Detecting Voice DeepFakes of Specific Speakers

Using Feature-Based and Transformer Models

Lucas T. Groot
14638274

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 900
1098 XH Amsterdam

Supervisor
Prof. dr. P.P.G. (Paul) Boersma

Phonetic Sciences
Faculty of Humanities
University of Amsterdam
Spuistraat 134
1012 VK Amsterdam

Semester 2, 2024-2025
Submission date: June 27, 2025

Abstract

DeepFake speech poses a growing threat to systems that rely on voice or voice recognition. As a result, the detection of voice DeepFakes and the development of spoofing countermeasures have received significant attention in literature. This study presents a comparison between traditional detection approaches, which involve extracting features from audio and training a detection model on these features, and a state-of-the-art transformer-based method (wav2vec2). Experiments were conducted using the WaveFake dataset. We found that transformer-based methods consistently outperform traditional feature-based approaches, both in terms of generalization to unseen DeepFake models and effective recognition of seen DeepFake models. Additionally, we found that among the seven DeepFake models in the dataset, the detection models performed poorest on HiFiGAN, both when this model was present in the train set and when it was not, possibly leading to the conclusion that this model creates the highest quality DeepFakes present.

Contents

Chapter 1

Introduction

Advancements in technology have made it increasingly difficult to distinguish between real and fake content on the internet. One notable development in this area is the rise of "DeepFakes": a term derived from *deep learning*−an AI method used to analyze and replicate patterns, and *fake*−indicating the synthetic nature of the content. DeepFake technology enables the creation of highly realistic audio and video, often making it challenging for humans to discern (Chadha et al., 2021). The negative implications of such a technology have already been observed, for instance, in 2019 then Speaker of the House Nancy Pelosi was featured in a manipulated video in which she appeared incoherent (Lee, 2019).

The goal of this research is to detect DeepFake speech. Earlier efforts have already been made to accomplish this, but these studies often used datasets with generic text-to-speech models that did not replicate the voice of any particular person. Kühne et al. (2020) found that humans rated text-to-speech voices significantly differently to human voices in terms of several characteristics, like eeriness. This indicates that the fake and real voices were quite easy to distinguish, even for humans. Examples of research with generic text-to-speech data include the works of Hamza et al. (2022) and Khochare et al. (2021): Both of these studies utilize the Fake or Real (FoR) dataset, in which fake speech samples are generated using generic text-to-speech models.

Of particular interest, however, are cases in which the speech of a specific individual is manipulated. This is often a high-profile individual, like in the case of Nancy Pelosi discussed previously. This is why in this research I propose using the WaveFake dataset: This dataset was created using several state-of-the-art DeepFake models trained on the LJSpeech dataset, which consists of 13,100 short audio clips (on average 6 seconds each; approximately 24 hours total) read by a single female speaker (Frank & Schönherr, 2021). Since this dataset is based on the speech of an actual person, it can serve as a more accurate benchmark for the effectiveness of DeepFake speech detection systems in real-world scenarios.

1.1 Background and Related Work

1.1.1 DeepFake Speech

DeepFake speech refers to audio in which a speaker's voice or message has been artificially generated or altered by AI. Recent DeepFake systems produce such high quality voice that they are indistinguishable from real speech (Wang et al., 2020). This capability enables convincing voice cloning and raises security concerns: attackers can impersonate public figures or private individuals with malicious intent (Reimao & Tzerpos, 2019). In fact, criminals have already exploited this threat: one case involved synthesized audio used to mimic a CEO's voice and authorize a $240,000 fraudulent transfer (Stupp, 2019). This makes robust detection of AI-generated speech critical, yet challenging due to the realism of modern DeepFakes. As a result, research on developing reliable detectors has intensified in recent years (Yi et al., 2023).

1.1.2 Datasets

To facilitate development of detection methods, researchers have compiled specialized datasets of synthetic and real speech. For example, early anti-spoofing challenges like ASVspoof and ADD provided benchmark datasets to evaluate DeepFake detection methods (T. Yang et al., 2025). Additional datasets have also emerged. The Fake-or-Real (FoR) corpus contains over 87,000 synthetic utterances paired with 111,000 genuine recordings collected from diverse open sources (Reimao & Tzerpos, 2019). Similarly, the WaveFake dataset provides tens of thousands of AI-generated clips produced by multiple state-of-the-art neural architectures and is accompanied by a dataset of respective real speech (Frank & Schönherr, 2021). These resources serve as important benchmarks for training and comparing detection algorithms.

1.1.3 Detection Approaches

Prior detection approaches have ranged from classical machine learning on hand-crafted features to end-to-end neural architectures. A common strategy is to extract acoustic descriptors—such as short-time spectral or cepstral features (MFCC, CQCC, etc. (Todisco et al., 2017)) and prosodic metrics (pitch, energy, duration)—that may capture artifacts of speech synthesis (Chadha et al., 2021). Early detectors typically used these features with Gaussian mixture models or support vector machines (Yi et al., 2023). In contrast, recent state-of-the-art systems utilize deep neural networks: convolutional or residual networks trained on spectrograms (or even raw waveforms) show promising results (Yi et al., 2023). Light CNNs and ResNet variants have been widely adopted as baselines in recent ASVspoof and ADD competitions (Yi et al., 2023). Transformers are a newer type of neural network and also show promising results, offering robust and accurate DeepFake detection (Goyal et al., 2025).

Chapter 2

The dataset

The foundation of the WaveFake dataset is the LJSpeech dataset. LJSpeech is a public dataset consisting of 13,100 short audio clips, which are 6 seconds in length on average, totaling around 24 hours. These audio clips consist of excerpts from seven non-fiction books read by a single female speaker and were captured using the built-in microphone of a MacBook Pro (Frank & Schönherr, 2021).

Raw audio has a very high temporal resolution—generally the sampling rate is at least 16000 Hz. Raw audio also has short-term (e.g., phonemes) and long-term (e.g., prosody) patterns, which makes it challenging to model (Kumar et al., 2019). Because of this, the usual practice for generating DeepFakes is to extract some low-dimensional intermediate representation from the provided text. This intermediate representation often consists of linguistic features or mel spectrograms (Frank & Schönherr, 2021). Then, an additional model, often called a vocoder, maps this intermediate representation to raw audio.

WaveFake extends the LJSpeech dataset by generating DeepFake versions of each clip using seven state-of-the-art vocoder models: MelGAN, Large MelGAN, Parallel WaveGAN (PWG), Multi-band MelGAN (MB-MelGAN), Full-band MelGAN (FB-MelGAN), HiFi-GAN, and WaveGlow (Frank & Schönherr, 2021). These models are briefly described in the following section.

2.1 DeepFake models used

The dataset includes six DeepFake vocoder models based on Generative Adversarial Networks (GANs) (Frank & Schönherr, 2021). GANs consist of two neural networks—a generator and a discriminator—which are trained in opposition to each other. The generator attempts to create realistic synthetic data, with the goal of fooling the discriminator. The discriminator simultaneously learns to distinguish between real and synthetic inputs (Creswell et al., 2018). This adversarial training allows GANs to produce highly convincing DeepFakes.

