Towards Robust, Explainable Deepfake Audio Detection

Georgia Channing

University of Oxford

A thesis presented for the degree of

Master of Science in Advanced Computer Science

Trinity 2024

The ability to disseminate large-scale disinformation to undermine scientifically established facts poses an existential risk to humanity and endangers democratic institutions and fundamental human rights.

— UN Report

Acknowledgements

I would like to thank my thesis supervisors, Prof. Ronald Clark, Prof. Philip Torr, and Dr. Christian Schroeder de Witt for their support and guidance in writing this thesis. I would also like to that the BBC for access to their compute credits and their journalists. None of this would be possible without the support of my friends, family, or partner, and–to them–I am eternally indebted. Lastly, I would like to thank the Chemical Library at the University of Ferrara for access to their `eduroam`, without which this document could never have been generated.

Abstract

The rise of deepfake audio presents a growing threat to election security, where realistic synthetic voices can be used to manipulate public opinion and spread disinformation. As deepfake audio generation techniques rapidly improve, detection systems struggle to keep pace. While numerous academic works address detection methods, they often overlook the challenges of generalization errors and the effects of data augmentation commonly encountered in real-world scenarios. In this work, we not only investigate these underexplored areas but also emphasize the importance of model explainability. We recognize that explainable AI is essential for decision-makers in election contexts, where trust and transparency are paramount. Additionally, we outline future directions for enhancing deepfake audio detection, with a focus on improving robustness and adaptability to evolving threats.

Keywords— deepfake audio - explainability - self-supervised learning

Table of Contents

List of Figures

List of Tables

1 | Introduction

1.1 Motivation

On June 2, 2024, Mexico held a presidential election. The front-runner, and ultimate winner, of the election was Claudia Scheinbaum, the former mayor of Mexico City. Scheinbaum resigned from her post as mayor of Mexico City on June 16, 2023, in order to contend in her party's internal process for selecting a presidential candiate. Upon her resignation, she appointed a man named Marti Batres to serve as interim mayor.

In early November 2023, a purportedly leaked audio clip of Marti Batres went viral [43]. In the clip, Batres appears to plan to manipulate the mayoral election process to favor a particular candidate, Clara Brugada, at the expense of another candidate, Omar Harfus. The alleged voice of Batres also suggests that a number of prominent journalists are colluding in this plot, undermining their credibility. Mexico has long suffered from political corruption and unreliable journalism, and, as a result, the leaked recording was quickly accepted as fact by many. The audio quality is also superb; the clip includes pauses, the sounds of breathing, and Batres' smooth Mexico City accent [43]. Batres' was quick to denounce the viral clip as fake, but many Mexicans were hesitant to accept this explanation.

A journalist at the British Broadcasting Company (BBC), Jack Goodman, got wind of this story and decided to try to get to the bottom of whether or not the clip was in fact fake. At the end of June 2024, Jack Goodman met with the author of this thesis and reported that he took the clip of Batres' to both academic and com-

mercial "experts" in deepfake audio to try to determine if the clip's authenticity. He also worked with a Mexican journalist at BBC Mundo reporter named Laura Garcia, his go-to Spanish-language and Mexican-politics specialist.

Shortly after the clip was discovered, it was run through an online deepfake audio detector called AI Voice Detector, which declared with 90% certainty that the clip was real. Jack then took the clip to Hany Farid, a leading researcher in this space at the University of California, Berkeley, who ran it through three models. All three came back with the pronouncement that the clip was fake. As a sanity check, Jack also asked Laura to record a sample of her voice, which he then sent to Farid. Two of Farid's models correctly labelled Laura's voice as real, but one did not. Farid, who worked closely with Jack on this project, was also unable to furnish a detailed explanation of what about an audio sample leads a model to classify it as real or fake. This makes it incredibly difficult for journalists, professional fact checkers, or savvy media consumers to judge the validity of a model's classification. In addition, Jack took the clip to a variety of commercial providers of deepfake audio detectors. Some of them correctly classified the audio as fake, but, when presented with Laura's voice sample, they also classified it as fake.

Given the conflicting signals encountered when trying to establish the authenticity of an audio sample and the lack of explanation for any given classification, it is almost impossible for a journalist or member of law enforcement to reliably or confidently determine that something is fake and respond to it.

1.2 Aim and Objectives

Given the new prevalence and dangers of deepfake audio, this thesis explores the current ability to detect deepfake audio and explain why something might be classified as fake. In short, this thesis attempts to account for and improve the state of robust, explainable deepfake audio detection.

As this thesis is motivated by impacts in election interference, research questions are driven by the needs of professionals attempting to combat it and what they report seeing in the wild.

Jack Goodman, the BBC journalist who spoke to the author in June 2024, also reported seeing an increasing number of spliced deepfake audio tracks, in which a large segment of the audio or audio-video content is real but some small portion is not. This small portion is nearly impossible to detect, but may drastically change the impact and meaning of an audio message. Jack pointed to an example that cropped up in the 2024 Pakistani election cycle, in which a video of a prominent politician speaking was circulated. This video was mostly real, but, in the segment of the audio where the politician discusses a few journalists, new names were painted, or spliced, into the video.

In this thesis, we evaluate multiple methods of deepfake audio detection, comparing traditional methods with more recent deep learning advancements. The author compares performance of these methods on different datasets, tests whether samples can still be reliably classified after undergoing significant compression or other manipulation, evaluates different models' generalization abilities, attempts to explain model classifications, and discusses successes and failures. Though the

author achieves state-of-the-art performance on a well-known deepfake benchmark, the author emphasizes that there is much work left to do.

1.3 Thesis Outline

The remainder of this report is organised as follows:

Chapter 2 — introduces the technical concepts critical to this study.

Chapter 3 — surveys recent related literature.

Chapter 4 — describes the datasets, models, hyperparameters, and hardware employed to conduct the experiments.

Chapter 5 — reports the experiments conducted in this study and their results.

Chapter 6 — reviews the results and synthesizes the authors' takeaways.

Note: While this report would not have been possible without the support of the many people mentioned in the Acknowledgements, it should be noted that all of the experiments presented in this thesis, their associated code, and subsequent analysis were written solely by the candidate. The use of the pronoun "we" throughout the report is for ease of reading and is figurative.

2 | Background

In this section, we review the concepts and technical material necessary for the remainder of this thesis. We introduce methods for generating deepfake audio, crafting audio features, training audio classifiers, evaluating model performance, and explaining model predictions.

2.1 Generating DeepFake Audio

Before we dive into the methods of detecting whether an audio sample is deepfake or not, it is worth discussing how deepfake audio samples are created. The two most prevalent methods for generating deepfake audio are text-to-speech (TTS) and voice conversion (VC) audio generation.

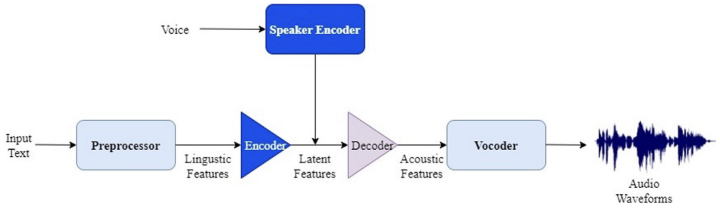


Figure 2.1: Generalized text-to-speech model architecture as produced by Alican et al.[36]

TTS architectures typically accept two inputs: an audio sample of the target speaker's voice and a target text. As seen in Figure 2.1, a TTS architecture is comprised of four primary mechanisms: the speaker encoder, the text encoder, the decoder, and the vocoder. A speaker encoder extracts features of the speaker's voice, such as timbre, cadence, and pitch. Those features are then combined with the text encoder, which uses the text to generate linguistic features and phoneme

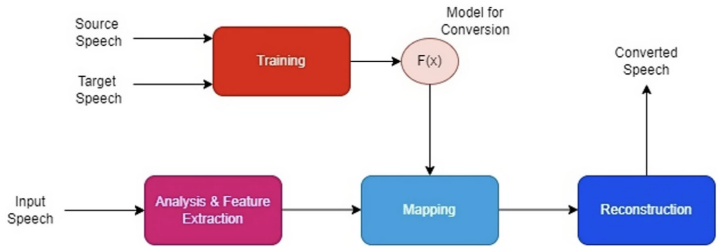


Figure 2.2: Generalized voice conversion model architecture as produced by Ali-can et al.[36]

mappings. Together, these two comprise the latent audio features. The decoder maps the latent features back to acoustic features, which are then translated to a waveform by the vocoder. Different TTS models take different approaches to each of these components, but most state-of-the-art TTS models now use EnCodec tokens and auto-regressive encoders for both text and speech encoding [32, 36].

VC methods, in contrast, train models to define a mapping between source and target speech. As seen in Figure 2.2, at both training and inference time, vocal, emotional, and tonal features are extracted, typically with a neural network. Then, during training, a function is learned to map source speech to associated target speech. Despite the relative simplicity of the VC method when compared to the TTS method, deepfake detection has historically been more difficult with deepfakes produced by VC methods than TTS methods [36].

2.2 Classical Signal Processing Features

The set of features we describe below are often referred to as “hand-crafted” features in machine learning literature. These features are not learned by neural networks but have instead been engineered by audio and signal processing ex-

perts. Though these features predate the advent of deep learning, they are still very popular for various tasks in audio classification.

Spectral Features Mel-Frequency Cepstral Coefficients, or MFCCs, are a set of typically ten to twenty features that broadly capture the *timbre* of audio samples. The Mel scale is a perceptual scale of pitches that human listeners have judged to be equidistant, which makes the Mel scale highly relevant in tasks related to human audio perception and understanding.

To calculate the MFCCs, the audio signal is first passed through a pre-emphasis filter that amplifies higher frequencies, to which the human ear is highly attentuated. The pre-emphasis filter is a simple first-order high-pass filter such that:

$$y(t) = x(t) - \alpha \cdot x(t-1), \tag{2.1}$$

where y is the computed pre-emphasis signal, x is the original signal, and α is a parameter typically chosen between 0.95 and 0.97 [1]. Then, the signal is divided into short overlapping frames of 20 milliseconds, and a Hann window is applied to each frame. The Hann window function tapers the beginning and end of each frame such that the abrupt discontinuities at the edges of the frame do not lead to inaccuracy when the signal is represented in the frequency domain [1]. The Hann window is defined by:

$$w(n) = 0.5 \cdot \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right), \tag{2.2}$$

where N represents the total number of points in the window and n is the index

of a point.

We then perform a Fast Fourier Transform (FFT) on each windowed frame to convert it to the frequency domain, such that:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2\pi kn/N}, \tag{2.3}$$

where $X(k)$ is the frequency spectrum for the frame k and N is the length of the FFT. Using the frequency spectra we have just computed, we compute the power spectrum of the signal:

$$P(k) = \frac{|X(k)|^2}{N}, \tag{2.4}$$

where $P(k)$ is the power spectral density for the frame k. To mimic the way that the human ear perceives frequencies on a non-linear scale, the power spectrum is passed through a Mel-scale filter bank [1]. The filters are triangular and spaced according to the Mel scale, which is calculated as:

$$m(f) = 2\,595 \cdot \log_{10}\left(1 + \frac{f}{700}\right), \tag{2.5}$$

where f is the frequency in Hertz and $m(f)$ is the corresponding Mel frequency. The log sum of the energy in each band after the Mel filters have been applied to the power spectrum is calculated by:

$$S_m = \log \sum_{k=f_{\text{min}}}^{f_{\text{max}}} P(k) \cdot H_m(k), \tag{2.6}$$

where S_m is the filtered spectrum in the m-th Mel band and $H_m(k)$ is the triangular filter for the m-th Mel filter. Finally, the MFCCs are generated by applying the

Discrete Cosine Transform (DCT) to the logarithmic Mel filter bank energies. The n-th MFCC is calculated by:

$$c_n = \sum_{m=0}^{M-1} \log(S_m) \cdot \cos\left[\frac{\pi n(2m+1)}{2M}\right], \tag{2.7}$$

where c_n is the n-th filter and M is the number of Mel filters, which is a controllable parameter.

Beyond MFCC features, other spectral shape features, such as the spectral centroid and bandwidth, can also be useful. The spectral centroid is a weighted average of the frequencies present in an audio signal, calculated by discrete Fourier transform, with their magnitudes used as weights. The centroid is calculated by:

$$C = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}, \tag{2.8}$$

where $x(n)$ represents the weighted frequency value of bin n and $f(n)$ represents the central frequency of that bin [2].

The spectral bandwidth is defined by the spectral spread and is calculated by:

$$B = \sqrt{\frac{\sum_{n=1}^{N-1}(f(n)-C)^2 x(n)}{\sum_{n=0}^{N-1} x(n)}}. \tag{2.9}$$

Finally, the spectral roll-off of an audio sample is the frequency index R below which the fraction γ, typically $\gamma = 0.85$, of the audio signal's frequencies resides:

$$\sum_{k=1}^{R} |X(k)|^2 \geq \gamma \sum_{k=1}^{B} |X(k)|^2. \tag{2.10}$$

Chroma Analagous to features based on the Mel scale, chroma features capture audio frequencies across the twelve pitch classes (i.e., C, C#, D, D#, etc.). Like MFCCs, chroma features are calculated through fast Fourier transforms [2]. However, they distinguish themselves from MFCC features by capturing harmonic and melodic elements of audio samples while remaining invariant to changes in timbre.

To calculate chroma features, we first compute the Short-Time Fourier Transform (STFT), which translates the time-domain signal $x(t)$ into the frequency domain:

$$X(t, f) = \sum_{n=0}^{N-1} x(n) \cdot w(n - t) \cdot e^{-2\pi i \frac{fn}{N}}, \tag{2.11}$$

where $X(t, f)$ is the complex-valued frequency spectrum at time t and frequency f, $w(n)$ is the Hann window function, and $x(n)$ is the discretized audio signal. We map the resulting spectrum to a log frequency scale that mimics human perception of musical pitch. As such, each frequency bin from the Fourier transform is mapped to one of the twelve chroma bins by:

$$p = \left(12 \cdot \log_2 \left(\frac{f}{f_0}\right)\right) \mod 12, \tag{2.12}$$

where f is the frequency of a given bin, f_0 is a reference frequency (usually $A4 = 440$ Hertz), and p is the pitch class (0 for C, 1 for C#, etc).

After assigning each frequency to a chroma bin, the energy for all frequencies in

each pitch class across different octaves is summed by:

$$C_k(t) = \sum_{f \in \mathcal{B}_k} |X(t, f)|^2, \tag{2.13}$$

where $C_k(t)$ is the energy in the k-th chroma bin at time t and $\mathcal{B}_k$ is the set of frequencies that map to the k-th pitch class[2].

