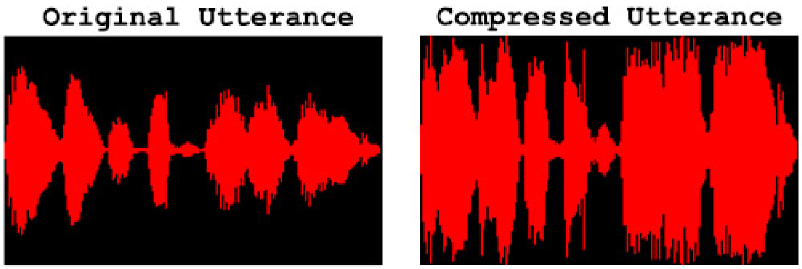


Figure 5.6: Example of Dynamic Range Compression

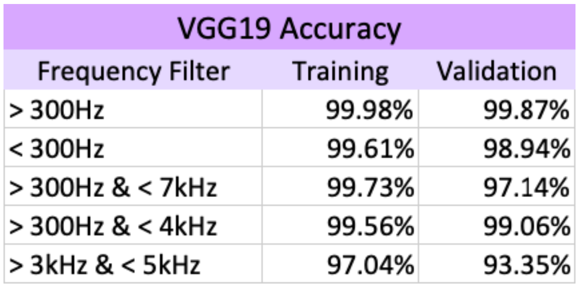
For this experiment, we use the gender-unbalanced 2-second version of the dataset (for-2seconds). Then, using the SoX tool we apply compression to the utterances using the following command: *sox input.wav output.wav compand 0.01,1 6:-70,-60,-0.1 5 -90 0.2*. The effect of this compression can be seen in Figure 5.6, where the lower parts of the utterances were amplified while keeping the loud parts intact. This procedure was applied to every utterance in the dataset.

The compressed dataset was transformed into waveform images (as in Section) that were used to train a VGG19 model. This model achieved a validation accuracy of 89.15%, which is slightly lower than the original waveform performance (89.79%). This result shows that volume discrepancies have little-to-none affect in the results presented.

C.3 Frequency Filter Analysis

As in the Experiments section we identified that high frequencies are the most important in the classification phase, it was decided to test if without those frequencies the model would learn different features. To test this hypothesis, we applied a variety of low-frequency and high-frequency cut filters to analyze the impact on the accuracy.

For this experiment, we use the gender-unbalanced version of the dataset cropped at two seconds (for-2seconds). Then, using the SoX tool we generated the spectrograms for the dataset. With the spectrograms created, we cropped the spectrogram image horizontally in different sections to simulate frequency filters. Figure 5.7 shows the list of filters applied and their accuracy using the VGG19 architecture and STFT audio representation.



VGG19 Accuracy		
Frequency Filter	Training	Validation
> 300Hz	99.98%	99.87%
< 300Hz	99.61%	98.94%
> 300Hz & < 7kHz	99.73%	97.14%
> 300Hz & < 4kHz	99.56%	99.06%
> 3kHz & < 5kHz	97.04%	93.35%

Figure 5.7: Frequency Filter Analysis

As seen in Figure 5.7, applying filters reduce the accuracy of the model. However, the performance is still high. This shows that the neural network was able to learn different features of the audio. To test this hypothesis, we generated the Average Classification Activation Maps for the utterances with low cut at 300hz and high cut at 4kKz. The resulting ACAM can be seen in Figure 5.8, in which it is possible to observe that the neural network learned features from different parts of a spectrogram.

C.4 Signal/Noise Ratio Analysis

An important factor widely analyzed by previous research in synthetic speech detection is the relation between noise and model accuracy. The idea is to investigate how noise impacts on the accuracy of the model by adding a variety of levels of pink noise to the utterances and observing the model performance.