The GANs used by Frank and Schönherr (2021) to create the DeepFakes are:
MelGAN: a GAN-based model that uses a fully convolutional network consisting of transposed convolutional layers to upsample the inputs, which are mel spectrograms. It uses a multi-scale architecture as the discriminator, D_1 operates on the scale of raw audio, and D_2 and D_3 operate on downsampled versions of the raw audio, having a downsampling factor of 2 and 4 respectively. This downsampling is performed to detect patterns at different scales. This downsampling is also achieved by convolutions (Kumar et al.,

2019).
Large MelGAN: a larger version of MelGAN described above, including a larger receptive field, which means that the region of the input on which an output unit depends is larger (Luo et al., 2016).
Parallel WaveGAN (PWG): A GAN-based model that uses convolutional layers to upsample the inputs, similar to MelGAN. However, instead of relying solely on time-domain waveform differences, it evaluates spectral consistency using STFTs computed at different resolutions (i.e., varying window sizes and hop lengths). This allows it to capture crucial speech features such as formants, thereby potentially improving the quality of DeepFake synthesis (Yamamoto et al., 2020).
Multi-band MelGAN (MB-MelGAN): An enhanced version of MelGAN that improves waveform generation speed and quality. It decomposes the audio into multiple frequency bands, processes each band separately, and then combines them to reconstruct the full audio. This approach reduces computational complexity and enhances the model's ability to capture fine-grained spectral details (G. Yang et al., 2021).
Full-band MelGAN (FB-MelGAN): A variant of MB-MelGAN that removes the sub-band decomposition, generating full-band audio directly. Unlike MelGAN, it incorporates multi-scale STFT losses (similar to Parallel WaveGAN) to improve audio fidelity by comparing generated and real waveforms at multiple resolutions (G. Yang et al., 2021).
HiFi-GAN: A model that uses a fully convolutional neural network as the generator and includes two discriminators, namely a multi-scale and multi-period discriminator. The multi-period discriminator captures audio patterns at different temporal scales using multiple sub-discriminators, each processing the input based on a distinct non-overlapping periodic reshaping of the waveform achieved by convolutions. The multi-scale discriminator solves the non-overlapping property of the multi-period discriminator by consecutively evaluating the entire audio sequence (Kong et al., 2020).

The dataset also includes one flow-based DeepFake model:
WaveGlow: WaveGlow is a flow-based generative model designed for high-quality speech synthesis from mel spectrograms. Unlike GAN-based models, WaveGlow employs invertible and differentiable transformations, enabling exact likelihood computation. This approach allows for efficient training using a single cost function: maximizing the likelihood of the training data. (Prenger et al., 2019).

The DeepFakes appear to have been generated using pre-trained versions of the HiFiGAN and WaveGlow models mentioned above (Frank & Schönherr, 2021), whereas the remaining DeepFake models are not explicitly mentioned to be pre-trained (Frank & Schönherr, 2021). The DeepFake models appear to be trained on the entire LJSpeech dataset, meaning the goal is recreating the training distribution (Frank & Schönherr, 2021). This should generate high-quality audio, but also comes with a risk of overfitting. However, since these models primarily follow a GAN structure, the generator only interacts with the discriminator, meaning that the real audio is never directly seen by the model (Frank & Schönherr, 2021). Therefore, the risk of (almost) exact replication is small, which is also apparent when looking at characteristics of the generated speech, e.g., average pitch is clearly lower in the DeepFake samples (Frank & Schönherr, 2021), which supports the fact that these models did not produce exact replications. WaveGlow also shows this difference, which is the only non-GAN model.

Chapter 3

Feature-based method

A strategy for detecting DeepFakes often present in literature is extracting features from the audio and training a machine learning model on these features (Almutairi & Elgibreen, 2022). This strategy will also be implemented in this research. However, before being able to implement such a strategy we must make sure that the audio from which these features will be extracted is of equal length. This is because an increase in signal length also comes with an increase in extracted features: many features are extracted using sliding windows, and longer audio contains more of these windows. In the LJSpeech dataset (and, consequently, the WaveFake dataset), the lengths of the clips vary widely. The shortest clip is 1.1 seconds long and the longest clip is 10.1 seconds long.

A variable size feature vector is not suitable for training, to solve this problem only the first 2 seconds of the clips will be included, which is also in line with the work of Khochare et al. (2021). Clips with a length of less than 2 seconds will be withheld from the data.

Features found to be effective for discerning between real and DeepFake audio in a number of papers will be included and these features are summarized here:

MFCCs: A feature found to be effective in several articles was MFCCs (Hamza et al., 2022)(Bird & Lotfi, 2023). MFCCs mimic human hearing by operating on a log scale (Hamza et al., 2022), possibly enabling a model to be more tuned to how humans perceive and create speech.

Spectral Features: Spectral features were also found to be effective in recognizing DeepFakes (Hamza et al., 2022)(Khochare et al., 2021). Spectral features attempt to summarize information about how energy is distributed across frequencies (Tzanetakis, 2011) using several aspects, such as roll-off point, centroid and bandwith. These will be included in the model.

The roll-off point is the frequency R_n below which 85% of the energy distribution of the magnitude spectrum is concentrated (Tzanetakis, 2011).

The spectral centroid is the center of gravity of the magnitude spectrum and is defined as (J. Li et al., 2024):

$$f_{centroid} = \frac{\sum_{n=0}^{N-1} f(n) * X(n)}{\sum_{n=0}^{N-1} X(n)}$$

Where $f(n)$ represents the frequency bins in the frequency domain and X(n) represents the corresponding magnitudes (J. Li et al., 2024).

The spectral bandwith quantifies the spread of a signal's spectrum around its centroid; a high bandwidth means energy is spread widely across frequencies (Tzanetakis, 2011).

Root Mean Square Energy: Root Mean Square Energy is a measure of signal loudness, which is also a feature that has found to be effective (Khochare et al., 2021). Root Mean Square Energy is defined as (Khochare et al., 2021):

$$x_{rms} = \sqrt{\frac{1}{n}(x_1^2 + x_2^2 + \ldots + x_n^2)}$$

Although the clips are not calibrated for loudness, the recording conditions remain similar, which might make this feature useful for classification.

Jitter and Shimmer Features: Jitter and shimmer can be a measure of speech quality and these features have been shown to be effective for DeepFake speech detection (K. Li et al., 2023). Jitter and shimmer are visualized below: jitter measures perturbations in the term or frequency (pitch) of the soundwave, while shimmer measures perturbations in the amplitude or loudness of the soundwave (K. Li et al., 2023).

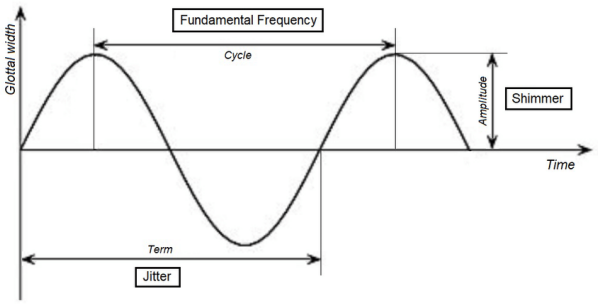


Figure 3.1: Visualization of Jitter and Shimmer from ResearchGate[1]

The features will be extracted using the OpenSMILE library, which is explained in the following section.

3.1 OpenSMILE

The openSMILE (Speech & Music Interpretation by Large-space Extraction) toolkit is a widely used open-source framework for audio feature extraction. It implements a broad library of low-level descriptors (LLDs) – including spectral and cepstral measures (e.g. MFCCs, PLP coefficients), prosodic features (pitch, energy/loudness), formant frequencies, and other voice-quality metrics – and can compute first-order derivatives (deltas) as well as a variety of higher-level transformations (Eyben et al., 2010).

Most importantly, openSMILE can apply statistical functionals (e.g. means, variances, percentiles, peak times) to LLD trajectories to produce utterance-level summary features. The toolkit is written in optimized C++, which enables fast and efficient computation of the features (Eyben et al., 2010).

Its flexibility and comprehensive feature set have made it a widely used toolkit in speech and paralinguistic research, e.g., openSMILE served as the official feature extractor in INTERSPEECH challenge tasks (Eyben et al., 2010).