Zero Crossing Rate Beyond spectral features, there are a few other popular acoustic features for audio classification tasks. One of the most useful for speech-related tasks is the zero crossing rate (ZCR). The ZCR refers to the rate at which the audio signal crosses between positive and negative [2]. It is defined as:

$$ZCR = \frac{1}{T-1} \sum_{t=1}^{T-1} \mathbf{1}_{\mathbb{R}_{<0}}(s_t s_{t-1}), \tag{2.14}$$

where s is a signal of length T and $\mathbf{1}_{\mathbb{R}_{<0}}$ is an indicator function. As the ZCR is essentially a primitive *pitch* detection metric, it is widely used in human voice activity detection, music information retrieval, and speech recognition.

Root Mean Square As an energy function, the root mean square (RMS) of an audio sample tends to capture the *loudness* of a moment n in an audio sample of length N. The RMS is calculated by:

$$E(n) = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} x(n+i)^2}. \tag{2.15}$$

These features are popular not only for their efficacy but also for the ease and speed with which they are computed. We will use these features with a traditional

ensemble model, introduced in the next section.

2.3 Gradient Boosting Decision Trees

Gradient boosting decision trees (GBDT) combine three core concepts in traditional machine learning: ensemble learning, boosting, and gradient descent. As an *ensemble* method, GBDT combine the predictions of several weak learners–typically decision trees–to produce a stronger overall prediction. The *boosting* aspect of GBDT means that the model is constructed sequentially, such that at each iteration a new weak learner is added to correct for the previous learners' mistakes. Finally, the *gradient* aspect of GBDT reflects the optimization technique used to find the best fit for each new weaker learner added to the ensemble.

At initialization, the GBDT is an ensemble containing a single weak learner that makes a prediction. For binary classification, the model is initialized with a constant value, the log-odds of the positive class:

$$\hat{F}_0(x) = \frac{1}{2} \log \left(\frac{p}{1-p} \right), \tag{2.16}$$

where p is the probability of the positive class. At each iteration, until the maximum number of weak learners permitted in the ensemble is reached, the pseudo-residuals are calculated, which are the negative gradients of the loss function:

$$r_{im} = - \left[\frac{\partial L(y_i, \hat{F}_{m-1}(x_i))}{\partial \hat{F}_{m-1}(x_i)} \right] = y_i - \hat{p}_{m-1}(x_i), \tag{2.17}$$

where r_{im} is the residual for the i-th instance at iteration m, $L(y_i, \hat{F}_{m-1}(x_i))$ is the

loss function to be minimized, $\hat{p}_{m-1}(x_i)$ is the predicted probability of the positive class for the i-th instance at iteration $m-1$, and y_i is the true label. A new weak learner, $h_m(x)$, is fitted to this residual by:

$$h_m(x) = \arg\min_h \sum_{i=1}^{n} \left(r_{im} - h(x_i)\right)^2 . \tag{2.18}$$

The prediction of each weak learner is scaled by a learning rate in order to avoid overfitting, and added to the ensemble, such that the model at iteration m is given by:

$$\hat{F}_m(x) = \hat{F}_{m-1}(x) + \nu \cdot h_m(x), \tag{2.19}$$

where ν is the learning rate. At the end of the learning process, the predicted probability of a data sample's membership in the positive class is given by:

$$\hat{p}(x) = \frac{1}{1 + \exp(-\hat{F}_m(x))}. \tag{2.20}$$

The final classification is given by thresholding the prediction probability p, where $p = 0.5$ is typical.

GBDT can be robust and versatile but can also be highly sensitive to overfitting. As a result, careful hyperparameter selection is critical to the performance of GBDT. In particular, the number of estimators in the ensemble and the complexity of each weak learner is critical to good performance. When the weak learners are decision trees, the metric of complexity is the depth of each decision tree.

2.4 Audio Transformers

Since the publication of *Attention is All You Need* [9] in 2017, transformer models have become increasingly widespread for a wide variety of tasks, though most notably text generation. As suggested by the title of that 2017 paper, *attention* is the core of the transformer architecture. Given a sequence of input embeddings

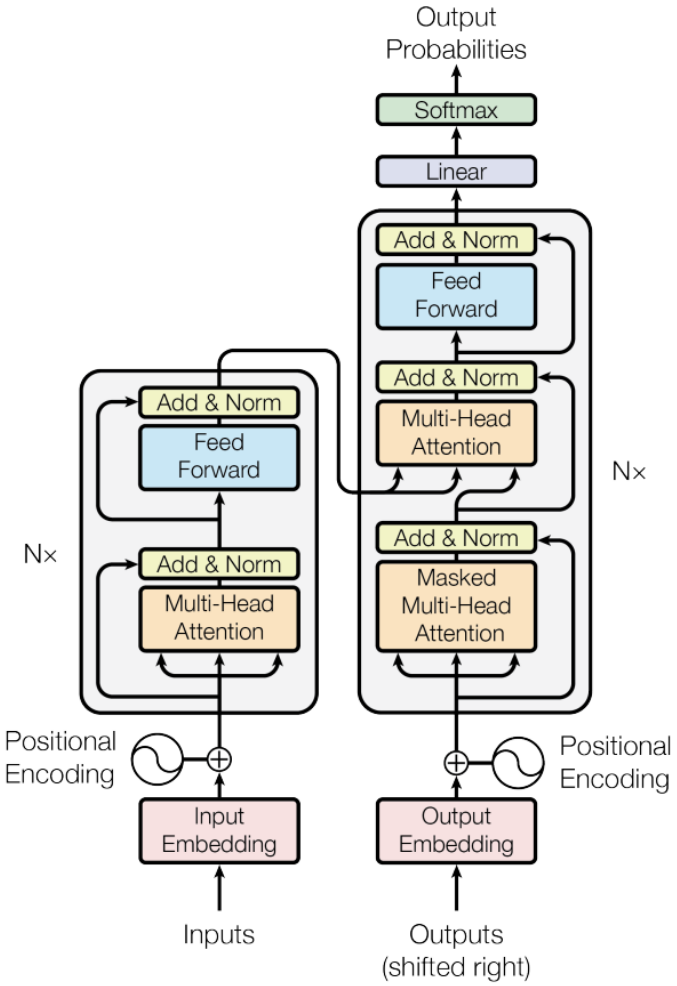


Figure 2.3: Diagram of the transformer architecture as presented in *Attention is All You Need* [9].

E, a transformer model encodes tokens using a *self-attention* mechanism, which allows the model to focus on different parts of the input sequence when encoding a particular token [9]. A single self-attention operation (or head) is defined by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.21}$$

where $\mathbf{X}$ is the matrix of input embeddings, $Q = \mathbf{X}W_Q$ is called the query matrix, $K = \mathbf{X}W_K$ is called the key matrix, $V = \mathbf{X}W_V$ is called the value matrix, W_Q, W_K, W_V are learned weight matrices, and d_k is the dimensionality of the keys [9]. In order to facilitate learning multiple different features, multiple self-attention heads are used. Multi-attention is then defined by:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O, \tag{2.22}$$

where $\text{head}_i = \text{Attention}(QW_{Q_i}, KW_{K_i}, VW_{V_i})$ and W_O is the output projection matrix. The output of the multi-head attention is passed into a Feed-Forward Neural Network (FFN) defined by:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2, \tag{2.23}$$

where W_1 and W_2 are learned weight matrices, b_1 and b_2 are biases, and ReLU is the activation function.

Each multi-head attention or FFN sub-layer is followed by layer normalization defined by:

$$\text{Layer Output} = \text{LayerNorm}(x + \text{Sub-layer}(x)). \tag{2.24}$$

As shown in Figure 2.3, the final transformer architecture stacks multiple of these multi-head attention and FFN layers to capture increasingly complex patterns.

The aforedescribed operations are universal to the transformer architecture, but methods of creating the input sequence vary widely. The transformer architecture relies on the creation of *tokens* from raw input data and the learned *attention* between those tokens. For natural language tasks, tokens typically represent individual words. For image tasks, tokens are typically pixel patches. For audio tasks, there are a variety of approaches. Here, we will review two popular mechanisms for generating input embeddings from audio data.

Wav2Vec One of the most popular feature generators is "Wav2Vec", produced and published by Meta in 2019 [15]. Wav2Vec uses a 7-layer convolutional neural network (CNN) generate latent feature encodings, which are then put into a quantization module to make the final tokens which will be fed to the transformer [15]. As speech is continuous, Wav2Vec strives to automatically infer discrete speech units with the quantization module, such that tokens can be formulated as they would be in natural language, representing complete but discrete data units [15]. Because of Wav2Vec tokens' similarity to that of natural language tokens, they are often thought of as *semantic* audio tokens.

Audio Spectrogram Transformer In contrast, the Audio Spectrogram Transformer (AST), introduced in 2021 by Gong et al [23], simplifies the audio token generation process. This model was the first fully transformer-based architecture and moved away from convolutional neural network approaches that were popular for years for audio feature generation.

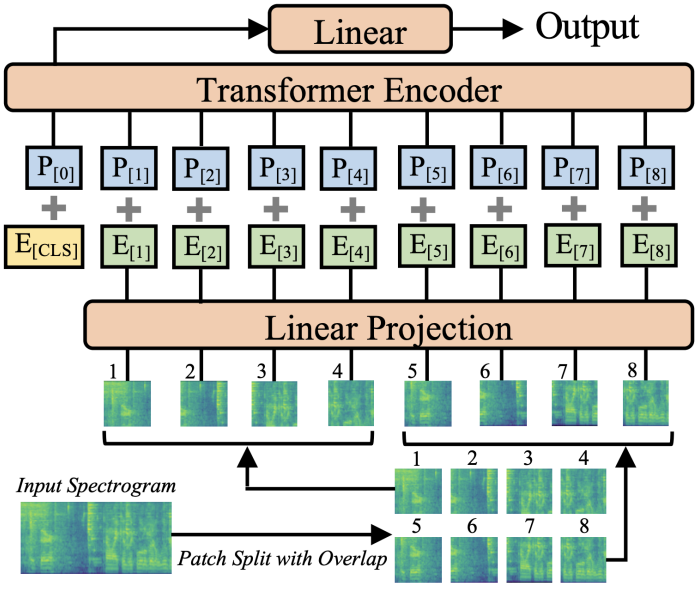


Figure 2.4: Diagram of the audio spectrogram transformer architecture introduced by Gong et al.[23]

As seen in Figure 2.4, the AST first transforms an input audio wave of length t seconds into a sequence of 128-dimensional Mel features, which are computed with a Hamming window every 10 milliseconds. The difference between the Hamming window and the Hann window discussed in the previous section is that the Hann window touches zero at both ends while the Hamming window doesn't quite reach zero and therefore contains slight discontinuity in the signal [1]. The resultant $128 \times 10t$ spectrogram is then used as input to the AST. The spectrogram is then split into a sequence of N 16×16 patches, with overlap in both time and frequency dimensions. Each 16×16 patch is flattened into a single-dimensional patch of size 768 with a linear layer. Since the transformer network does not maintain input order, an additional positional embedding token is appended to each patch to allow the model to maintain the spatial structure of the input spectrogram [23]. We also often prepend a `[CLS]` token to the input. After the input has passed

through all the transformer layers, the final representation of the `[CLS]` token will contain information aggregated from the entire input sequence and will be used to make the final prediction. Once the audio is formatted in this way, the AST effectively feeds the input sequence to a Vision Transformer (ViT), an image transformer model trained on ImageNet [23]. This approach essentially translates the audio signal into an image and then uses a transformer pre-trained on image data to make classifications.

The AST tokens, in comparison to Wav2Vec tokens, are much easier to visualize and relate to the original audio sample. Irrespective of token generation strategy, either set of tokens can be used with transformers for a variety of downstream tasks. Both Wav2Vec and AST models can be finetuned for specific downstream tasks, such as deepfake audio detection.

2.5 Metrics & Losses

In this work, we concern ourselves with a set of classification problems. We train our models to minimize the log loss, or logarithmic loss. The log loss, which measures the accuracy of the probability predictions by comparing the predicted probability of the true class label with the actual label, is defined by:

$$\text{Log Loss} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{p}_i) + (1-y_i)\log(1-\hat{p}_i)\right], \tag{2.25}$$

where n is the number of observations, y_i is the true label for the i-th observation, and $\hat{p}_i$ is the predicted probability of the positive class for the i-th observation.

We evaluate models trained for binary classification tasks with accuracy, preci-

sion, recall, and F1-score. We define four classes of potential predictions: true positives, true negatives, false positives, and false negatives. A true positive is a prediction where $y_{pred} = y_{true} = 1$, or both true label and predicted label are positive. Similarly, a true negative is a prediction where $y_{pred} = y_{true} = 0$, or both true label and predicted label are negative. A false positive is a prediction where $y_{pred} \neq y_{true}$ and $y_{true} = 0$. Finally, a false negative is a prediction where $y_{pred} \neq y_{true}$ and $y_{true} = 1$.

Accuracy measures the overall correctness of the model by calculating the ratio of correct predictions (both true positives and true negatives) to the total number of predictions, and is defined by:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.26}$$

Precision and recall are particularly important when evaluating the performance of a model on imbalanced data, as they capture performance on different types of input data. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positives that are correctly identified by the model. Precision and recall are defined by:

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}. \tag{2.27}$$

The F1-score is the harmonic mean of precision and recall. It provides a single metric that balances the trade-off between precision and recall and is especially

useful when the class distribution is imbalanced. F1-score is defined by:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.28}$$

Accuracy is an appropriate metric for binary classification only when the classes are balanced, so we rely more heavily on F1-score when we train and evaluate models with imbalanced classes.

Another useful metric is the Receiver Operating Characteristic - Area Under the Curve, or ROC AUC. The ROC AUC reflects how well a model is able to separate the classes by measuring the trade-off between the recall, also known as the true positive rate (TPR), and the false positive rate (FPR) across various classification thresholds. The TPR and FPR are defined by:

$$\text{TPR} = \frac{TP}{TP + FN}, \qquad \text{FPR} = \frac{FP}{FP + TN}. \tag{2.29}$$

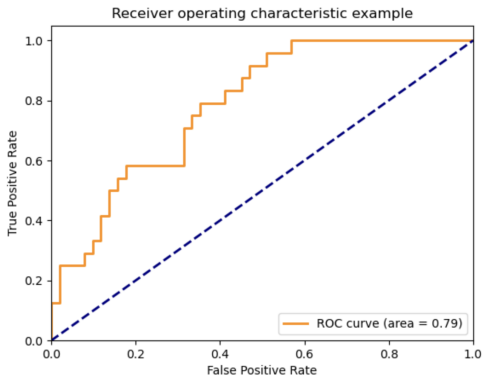


Figure 2.5: Graphical illustration of the ROC curve provided by `sklearn` [3].

