AUDIO DEEPFAKE DETECTION*

Mark He Huang† **Peiyuan Zhang**† **James Raphael Tiovalen**† **Madhumitha Balaji**†
Information Systems Technology and Design (ISTD) Pillar
Singapore University of Technology and Design
Singapore, Singapore
{he_huang, peiyuan_zhuang, james_raphael, madhumitha_balaji}@mymail.sutd.edu.sg

Shyam Sridhar†
Engineering Systems and Design (ESD) Pillar
Singapore University of Technology and Design
Singapore, Singapore
shyam_sridhar@mymail.sutd.edu.sg

1 Introduction

DeepFake audio refer to those that are fabricated and are underpinned by deep neural networks (DNN) that learn the movements of sound recordings to the extent that they can produce realistic-sounding fake audio. They are generally used to imitate people's voices. While they could, at times, be amusing, such techniques can be misused to spread misinformation which can lead to detrimental consequences. Some of the ways in which DeepFake techniques can be exploited include online harassment, influencing political movements, and people impersonation. For example, audio DeepFake coupled with image DeepFake might potentially be used as a tool for information warfare between Russia and Ukraine[1]. Therefore, there has been tremendous attention devoted to designing and implementing realistic audio DeepFake detection systems in recent years. In this project, we focus on distinguishing DeepFake audio that are generated by various state-of-the-art GAN-based DNN apart from the real ones. We explore different sequential modelling architectures as well as different base features for DeepFake audio detection. We release code and results on GitHub: `https://github.com/MarkHershey/AudioDeepFakeDetection`

2 Related Works

Audio DeepFake detection. Plenty of models was proposed to filter out fake audio from the real ones. [2] tries to provide a more fine-grained supervision signal by having two classification blocks. The first is to predict the binary fake or real label, while the second tries to output the type of audio synthesis technique used to generate the fake audio. Instead of training a model from scratch, [3] proposes to use the hidden layer neuron activation map from an existing speaker recognition (SR) model as the feature representation of an audio file. They then feed this representation to a lightweight classifier to output the final prediction. Meanwhile, [4] proposes to utilize a shallow CNN-based approach in differentiating real voice audio from synthetic ones. [5] proposes a brand new framework for locating the fake regions in the overall input audio. They do so by equipping the fake span discovery strategy with a self-attention mechanism in order to enhance their quality of detection. [6] aims to provide a solution to replay attacks, where an attacker acquires a target speaker's voice using a recording device and then replays using a playback device. The attacker does this using different combinations of replay and playback devices with background environments. In order to defend against this class of attacks, the authors propose using an end-to-end DNN without knowledge-based intervention. [7] also proposes another method to conduct replay and speech synthesis detection via the usage of two blocks: a Res2Net block and a squeeze-and-excitation block.

A Course Project for SUTD 50.039 Theory and Practice of Deep Learning (2022 Spring)
†These authors have contributed equally.

Audio Processing. Digital audio signals are records of the original sound waves at a certain sample frequency, i.e. sampling rate. Traditional statistical models [8] usually take pre-processed audio features, such as the Mel Frequency Cepstral Coefficients (MFCC) or the Linear Frequency Cepstral Coefficients (LFCC) as the model input. MFCC is one of the most commonly used audio features because its frequency bands are designed to be close to the human auditory system. To derive the MFCC features from waveform signals, we first transform the waveform signal into a Mel-scaled spectrogram, then followed by taking an logarithm and applying a discrete cosine transform as shown in Equation 1. LFCC is similar to MFCC, but it uses a linear scale filter, which will result in more of the high frequency signal being retained compared to the MFCC feature. Deep learning-based models take in various kind of audio inputs, including raw waveform, spectrogram, Mel-spectrogram, MFCC/LFCC, etc. In addition, Double Delta augmentation are often used on top of MFCC/LFCC, which takes the first and second derivatives as shown by Equation 2 to further enhance temporal representation.

$$c(t,r) = \sum_{s=0}^{S-1} \log\left[X_{\text{mel}}(t,s)\right] \cdot cos\left[\frac{\pi \cdot r \cdot (s+0.5)}{S}\right] \quad \forall r = 0,\ldots,R-1, \tag{1}$$

$$d(t) = \frac{\sum_{n=1}^{N} n \cdot \left[c\left(t+n\right) - c\left(t-n\right)\right]}{2 \cdot \sum_{n=1}^{N} n^2} \quad \forall t = 0,\ldots,T-1, \tag{2}$$

In our work, we experiment with various model architectures, equipped with different audio features, to compare their performance. All MFCC or LFCC features used in our project are augmented using Double Delta.