For this experiment, we use the gender-unbalanced version of the dataset cropped at two seconds (for-2seconds). Then, using the SoX tool we apply pink noise in a

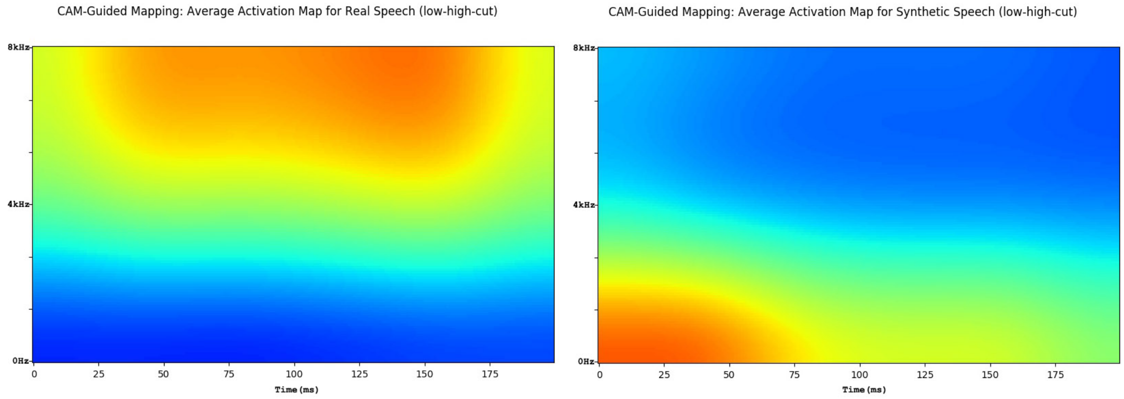
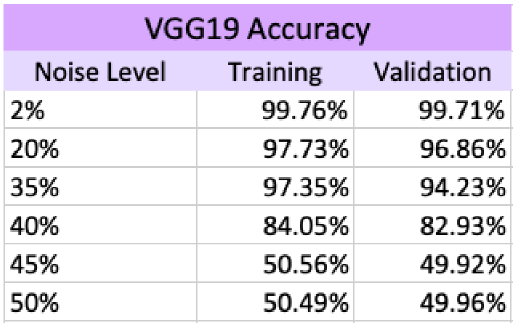


Figure 5.8: Averaged CAM for frequency-filtered spectrograms

variety of volume levels, from 2% to 50% of the resulting audio. This results in six sub-datasets, each one related to a level of noise. We then use each of the six sub-datasets to train an independent VGG19 model (with STFT audio representation). The noise levels and related accuracies can be seen in Figure 5.9.



VGG19 Accuracy		
Noise Level	Training	Validation
2%	99.76%	99.71%
20%	97.73%	96.86%
35%	97.35%	94.23%
40%	84.05%	82.93%
45%	50.56%	49.92%
50%	50.49%	49.96%

Figure 5.9: Noise ratio and accuracy on VGG19

Figure 5.10 shows the data plotted into a graph in which it is possible to observe that the higher the noise level, the lower the accuracy is. It is also possible to note that up to 35% noise volume the accuracy of the model is still high, showing that the architecture is fairly robust against noise. With a noise volume higher than 40% the accuracy starts to drastically decrease. When the noise level is equal or higher than 45%, the VGG19 is not able to distinguish between synthetic or real. It is important to note that a 45% noise ratio generates a poor quality utterance, making it hard even for humans to distinguish between a real and a synthetic utterance.