[1]https://www.researchgate.net/figure/Graphical-representation-of-F-0-Jitter-and-Shimmer_fig7_321351129

3.1.1 The ComParE 2016 featureset

A prominent openSMILE configuration is the INTERSPEECH 2016 Computational Paralinguistics (ComParE) Challenge feature set. This is an especially large "brute-force" paralinguistic feature collection. The ComParE 2016 set produces 6,373 utterance-level features (J. Chen et al., 2021). These are derived by first extracting 65 acoustic LLDs (covering energy, spectral and voicing characteristics, for example, MFCC bands, spectral flux/entropy, F0 (pitch), intensity, jitter/shimmer, etc.) and their 65 first-order deltas (audEERING, 2016b). By default these LLDs are computed on overlapping windowed frames of 20 ms length with a 10 ms step size (audEERING, 2016b), which is a common setting for speech analysis. After frame-level extraction, a large battery of statistical functionals (including means, standard deviations, percentiles, linear regression coefficients, and contour-related measures such as rise/fall times of peaks) is applied to each LLD and delta track (audEERING, 2016a). The result is a high-dimensional feature vector that encapsulates broad acoustic patterns over the entire utterance. Because of its exhaustive coverage, the ComParE feature set is able to be used in a broad range of tasks (Schuller et al., 2016).

3.1.2 Selecting from the ComParE 2016 featureset

We will select the features mentioned at the start of section 3 from the ComParE 2016 featureset. This is done by, for example, selecting all features which contain "MFCC" for selecting the MFCC features. We thereby reduce the total amount of features from 6373 to 2300 features. The total amount of instances of each of the selected features is summarized here:

Feature Type	**Amount of Features (ComParE 2016 Feature Set)**
MFCC	1400
Spectral	566
RMS Energy	100
Jitter & Shimmer	234
Total	**2300**

Table 3.1: Feature breakdown from the ComParE 2016 feature set

3.2 Models

We will use several machine learning models to evaluate and compare performance on our dataset for the feature-based method. These models have been chosen because of their effectiveness in earlier studies. The selected models include XGBoost (Hamza et al., 2022), Random Forest (Khochare et al., 2021) and a slightly older model: Support Vector Machine (SVM) (Yi et al., 2023).

3.2.1 XGBoost

XGBoost (Extreme Gradient Boosting) is a gradient tree boosting algorithm, meaning it builds an ensemble of weak learners (decision trees) in a sequential manner where each subsequent model attempts to correct the errors of its predecessor (Natekin & Knoll,

2013). XGBoost achieves state-of-the-art results in many machine learning challenges. XGBoost also includes many optimization techniques, in order to create a scalable algorithm (T. Chen & Guestrin, 2016).

3.2.2 Random Forest

Random Forest is an ensemble learning method that generates many decision trees, each created with a random subset of the columns. The final class prediction for an instance is the most predicted class, i.e., the mode of the predictions made by the decision trees (Biau & Scornet, 2015).

3.2.3 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning algorithm that constructs a hyperplane in a high-dimensional space to separate different classes. SVMs can use kernel functions to handle non-linear classification tasks, making them versatile and powerful for complex datasets (Suthaharan, 2016).

Chapter 4

Transformer method

Transformers are a newer type of neural network based on attention layers. Transformers were introduced in 2017 and have proven to be effective on a wide range of challenging tasks (Vaswani et al., 2017). wav2vec2 is a self-supervised transformer and has demonstrated excellent performance, even with limited labeled data, thus making it well-suited for our problem (Baevski et al., 2020).

Wav2vec2 takes raw audio waveforms as input and uses a Convolutional Neural Network (CNN) to extract representations from this raw audio. CNNs are especially powerful, since they make strong assumptions about the nature of the audio, for instance, that audio in close vicinity matters more to local structure than distant audio. After this, a transformer takes these representations as input and processes them to produce contextualized representations that capture information from the entire sequence. This is achieved through a Transformer-based network that models dependencies across the entire sequence of latent representations by using attention layers (Baevski et al., 2020). The architecture is visualized below:

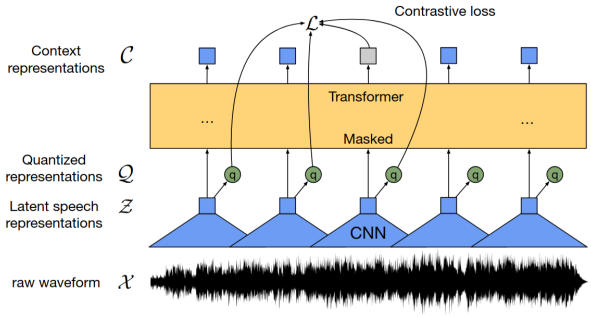


Figure 4.1: Architecture of wav2vec2 (from original paper)

During fine-tuning for downstream classification tasks, a classification head is added. The CNN part of wav2vec2, also known as the feature extractor, is usually frozen (Conneau et al., 2020), meaning that this part of the model will not be trained further.

Chapter 5

Results

The data we will use is very imbalanced: For each real instance in the LJSpeech dataset there are 7 DeepFake instances in the WaveFake dataset. If we train on such data, we risk creating a model that only classifies audio as DeepFake, since this would still result in a very high percentage of correct predictions.

This problem has been solved by making perfectly balanced datasets with a subset of the data, each of these subsets having the goal of providing a fair metric for the two evaluation methods described below. Having a perfectly balanced dataset will also prevent the need of using more complicated evaluation metrics that can account for unbalanced datasets, such as F1-score. The main evaluation metric used will be the simple accuracy score, defined as the ratio of correctly classified instances to the total number of instances.

Using a real clip in the train set and a DeepFake version of the clip in the test set or vice versa is problematic: if you know the clip in the train set is real/fake you can infer that the clip in the test set is fake/real. Whether models actually have this capability is questionable, but it nonetheless seems unsound. Therefore, if a real version of the clip is in either the train or test set, the DeepFake version will also be in that same set.

Two evaluation methods will be implemented: A model-seen test, meaning that the DeepFake models in the test set also occur in the train set, and a leave-one-model-out test, meaning that the Deepfake model in the test set does not occur in the train set. This will allow testing of both the accuracy of representation of seen DeepFake models and the generalization of our detection models to unseen DeepFake models. It will simultaneously allow us to test to what degree the DeepFake generation methods are resistant to generalization and representation.

We chose not to shuffle the LJSpeech-based WaveFake dataset prior to splitting. This preserves the natural ordering of the source material. This ordering corresponds to meaningful structure, such as progression through book chapters, books, or recording session continuity. By maintaining this sequence, we aim to evaluate model performance in a setting that more closely resembles real-world conditions, where temporal or thematical consistency may influence detection. This approach also ensures that a sentence in the test set is not directly adjacent to a sentence in the train set, which would be very likely when shuffling, since the excerpts are segmented by simply splitting on silence (Ito, 2017). Adjacency of sentences might lead to unrepresentative levels of performance, since the "tells" differentiating between real and DeepFake audio might be overly similar.

5.1 Model-seen results

The first testing that will be done is model-seen testing, meaning that all DeepFake methods occur in both the train and test sets. This will be done by making 10 separate datasets for cross-validation. The exact method to create these datasets is described here:

1. For each of the real clips, randomly select one DeepFake replica to also be included in the dataset.
    - For the first two datasets, the first 20% of the dataset will be the test set.
    - For the second two datasets, the second 20% of the dataset will be the test set.
    - Continue this for all the 10 datasets (There are 5 test sets in total).
2. We report average performance across these splits.

5.1.1 Feature-based method

We first begin by testing the feature-based method using the features and detection models described in Chapter 3. As mentioned, performance is averaged across the ten splits. We also report performance across vocoder models. This performance is calculated as the accuracy across real/DeepFake pairs of which the DeepFake method is the same as the vocoder model. Accuracies across splits as well as model distribution can be found in Appendix A.

Model	**Total Acc.**	**Mel GAN**	**MelGAN (L)**	**MB-MelGAN**	**FB-MelGAN**	**PWG**	**HiFi GAN**	**Wave Glow**
SVM (RBF)	0.5204 ±0.0053	0.5337 ±0.0119	0.5261 ±0.0134	0.5212 ±0.0078	0.5173 ±0.0092	0.5233 ±0.0133	0.5182 ±0.0098	0.5035 ±0.0116
Random Forest	0.7919 ±0.0084	0.8899 ±0.0245	0.8714 ±0.0215	0.8332 ±0.0203	0.6565 ±0.0202	0.7579 ±0.0139	0.7284 ±0.0194	0.8100 ±0.0119
XG-Boost	0.8483 ±0.0066	0.9210 ±0.0165	0.8981 ±0.0103	0.8939 ±0.0168	0.7238 ±0.0192	0.8261 ±0.0086	0.7984 ±0.0168	0.8787 ±0.0089

Table 5.1: Model-seen accuracies (Including XGBoost with default parameters: max_depth=3, n_estimators=100, lr=0.1)

We can see that XGBoost clearly performs the best out of the 3 detection models. The DeepFake models on which XGBoost performs poorest are FB-MelGAN and HiFiGAN, possibly leading to the conclusion that these create the highest quality DeepFakes. This is supported by the fact that all models were seen in the train set, meaning that the architectural differences among the DeepFake models should make only little difference.