3 Dataset

For the source of real audio data, we use LJSpeech [9], which is a high-quality human speech dataset consisting of 13,100 short audio clips (ranging from approximately 1-10 seconds, 24 hours in total) of a single female speaker reading passages from 7 non-fiction books. For the source of deep fake audio data, we use WaveFake [10], a dataset based on LJSpeech [9], introduced in 2021 at NeurIPS. It consists of 117,985 synthetic audio clips (196 hours in total) generated by six different deep learning-based generative network architectures (i.e., **MelGAN** [11], **Parallel WaveGAN (PWG)** [12], **Multi-band MelGAN (MB-MelGAN)** [10], **Full-band MelGAN (FB-MelGAN)** [10], **HiFi-GAN (HiFi-GAN)** [13], and **WaveGlow** [14]). Each of the six deep neural networks generates 13,100 audio clips corresponding to the real audio clips in LJSpeech [9].

4 Methods

4.1 Gaussian Mixture Model

Motivated by previous works such as [15, 16, 17], we developed a GMM based approach to deepfake audio detection. Specifically, our classifier is composed of 2 Gaussian Mixture Models that were trained individually on real and generated audio samples. Each model is composed of 128 single Gaussian models. The Gaussian distribution is chosen here as the choice of the probability distribution for the mixture model, given its unique mathematical properties and its good computational performance.

The two models used the MFCC features that we extracted from the audio files as inputs. The final classification is made according to the likelihood function:

$$f(x) = \log p(x|\theta_r) - \log p(x|\theta_g) \tag{3}$$

where θ_r and θ_g are the Gaussian parameters for the real and generated audio distributions respectfully with x being the input MFCC feature.

4.2 Vanilla Recurrent Neural Network

Audio is by nature a kind of sequential data, in which the waveform manifests different frequencies and amplitudes in different timestamps. Vanilla Recurrent Neural Network (RNN) is the simplest architecture proposed to process and learn from sequential data. In our implementation of Vanilla RNN, there is a hidden state to keep track of all the historical information from the previous timestamps. Given an input in timestamp t, the current hidden state is updated by the hidden state from the last timestamp and the current input through fully-connected layers and an

activation function. We will use the hidden state in the last timestamp as the representation of the entire audio. Then, this representation will be fed to two fully connected layers to produce the final classification result for this audio file.

$$h_t = tanh(W^i i_t + W^h h_{t-1}) \tag{4}$$

$$o_t = W^{o2} relu(W^{o1} h_{last}), \tag{5}$$

where W^i, W^h, W^{o2}, W^{o1} refers to the parameters for the input layer, hidden state transition, and the two fully connected layers for output respectively.

4.3 Bidirectional Long Short-Term Memory Network

While current works including [6] employ Gated Recurrent Unit (GRU) in their design, we want to check how Long Short-Term Memory (LSTM) module will perform compared to GRU and vanilla RNN. The nature of this task relies on some temporal information, but the later information in the audio is not necessarily more important or less important than the earlier information. Based on this observation, we design a double-layered bidirectional LSTM layer followed by a 1-dimensional convolution layer and a fully-connected layer. The model takes in pre-processed LFCC or MFCC features shaped (B, N, T) where B is the batch size, N is the number of linear/Mel-frequency cepstral coefficients, and T is the number of frames in the temporal dimension.