As shown in Figure 2.5, the ROC curve is the plot of the TPR against the FPR at

different classification thresholds. The AUC is, as the name suggests, the scalar area underneath this curve. The ROC AUC is a robust metric that can provide a comprehensive view of how a model handles different precision and recall trade-offs.

2.6 Explainability

Explainability is a challenge across all media in machine learning, but particularly so for audio data. A core challenge in explainability for audio is that there is limited natural language for describing audio features [42]. Unlike descriptions for text, image, and video, for which we have an expansive lexicon, there is much less vocabulary dedicated to the description of audio. We can discuss pitch, tone, cadence, loudness, but it can be very difficult to semantically capture the difference between the two speakers of the same sex from the same region speaking the same language. So, in trying to explain the behavior of audio models, we turn to a variety of quantitative and visual approaches. In this section, we introduce explainability methods specific to the models introduced in Section 2.3.

Gradient Boosting Decision Trees With a gradient boosting decision trees, or any other tree-based ensemble method, feature importances can be calculated. As described in Algorithm 1, feature importances are calculated by measuring the model's change in performance after each feature is permuted. To stabilize results, we permute each feature multiple times and use the mean and standard deviation of each feature's importance.

However, the importance of a given feature may be obscured by the permuta-

Algorithm 1 Permutation Importance Algorithm

Input: fitted model m, dataset D, metric a, repeats R
  $s \leftarrow a(m, D)$
  for each feature j in D **do**
    for repeat in $1 \dots R$ **do**
      Randomly shuffle column j in D to create corrupted $\hat{D}_{r,j}$
      $s_{r,j} \leftarrow a(m, \hat{D}_{r,j})$
    end for
    $i_j = s - \frac{1}{R} \sum_{r=1}^{R} s_{r,j}$
  end for

tion feature importance algorithm if multiple features are multicollinear, as is the case for the audio signal features described in the previous section. Intuitively, permuting one feature will have little impact on the model's performance if the same, or very similar, information is available to the model through another non-permuted feature. To combat this issue, we can perform hierarchical clustering on the Spearman rank-order correlations between features and keep a single feature from each cluster. This way, when a feature is permuted, there should be no other non-permuted feature containing duplicate information.

Occlusion Occlusion is a technique used for model explainability, particularly with deep learning models that might otherwise be considered "black boxes". The core idea is to iteratively occlude, or block from view, parts of the input data, measure how the model's prediction changes, and, ideally, identify which parts of the input data are most important.

Consider some input $X = [x_1, x_2, \dots, x_n]$, where X represents the original input and each x_i represents a subsection (perhaps a pixel, patch, or token) in X, and some model f. First, we generate a baseline prediction $\hat{y} = f(X)$, which will serve as the point of comparison. Then, we iteratively mask each x_i from the

input, such that:

$$\mathbf{X}^{(i)}_{\text{occluded}} = \mathbf{X} \odot \mathbf{M}_i, \tag{2.30}$$

where $\mathbf{M}_i$ is a mask that occludes the i-th subregion of the input and $\odot$ represents element-wise multiplication. After this operation, the occluded region will be replaced with some specified value (e.g., 0, 1, or a mean of the feature across all samples). For each occluded input, the model makes a new prediction given by:

$$\hat{y}^{(i)} = f(\mathbf{X}^{(i)}_{\text{occluded}}). \tag{2.31}$$

The intuition is that if a change in the model's prediction is observed when a region is occluded, that region is likely important. After occluding different regions and observing changes in the model's behavior, the results can be visualized in a heatmap.

Attention Visualization As discussed in Section 2.4, transformer models rely on a self-attention mechanism to understand the relationships between different parts of the input sequence. The attention mechanism assigns a weight to each token, which reflects the importance of each token in relation to every other token.

Recall that attention is defined by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.32}$$

where $\mathbf{X}$ is the matrix of input embeddings, $Q = \mathbf{X}W_Q$ is called the query matrix, $K = \mathbf{X}W_K$ is called the key matrix, $V = \mathbf{X}W_V$ is called the value matrix, W_Q, W_K, W_V are learned weight matrices, and d_k is the dimensionality of the

keys [9]. After applying a softmax on the unnormalized attention weights, we are left with a normalized $d_k \times d_k$ matrix of weights that can be visualized as a heatmap. In such a visualization, the $x - axis$ represents position in the input sequence (or token ID), the $y - axis$ represents the tokens for which attention is computed, and the color intensity at some (i, j) represents the attention weight for token i on token j, where greater attention is read to reflect higher importance.

A limitation of attention visualization is that it is done per layer per head, which can make it difficult to observe the overall model's attention. To combat this, in 2021, Abnar et al. proposed "attention rollout" to trace the distribution of attention across multiple or all of the model's layers [18]. This method gives us a more complete view the model's distribution of attention.

Consider an L-layer transformer with attention matrices W_l for $l \in \{1, 2, \ldots, L\}$, where each W_l represents the attention between different tokens at that layer. We compute a cumulative attention matrix by multiplying the attention matrices across the layers, such that:

$$W^{\text{rollout}} = W^{(1)} W^{(2)} \ldots W^{(L)}. \tag{2.33}$$

Once we have computed the cumulative attention matrix W^{rollout}, it can be visualized similarly to the single attention weights. We can also extract the attention weights specific to the `[CLS]` token to understand what parts of the input sequence were most relevant for the final classification.

3 | Related Work

The current state-of-the-art for deepfake audio detection is divided decisively into two streams: the academic and the commercial. As this is an academic work, we will focus largely on recent academic advancements in deepfake audio detection, but we also include a brief overview of the commercial landscape.

Progress in deepfake audio detection coalesces around two benchmark challenges: ASVspoof and Audio Deepfake Detection (ADD). As a result, almost all the work in this field is highly biased towards succeeding in these competitions, which furnish clean, high-quality audio tracks.

The biannual ASVspoof competition series began in 2015 to address vulnerabilities in automatic speaker verification (ASV) systems caused by spoofing attacks. The first few editions, ASVspoof 2015, 2017, and 2019, focused on speech synthesis and replay attacks transmitted over telecommunication channels. Deepfake audio detection was added at ASVspoof 2021, and ASVspoof5 (2024) includes a wide variety of adversarial attacks and explores new challenges with spoofing-robust automatic speaker verification.

The ADD challenge series began in 2022 and intends to be an annual challenge. Unlike the ASVspoof challenge series, whose primary focus is automatic speaker verification, ADD concerns itself only with deepfake audio.

Academic work tends not to account for on-the-ground issues in deepfake audio detection, such as low quality audio and common techniques used by bad actors to circumvent detection [36]. Commercial purveyors of deepfake audio detectors

are, naturally, more focused on real-world use cases of economic value, but their methods are difficult to verify and compare as they typically do not release many details about their implementations, training sets, or metrics. This opacity makes it unclear how meaningful their classifications truly are. In private communication with a senior audio professional employed by a leading media company, we learned that such commercial offerings are commonly seen as 'intransparent' and 'untrustworthy' in the field. The same senior audio professional reported the us that, when supplied with an audio track, most of the commercial purveyors will return to the user a "percentage fake score". However, discussions with salespeople from two commercial purveyors betrayed that they are unable to define exactly what this score means. It is not clear, for example, whether this means that some subset percentage of the audio is deepfake, that their model is some percent sure that the sample contains deepfake audio, or something else entirely. The commercial purveyors then seem to be preying on a layperson's inexperience or inability to closely inspect the meaning of the produced score.

3.1 Academic Approaches to Deepfake Detection

In the past, academic approaches have focused on achieving best performance on narrow tasks, such as classifying one specific type of deepfake audio, and few have attempted to measure a model's generalization ability or performance given low-quality audio [40, 36]. As a result, the academic state-of-the-art is likely not particularly useful for the task of trying to identify deepfake audio in the wild. Nevertheless, we review current methods in deepfake audio classification.

Traditional Machine Learning Approaches The early days of deepfake audio detection were dominated by traditional classifiers, such as support vector machines (SVM) and Gaussian mixture models (GMM), and traditional signal processing features. These works generally use different subsets of the hand-crafted features described in the previous chapter, and they generally perform well with academic datasets, in which the distributions of features between real and deepfake audio are relatively easily separable. Observing that these models will likely not generalize well when presented with deepfake audio from unseen distributions, Zhang et al. propose using an SVM to learn a tight boundary around the features of real audio. In their work *One-class Learning Towards Synthetic Voice Spoofing Detection*, they propose learning this boundary by only training on real audio [29]. This method works impressively well, out-performing almost all other methods on the ASVspoof 2019 data, but the authors hypothesize that it is unlikely to perform as well when the true audio varies more widely (e.g., with non-English speech, with speakers with accents, with non-adult speakers, etc.).

Recent work also suggests that the GBDT may be more robust to unseen data as well as boast faster inference times than both SVM and GMM [35]. In a 2023 work by Bird et al., the authors demonstrate the power of the GBDT, reporting accuracies of 99.3% on the DeepVoice dataset and inference times of 0.004 seconds for 1 second of input speech [35]. In their work *Real-Time Detection of AI-Generated Speech for Deepfake Voice Conversion*, they also explore features importances and the statistical characterizations of real and deepfake audio. They observe that the fifth MFCC feature is critical to their GBDT's performance on the DeepVoice dataset. They also observe that the features of real audio samples

have a greater standard deviations than those of deepfake audio samples. Another recent work by Togootogtokh et al., published in January 2024, employs a GBDT for deepfake audio classification task with a custom dataset comprised of true samples from the LJ Speech Dataset and deepfake samples generated with various HuggingFace TTS models [44].

CNN-Based Classifiers The most popular architecture for deepfake audio detection is by far the convolutional neural network (CNN) based classifier [36, 21, 11, 17, 7]. CNN classifiers have been historically favored for their ability to combine spatial and temporal information through convolution. The most successful of these is called *Light Convolutional Neural Network* (LCNN) and was proposed by Wu et al. in 2020 [21]. LCNN consists of convolutional and max-pooling layers with Max-Feature-Max (MFM) activation. Wu et al. choose the MFM activation function instead of the arguably standard ReLU function because they observe that the MFM learns more compact features than the sparse high-dimensional ones learned with ReLU [21]. This distinction is what makes their CNN "light". The LCNN has been incredibly successful in recent ASVspoof and ADD competitions; it was the best system at ASVspoof 2017 and continued to be the best system in one of the subtasks of ASVspoof 2019. Another successful CNN architecture, proposed by Dinkel et al., uses raw waveforms as input to a convolutional long short-term neural network. Their model combines time-convolving layers with frequency-convolving layers to "reduce time and spectral variations" with long-term temporal memory layers to capture longer-term temporal relationships [36, 7]. A variety of other CNN architectures have been proposed, but all of them generalize poorly to unseen attacks [36].

Self-Supervised Embedding-Based Classifiers There are a few recent works that use self-supervised embedding features as the basis of their classification algorithms, a number of which use Wav2Vec features [26]. For example, Xie et al. use Wav2Vec features as input to a Siamese neural network that they train to distinguish whether the speech samples in a pair belong to the same category. This work, published at INTERSPEECH 2021, reported state-of-the-art results on the ASVspoof 2019 dataset. Martin-Donas et al. use deep features extracted from pretrained XLS-R, a variation of Wav2Vec, and a finetuned transformer classifier to win the ADD 2022 challenge [31]. Some other recent works uses HuBERT features, and Wang et al. compare Wav2Vec-, XLS-R-, and HuBERT-based features [33].

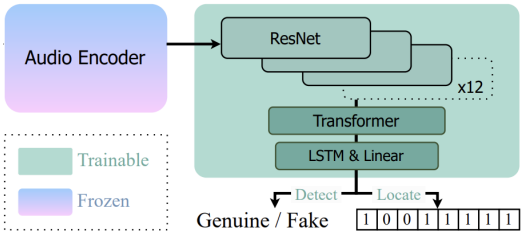


Figure 3.1: Illustration of the FakeSound model architecture proposed by Xie et al.[46]

Deepfake Audio In-Painting Another relevant research area is that of deepfake audio in-painting detection. In-painting, or splicing, refers to the practice of mixing real and fake audio such that only small portions of an audio sample are actually manipulated. Detecting deepfake audio in-painting requires not only identifying that a sample contains some corrupted audio but also identifying the timestamps at which the corruption begins and ends. The most recent work in this field was is Xie et al. and is to be published as INTERSPEECH 2024 (Sept 1-5,

2024). In this work, the authors propose a framework called EAT, which incorporates a ResNet, a two-layer transformer encoder, a single-layer bidirectional Long Short-Term Memory network (LSTM), and a final classification layer, as seen in Figure 3.1.

The EAT framework achieves an F1 score of 98% when classifying segments at 20 millisecond resolution [46]. However, Xie et al. only evaluate their method on a custom dataset, do not attempt to evaluate their architecture's performance on any of the standard deepfake audio benchmarks, and focus primarily on deepfake environmental sounds and background audio [46]. A slightly older but broader work, from 2022, that investigates in-painted deepfake audio is that of Cai et al. in *Waveform Boundary Detection for Partially Spoofed Audio* [30]. Cai et al. use a combination of Wav2Vec and MFCC features as their input to a series of single-dimensional CNNs, a transformer encoder, a bidirectional long short-term memory network (BiLSTM), and a final linear layer for classification [30]. With their method, they achieve the best performance in the locating manipulated clips task of the ADD 2022 challenge.

3.2 Commercial Services for Deepfake Detection

While we have reviewed popular methods in academic literature for deepfake audio detection, most journalists, law enforcement officials, or other concerned voters will generally not be able to implement the academic state-of-the-art. Instead, they will rely on commercially-available deepfake detectors.

Resemble AI is the leading commercial purveyor of deepfake audio detection

technology, and they market themselves largely to law enforcement and other governmental bodies. Their model, DETECT-2B, is an ensemble method that combines multiple self-supervised audio representation models (such as EnCodec and Wav2Vec) with transformers they have finetuned for the deepfake audio detection task [41]. Beyond finetuning an unspecified number of pre-trained models, they add a final sequence-modelling layer as the final decision layer. At inference time, the ensemble method makes predictions for overlapping 6-second time slices across the duration of the entire audio clip. They then aggregate those predictions to make a final classification depending on the user's desired specificity [41]. Recently, Resemble AI has attempted to integrate state-space modelling, an approach related to transformer methods that incorporates stochasticity in the interest of greater throughput, but they have not found that that approach performs as well as their ensemble of pre-trained transformers method [41]. According to a Resemble AI spokesperson, they require 6 seconds of audio to make a classification.