The performance of Support Vector Machine is poor and even very close to random guessing. Since the kernel used (Radial Basis Function) is already the most expressive available, Support Vector Machine will be excluded from any further testing. Random Forest is significantly outperformed by XGBoost across all DeepFake models, leading to the decision that fine-tuning XGBoost is more fruitful for possible performance gains.

Hyperparameter Tuning We will now also find the best set of parameters for the best-performing model (XGBoost). We will do this using a grid search. The grid search will evaluate combinations of the following three hyperparameters:

1. **n_estimators**: which is the number of trees generated by the XGBoost model, with values of [100, 200].
2. **tree_depth**: which is the maximum height (or alternatively depth) of a tree, with values of [3, 5, 7].
3. **learning_rate**: which is the learning rate of the XGBoost model, with values of [0.01, 0.1, 0.2].

This grid search will optimize average accuracy across all models and shifts, since the accumulated accuracy is equivalent to this metric, this will be optimized. The best-performing model for this metric was the cofiguration n_estimators=200, max_depth=3, lr=0.2.

Model	**Total Acc.**	**Mel GAN**	**MelGAN (L)**	**MB-MelGAN**	**FB-MelGAN**	**PWG**	**HiFi GAN**	**Wave Glow**
XG-Boost	0.8586 ±0.00464	0.9278 ±0.01205	0.9085 ±0.01913	0.9095 ±0.01433	0.7287 ±0.02005	0.8454 ±0.01017	0.8097 ±0.01643	0.8831 ±0.00855

Table 5.2: Model-seen accuracies (XGBoost, n_estimators=200, max_depth=3, lr=0.2)

We can see that performance improved slightly for each model and in terms of the total accuracy across the test set, but no significant changes occured.

5.1.2 XGBoost Feature Importances

We will now also plot the 20 most important features for this best-performing model below. We report feature importances in terms of gain, which is the average reduction in loss when the feature is used. In simple terms, this means how much the model improves on average each time the feature is used to split on. We use the term split since XGBoost is a decision tree-based model.

We can see that the 20 most important features are primarily MFCC features, but also contain jitter features, with two of these jitter features being the most important features present. It is also apparent that all of these features are smoothed averages, possibly since this reduces any noise present. We briefly explain both the MFCC features and the jitter features present:

MFCC features: The MFCC features are all similar:

- They are either computed using smoothing averages or their deltas.
- The MFCC's used mostly range between 8 and 12 (out of 14 total)
- They are either LPGain (Linear Predictive Gain) or LPC (Linear Predictive Coding) features.

The Linear Predictive Coding Coefficients (lpc1 and lpc2) and Linear Predictive Gain features are closely related, both being part of the Linear Predictive Model: Linear Predictive Coding models the vocal tract using several coefficients (Sawant et al., 2010).

The gain factor refers to the amount of excitation driving the vocal tract (Sawant et al., 2010). Creating a model of the vocal tract enables the detection of inconsistencies, which appears to be valuable for detecting DeepFake speech.

Jitter features: There are 3 jitter features in this set:

- The first precentile of the smoothed average of jitterDDP.
- The first quartile of the smoothed average of jitterDDP.
- The flatness of the smoothed average of jitterDDP.

JitterDDP refers to difference of differences of periods, a robust measure of pitch instability. It calculates the average absolute difference between the differences of successive pitch periods, and then normalizes this value by dividing it by the average pitch period length over the frame (audEERING GmbH, 2025).

It is clear that the lower percentiles of jitter are an important factor, possibly since DeepFake speech sustains a more constant jitter throughout the entire excerpt than real speech. Jitter flatness is also a factor, which means significant changes in jitter might also differentiate between real and DeepFake audio.

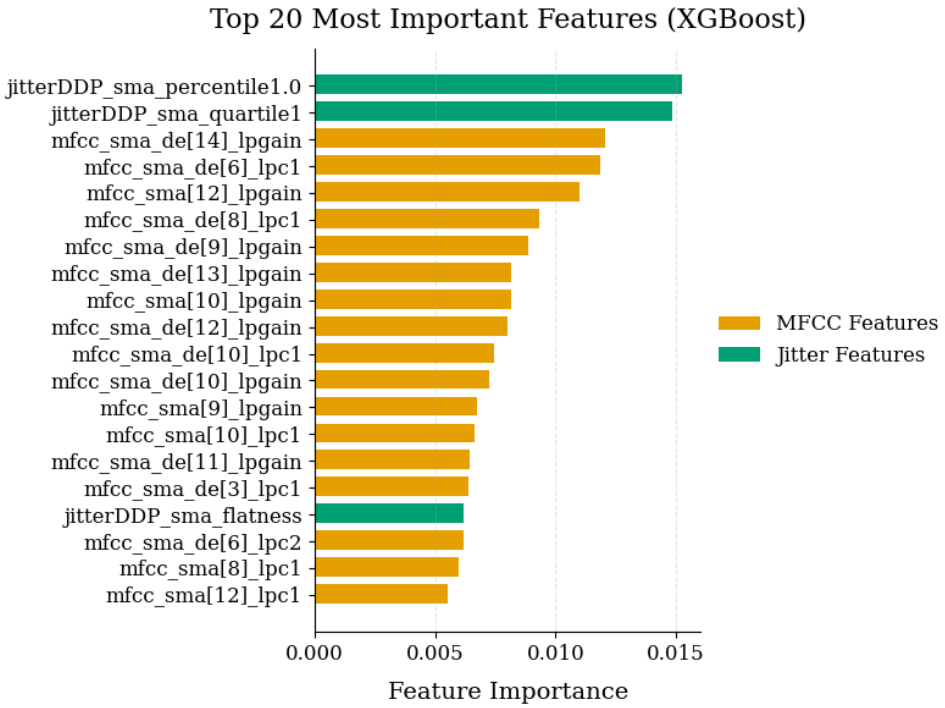


Figure 5.1: Feature importances of best performing XGBoost model

5.1.3 Transformer method

We now also implement the same strategy for the transformer method (using wav2vec2). We use the base wav2vec2 model and only clips of more than 2 seconds are included, which are truncated to 2 seconds. This is mainly done to save time and resources. Fine-tuning was performed for 5 epochs with a learning rate of 10^{-5} The results are summmarized below; again, complete results and DeepFake model distributions can be found in appendix A:

Model	Total Acc.	Mel GAN	MelGAN (L)	MB-MelGAN	FB-MelGAN	PWG	HiFi GAN	Wave Glow
Wav2-Vec2	0.9852 ±0.0021	0.9959 ±0.0016	0.9951 ±0.0017	0.9988 ±0.0009	0.9687 ±0.0083	0.9952 ±0.0024	0.9496 ±0.0072	0.9941 ±0.0032

Table 5.3: Model-seen accuracies (fine-tuned wav2vec2, 10^{-5} learning rate, trained for 5 epochs)