4.4 Shallow Convolutional Neural Network

We also adopted a shallow CNN approach, inspired by a paper written by Lieto et al. [4] that originally proposed this idea. It is possible to utilize CNN applied to some kind of 2D-image representation of the audio signals. The aforementioned paper leveraged a CNN model applied to the Mel-spectrogram representation of the audio dataset, which performed relatively well. For this project, to ensure some uniformity and standardization with the rest of the models, we used the MFCC and the LFCC features of the audio signals instead. The MFCC feature is obtained by performing a logarithmic scaling on the Mel-spectrogram, followed by computing the DCT on the resultant of the previous operation. Furthermore, while MFCC is calculated from the DB-scaled Mel-spectrogram, LFCC is similar but derived from the DB-scaled linear filtered spectrogram instead. Due to some differences in the audio dataset that we use compared to the original paper, we have also slightly modified the network architecture to be as such:

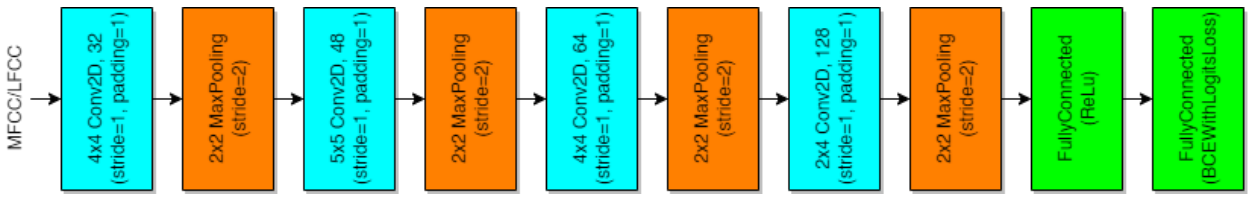

Figure 1: Shallow CNN Architecture.

The proposed shallow CNN architecture takes as input the 2D representation of the audio signal $\overline{\mathbf{X}}_w$ (either MFCC or LFCC) and outputs a single-element vector indicating the likelihood of the analyzed audio belonging to each class. The architecture, as shown in Figure 1, is composed of four 2D convolutional layers, each one followed by a max-pooling layer (2×2 pool size and 2×2 stride). The first and third convolutional layers are composed of 32 and 64 filters with size 4×4, respectively. The second convolutional layer consists of 48 filters with size 5×5. The last convolutional layer is composed of 128 filters with size 2×4. All the four convolutional layers have stride size 1×1 and padding size 1×1. The final convolutional layer is then flattened and connected to a fully connected layer with the ReLu activation function outputting a 128-element vector. A last fully connected layer outputs a 1-element vector used to get the final binary classification result, i.e., human or bot. A label of 0 would imply that the audio signal is classified as fake (generated by a bot), while a label of 1 would imply that the audio signal is classified as real (generated by a real human).

4.5 Time-Domain Synthetic Speech Detection

Hua et al. [18] proposed Time-Domain Synthetic Speech Detection (TSSD), an end-to-end synthetic speech detection framework that uses deep neural networks (DNN) for feature extraction as well as classification. This lightweight neural network takes in raw audio waveforms without any pre-transforms or hand-crafted feature engineering. Its

architecture is described in Figure 2. The first block is a 1x7 1D convolutional layer with 16 channels, followed by Batch Normalization (BN), ReLU activation and max-pooling with kernel size 4. Next, ResNet-style modules [19] are stacked M times (in our network, $M = 4$) with a max-pool layer of kernel size 4. Each of these modules has 3 1x3 convolution layers whose output is concatenated with the original input transformed by a 1x1 convolution (i.e., a skip connection), with BN and ReLU applied at the end of each of these layers. In our model, C_R, the number of channels, is 16 for the first ResNet block, 32 for the second, 64 for the third and 128 for the last. Finally, this is followed by 3 fully connected layers (where C_L, the output dimension, is 128 for the first linear layer, 64 for the second, and 32 for the last) and the output is produced by a softmax layer.