As far as we know, none of the commercial purveyors return any additional insight or explanation of the classification to the user. The combination of opaque methods and lack of explainability make it incredibly difficult for a non-technical person to get any kind of trustworthy classification or actionable insight into deepfake audio they may have encountered.

3.3 Audio Explainability

Beyond the model-specific explainability methods introduced in Section 2, the two most common methods for explainable AI (XAI) are LIME and SHAP [42].

Introduced by Ribeiro et al. in 2016, LIME, or Local Interpretable Model-agnostic Explanations, endeavours to explain a complex model's behavior by approximating it around the prediction of interest with a simpler, more interpretable model [6].

This technique involves generating perturbed variations of the original instance by randomly sampling the feature space for some subset of the features. The model is then queried with the perturbed data, and the response is recorded. LIME then fits a local surrogate model, generally a weighted linear regression model, to approximate the behavior of the black-box model around the specific instance. The closer the sample is to the original instance, the higher the weights are. This is typically achieved with an exponential kernel:

$$w(z, x) = \exp\left(-\frac{D(z, x)^2}{\sigma^2}\right), \tag{3.1}$$

where z is a perturbed instance, x is the original instance, $D(z, x)$ is the distance between z and x, and σ is the kernel width parameter.

Introduced by Lundberg et al. in 2017, SHAP, or SHapley Additive exPlanations, leverages Shapley values, originally developed in cooperative game theory to fairly distribute a total payout among players based on their contributions [8]. SHAP assigns each feature an importance value based on its contribution to the model's prediction [42]. Mathematically, a feature i's importance is calculated as the average marginal contribution of feature i across all possible subsets S of the

features:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left(f(S \cup \{i\}) - f(S) \right), \tag{3.2}$$

where F is the set of all features, S is a subset of F, $f(S)$ is the model's output for the feature subset S, $f(S \cup \{i\})$ is the model's output when feature i is added to subset S, and ϕ_i is the SHAP value for feature i. Thus, the importance of feature i is measured as the average change in model output when feature i is added to all possible subsets of features.

An issue with both the SHAP and LIME techniques is that they both tend to perform poorly when the features are highly correlated, which is almost always true when working with audio data [30]. As a result, we focus on the model-specific explainability methods introduced in Section 2.

4 | Methods

In this section, we describe the datasets, models, libraries, and hardware used to conduct experiments. We also describe relevant model hyperparameters and settings, which we believe should enable the reproducibility of these experiments.

4.1 Data Sources

We employ a variety of publicly available datasets to conduct our experiments. In some cases, we combine and customize them in order to better mimic real-world use cases and challenges.

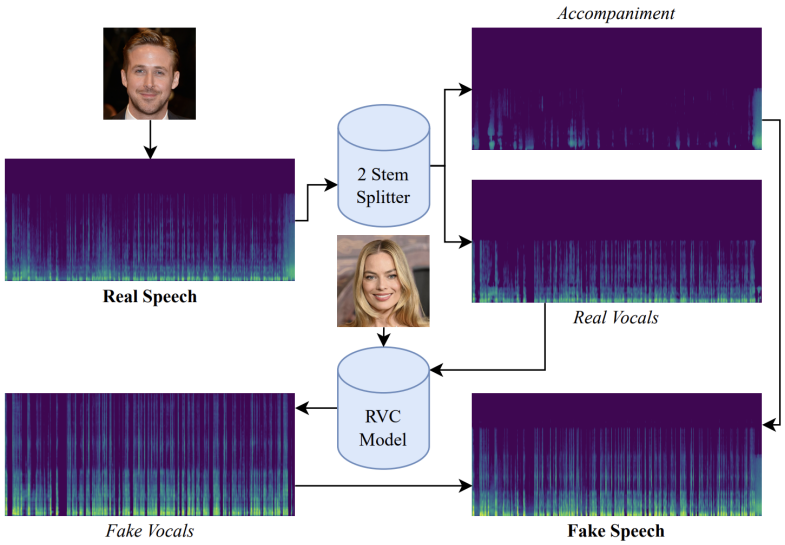


Figure 4.1: Diagram of the retrieval-based voice conversion and stem splitter method employed Bird et al. to generate the DeepVoice dataset[35].

DeepVoice The DeepVoice dataset contains English-language human speech from eight different speakers, as well as deepfake versions of those same speeches

created with Retrieval-based Voice Conversion (RVC), an open-source voice conversion algorithm. The DeepVoice dataset contains real speech from Joe Biden, Ryan Gosling, Elon Musk, Barack Obama, Margot Robbie, Linus Sebastian, Taylor Swift, and Donald Trump. Figure 4.1 depicts the process by which the deepfake speech is generated. Each real speech also contains "accompaniment" (e.g., applause, microphone feedback), which the creators of DeepVoice strove to maintain but not amplify in the RVC-generated deepfake speech. To that end, before using the RVC model to generate deepfake speech, the accompaniment is separated from the vocals with a 2-stem splitter and then recombined with the deepfake vocals [35]. As the speech samples vary widely in length, each sample was cut down to segments of 3 seconds for use in this study.

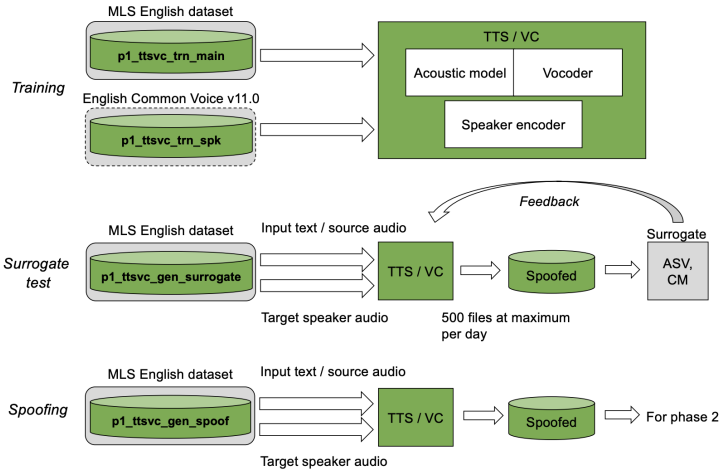


Figure 4.2: Diagram capturing the training, testing, and spoofing methods through which ASVspoof5 data is generated[45].

ASVspoof5 The ASVspoof5 dataset is a state-of-the-art dataset containing eighteen different varieties of deepfake audio as well as true speech samples [45]. Each sample is labelled with a classification as “bonafide” or “spoof”. If the sample

is spoofed, the attack method that was used to generate the sample is specified. The dataset, which was released in June 2024, contains 182 357 train samples and 142 134 test samples. As seen in Figure 4.2, each deepfake (or spoofed) audio sample is generated with a novel VC or TTS method, which were trained on two English-language datasets. The final deepfake samples are made using the English-subset of the Multilingual LibriSpeech (MLS) dataset [45]. The 18 different attack types included in this dataset make it the most attack diverse of the datasets we consider in this study. We take this dataset to represent the state-of-the-art in deepfake audio generation and, in subsequent sections, refer to it simply as the ASVspoof dataset.

FakeAVCeleb The FakeAVCeleb dataset is a standard in the deepfake audio detection repertoire, but it is now slightly out-of-date, as it was released in 2021 [24]. It contains both deepfake audio and video, but we use only the audio component in this study. The audio subset of the FakeAVCeleb dataset contains 9 712 real audio samples and 10 843 deepfake audio samples. The language of each audio sample is English, but FakeAVCeleb includes balanced classes of male and female speakers as well as speakers who self-identify as African, East Asian, South Asian, Caucasian (American), and Caucasian (European) [24]. This makes FakeAVCeleb the most linguistically diverse of the datasets considered in this study.

4.1.1 Custom Configurations

We augment the DeepVoice and ASVspoof datasets in order to study model behavior in more diverse data settings. When shortening or mixing audio samples, we are careful to use each audio sample only once and only for either training or

testing. In other words, if we use a 1-second sample of a longer audio track, no other sample will be extracted from that same audio track.

DeepVoice The DeepVoice dataset is useful for creating a dataset with "spliced" fakes, in which most of an audio sample is real but a small subset of it is fake. The DeepVoice dataset was well-suited to this task because it includes only eight speakers. To make the spliced, or in-painted, samples, 0.3 seconds of true audio from each 3-second sample was replaced by 0.3 seconds of fake audio with the sample speaker. For example, to generate a spliced sample, we might use a 3-second bonafide sample of Margot Robbie, remove a random 0.3 second subset, and replace that 0.3 seconds of real audio with 0.3 seconds of voice-cloned Margot Robbie. This dataset is used to evaluate the various models abilities to detect audio samples with deepfake subsamples.

ASVspoof The audio samples provided by ASVspoof are clean. The ASVspoof bonafide samples were created in recording studios with high quality microphones; The ASVspoof fake samples were generated with TTS algorithms that add no additional noise. When deepfake audio is circulated over the social media, it undergoes compression–often multiple times. In order to identify fake audio downstream, our models must be robust to this kind of distribution modification. Additionally, bad actors will try to obscure as much as possible that an audio sample has been faked. A common approach to obscure fake audio is to play to audio aloud in a room and re-record the audio before disseminating it. We create two additional datasets from the ASVspoof dataset for an additional challenge.

Compressed ASVspoof To tackle the issue of compression, we write all of the audio samples in the ASVspoof dataset to the lossy MP3 format (from the lossless FLAC format) with a bitrate of 128k. This shrinks each file, on average, to 33.7% of its original size.

Rerecorded ASVspoof To tackle the issue of re-recording, we re-recorded audio samples in the ASVspoof dataset by playing them aloud on a 2021 MacBook Pro in a large, closed stone-walled room while simultaneously recording.

4.2 Models & Hyperparameters

In this thesis, we experiment with three models: gradient boosting decision trees (GBDT), the Audio Spectrogram Transformer (AST), and a Wav2Vec-based transformer (Wav2Vec). Here, we make explicit the hyperparameters and settings relevant for our experiments.

4.2.1 Gradient Boosting Decision Trees

Given its recent popularity for deepfake audio classification tasks [35, 44], we use the gradient boosting classifier as the baseline in this study. The features for these tests are the hand-crafted, signal processing features described in Section 2. We use a total of 37 features, which include 20 MFCC features, 12 chroma features, spectral bandwidth, spectral roll-off, spectral centroid, ZCR, and RMS. The features for each audio sample are calculated by averaging each feature's values across all frames. For example, the final ZCR is calculated by averaging the ZCR for each frame across the length of the entire audio sample.

To evaluate best possible performance with the GBDT, we do a hyperparameter search across maximum tree depths and maximum number of estimators. We consider maximum tree depth in $\{3, 8, 10, 15, 25\}$ and number of estimators in $\{10, 100, 200, 400, 600\}$. We do our hyperparameter search with the ASVspoof dataset. We conduct tests with $10\,000$ total data samples, 33% of which are held out as a test set. As GBDTs are highly sensitive to unbalanced data classes, we balance classes before conducting experiments with the GBDT.

Table 4.1: Accuracy of gradient boosting classifier for various maximum tree depths and number of estimators.

		Number of Estimators					
		10	100	200	400	600	1000
Max Depth	3	0.748	0.822	0.838	**0.850**	0.856	0.859
	8	**0.792**	**0.840**	**0.854**	**0.860**	**0.859**	**0.862**
	10	0.787	0.836	0.845	**0.848**	0.848	0.848
	15	0.745	0.791	0.814	**0.814**	0.814	0.814
	25	0.725	0.732	0.736	**0.736**	0.736	0.736

As shown in Table 4.1, the classification accuracy is highest with shorter decision trees and more estimators when trained with 3-second audio samples. The performance stabilizes with 400 or more estimators, so we take 400 estimators and maximum depth of 8 as our optimal hyperparameter setting. We find that this configuration also offers best performance for other audio sample lengths.

As seen in Figure 5.1, performance is also dependent on the number of estimators. We observe that performance improves with the number of estimators, but we also see diminishing returns after 400 estimators. We conclude that 400 estimators and a maximum tree depth of 8 are the optimal hyperparameters for this training scenario.

Table 4.2: Hyperparameters used in the finetuning of Wav2Vec and AST transformers.

Hyperparameter	Value
Batch Size	32
Learning Rate	3×10^{-5}
Training Steps	40
Train-Test Split	0.33
Weight Decay	0.0
Warm Up Ratio	0.1
Optimizer	$AdamW$ [12]

4.2.2 Transformer Models

Both AST and Wav2Vec transformer models use their own specialized encodings. Otherwise, they are finetuned very similarly. Table 4.2 reports the hyperparameters used when finetuning the Wav2Vec and AST models, as well as the experiments performed with each model and dataset combination.

4.2.3 Libraries & Packages

This thesis builds upon many open-source libraries and packages. To load audio data and generate traditional signal-processing features, we use the `librosa` library [5]. The GBDT implementation is a product of `sklearn` [3]. We also use `sklearn` to calculate the performance metrics (accuracy, precision, recall, F1 score, and ROC AUC). To load the AST and Wav2Vec feature encoders and pretrained models, we use `huggingface` [20]. To do the many matrix operations involved in this thesis, we use `pytorch` [13].

4.2.4 Hardware

All the experiments for this thesis were conducted on the Microsoft Azure Platform with credits supplied by the BBC.

The GBDT experiments were conducted with `NCasT4_v3`-series virtual machines (VM), which are powered by Nvidia Tesla T4 GPUs and AMD EPYC 7V12(Rome) CPUs. These VMs feature 4 NVIDIA T4 GPUs with 16 GB of memory each, 64 non-multithreaded AMD EPYC 7V12 (Rome) processor cores, and 440 GiB of system memory. The explainability experiments and visualizations were also completed on a `NCasT4_v3` VM.

Both AST and Wav2Vec finetuning was conducted with `NC_A100_v4`-series VMs, which boast up to 4 NVIDIA A100 PCIe GPUs with 80 GB memory each, up to 96 non-multithreaded AMD EPYC Milan processor cores and 880 GiB of system memory. For these experiments, we used only 1 NVIDIA A100 PCIe GPU.

5 | Experiments

In this study, we work most often with the ASVspoof data as it represents the state-of-the-art for deepfake audio generation. However, as the ASVspoof 5 competition has not yet completed and there is not yet a benchmark for that data, we demonstrate the comparative performance of our AST and Wav2Vec models on the FakeAVCeleb dataset.