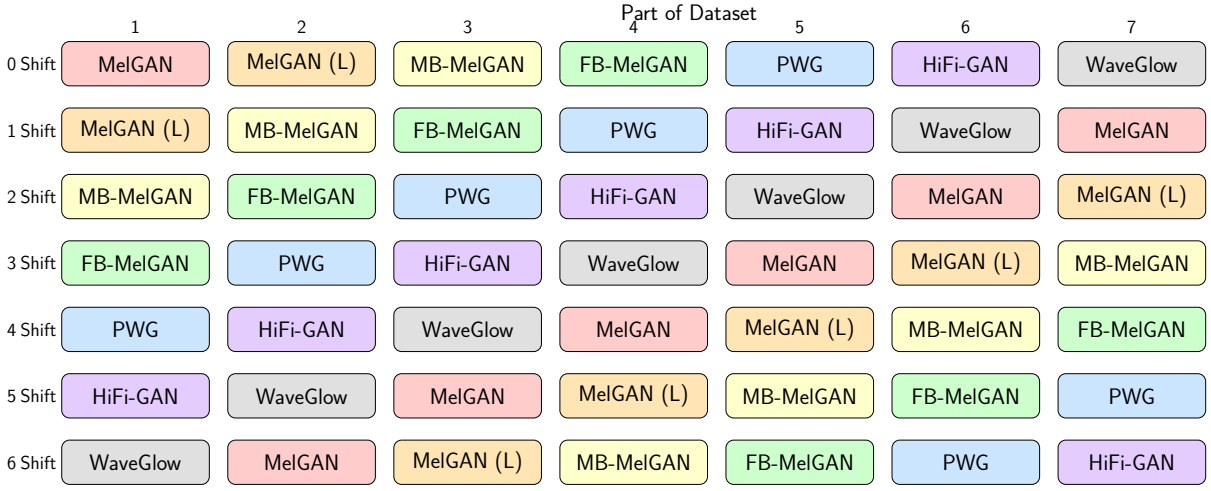
Again, DeepFakes created by FB-MelGAN and HiFiGAN result in the poorest classification results of the detection model (wav2vec2). We can see that wav2vec2 significantly outperforms the best feature-based model (XGBoost) across all DeepFake models.

5.2 Leave-one-model-out results

The previous training procedure involved all DeepFake generation models being in both the train and the test set. We will now also try keeping one model out of sample by using this model as the test set and training on all other models. For this, 7 alternative datasets will be created. For the first dataset, with 0 "shift", the first $\frac{1}{7}$th part will include DeepFakes from the standard MelGAN model, the second $\frac{1}{7}$th will include DeepFakes from the larger MelGAN version, and so on in the order visible in the figure below.

Then, for the second dataset, a "shift" is introduced, which means now the first $\frac{1}{7}$th part of the dataset will be faked by the larger MelGAN version, while the standard MelGAN version fakes the last $\frac{1}{7}$th part of the dataset. The exact shifts are displayed in the figure below.

Then, for evaluation, each of the cells in the figure is taken as the test set exactly once, and the remaining cells in the row are the train set. This will allow us to test generalization to an unseen model 7 times for each of the 7 models.



Visualization of the seven different datasets. (used for cross-validation)

Leaving one model out of the train set simulates the real-world scenario of encountering previously unseen DeepFake generation techniques, allowing us to evaluate the generalization capability of our detection system. It will also allow us to compare the DeepFake generation models in terms of resistance to generalization. Since XGBoost was clearly the best-performing model in the previous experiments, we chose to only report the performance of this model for the feature-based case.

5.2.1 Feature-based method (XGBoost)

We again begin with the feature-based method, but now only consider the best-performing model in previous experiments, considering we thought it unlikely that the performance of Random Forest could rise so significantly that it outperforms XGBoost.

Shift	Mel GAN	MelGAN (L)	MB-MelGAN	FB-MelGAN	PWG	HiFi-GAN	Wave Glow
0	0.9031	0.8066	0.8955	0.7097	0.7799	0.7756	0.8703
1	0.9294	0.8043	0.8635	0.7018	0.7640	0.7659	0.8529
2	0.9215	0.8115	0.8702	0.6956	0.7922	0.8001	0.8409
3	0.9203	0.7928	0.9077	0.7022	0.7689	0.7908	0.8513
4	0.8934	0.7603	0.8972	0.6904	0.8013	0.7782	0.8515
5	0.9255	0.7954	0.8764	0.6815	0.8038	0.7868	0.8337
6	0.8974	0.8346	0.8793	0.6589	0.7570	0.7893	0.8500
Avg	0.9031 ±0.01287	0.8066 ±0.01678	0.8793 ±0.01571	0.6841 ±0.01378	0.7839 ±0.01613	0.7814 ±0.01075	0.8498 ±0.01451

Table 5.4: leave-one-model-out accuracies (XGBoost with default parameters: max_depth=3, n_estimators=100, lr=0.1)

We can see that the detection accuracies across all of the DeepFake models have dropped significantly now that the model in the test set is not present in the training data. The ordering of accuracies has not changed drastically; still, the poorest accuracies are achieved on HiFiGAN and FB-MelGAN, suggesting that these models produce the highest quality DeepFakes, or are more different than the other DeepFake generation techniques.

Hyperparameter tuning We will perform hyperparameter tuning to identify the best-performing XGBoost configuration using a grid search. The grid search will evaluate combinations of the following three hyperparameters:

1. **n_estimators**: which is the number of trees generated by the XGBoost model, with values of [100, 200].

2. **tree_depth**: which is the maximum height (or alternatively depth) of a tree, with values of [3, 5, 7].

3. **learning_rate**: which is the learning rate of the XGBoost model, with values of [0.01, 0.1, 0.2].

   This grid search will optimize average accuracy across all models and shifts, since the accumulated accuracy is equivalent to this metric, this will be optimized. The best-performing model for this metric was the configuration n_estimators=200, max_depth=3, lr=0.2.

Shift	Mel GAN	MelGAN (L)	MB-MelGAN	FB-MelGAN	PWG	HiFi-GAN	Wave Glow
1	0.9076	0.8269	0.9032	0.7116	0.7896	0.7772	0.8846
2	0.9354	0.8436	0.8706	0.6950	0.7743	0.7708	0.8647
3	0.9322	0.8418	0.8807	0.7019	0.8077	0.8022	0.8502
4	0.9300	0.8362	0.9187	0.6963	0.7899	0.7965	0.8500
5	0.9037	0.8067	0.9101	0.6893	0.8107	0.7806	0.8602
6	0.9249	0.8223	0.8952	0.6780	0.8093	0.7948	0.8345
7	0.9048	0.8637	0.8866	0.6671	0.7674	0.7926	0.8643
Avg	0.9198 ±0.01287	0.8347 ±0.01678	0.9093 ±0.01571	0.6913 ±0.01378	0.7927 ±0.01613	0.7735 ±0.01075	0.8584 ±0.01451

Table 5.5: Leave-one-model-out accuracies (XGBoost, n_estimators=200, max_depth=3, lr=0.2)

We can see in the table that improvements in accuracy were achieved across all models with the exception of HiFi-GAN, for which the performance decreased slightly.

We choose not to show feature importances in this part, as one model is always missing from the training data and the features that predict that DeepFake technique might consequently not be present in the important features. A comparison of feature importances when leaving out certain DeepFake models is certainly possible, but is beyond the scope of this research.

5.2.2 Transformer Method

In this part we fine-tune wav2vec2. As mentioned earlier in Chapter 4, we freeze the CNN part during fine-tuning. Consequently, we fine-tune the transformer part and the classification head added to wav2vec2 on the training data. Performance will be evaluated using the same procedures as the feature-based method to enable direct comparison.

The base model of wav2vec2 is used and the inputs are truncated to 2 seconds, while clips less than two seconds are again excluded from the dataset, this is mainly done due to memory and time constraints. The model is fine-tuned to the training data for 5 epochs with a learning rate of 10^{-5}. Results are visible in the table below.