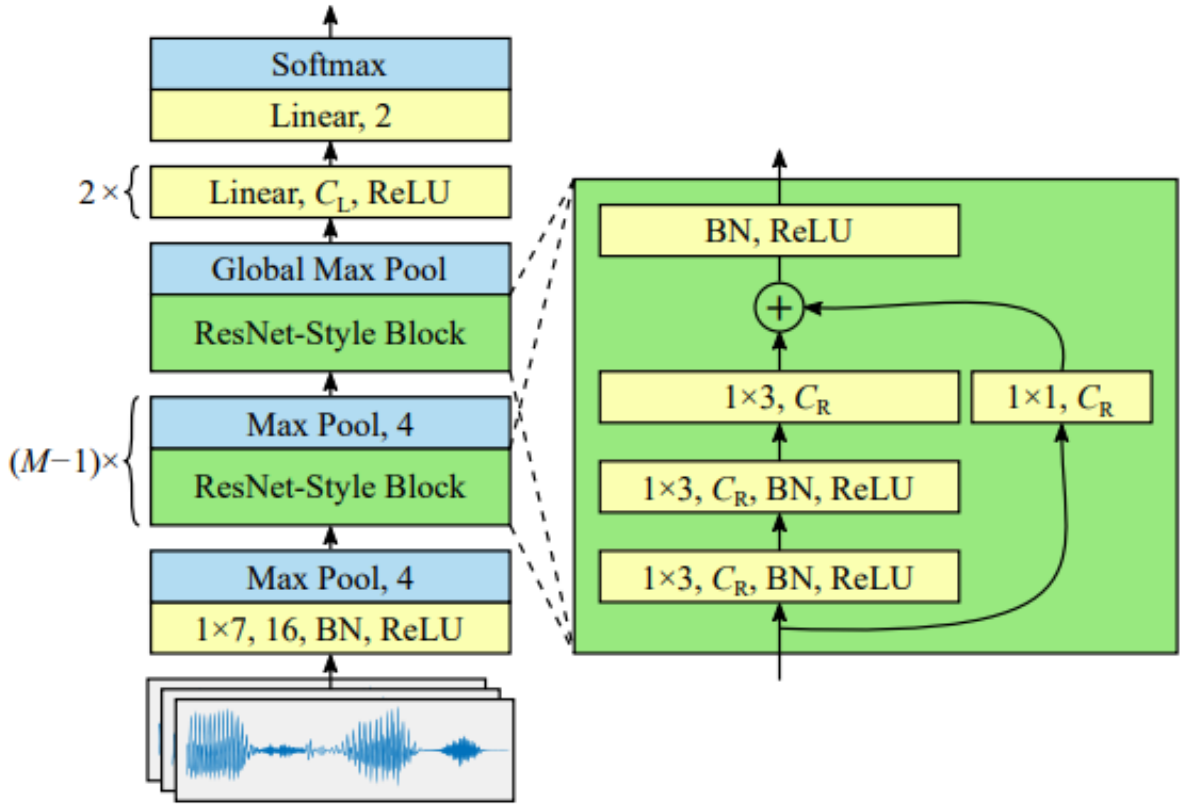

Figure 2: TSSD Architecture.

5 Experiments and Result

Training and Testing Setup. We choose to conduct experiments using the two most challenging experiment setups described in WaveFake [10].

- **In-distribution Setup**: We use 80% MelGAN + 80% LJSpeech for training, and 20% MelGAN + 20% LJSpeech for testing. i.e., the Real-to-Fake ratio in training is 1:1.
- **Out-of-distribution Setup**: We use everything except MelGAN + 80% LJSpeech for training, and 20% MelGAN + 20% LJSpeech for testing. i.e., the Real-to-Fake ratio in training is roughly 7.4:1.

Loss Function. We use the Binary Cross Entropy Loss (BCELoss) as the training loss for this binary classification problem. In particular, we use the `torch.nn.BCEWithLogitsLoss` function from PyTorch [20], which combines the Sigmoid function with the BCELoss in a single layer to achieve better numerical stability. When the model is trained in the out-of-distribution setting, we calculate the ratio of the number of positive samples to the number of negative samples, i.e. pos_weight, and use this as the additional weight for the loss. By doing so, the loss acts as if there are an equal amount of positive and negative samples.

$$l_n = -w_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right] \tag{6}$$

$$\ell(x, y) = mean(L) = mean(\{l_1, \ldots, l_N\}^\top) \tag{7}$$

Hyper-parameters. For most of the experiments, models are trained on a single NVIDIA RTX 3090 graphics processing unit with a batch size of 256. We use the Adam [21] algorithm with an initial learning rate of 0.0005 and a weight decay of 0.0001 for stochastic optimization.