Table 5.1: Accuracy and ROC AUC comparison of various competitor models.

Model	Accuracy	ROC AUC
EfficientNet-B0 [16]	0.500	-
MesoInception-4[10]	0.540	-
VGG16 [4]	0.671	-
Xception [14]	0.763[1]	0.853
AD DFD	-	0.881
LipForensics [19]	-	0.911
FTCN [25]	-	0.931
AVAD [38]	-	0.945
RealForensics [34]	-	0.971
FACTOR [37]	-	**0.974**
AST (ours)	0.979	0.985
Wav2Vec (ours)	**0.991**	**0.990**

Table 5.1 lists the ROC AUC or accuracy, depending on metric reported, of competitor models on the FakeAVCeleb data. The AST and Wav2Vec models outperform all other models that report results on this dataset.

We want to emphasize that this is a state-of-the-art result on the FakeAVCeleb dataset. Given the impressive performance of the Wav2Vec and AST models with the FakeAVCeleb dataset, we reflect on why these results have not yet been pub-

[1] Unimodal (audio-only) result.

lished. In Section 3, we described works that rely on self-supervised feature embeddings, such as Wav2Vec, but these works also combine self-supervised feature embeddings with other model architectures. Recall, for example, the work by Xie et al., which uses Wav2Vec embeddings as input to a Siamese neural network [28]. We suspect that other researchers avoid publishing results as simple as those we have just presented because they fear the contribution is not sufficiently novel. In this thesis, we find that using AST and Wav2Vec off-the-shelf is sufficient for state-of-the-art performance. In the remainder of this thesis, we pivot to adjacent research questions.

In the following sections, we explore the effects that manipulations of the input data have on model performance. We also measure our models' abilities to generalize to unseen data. For convenience, we account for all experiments in Table 5.2.

Table 5.2: Transformer experiments conducted with various datasets and audio sample lengths. The sets represent the length of audio samples in seconds considered for each dataset.

Dataset	GBDT	Wav2Vec	AST
ASVspoof	$\{0.1, 0.5, 1, 3, 6\}$	$\{0.1, 0.5, 1, 3, 6\}$	$\{0.1, 0.5, 1, 3, 6\}$
Compressed ASVspoof	$\{0.1, 0.5, 1, 3, 6\}$	$\{0.1, 0.5, 1, 3, 6\}$	$\{0.1, 0.5, 1, 3, 6\}$
Rerecorded ASVspoof	$\{0.1, 0.5, 1, 3, 6\}$	$\{0.1, 0.5, 1, 3, 6\}$	$\{0.1, 0.5, 1, 3, 6\}$
DeepVoice	$\{3\}$	$\{3\}$	$\{3\}$
Spliced DeepVoice	$\{3\}$	$\{3\}$	$\{3\}$
FakeAVCeleb	$\{6\}$	$\{6\}$	$\{6\}$

5.1 In-Painted Deepfake Audio

Responding to the situation that arose in this year's Pakistani elections, in which a recording of politician with in-painted deepfake audio was circulated, we attempt

to detect the presence of in-painted deepfake audio in 3-second samples, in which 10% of the sample is fake. We use the spliced DeepVoice dataset for the following experiments.

Table 5.3: GBDT evaluation performance on in-painted audio samples.

Class	Precision	Recall	F1
Bonafide	0.64	0.65	0.64
Spoof	0.64	0.63	0.64

Attempts to detect the presence of in-painted audio samples with the GBDT, perhaps unsurprisingly, were not successful. The GBDT achieved evaluation accuracy of 64%, and Table 5.3 reports the similarly disappointing precision, recall, and F1 scores for this experiment. As each GBDT feature is generated by taking the mean across all the frames in an input, it is likely that the variance in features that could be observed from the small subsets of in-painted deepfake audio are drowned out by the features of the rest of the legitimate audio. In contrast, the AST and Wav2Vec models are quite successful at this task, achieving 97.3% and 96.7% evaluation accuracy respectively, when trained for 20 epochs and evaluated with balanced classes. However, in line with the findings of Cai et al. [30], neither of the transformer models were able to produce reliable time-stamps for the beginning and end of the in-painted video. So, in order to detect and report the locations of small regions of deepfake audio, we turn to evaluating the length at which deepfake audio can no longer be detected reliably.

In addition, we hope that by studying the length of audio sample required to make a meaningful classification, we can develop intuition about the relative importance of temporal features and acoustic features in deepfake audio detection. In other

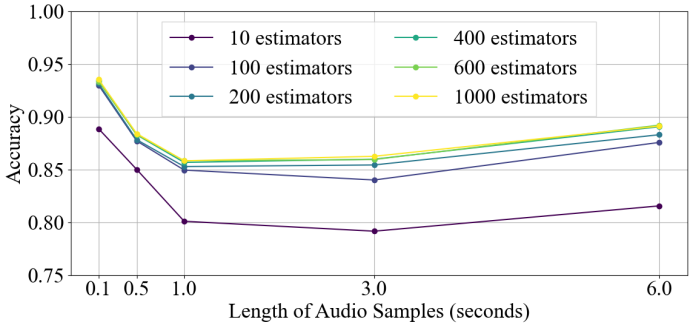


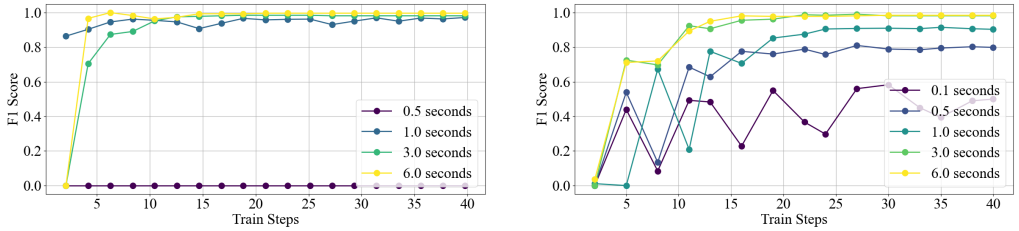
Figure 5.1: Accuracy across number of estimators and audio sample length with fixed tree depth of 8.

words, is the trace of deepfake audio only discernible when evaluated relative to the rest of a longer audio sample? Or, does deepfake audio leave a signature observable in even the smallest clips of audio?

5.2 Variable Audio Sample Lengths

We conduct tests with various lengths of audio samples and evaluate the performance of the GBDT, AST, and Wav2Vec models. Following the optimal hyperparameter settings discussed in Section 4, we employ a GBDT with maximum tree depth of 8 and 400 estimators to set baseline results for the following experiments. Figure 5.1 reports the performance of the classifier on audio samples of different lengths. Recall that when training the gradient boosting classifier, we balanced the classes between real and fake audio, so the accuracy baseline is 50%. The GBDT classifier is able to achieve, at best, 94% evaluation accuracy. The model performs best with audio samples of 100 milliseconds, the length of approximately one frame, and then performance declines for samples of length between 0.5 and 3 seconds. Model performance improves again given 6-second audio samples.

Recall that the final features for this model are generated by taking the mean of the features calculated for each frame along the entire length of the audio sample. It seems that there may be a "sweet spot" of including enough information in the sample that the model can make a meaningful observation but not including so much information that meaningful features are diluted.



(a) Wav2Vec (b) AST

Figure 5.2: F1 scores over train steps for the fine-tuned transformer models.

We compare the performance of the GBDT with that of the transformer models. As seen in Figure 5.2, the AST performs quite well for audio samples with length 0.5 seconds or longer. The same is true for the Wav2Vec model for audio samples of length 1 second or longer.

Table 5.4 reports the evaluation performance of the AST and Wav2Vec models for each of the audio sample lengths considered. The Wav2Vec model is not able to make predictions on 0.1-second audio samples as the Wav2Vec feature generator cannot make enough tokens for the required input sequence length from such a short audio sample. In Table 5.4, this is reflected in the Wav2Vec row for the 0.1-second experiment with NaNs. With 0.5-second samples, the Wav2Vec model is able to make tokens, but it is not able to learn anything from them; The Wav2Vec model makes predictions only for one class over the entirety of the training period.

Table 5.4: Comparing the precision, accuracy, recall, and F1 of the models AST and Wav2Vec for various audio lengths measured in seconds.

Audio Length (s)	Model	Precision	Accuracy	Recall	F1
0.1	AST	0.640	0.806	0.410	**0.500**
	Wav2Vec	NaN	NaN	NaN	NaN
0.5	AST	0.825	0.903	0.774	**0.799**
	Wav2Vec	0.000	0.773	0.000	0.000
1.0	AST	0.946	0.955	0.864	0.903
	Wav2Vec	0.986	0.958	0.958	**0.972**
3.0	AST	0.994	0.991	0.969	0.981
	Wav2Vec	0.964	0.991	1.000	**0.982**
6.0	AST	0.981	0.992	0.987	0.984
	Wav2Vec	0.993	0.999	1.000	**0.997**

The AST is able to make and classify the input sequence, but the poor performance on the 0.1-second samples suggest that there may simply not be enough information available to make meaningful classifications. This poses an interesting contrast the the performance of the GBDT with 0.1-second audio samples. The AST model, in contrast to the Wav2Vec model, is able to learn something from the 0.5-second samples, which suggests that many deepfake audio samples have a signature that can be detected locally without greater temporal context. With 1-second samples or longer, the Wav2Vec model outperforms the AST model, but both models perform extremely well on the evaluation data.

5.3 Augmentation and Degradation

In this section, we evaluate the impact that file degradation has on classification performance. Recall that audio files typically undergo multiple rounds of com-

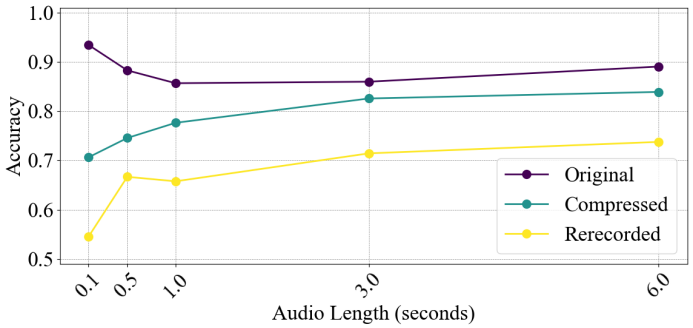


Figure 5.3: Accuracy over various audio lengths for the original, compressed, and rerecorded audio data for booster graphs with 10 000 datapoints, max depth of 8, and 400 estimators.

pression when circulated on social media and that a common method of avoiding deepfake audio detection is to rerecord deepfake audio samples. Thus, we are interested in the effect on classification performance that these data augmentations have. Here, we focus exclusively on the ASVspoof data, with modifications for compression and rerecording.

We evaluate the performance of the gradient boosting classifier on compressed audio samples with the optimal hyperparameters of 400 estimators and maximum tree depth of 8. Figure 5.3 shows that performance is significantly worsened by both compression and rerecording. The difference in performance is particularly stark on short audio samples, where the gradient boosting classifier excels with the original, unmodified ASVspoof data.

We then evaluate the AST and Wav2Vec models performance with the augmented ASVspoof data. Table 5.5 reports the classification accuracy of all three models' performances on the original, compressed, and rerecorded ASVspoof data. Impressively, the Wav2Vec model suffers from no performance degradation when

Table 5.5: Comparing the precision, accuracy, recall, and F1 of the models GBDT, AST, and Wav2Vec for 6.0-second samples of original, compressed, and rerecorded data.

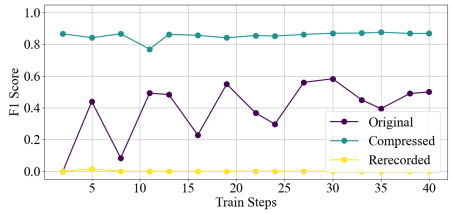
augmentation	model	precision	accuracy	recall	F1
original	GBDT	0.903	0.896	0.891	0.894
	AST	0.981	0.992	0.987	0.984
	Wav2Vec	0.993	0.998	1.000	**0.997**
compressed	GBDT	0.853	0.841	0.837	0.841
	AST	0.994	0.994	0.982	0.988
	Wav2Vec	0.995	0.998	1.000	**0.997**
rerecorded	GBDT	0.589	0.589	0.589	0.589
	AST	0.968	0.991	0.994	0.981
	Wav2Vec	0.998	0.995	0.996	**0.997**

trained with the augmented 6-second audio samples.

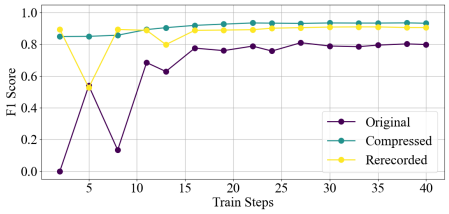
We also evaluate the effect that data augmentation has on the models' abilities to classify audio samples of different lengths. Figures 5.4 and 5.5 illustrate the performance of the AST model and Wav2Vec models across different training steps for original, compressed, and rerecorded audio data, respectively. The F1 score is used as the evaluation metric, with higher values indicating better performance. Interestingly, the AST model performs better with the compressed data for every sample length. This gap is particularly pronounced with the shorter sample lengths of 0.1, 0.5, and 1.0 seconds, as seen in Figures 5.4a, 5.4b, and 5.4c.