Shift	Mel GAN	MelGAN (L)	MB-MelGAN	FB-MelGAN	PWG	HiFi-GAN	Wave Glow
0	0.9919	0.9877	0.9965	0.9289	0.9661	0.8422	0.9671
1	0.9909	0.9911	0.9986	0.8155	0.9948	0.8143	0.9724
2	0.9962	0.9728	0.9992	0.9439	0.9894	0.8192	0.9763
3	0.9939	0.9839	0.9992	0.9569	0.9907	0.7474	0.9824
4	0.9924	0.9779	0.9973	0.8165	0.9930	0.8785	0.9513
5	0.9605	0.9807	0.9964	0.8859	0.9803	0.8699	0.9929
6	0.9970	0.9445	0.9973	0.9249	0.9839	0.8683	0.9844
Avg	0.9890 ±0.05046	0.9769 ±0.07927	0.9978 ±0.05873	0.8961 ±0.08392	0.9855 ±0.06473	0.8343 ±0.04848	0.9753 ±0.04432

Table 5.6: Leave-one-model-out accuracies (fine-tuned wav2vec2, 10^{-5} learning rate, trained for 5 epochs)

We can see that performance of wav2vec2 reduces significantly when the tested DeepFake model is not present in the train set, especially for the models on which the lowest

scores were achieved (FB-MelGAN and HiFiGAN). The classification accuracy of FB-MelGAN reduced from 0.9867 to 0.8961, while the classification accuracy of HiFiGAN reduced from 0.9495 to 0.8343. This 0.8343 accuracy score almost makes XGBoost competitive on this particular DeepFake model, since this model achieved a score of 0.7814 on this test, which is only a slight reduction from the 0.7984 accuracy score achieved on HiFiGAN in the model-seen test. This could mean that while XGBoost performs worse overall, its generalization capabilities to unseen high quality DeepFakes are substantially better than wav2vec2, considering the more extreme performance reductions of wav2vec2 from model-seen to leave-one-model-out testing.

Chapter 6

Discussion

While the transformer-based method demonstrated promising results in detecting DeepFake speech, the generalizability of these findings is constrained by the limitations of the dataset used. Specifically, the dataset contains speech excerpts and corresponding DeepFake replications from a single speaker, all of which originate from audiobook recordings (Frank & Schönherr, 2021). The WaveFake dataset we used was also introduced in 2021. While this is a useful benchmark, this raises several important points.

The use of a single speaker limits the phonetic, prosodic, and expressive variability present in the data. The model might learn speaker-specific patterns that are good indicators of a clip being real or DeepFake which are not indicative of DeepFakes on a broader scale. Audiobooks speech is also often very clean, with little hesitation or pauses. The presence of background noise is also nearly non-existent. This contrasts with the reality that these things do appear in real world settings.

The WaveFake dataset was introduced in 2021, meaning that the DeepFake technology used was also no newer than 2021. DeepFake technology is rapidly advancing (Goyal et al., 2025), making a relatively recent dataset from 2021 possibly outdated already.

To enhance the robustness and practical relevance of DeepFake speech detection systems, future research should prioritize evaluation on multi-speaker datasets, preferably with diverse and maybe also multi-language speakers. The real and DeepFake content should also include the same speaker, as detecting such data is more in line with the actual requirements of DeepFake detection systems. Training on a dataset in which the real and DeepFake content are concerning different speakers might also devolve into the more simple task of speaker recognition.

Recent datasets have begun to address this gap. For example, the MLAAD dataset offers promise for future research of this type: This dataset consists of DeepFakes in many different languages across different speakers (Müller et al., 2024). However, experimenting on such datasets was beyond the scope of the present study, primarily due to their large scale and the significant computational resources required for training. The use of a single-speaker dataset also enabled better experimental controls. Nevertheless, experimenting on larger and more diverse datasets like MLAAD remains a valuable direction for future work, particularly for exploring generalization across speakers, languages and DeepFake generation methods.

Chapter 7

Conclusion

The transformer method tried in this research (wav2vec2) achieved far superior performance compared to the more traditional feature-based method, both when the DeepFake model which it is tested on is present and absent in the train set. This indicates that this method of DeepFake detection is more effective.

Out of the DeepFake models, HiFiGAN and Full-Band MelGAN were the hardest to detect, both when they are present in the training data and when they are not, possibly leading to the conclusion that these models produce the highest quality DeepFakes.

The most important features used by the feature-based model were all computed using smoothed averages. Among these were a few jitter features, primarily the lower percentiles of jitter, possibly since DeepFake speech maintains a higher jitter throughout the entire utterance than real speech. The flatness of jitter was also important, which might be due to more rapid jitter changes in either real or DeepFake speech. Linear Predictive Modelling was also important: LPC (Linear Predictive Coding), which is a technique that is able to analyze speech characteristics such as vocal tract shape, and Linear Predictive Gain (LPGain), which measures vocal tract excitation are important features.

The high-accuracy detection of wav2vec2 highlights that it might be possible to classify speech as real and DeepFake based on real and DeepFake samples of a single speaker. Aside from this speaker, it might also be possible to, for example, create a "Nancy Pelosi" dataset containing real and DeepFake clips of Nancy Pelosi to detect DeepFakes of this speaker.

Bibliography

Almutairi, Z., & Elgibreen, H. (2022). A Review of Modern Audio Deepfake Detection Methods: Challenges and Future Directions. *Algorithms*, *15*(5). https://doi.org/10.3390/a15050155

audEERING. (2016a). openSMILE compare_2016_core.func.conf.inc [Configuration file for functionals in the ComParE 2016 feature set].

audEERING. (2016b). openSMILE compare_2016_core.lld.conf.inc [Configuration file for low-level descriptors in the ComParE 2016 feature set].

audEERING GmbH. (2025). openSMILE Documentation: cPitchJitter Component [Accessed: 2025-06-22]. https://audeering.github.io/opensmile/_components/cPitchJitter.html

Baevski, A., Zhou, H., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. https://arxiv.org/abs/2006.11477

Biau, G., & Scornet, E. (2015). A Random Forest Guided Tour. https://arxiv.org/abs/1511.05741

Bird, J. J., & Lotfi, A. (2023). Real-time Detection of AI-Generated Speech for DeepFake Voice Conversion. https://arxiv.org/abs/2308.12734

Chadha, A., Kumar, V., Kashyap, S., & Gupta, M. (2021). Deepfake: an overview. *Proceedings of second international conference on computing, communications, and cyber-security: IC4S 2020*, 557–566.

Chen, J., Ye, J., Tang, F., & Zhou, J. (2021). Automatic Detection of Alzheimer's Disease Using Spontaneous Speech Only. *Proceedings of Interspeech 2021*, 3830–3834. https://doi.org/10.21437/Interspeech.2021-2002

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Conneau, A., Baevski, A., Collobert, R., Mohamed, A., & Auli, M. (2020). Unsupervised Cross-lingual Representation Learning for Speech Recognition. https://arxiv.org/abs/2006.13979

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, *35*(1), 53–65. https://doi.org/10.1109/MSP.2017.2765202

Eyben, F., Wöllmer, M., & Schuller, B. (2010). Opensmile: the munich versatile and fast open-source audio feature extractor. *Proceedings of the 18th ACM International Conference on Multimedia*, 1459–1462. https://doi.org/10.1145/1873951.1874246

Frank, J., & Schönherr, L. (2021). WaveFake: A Data Set to Facilitate Audio Deepfake Detection. https://arxiv.org/abs/2111.02813

Goyal, H., Wajid, M. S., Wajid, M. A., Khanday, A. M. U. D., Neshat, M., & Gandomi, A. (2025). State-of-the-art AI-based Learning Approaches for Deepfake Generation and Detection, Analyzing Opportunities, Threading through Pros, Cons, and Future Prospects. https://arxiv.org/abs/2501.01029

Hamza, A., Javed, A. R. R., Iqbal, F., Kryvinska, N., Almadhor, A. S., Jalil, Z., & Borghol, R. (2022). Deepfake Audio Detection via MFCC Features Using Machine Learning. *IEEE Access*, *10*, 134018–134028. https://doi.org/10.1109/ACCESS.2022.3231480

Ito, K. (2017). The LJ Speech Dataset [Accessed: 2025-06-23]. https://keithito.com/LJ-Speech-Dataset/

Khochare, J., Joshi, C., Yenarkar, B., Suratkar, S., & Kazi, F. (2021). A deep learning framework for audio deepfake detection. *Arabian Journal for Science and Engineering*, 1–12.