Evaluation Metrics. In Table 1, we follow WaveFake [10] to report the Equal Error Rate (EER) (a.k.a. Crossover Error Rate), which is commonly used to measure the overall accuracy of a biometric system. EER is defined as the common value where the False Positive Rate (FPR) equals the False Negative Rate (FNR). Lower EER values indicate better performance. Additionally, in Table 2, we also report the F1 score (a.k.a. balanced F-score) and the Area Under the Receiver Operating Characteristic Curve (ROC AUC). Higher F1 score and larger AUC indicate better performance.

Table 1: Experimental Results Reported in EER

Model (input feature type)	In-dist Setup	Out-of-dist Setup
GMM (w/ LFCC) [10]	0.148	0.220
RawNet2 (w/ wave) [22]	0.001	**0.008**
VanillaRNN (w/ wave)	0.350	$--$
Bi-LSTM (w/ wave)	0.264	$--$
Bi-LSTM (w/ MFCC)	0.040	$--$
Bi-LSTM (w/ LFCC)	0.004	**0.044**
ShallowCNN (w/ MFCC)	0.004	$--$
ShallowCNN (w/ LFCC)	**0.000**	0.093
TSSD (w/ wave)	0.001	0.056

GMM (w/ LFCC) and RawNet2 (w/ wave) results are provided by WaveFake [10]. We round all results to three decimal places following [10], hence, 0.000 implies the actual value is less than 0.0005. The dashed line indicates that the experiment was not performed due to time constraints.

Table 2: Experimental Results Reported in F1 Score and AUC

	In-dist Setup		Out-of-dist Setup	
Model (input feature type)	F1-score	ROC AUC	F1-score	ROC AUC
VanillaRNN (w/ wave)	0.649	0.653	$--$	$--$
Bi-LSTM (w/ wave)	0.742	0.750	$--$	$--$
Bi-LSTM (w/ MFCC)	0.960	0.960	$--$	$--$
Bi-LSTM (w/ LFCC)	0.996	0.996	**0.965**	**0.9651**
ShallowCNN (w/ MFCC)	0.997	0.997	$--$	$--$
ShallowCNN (w/ LFCC)	**1.000**	**1.000**	0.939	0.937
TSSD (w/ wave)	0.999	0.999	0.957	0.956

GMM (w/ LFCC) and RawNet2 (w/ wave)'s F1 and AUC results are not reported by WaveFake [10]. We round all results to three decimal places, hence, 1.000 implies the actual value is larger than or equal to 0.9995. The dashed line indicates that the experiment was not performed due to time constraints.

6 Analysis

We expect the models trained with the out-of-distribution setup to take significantly more time during training, and this was indeed the case. However, in terms of performance, both models trained with the in-distribution setup and the out-of-distribution setup performed similarly well.

Two anomalous data points of interest that we can manually inspect further are *LJ048-0107* and *LJ050-0267*. Five of our classifier models wrongly predicted the categories of those two data points. More specifically, *LJ048-0107* was wrongly

classified by *ShallowCNN_lfcc_O*, *SimpleLSTM_mfcc_I*, *TSSD_wave_I*, *WaveLSTM_wave_I*, and *WaveRNN_wave_I*, while *LJ050-0267* was wrongly classified by *ShallowCNN_lfcc_O*, *ShallowCNN_mfcc_I*, *SimpleLSTM_mfcc_I*, *WaveLSTM_wave_I*, and *WaveRNN_wave_I*.

A more interactive and exploratory version of this qualitative analysis is provided here: `https://markhh.com/AudioDeepFakeDetection`

As demonstrated on the aforementioned interactive GUI page, all of the features of *LJ048-0107* and *LJ050-0267* that we have used for our models are quite similar. The main blocks in each of the features are mostly the same, and they only differ in the finer details. Therefore, this explains why the models have a harder time in differentiating them.