5.4 Generalization

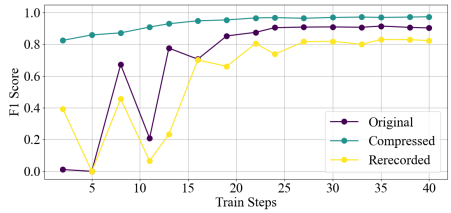
In this section, we investigate the models' abilities to generalize to unseen data and attacks. We use 6-second audio samples for the following experiments. As the ASVspoof data is the most diverse from an attack perspective, we use models



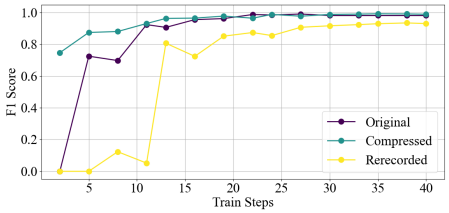
(a) 0.1 seconds



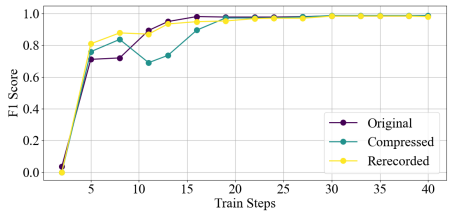
(b) 0.5 seconds



(c) 1.0 second



(d) 3.0 seconds

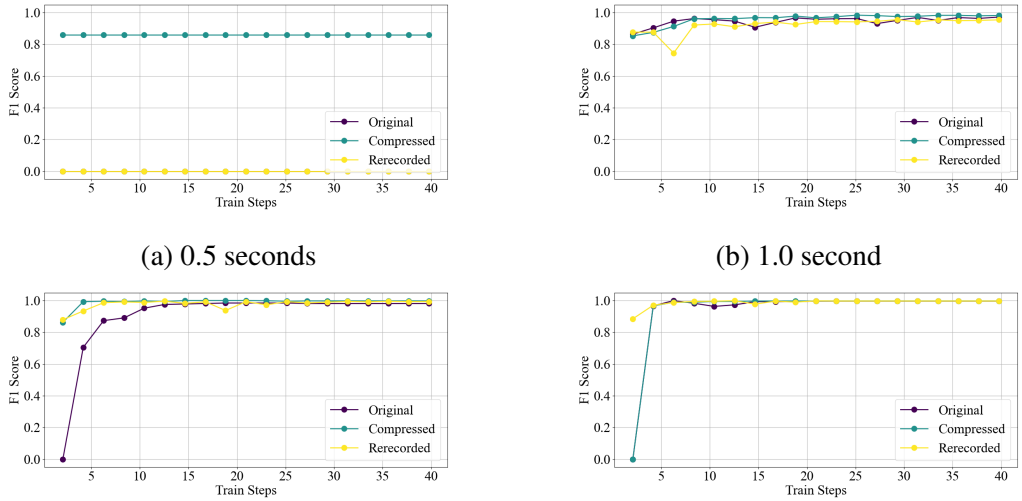


(e) 6.0 seconds

Figure 5.4: AST evaluation F1 scores across data augmentation and sample length.

trained with ASVspoof data to classify FakeAVCeleb data. Again, we compare performance between our baseline classifier, the GBDT, and the two transformer methods. We use $3\,000$ evaluation samples from FakeAVCeleb, balancing the classes such that there are $1\,500$ bonafide audio samples and $1\,500$ spoof audio samples.

When using the GBDT classifier trained on 6-second samples of the original ASVspoof dataset, we observe that an overall accuracy of 51%, which indicates that the GBDT does essentially no better than random guessing between the two



(a) 0.5 seconds

(b) 1.0 second

(c) 3.0 seconds

(d) 6.0 seconds

Figure 5.5: Wav2Vec evaluation F1 scores across data augmentation and sample length.

classes. Table 5.6 reports the precision, recall, and F1 scores for both bonafide and spoof audio for all three models. The transformer methods perform much better: the AST and Wav2Vec models achieve an overall accuracy of 85% and 81%, respectively, on the FakeAVCeleb evaluation data.

Interestingly, the Wav2Vec model's predictions on the unseen data are not at all balanced. The high precision but low recall on the spoof class reveal that the Wav2Vec model missed many of the deepfake instances in the FakeAVCeleb data. The AST model is better balanced and achieves F1 scores of 85%.

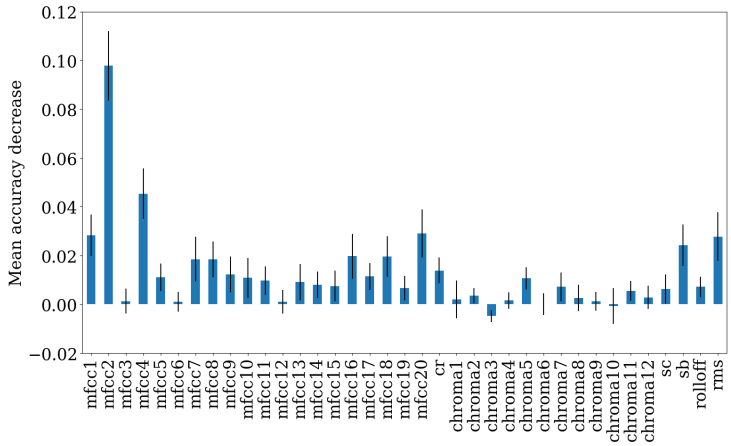
5.5 Explainability

In this section, we try to explain the behavior of our models in the previous sections. The methods presented in this section are model-specific; they all leverage

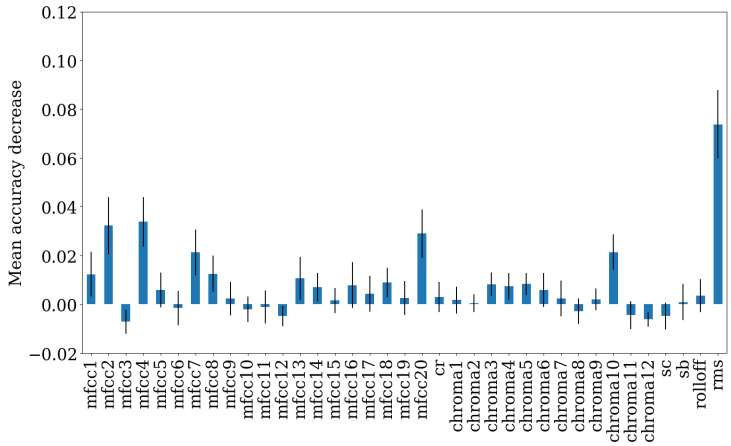
Table 5.6: Performance comparison of ASVspoof-trained models on FakeAVCeleb data.

Model	Class	Precision	Recall	F1
GBDT	bonafide	0.50	0.58	0.54
	spoof	0.51	0.51	0.51
AST	bonafide	0.85	0.84	**0.85**
	spoof	0.85	0.86	**0.85**
Wav2Vec	bonafide	0.73	**0.98**	0.84
	spoof	**0.97**	0.63	0.77

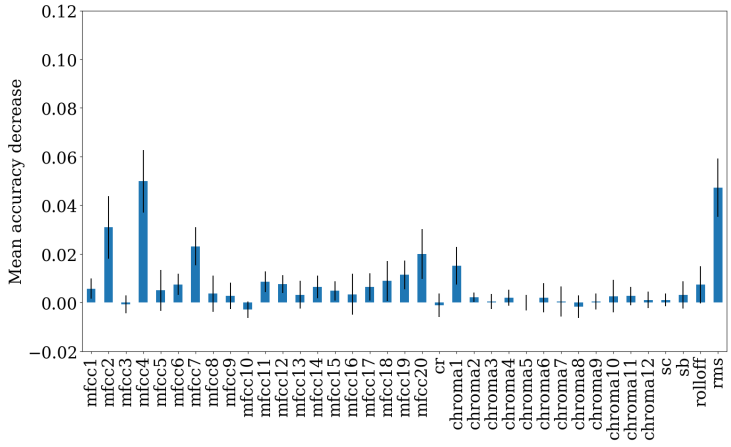
specific attributes of the model they endeavour to demystify. They are also highly visual as we posit that this is the best method of conveying highly technical information to non-technical readers.



(a) 1.0 second



(b) 3.0 seconds



(c) 6.0 seconds

Figure 5.6: GBDT feature importances as measured by mean accuracy decrease with standard deviations for the 1.0, 3.0, and 6.0-second classifiers.

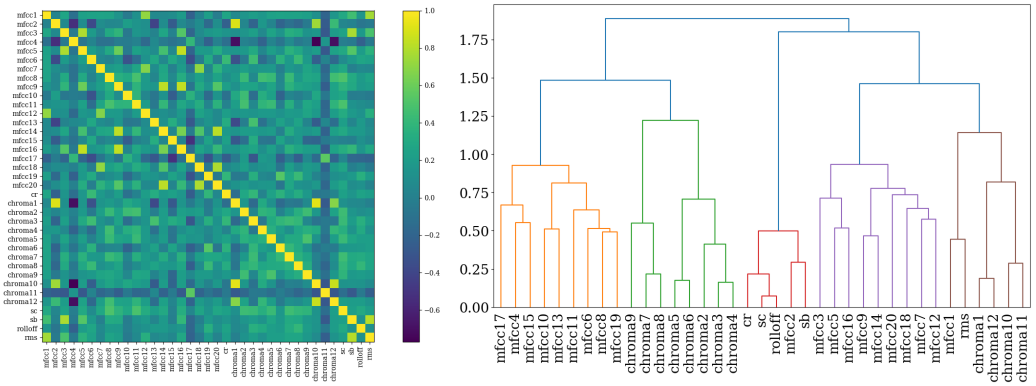
5.5.1 Feature Importances

As discussed in Section 2, an advantage of ensembles of decision trees, as the GBDT is, is the ability to calculate feature importances. In their recent deepfake audio classification with GBDT work, Bird et al. report feature importances and draw meaning from them [35]. In this thesis, we attempt to do the same thing–to explain the behavior of the GBDT and to isolate some aspect of the audio sample as a signature of its deepfake classification.

Following the permutation importance algorithm described in Algorithm 1, we compare the GBDT feature importances for models trained with 1.0, 3.0, and 6.0-second audio samples. As shown in Figure 5.6, in all three cases, the second MFCC, fourth MFCC, and RMS features are the most influential in the GBDT's decision-making. Recall that the RMS feature is most closely associated with the loudness of an audio sample. We find the high importance of the RMS troubling as our intuition is that loudness is not inherently a characteristic of deepfake audio.

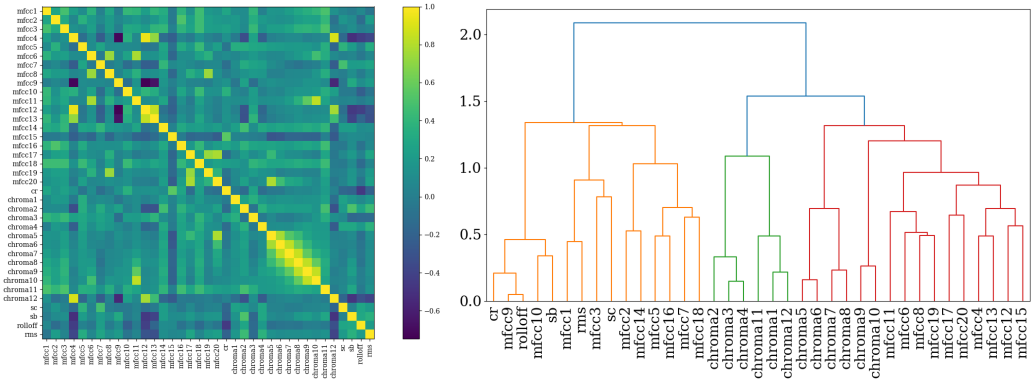
To be thorough, we retrain the GBDT with only the three most important features, the second MFCC feature, the fourth MFCC feature, and the RMS, and evaluate the model's performance when given only these three features. We observe some performance degradation; When asked to classify 6.0-second audio samples, the model is only able to achieve 70.0% precision, recall, and accuracy (compared to 89.0% precision, recall, and accuracy when trained with all features).

As the vast majority of features are estimated to have a less than 2% impact on overall accuracy, we also calculate feature correlations. As shown in Figure 5.7, many of the features are correlated. We perform a hierarchical clustering of the



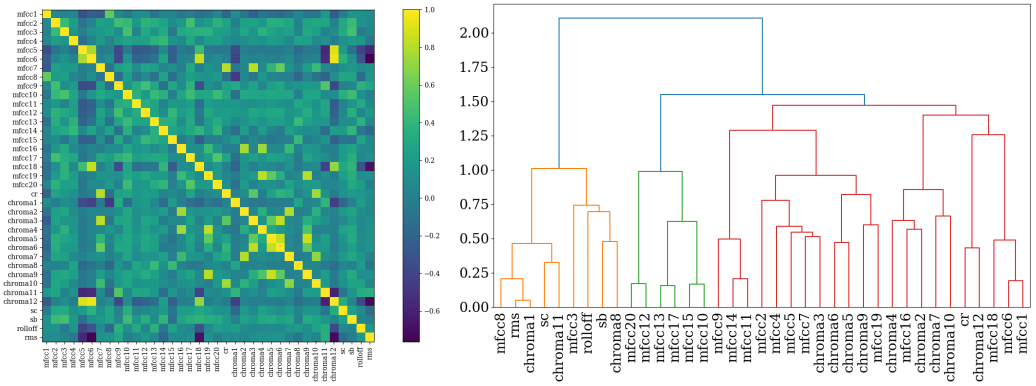
(a) Feature Correlation (1.0 second)

(b) Feature Clusters (1.0 second)



(c) Feature Correlation (3.0 seconds)

(d) Feature Clusters (3.0 seconds)



(e) Feature Correlation (6.0 seconds)

(f) Feature Clusters (6.0 seconds)

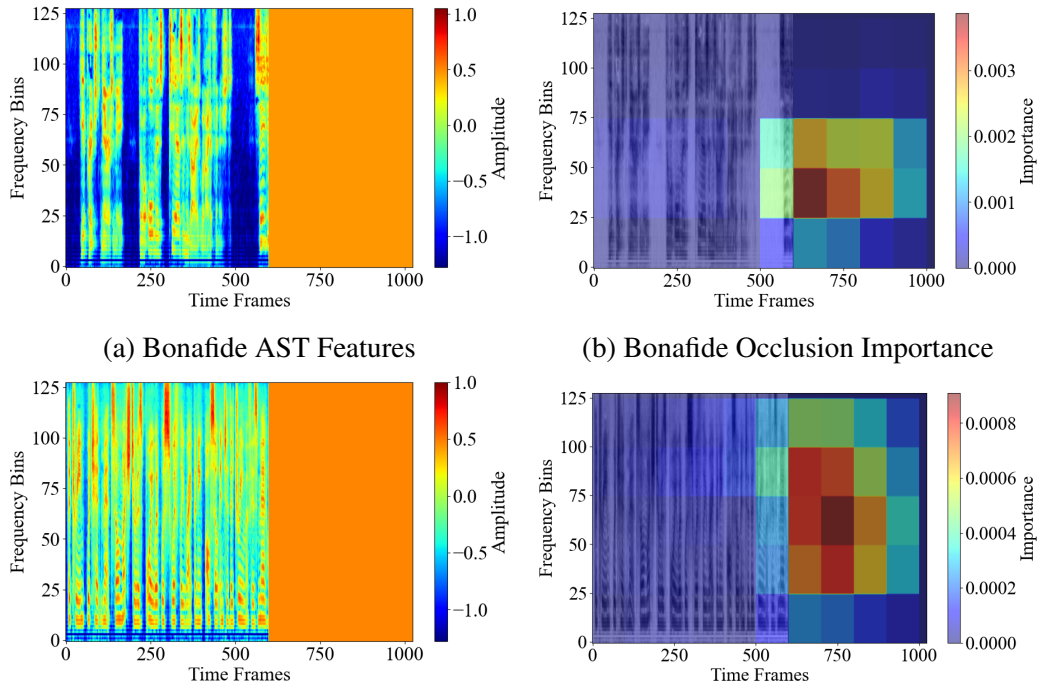
Figure 5.7: GBDT feature correlations and clusters for the 1.0, 3.0, and 6.0-second classifiers. In these figures, `sc` refers to the spectral centroid, `sb` refers to the spectral bandwidth, `cr` refers to the ZCR, mfcc$i$ refers to the i-th MFCC feature, and chroma$i$ refers to the i-th chroma feature.

features using Ward's linkage, and observe that there are only a few clusters of features. Perhaps interestingly, there is no clustering of the features in which our three most important features, MFCC2, MFCC4, and RMS, are all in different clusters.