Kong, J., Kim, J., & Bae, J. (2020). HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 17022–17033, Vol. 33). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/c5d736809766d46260d816d8dbc9eb44-Paper.pdf

Kühne, K., Fischer, M. H., & Zhou, Y. (2020). The human takes it all: Humanlike synthesized voices are perceived as less eerie and more likable. evidence from a subjective ratings study. *Frontiers in neurorobotics*, *14*, 593732.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brébisson, A., Bengio, Y., & Courville, A. C. (2019). MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/6804c9bca0a615bdb9374d00a9fcba59-Paper.pdf

Lee. (2019). Nancy Pelosi clip shows misinformation still has a home on Facebook. *BBC*. Retrieved April 9, 2025, from https://www.bbc.com/news/technology-48405661

Li, J., Guo, Y., Dou, Y., Wang, J., Qiu, B., & Liu, X. (2024). An approach to bearing fault diagnosis based on ensemble learning and case-based reasoning. *Journal of Physics: Conference Series*, *2787*, 012042. https://doi.org/10.1088/1742-6596/2787/1/012042

Li, K., Lu, X., Akagi, M., & Unoki, M. (2023). Contributions of Jitter and Shimmer in the Voice for Fake Audio Detection. *IEEE Access*, *11*, 84689–84698. https://doi.org/10.1109/ACCESS.2023.3301616

Luo, W., Li, Y., Urtasun, R., & Zemel, R. (2016). Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 29). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/c8067ad1937f728f51288b3eb986afaa-Paper.pdf

Müller, N. M., Kawa, P., Choong, W. H., Casanova, E., Gölge, E., Müller, T., Syga, P., Sperl, P., & Böttinger, K. (2024). MLAAD: The Multi-Language Audio Anti-Spoofing Dataset. *2024 International Joint Conference on Neural Networks (IJCNN)*, 1–7. https://doi.org/10.1109/IJCNN60899.2024.10650962

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, *Volume 7 - 2013*. https://doi.org/10.3389/fnbot.2013.00021

Prenger, R., Valle, R., & Catanzaro, B. (2019). Waveglow: A Flow-based Generative Network for Speech Synthesis. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3617–3621. https://doi.org/10.1109/ICASSP.2019.8683143

Reimao, R., & Tzerpos, V. (2019). FoR: A Dataset for Synthetic Speech Detection. *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, 1–10. https://doi.org/10.1109/SPED.2019.8906599

Sawant, G., Singh, A., Kadam, K., Mazarello, R., & Dumane, P. (2010). Speech synthesis using LPC. *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*, 515–517. https://doi.org/10.1145/1741906.1742022

Schuller, B., Steidl, S., Batliner, A., Hirschberg, J., Burgoon, J. K., Baird, A., Elkins, A., Zhang, Y., Coutinho, E., & Evanini, K. (2016). The INTERSPEECH 2016 computational paralinguistics challenge: deception, sincerity and native language.

Stupp. (2019). Fraudsters Used AI to Mimic CEO's Voice in Unusual Cybercrime Case. *Wall Street Journal*. Retrieved June 2, 2025, from https://www.wsj.com/articles/fraudsters-use-ai-to-mimic-ceos-voice-in-unusual-cybercrime-case-11567157402

Suthaharan, S. (2016). Support Vector Machine. In *Machine learning models and algorithms for big data classification: Thinking with examples for effective learning* (pp. 207–235). Springer US. https://doi.org/10.1007/978-1-4899-7641-3_9

Todisco, M., Delgado, H., & Evans, N. (2017). Constant Q cepstral coefficients: A spoofing countermeasure for automatic speaker verification. *Computer Speech Language*, *45*, 516–535. https://doi.org/https://doi.org/10.1016/j.csl.2017.01.001

Tzanetakis, G. (2011). Audio feature extraction. *Music data mining*, 49–74.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wang, R., Juefei-Xu, F., Huang, Y., Guo, Q., Xie, X., Ma, L., & Liu, Y. (2020). DeepSonar: Towards Effective and Robust Detection of AI-Synthesized Fake Voices. *Proceedings of the 28th ACM International Conference on Multimedia*, 1207–1216. https://doi.org/10.1145/3394171.3413716

Yamamoto, R., Song, E., & Kim, J.-M. (2020). Parallel Wavegan: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6199–6203. https://doi.org/10.1109/ICASSP40776.2020.9053795

Yang, G., Yang, S., Liu, K., Fang, P., Chen, W., & Xie, L. (2021). Multi-Band Melgan: Faster Waveform Generation For High-Quality Text-To-Speech. *2021 IEEE Spoken Language Technology Workshop (SLT)*, 492–498. https://doi.org/10.1109/SLT48900.2021.9383551

Yang, T., Sun, C., Lyu, S., & Rose, P. (2025). Forensic deepfake audio detection using segmental speech features. https://arxiv.org/abs/2505.13847

Yi, J., Wang, C., Tao, J., Zhang, X., Zhang, C. Y., & Zhao, Y. (2023). Audio Deepfake Detection: A Survey. https://arxiv.org/abs/2308.14970

Appendix A

Link to code and datasets used (WaveFake Part 1 contains feature-based experiments, WaveFake Part 2 contains transformer-based experiments):
https://drive.google.com/drive/folders/1pWzg_qxvAlmtxUgFYx0nOpvqtq8ibLwv?usp=sharing

Train Set							
Split	**MelGAN**	**MelGAN-L**	**MB-MelGAN**	**FB-MelGAN**	**PWG**	**HiFi-GAN**	**Wave Glow**
1	1439	1492	1415	1445	1517	1441	1513
2	1477	1421	1390	1452	1516	1533	1473
3	1469	1545	1423	1450	1455	1499	1422
4	1505	1457	1453	1504	1458	1417	1469
5	1491	1447	1456	1434	1468	1498	1468
6	1535	1472	1467	1422	1460	1478	1428
7	1478	1519	1440	1458	1436	1441	1491
8	1493	1470	1502	1499	1420	1423	1456
9	1510	1405	1534	1467	1466	1434	1446
10	1438	1465	1439	1528	1434	1443	1515
Test Set							
Split	**MelGAN**	**MelGAN-L**	**MB-MelGAN**	**FB-MelGAN**	**PWG**	**HiFi-GAN**	**Wave Glow**
1	372	380	379	361	331	359	384
2	352	348	384	348	398	363	373
3	351	357	367	387	360	388	355
4	367	371	356	390	348	350	383
5	377	340	371	374	370	380	354
6	346	369	370	334	365	390	392
7	362	367	376	365	359	349	387
8	346	362	368	350	398	362	379
9	382	355	362	371	363	371	362
10	331	352	344	407	363	378	380

Table 1: Model-seen DeepFake model distribution of feature-based models

Split	**Total Acc.**	**MelGAN**	**MelGAN Large**	**MB-MelGAN**	**FB-MelGAN**	**PWG**	**HiFi-GAN**	**WaveGlow**
1	0.5170	0.5296	0.5197	0.5158	0.5069	0.5242	0.5056	0.5169
2	0.5281	0.5540	0.5244	0.5195	0.5201	0.5352	0.5248	0.5188
3	0.5137	0.5171	0.5448	0.5123	0.5116	0.5083	0.5142	0.4887
4	0.5227	0.5381	0.5445	0.5112	0.5167	0.5201	0.5157	0.5117
5	0.5188	0.5279	0.5235	0.5162	0.5160	0.5297	0.5112	0.5071
6	0.5171	0.5202	0.5041	0.5345	0.5030	0.5219	0.5333	0.5013
7	0.5270	0.5497	0.5368	0.5293	0.5288	0.5292	0.5272	0.4910
8	0.5262	0.5246	0.5262	0.5217	0.5329	0.5377	0.5318	0.5092
9	0.5120	0.5301	0.5056	0.5325	0.5108	0.4917	0.5054	0.5069
10	0.5211	0.5453	0.5312	0.5190	0.5258	0.5344	0.5132	0.4829