7 Conclusion

In this project, we implemented, tested, and compared several deep learning architectures to classify human and bot speech with the WaveFake and LJSpeech datasets as the common benchmark in an attempt to measure their performance. From our results, we conclude that while the various networks included in this research are able to perform relatively well even for data generated by very strong GANs, there are still certain audio data inputs that potential adversaries can craft to fool these networks, such as the ones demonstrated in our analysis. As such, further improvement can still be made to increase the detection strength of these models. The results of this research will be useful in dealing with potential new privacy and security issues due to automatic speech and deepfake voice generation tools. We hope that the achieved results will inspire the research community to further investigate this deepfake detection issue in the near future.

Acknowledgments

We would like to thank our course instructors, Dr Matthieu De Mari and Prof Berrak Sisman, for the opportunity to conduct this project. Additionally, we would also like to thank Prof Liu Jun for providing the GPU workstations for this project to conduct experiments.

References

[1] Deepfake video of Volodymyr Zelensky surrendering surfaces on social media. `https://www.youtube.com/watch?v=X17yrEV5sl4`. Accessed: 2022-04-10.

[2] Nishant Subramani and Delip Rao. Learning Efficient Representations for Fake Speech Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:5859–5866, 04 2020.

[3] Run Wang, Felix Juefei-Xu, Yihao Huang, Qing Guo, Xiaofei Xie, Lei Ma, and Yang Liu. DeepSonar: Towards Effective and Robust Detection of AI-Synthesized Fake Voices, 2020.

[4] A. Lieto, D. Moro, F. Devoti, C. Parera, V. Lipari, P. Bestagini, and S. Tubaro. "Hello? Who Am I Talking to?" A Shallow CNN Approach for Human vs. Bot Speech Classification. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2577–2581, 2019.

[5] Haibin Wu, Heng-Cheng Kuo, Naijun Zheng, Kuo-Hsuan Hung, Hung-Yi Lee, Yu Tsao, Hsin-Min Wang, and Helen Meng. Partially Fake Audio Detection by Self-attention-based Fake Span Discovery, 2022.

[6] Jee-weon Jung, Hye-jin Shim, Hee-Soo Heo, and Ha-Jin Yu. Replay attack detection with complementary high-resolution information using end-to-end DNN for the ASVspoof 2019 Challenge, 2019.

[7] Xu Li, Na Li, Chao Weng, Xunying Liu, Dan Su, Dong Yu, and Helen Meng. Replay and synthetic speech detection with res2net architecture, 2020.

[8] Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. HMM-based audio keyword generation. In *Pacific-Rim Conference on Multimedia*, pages 566–574. Springer, 2004.

[9] Keith Ito and Linda Johnson. The LJ Speech Dataset. `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[10] Joel Frank and Lea Schönherr. WaveFake: A Data Set to Facilitate Audio Deepfake Detection. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[11] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32, 2019.

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

[13] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

[14] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[15] Shentong Mo, Haofan Wang, Pinxu Ren, and Ta-Chung Chi. Automatic Speech Verification Spoofing Detection. 12 2020.

[16] K. N. R. K. Raju Alluri and Anil Kumar Vuppala. IIIT-H Spoofing Countermeasures for Automatic Speaker Verification Spoofing and Countermeasures Challenge 2019. In Gernot Kubin and Zdravko Kacic, editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 1043–1047. ISCA, 2019.

[17] Bhusan Chettri, Daniel Stoller, Veronica Morfi, Marco A. Martínez Ramírez, Emmanouil Benetos, and Bob L. Sturm. Ensemble Models for Spoofing Detection in Automatic Speaker Verification, 2019.

[18] Guang Hua, Andrew Beng Jin Teoh, and Haijian Zhang. Towards End-to-End Synthetic Speech Detection. *IEEE Signal Processing Letters*, 28:1265–1269, 2021.

[19] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, December 2014.

[22] Hemlata Tak, Jose Patino, Massimiliano Todisco, Andreas Nautsch, Nicholas Evans, and Anthony Larcher. End-to-End anti-spoofing with RawNet2. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373, 2021.