For consistency's sake, we continue to experiment with the 6.0-second classifier. We select the most important feature from each cluster and retrain the GBDT. Retrained with RMS, MFCC 20, and MFCC 4, the GBDT achieves precision of 64.3%, recall of 64.0%, and accuracy of 63.8%. As performance was better when using the three most important features, compared to using important features with more spread, it does seem that the second MFCC feature is actually critical to the model's decision-making. Though it is not feasible to attribute a single frequency to a single MFCC, as an MFCC is a compact representation of a spectral shape across the Mel-scale filterbank, the second MFCC captures low-frequency details, such as overall spectral slope and formant information. Each formant corresponds to a resonance in the vocal tract, and it is intuitive that deepfake audio would have anomalous resonance. While this explanation points to a potential source of inherent distinction between real and deepfake audio, it only provides a clue as to what the model is attentive to in general rather than sample-level specificity.

5.5.2 Occlusion

As the AST model converts the raw audio signals into a spectrogram input, it is well-suited to visualization. Following the occlusion method described in Section 2, we perform occlusion with box size $(200, 50)$ and stride size $(100, 25)$. Importance is measured by the magnitude of change in the predicted probability



(a) Bonafide AST Features

(b) Bonafide Occlusion Importance

(c) Spoof AST Features

(d) Spoof Occlusion Importance

Figure 5.8: Importance measured by occlusion for 6.0-second audio samples.

of the sample being in the positive class when a section is occluded.

As shown in Figure 5.8, the importance is greatest for the padded regions–regions that theoretically contain no predictive information. The audio samples shown in Figure 5.8 are of length 6.0-seconds, but we observe this phenomenon when calculating importance by occlusion for all sample lengths. For each sample length, the most important regions (as measured by the occlusion method) are always placed at the beginning of the padded region. This result is obviously unhelpful in explaining the model's decision-making, but it is suggestive of a phenomenon posited by Wu et al., in which transformer models store global information at locations in the feature space which are consistent, such that the weight information there is always propagated [27].

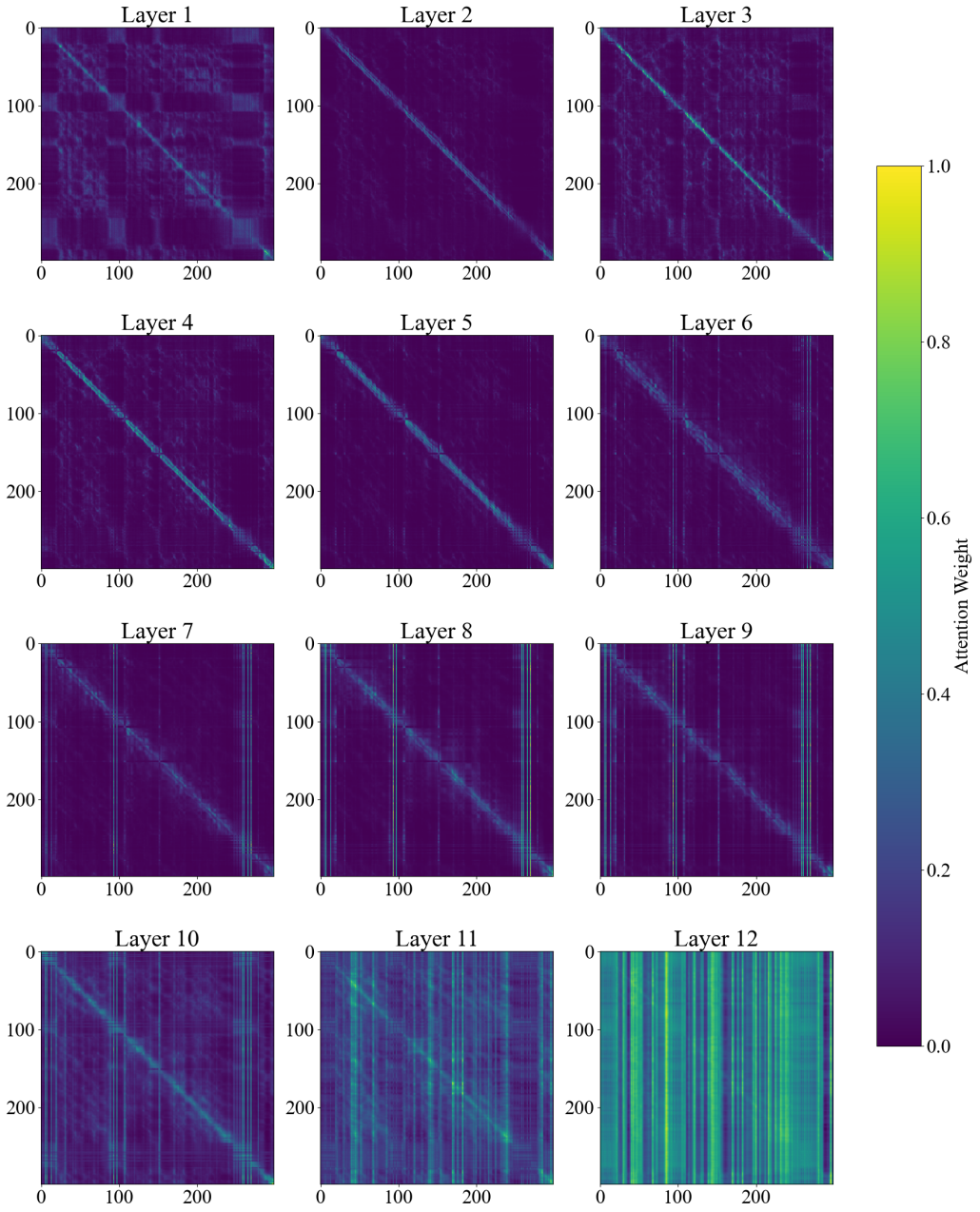


Figure 5.9: Normalized attention visualized for a bonafide 6.0-second sample where axes represent input token ID.

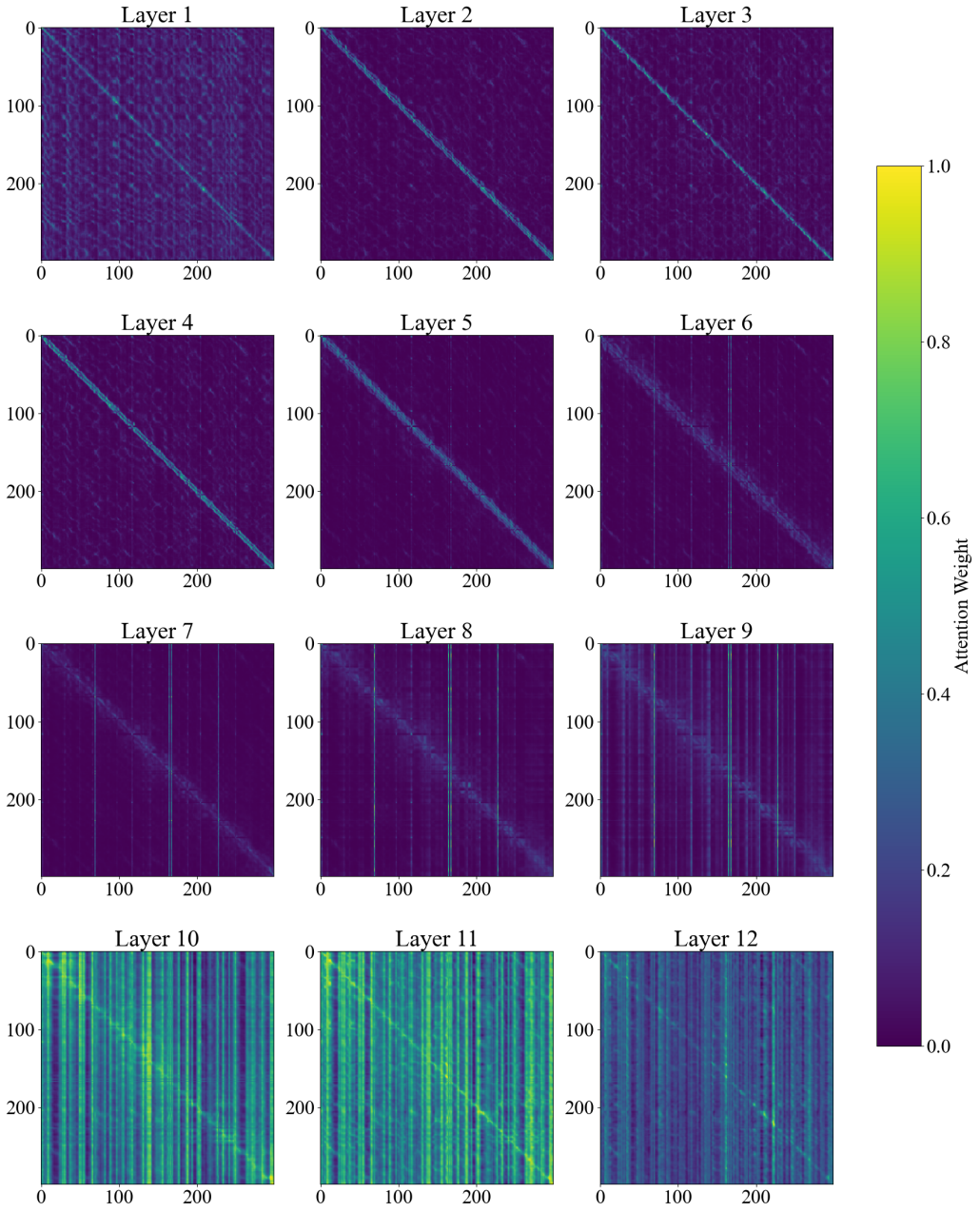


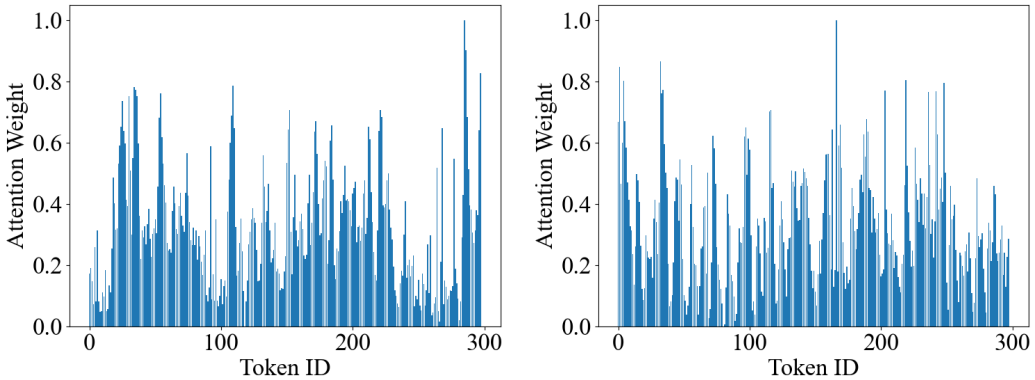
Figure 5.10: Normalized attention visualized for a spoof 6.0-second sample where axes represent input token ID.

5.5.3 Attention Visualization

One approach to explaining transformer models is visualizing the distribution of the attention weights over the input data. For each layer, as described in Section 2, there is an attention matrix that represents the amount of attention between each pair of tokens. This method has been employed for image and text data, but not, as of yet, to audio data [22].

In Figures 5.9 and 5.10, we visualize the each layer's attention matrix for the same bonafide and spoofed samples shown in Figure 5.8. Similar to results recorded on Vision Tranformer (ViT) by Dosovitskiy et al., we observe that at the attention at early layers is quite local with a relatively small receptive field while the attention at later layers is widely distributed [22].

We compute the attention roll-out, proposed by Abnar et al. and introduced in Section 2, which allows us to observe the overall attention flow on each of the input tokens by recursively multiplying the weight matrices of all the layers [18].



(a) Bonafide Normalized Token Attention (b) Spoof Normalized Token Attention

Figure 5.11: Distribution of attention for 6.0-second audio samples.

By normalizing the attention for the `[CLS]` classification token, we are able to

visualize which input tokens are most important for the model's overall classification, as seen in Figure 5.11. As each Wav2Vec token represents about 20 milliseconds of audio signal, we can pinpoint specific frames that were instrumental in the classification and perhaps inspect them more closely.

5.5.4 Attack Classification

Another potentially useful piece of information is what attack likely generated a given deepfake audio segment. Though this type of "explainability" does not give insight into a model's decision-making, it may provide a clue to the origin of a deepfake audio sample–something likely very important to a journalist or member of law enforcement. We therefore evaluate the abilities of the GBDT and transformer methods to identity which attack, if any, was used to generate an audio sample. Using 6-second audio samples, the GBDT achieved an overall accuracy of 51.9% with average precision of 53.0% and average recall of 52.1% over all 18 classes, with relatively stable performance across the attack types. The Wav2Vec model achieved overall evaluation accuracy of 91.8% with average precision of 91.7% and average recall of 91.8% across the 18 classes. The AST model performed similarly, achieving overall evaluation accuracy of 91.1%, average precision of 91.1%, as well as average recall of 91.1%. Though the models are quite successful at this task, previous exposure to deepfake audio generated with each attack is essential. It is not clear how well a model could detect an out-of-distribution sample.

6 | Discussion

Here, we synthesize the results presented in the previous section. First, we find that the AST and Wav2Vec models give state-of-the-art performance on a benchmark dataset in deepfake audio detection, FakeAVCeleb. Our conclusion from this result is that modern neural audio encoders, despite the fact that they are trained only on real audio, are better suited to deepfake audio detection than the other methods popular in deepfake literature. We believe that the continued interest in CNN-based classifiers and traditional machine learning approaches may be less effective in this domain and could potentially divert attention from more promising methods.