Table 2: Model-seen accuracies for Support Vector Machine

Split	Total Acc.	MelGAN	MelGAN Large	MB-MelGAN	FB-MelGAN	PWG	HiFi-GAN	WaveGlow
1	0.7800	0.8508	0.8434	0.8047	0.6454	0.7598	0.7382	0.8073
2	0.7876	0.8707	0.8506	0.8203	0.6695	0.7751	0.7204	0.8056
3	0.8035	0.8960	0.8936	0.8147	0.6886	0.7778	0.7500	0.8197
4	0.7971	0.8924	0.8585	0.8427	0.6577	0.7672	0.7543	0.8120
5	0.7805	0.8714	0.8471	0.8154	0.6604	0.7500	0.7325	0.7938
6	0.7848	0.8671	0.8604	0.8146	0.6572	0.7438	0.7333	0.8112
7	0.7909	0.9240	0.9114	0.8524	0.6137	0.7326	0.7063	0.7894
8	0.7922	0.9335	0.8854	0.8533	0.6329	0.7462	0.6892	0.8087
9	0.8045	0.9005	0.8817	0.8479	0.6712	0.7686	0.7439	0.8191
10	0.7979	0.8927	0.8821	0.8660	0.6683	0.7576	0.7156	0.8329

Table 3: Model-seen accuracies for Random Forest

Split	Total Acc.	MelGAN	MelGAN Large	MB-MelGAN	FB-MelGAN	PWG	HiFi-GAN	WaveGlow
1	0.8445	0.9126	0.8816	0.8852	0.7230	0.8293	0.8008	0.8698
2	0.8402	0.8864	0.8980	0.8594	0.7385	0.8291	0.7934	0.8753
3	0.8521	0.9103	0.9146	0.8815	0.7403	0.8375	0.8170	0.8761
4	0.8501	0.9292	0.8908	0.8876	0.7333	0.8348	0.8000	0.8786
5	0.8437	0.9098	0.8838	0.8881	0.7286	0.8176	0.8155	0.8672
6	0.8470	0.9147	0.8957	0.9039	0.7081	0.8247	0.8000	0.8737
7	0.8454	0.9351	0.9114	0.9043	0.6973	0.8106	0.7751	0.8773
8	0.8417	0.9321	0.9019	0.8940	0.6886	0.8204	0.7666	0.8865
9	0.8624	0.9463	0.8972	0.9156	0.7534	0.8196	0.8221	0.8826
10	0.8560	0.9335	0.9063	0.9196	0.7273	0.8375	0.7937	0.9000

Table 4: Model-seen accuracies for XGBoost with base parameters

Train Set							
Split	**MelGAN**	**MelGAN-L**	**MB-MelGAN**	**FB-MelGAN**	**PWG**	**HiFi-GAN**	**Wave Glow**
1	1469	1464	1402	1458	1506	1502	1461
2	1508	1436	1458	1436	1417	1474	1533
3	1463	1402	1512	1507	1475	1432	1472
4	1439	1472	1444	1462	1483	1501	1462
5	1413	1439	1497	1551	1483	1471	1408
6	1423	1491	1453	1447	1491	1457	1500
7	1487	1504	1408	1406	1531	1481	1446
8	1466	1482	1496	1447	1491	1477	1403
9	1436	1499	1473	1499	1473	1477	1441
10	1450	1506	1478	1449	1476	1405	1498
Test Set							
Split	**MelGAN**	**MelGAN-L**	**MB-MelGAN**	**FB-MelGAN**	**PWG**	**HiFi-GAN**	**Wave Glow**
1	372	388	355	368	323	374	386
2	364	393	365	359	373	373	339
3	370	363	373	410	362	354	333
4	384	367	350	368	339	362	395
5	380	418	333	367	332	369	367
6	368	357	390	367	377	357	350
7	376	335	386	386	320	364	398
8	332	359	383	387	361	363	380
9	371	358	343	369	363	385	377
10	377	378	365	386	351	337	372

Table 5: Model-seen DeepFake model distribution of XGBoost grid search

Split	Total Acc.	MelGAN	MelGAN Large	MB-MelGAN	FB-MelGAN	PWG	HiFi-GAN	WaveGlow
1	0.8566	0.9126	0.8905	0.8873	0.7514	0.8514	0.8222	0.8782
2	0.8554	0.9135	0.9109	0.8973	0.7270	0.8552	0.8003	0.8805
3	0.8552	0.9324	0.9091	0.8954	0.7293	0.8564	0.8051	0.8724
4	0.8601	0.9258	0.9114	0.9014	0.7446	0.8510	0.8149	0.8684
5	0.8599	0.9157	0.8744	0.9204	0.7561	0.8404	0.8347	0.8774
6	0.8527	0.9158	0.8908	0.8974	0.7357	0.8196	0.8149	0.8943
7	0.8538	0.9455	0.9075	0.9197	0.6982	0.8422	0.7747	0.8907
8	0.8636	0.9413	0.9457	0.9282	0.7054	0.8407	0.8030	0.8934
9	0.8686	0.9393	0.9162	0.9271	0.7398	0.8471	0.8286	0.8886
10	0.8603	0.9363	0.9285	0.9205	0.6995	0.8504	0.7982	0.8871

Table 6: Model-seen accuracies for best-performing XGBoost model

Train Set							
Split	**MelGAN**	**MelGAN-L**	**MB-MelGAN**	**FB-MelGAN**	**PWG**	**HiFi-GAN**	**Wave Glow**
1	1446	1489	1520	1439	1453	1469	1446
2	1494	1438	1480	1480	1404	1444	1522
3	1459	1470	1492	1488	1494	1382	1476
4	1465	1484	1460	1461	1428	1489	1474
5	1463	1441	1520	1412	1450	1507	1469
6	1461	1489	1479	1511	1417	1451	1454
7	1529	1548	1439	1371	1469	1412	1493
8	1447	1365	1483	1541	1467	1512	1446
9	1447	1427	1545	1486	1494	1445	1418
10	1443	1528	1459	1464	1453	1509	1406
Test Set							
Split	**MelGAN**	**MelGAN-L**	**MB-MelGAN**	**FB-MelGAN**	**PWG**	**HiFi-GAN**	**Wave Glow**
1	354	354	376	370	387	387	337
2	349	366	376	368	355	362	389
3	365	364	402	349	353	384	349
4	365	334	378	369	393	367	360
5	369	407	348	362	360	365	354
6	348	379	357	382	373	373	353
7	346	386	392	381	376	363	322
8	361	361	347	357	388	392	360
9	367	388	361	377	352	358	362
10	362	382	337	366	363	373	382

Table 7: Model-seen DeepFake model distribution of transformer method

Split	Total Acc.	MelGAN	MelGAN Large	MB-MelGAN	FB-MelGAN	PWG	HiFi-GAN	WaveGlow
1	0.9832	0.9958	0.9944	1.0000	0.9649	0.9961	0.9393	0.9955
2	0.9827	0.9928	0.9959	1.0000	0.9552	0.9915	0.9475	0.9949
3	0.9866	0.9986	0.9959	0.9975	0.9742	0.9972	0.9557	0.9871
4	0.9846	0.9973	0.9985	0.9987	0.9715	0.9962	0.9387	0.9917
5	0.9887	0.9973	0.9939	0.9986	0.9793	0.9972	0.9603	0.9944
6	0.9869	0.9971	0.9934	0.9972	0.9751	0.9960	0.9531	0.9986
7	0.9867	0.9957	0.9922	0.9987	0.9738	0.9973	0.9559	0.9938
8	0.9819	0.9958	0.9958	0.9986	0.9524	0.9961	0.9439	0.9930
9	0.9844	0.9946	0.9961	1.0000	0.9708	0.9901	0.9455	0.9931
10	0.9866	0.9945	0.9948	0.9985	0.9699	0.9945	0.9558	0.9987

Table 8: Model-seen accuracies for wav2vec2