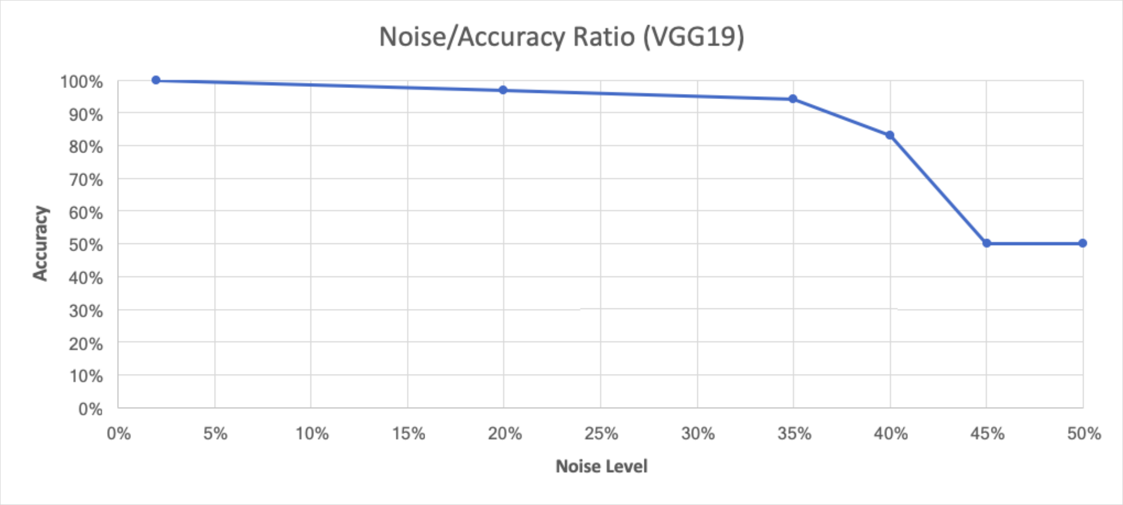


Figure 5.10: Noise ratio and accuracy chart

C.5 Overall Results

To summarize the experiments of this section, we create a diagram, shown in Figures 5.4, presenting all the additional experiments described in this appendix. From those results it is possible to observe and compare the impact of the experiments in the overall accuracy.

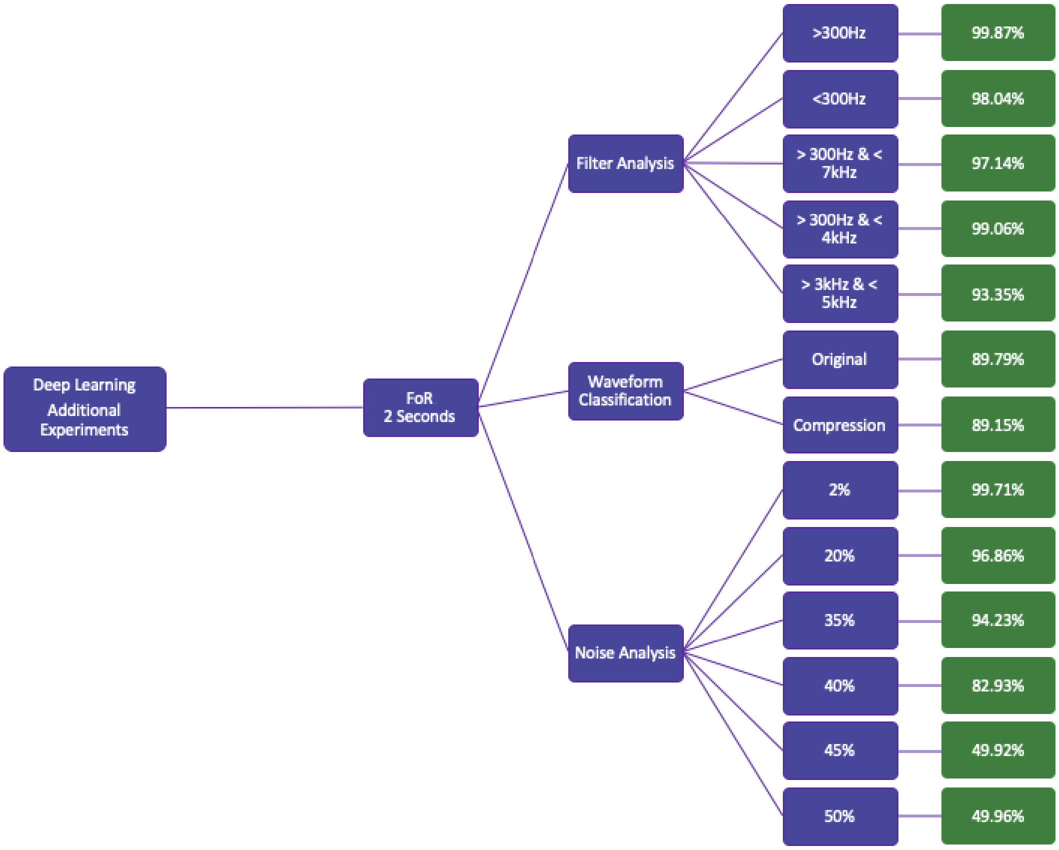


Figure 5.11: Additional Deep Learning Experiments Results