We find, however, that both the AST and Wav2Vec models require at least 1.0 seconds of audio to achieve good performance, while the GBDT performed much better when classifying small audio segments. We are unable to conclude whether the traces of deepfake audio lie more in the temporal or acoustic features. The GBDT's performance with very short audio samples leaves the impression that there is some feature of deepfake audio present even in the smallest samples. Yet, the transformer models perform poorly on short segments.

One possible explanation is that as the transformer models are designed to work with fixed input lengths, they are unable to make meaningful predictions if too few tokens are supplied. This might be a failure of the architecture or the feature encoding process, rather than the lack of signal.

Another possible explanation is that the GBDT has learned to separate the bonafide and spoofed samples by some feature that is not intrinsic to the classes, but to the

datasets themselves. For example, the GBDT's reliance on RMS for classification seems to support this hypothesis. All that said, 1.0 second of audio is a relatively small sample and considerably less than the commercial purveyors demand in order to make a classification. Given the Wav2Vec model's evaluation F1 of 97.2% on the ASVspoof data, we are optimistic about its potential for deepfake audio classification.

Despite the GBDT's recent popularity, its poor performance on the rerecorded ASVspoof furthers the point that they should not be used for deepfake audio detection in the wild. In contrast, the authors were impressed by both the AST and Wav2Vec models' performances on the compressed and rerecorded data. We conclude that these manipulations are not particularly influential with sufficiently sophisticated models. We feel confident that as deepfake audio detection moves towards transformer models that these methods of obscuring deepfake audio will no longer be threats.

In most of our experiments, the Wav2Vec model outperformed the AST model, which explains its relative popularity. However, in our generalization tests, we found the AST model's performance preferable. The ASVspoof-trained AST model caught considerably more of the spoof samples from the FakeAVCeleb data than the ASVspoof-trained Wav2Vec model, which missed many. We posit that the AST model learned more obscure but consistent features of deepfake audio, while the more semantically oriented Wav2Vec model learned more vocal and linguistic features specific to the data it was finetuned with. Despite Wav2Vec's superior performance in most experiments, we are hopeful that feature encoders like AST will become more prevalent as they are much less of a "black-box".

Ideally, in this thesis, we would have been able to provide confidence intervals to assess the statistical significance of our detector results. However, bootstrapping confidence intervals is computationally expensive and has not generally been performed in prior literature on related problems. A preliminary analysis on a random sample of results suggests that the confidence intervals would likely have been small compared to both the relative performances and absolute means. Therefore, the absence of confidence intervals does not affect our overall conclusions.

Finally, there is a shortage of academic text dedicated to explainability in the audio domain, and we suspect that this is due to its incredible difficulty [42]. It is challenging to interpret audio features; it is challenging to report empirical results; and it is challenging to measure success. Nevertheless, we endeavour to explain what we can and record our progress so that it can be built upon further.

Despite the relative transparency of the GBDT algorithm, we find that the feature importances it provides are not sufficiently meaningful in and of themselves to claim that the GBDT is "explainable". We also observe that the features we calculate to be important on the ASVspoof data are different than those observed to be important by Bird et al. on the DeepVoice data [35].

Though the occlusion technique did not yield explanations about the classification mechanism of the AST model, the findings of the method are scientifically interesting. We hope that advancements in the field of feature engineering and XAI will enable us to observe what global information is being perpetually passed through the network.

Lastly, we are optimistic that the attention rollout technique might be useful to

technical and non-technical people alike. Inspecting Wav2Vec tokens is challenging given the convolved manner in which they are generated, but we hope that with more interpretable features this method might generate truly meaningful insight.

Future Directions In the course of this thesis work, we considered other methods as well. We were not able to evaluate these methods fully, largely due to a constraint on computational resources, but would recommend further study.

First, we have begun to build a cascaded classifier, which we hope could be deployed to social media platforms. The cascaded classifier we envision will have an initial, quick-to-compute classifier that would simply identify whether an audio recording contains human speech. When experimenting with such a classifier, we found that thresholding the ZCR at 0.09 correctly separated all instances of bonafide human speech and most instances of spoof human speech from recordings containing music or environmental noise. We also found that when a ZCR threshold suggests that there is human speech in a recording, but a speech-to-text algorithm does not, then the audio sample was deepfake in all of the tests we conducted. These tests were random samples, but the results were consistent enough that we recommend further exploration.

Second, the authors were impressed by the performance of "one-class" classifiers, where models attempt to learn a tight boundary around what real speech is like [29]. The AST and Wav2Vec feature encoders are also similar in some ways to the "one-class" approach, as they are only trained on real audio samples. This also explains their relatively high generalization ability, as they are not specialized to specific deepfake audio attacks. However, AST is downstream task

agnostic, and Wav2Vec is specialized for audio-linguistic tasks (such as speech-to-text tasks). We propose that research should be done on feature generators that are specific to deepfake audio detection. In particular, we propose the design of a feature extractor that looks to amplify the following features, which we suspect would be particularly difficult to replicate:

- Jitter: the variability in pitch.
- Shimmer: the variability in amplitude across frames.
- Voice onset time: the time it takes for the vocal cords to start vibrating after the release of a consonant.
- Formants: Frequency peaks that correspond to the resonant frequencies in the vocal tract.

Third, we are interested in applying methods for the detection of AI-generated text to AI-generated audio. In *DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature*, Mitchell et al. propose using perplexity loss to evaluate the likelihood that a specific piece of text was generated by a model they query [39]. In this method, one gives the preceding context to the generative text model and measures how similar the model's subsequent generated text is to the text you have at hand. Though we recognize that this method is limited by the number of models one has access to, we propose that one could apply a similar approach with text-to-speech models: comparing the EnCodec tokens they generate to ones for an audio sample in question.

7 | Conclusion

In conclusion, the rapid advancement of deepfake audio technology poses a serious risk to election security, as detection systems struggle to keep up with increasingly sophisticated audio manipulation methods. Our study emphasizes the need to address generalization errors and the challenges of real-world data augmentation, areas that have been largely overlooked in existing research. To further empower decision-makers and journalists in election scenarios, we also focused on the creation of visual explanations. We hope that these are a step towards complete model transparency and deeper insight into the characteristics of deepfake audio.

Additionally, we conclude that the field must shift towards self-supervised learned representations, which offer greater adaptability and robustness compared to traditional feature-based models. These self-supervised approaches have the potential to more effectively capture the complexities of deepfake audio, reducing reliance on handcrafted features and improving performance in diverse, real-world scenarios. Moving forward, further research on improving generalization, model explainability, and adopting self-supervised learning will be essential to counter the evolving threats posed by deepfake audio.

Bibliography

[1] D. O'Shaughnessy. *Speech Communications: Human And Machine (ieee)*. Universities Press (India) Pvt. Limited, 1987. ISBN: 9788173713743. URL: https://books.google.it/books?id=UWmD8aY_448C.

[2] Anssi Klapuri and Manuel Davy. *Signal Processing Methods for Music Transcription*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387306676.

[3] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[5] Brian McFee et al. "librosa: Audio and Music Signal Analysis in Python". In: *SciPy*. 2015. URL: https://api.semanticscholar.org/CorpusID:33504.

[6] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should I trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 1135–1144.

[7] Heinrich Dinkel et al. "End-to-end spoofing detection with raw waveform CLDNNS". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Mar. 2017. DOI: 10.1109/icassp.2017.7953080. URL: http://dx.doi.org/10.1109/ICASSP.2017.7953080.

[8] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Proceedings of the 31st international conference on neural information processing systems*. 2017, pp. 4768–4777.

[9] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[10] Darius Afchar et al. "Mesonet: a compact facial video forgery detection network". In: *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2018, pp. 1–7.

[11] Galina Lavrentyeva et al. *STC Antispoofing Systems for the ASVspoof2019 Challenge*. 2019. arXiv: 1904.05576 [cs.SD]. URL: https://arxiv.org/abs/1904.05576.

[12] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations (ICLR)*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[13] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG]. URL: https://arxiv.org/abs/1912.01703.

[14] Andreas Rossler et al. "FaceForensics++: Learning to Detect Manipulated Facial Images". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[15] Steffen Schneider et al. *wav2vec: Unsupervised Pre-training for Speech Recognition*. 2019. arXiv: 1904.05862 [cs.CL]. URL: https://arxiv.org/abs/1904.05862.

[16] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 6105–6114.

[17] Hossein Zeinali et al. *Detecting Spoofing Attacks Using VGG and SincNet: BUT-Omilia Submission to ASVspoof 2019 Challenge*. 2019. arXiv: 1907.12908 [cs.CV]. URL: https://arxiv.org/abs/1907.12908.

[18] Sara Abnar and Willem Zuidema. "Quantifying attention flow in transformers". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 4190–4197.

[19] Alexandros Haliassos et al. "Lips Don't Lie: A Generalisable and Robust Approach to Face Forgery Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[20] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL]. URL: https://arxiv.org/abs/1910.03771.

[21] Zhenzong Wu et al. *Light Convolutional Neural Network with Feature Genuinization for Detection of Synthetic Speech Attacks*. 2020. arXiv: 2009.09637 [eess.AS]. URL: https://arxiv.org/abs/2009.09637.

[22] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[23] Yuan Gong, Yu-An Chung, and James Glass. *AST: Audio Spectrogram Transformer*. 2021. arXiv: 2104.01778 [cs.SD]. URL: https://arxiv.org/abs/2104.01778.

[24] Hasam Khalid et al. “FakeAVCeleb: A Novel Audio-Video Multimodal Deepfake Dataset”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*. 2021. URL: https://arxiv.org/abs/2108.05080.

[25] Yuezun Qian et al. “Exploring Temporal Coherence for More General Video Face Forgery Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[26] Yun Shi and Junichi Yamagishi. “Investigating self-supervised front ends for speech spoofing countermeasures”. In: *arXiv preprint arXiv:2111.07725* (2021).

[27] Kan Wu et al. *Rethinking and Improving Relative Position Encoding for Vision Transformer*. 2021. arXiv: 2107.14222 [cs.CV]. URL: https://arxiv.org/abs/2107.14222.

[28] Qin Xie et al. “Real-Time Conformer for Streaming Speech Recognition”. In: *Proc. Interspeech 2021*. 2021, pp. 2257–2261.

[29] You Zhang, Fei Jiang, and Zhiyao Duan. “One-Class Learning Towards Synthetic Voice Spoofing Detection”. In: *IEEE Signal Processing Letters* 28 (2021), 937â941. ISSN: 1558-2361. DOI: 10.1109/lsp.2021.3076358. URL: http://dx.doi.org/10.1109/LSP.2021.3076358.

[30] Zexin Cai, Weiqing Wang, and Ming Li. *Waveform Boundary Detection for Partially Spoofed Audio*. 2022. arXiv: 2211.00226 [eess.AS]. URL: https://arxiv.org/abs/2211.00226.

[31] Ting Chen et al. “Large Speech Supervision Improves Self-Supervised Speech Representation Learning”. In: *arXiv preprint arXiv:2203.01573* (2022).

[32] Alexandre DÃ©fossez et al. “High Fidelity Neural Audio Compression”. In: *arXiv preprint arXiv:2210.13438* (2022). URL: https://arxiv.org/abs/2210.13438.

[33] Xin Wang and Junichi Yamagishi. *Investigating self-supervised front ends for speech spoofing countermeasures*. 2022. arXiv: 2111.07725 [eess.AS]. URL: https://arxiv.org/abs/2111.07725.

[34] Shangzhe Zhao, Zhaoyang Cheng, and Chen Change Loy. “RealForensics: Leveraging Real Talking Faces via Self-Supervision for Robust Forgery Detection”. In: *European Conference on Computer Vision (ECCV)*. 2022.

[35] Jordan J. Bird and Ahmad Lotfi. *Real-time Detection of AI-Generated Speech for DeepFake Voice Conversion*. 2023. arXiv: 2308.12734 [cs.SD]. URL: https://arxiv.org/abs/2308.12734.

[36] Abhishek Dixit, Nirmal Kaur, and Staffy Kingra. “Review of audio deepfake detection techniques: Issues and prospects”. In: *Expert Systems* 40.8 (2023), e13322. DOI: https://doi.org/10.1111/exsy.13322. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/exsy.13322. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.13322.

[37] Adnan Dzanic et al. “Detecting Deepfakes Without Seeing Any: Face Forgery Detection via Adversarially-Filtered Frequency Domains”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023.

[38] Xiaodan Gu et al. “Self-Supervised Video Forensics by Audio-Visual Anomaly Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023.

[39] Eric Mitchell et al. “DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature”. In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 24950–24962. URL: https://proceedings.mlr.press/v202/mitchell23a.html.

[40] Jiangyan Yi et al. “Audio Deepfake Detection: A Survey”. In: *arXiv preprint arXiv:2308.14970* (2023).

[41] Resemble AI. *DETECT-2B | Resemble AI — resemble.ai*. https://www.resemble.ai/detect2b/. [Accessed 28-08-2024]. 2024.

[42] Alican Akman and BjÃ¶rn W. Schuller. “Audio Explainable Artificial Intelligence: A Review”. In: *Intelligent Computing* 3 (2024), p. 0074. DOI: 10.34133/icomputing.0074. eprint: https://spj.science.org/doi/pdf/10.34133/icomputing.0074. URL: https://spj.science.org/doi/abs/10.34133/icomputing.0074.

[43] Jack Goodman. *Trending - The Mexican mayor and a deepfake scandal - BBC Sounds — bbc.co.uk*. https://www.bbc.co.uk/sounds/play/w3ct5d9g. [Accessed 01-08-2024]. 2024.

[44] Enkhtogtokh Togootogtokh and Christian Klasen. “AntiDeepFake: AI for Deep Fake Speech Recognition”. In: *arXiv preprint arXiv:2402.10218* (2024). URL: https://arxiv.org/abs/2402.10218.

[45] Xin Wang et al. “ASVspoof 5: Crowdsourced Speech Data, Deepfakes, and Adversarial Attacks at Scale”. In: *ASVspoof 5 Workshop (Interspeech 2024 Satellite)*. 2024, pp. 1–8. DOI: 10.48550/arXiv.2408.08739. URL: https://arxiv.org/abs/2408.08739.

[46] Zeyu Xie et al. *FakeSound: Deepfake General Audio Detection*. 2024. arXiv: 2406.08052 [cs.SD]. URL: https://arxiv.org/abs/2406.08052.