Efficient Classification of Partially Faked Audio Using Deep Learning

Abdulazeez AlAli
School of Computer Science and Informatics
Cardiff University
Cardiff, UK
aliaa8@cardiff.ac.uk

George Theodorakopoulos
School of Computer Science and Informatics
Cardiff University
Cardiff, UK
theodorakopoulosg@cardiff.ac.uk

Abdullah Emad
School of Science and Technology
Zewail City
6th of October, Egypt
t-abdullahsabry@zewailcity.edu.eg

***Abstract*—The rise of synthetic and manipulated audio content, especially partial fake speech, presents significant challenges for verifying audio authenticity. Partial fake speech refers to segments of audio in which only certain parts have been altered or synthesized, making it more difficult to detect compared to fully synthetic speech. This paper introduces a novel detection model specifically designed to identify partial fake speech. Our approach incorporates Wav2Vec 2.0 as a feature extractor, along with max pooling, conformer blocks, attention-based pooling, and fully connected layers. Experimental results on two datasets demonstrate the model's effectiveness in detecting partial fake speech. Our models outperforms existing methods in terms of Equal Error Rate (EER), achieving 0% on the RFP dataset and 2.99% on the ASVSpoof 2019 LA dataset.**

***Index Terms*—Biometric security, deepfake audio, neural networks, partial fake audio, speech synthesis.**

I. Introduction

The rapid advancement of speech synthesis and manipulation technologies, including deep learning-based text-to-speech (TTS) [1]–[4] and voice conversion (VC) [5]–[7] systems, has enabled the creation of highly realistic synthetic speech. While these technologies offer beneficial applications, they also pose significant risks, particularly with the rise of partial fake speech attacks [8]. In these attacks, only specific segments of an audio recording are manipulated or replaced with synthetic content, while the rest of the audio remains real. This selective alteration makes partial fake speech more deceptive and harder to detect than fully synthetic speech, as the modified segments are often seamlessly integrated into the original recording.

Partial fake speech attacks pose significant risks across multiple domains, including misinformation campaigns, identity theft, fraud, and the deterioration of trust in digital media. For example, attackers can alter phrases within a speech to change its meaning, create fake evidence, or impersonate individuals for harmful purposes [8]–[10]. Consequently, there is an urgent need for effective detection mechanisms that can identify these sophisticated manipulations.

Several techniques were proposed for detecting partial fake speech [10]–[12]. However, these techniques were only evaluated on the ADD 2023 challenge [13] dataset, which is not publicly available.

Detecting partial fake speech is inherently more complex than identifying fully synthetic audio. Recent research has focused on utilizing deep learning models to analyze entire audio files and localize the fake segments [9], [11].

II. Motivation

The existing tools for detecting fakes and partial fakes (PF) are often optimized for performance on specific datasets. However, their effectiveness may significantly decrease when they are applied to different datasets, such as the RFP dataset [14]. For example, in previous research [8] we conducted partial fake attack experiments against three state-of-the-art (SOTA) detection methods, and we found that the Equal Error Rate (EER) for these experiments ranged from 3.70% to 50.16%. In contrast, the EER for the original datasets, containing entirely fake audio, was much lower, ranging from 1.34% to 4.9%.

Our objective in this paper is to answer the following question:

> Can we create a PF detection tool that performs effectively regardless of the dataset used for training and testing?

III. Partial Fake Speech Detection Model

To investigate the above-proposed question for PF detection, we used the RFP dataset [14] and ASVSpoof 2019 LA dataset [15]. Next, we develop our detection model, which consists of five main components, as illustrated in Figure 1.

Our model, named Partial Fake Detection Model (PFDM), is designed to classify partially faked audio data into fake

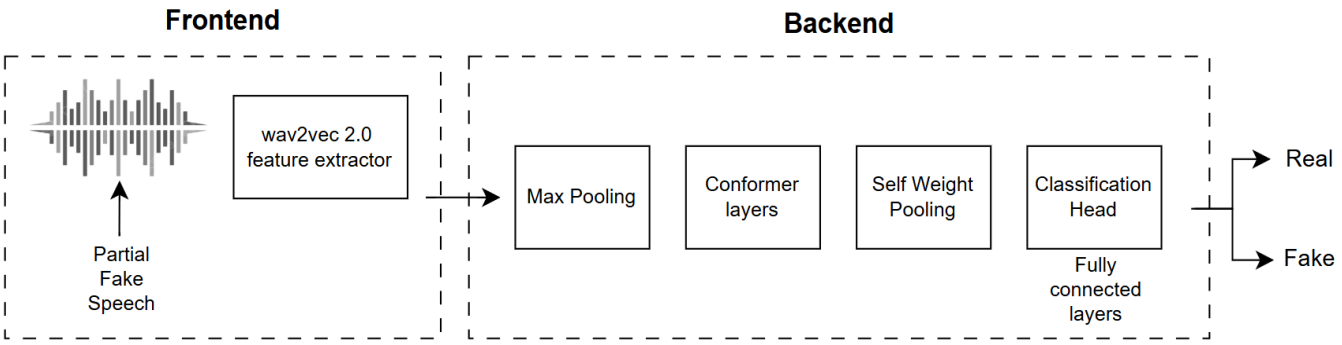

Fig. 1. Our proposed detection model

and real. We begin by integrating several ML components into the model and assessing their detection performance until we identify the optimal combination. This involves removing components that negatively impact the detection results. Our neural network model uses Wav2Vec 2.0 [16] as a feature extractor, and it is the first model that integrates this specific combination of several components, explained below. These components include Max Pooling, a Conformer block, Self-Weighted Pooling, and a fully connected refinement block for classification. This architecture is specifically designed to handle audio-based spoofing detection effectively. It extracts features from the raw audio signal and utilizes advanced deep learning components for precise classification.

A. Wav2Vec 2.0 (Feature Extractor)

The model utilizes Wav2Vec 2.0 [16] as the initial component in the pipeline, functioning as a feature extractor. Wav2Vec 2.0 is a pre-trained neural network that learns to represent raw audio data. It transforms the raw waveform input into high-level feature vectors that capture the essential acoustic characteristics of the audio. These feature vectors then serve as input for the subsequent layers of the model. This pre-trained feature extractor significantly minimizes the need for manual feature engineering while providing high-quality, context-aware representations of speech signals.

Input: The Wav2Vec components receive two inputs: the batch size and the time steps. The batch size refers to the number of data samples processed together in one training and testing iteration in machine learning. Time steps indicate the individual units of time in an audio signal during which features are extracted and analyzed. The raw waveform is processed across these time steps, and the model typically makes predictions based on how these features change over time.

Output: The output of wav2vec includes the batch size, time resolution, and feature dimension. The time resolution is determined by wav2vec and specifies the smallest time unit used to extract features from the audio signal. A higher time resolution allows the model to analyze more detailed segments of the audio over time, while a lower time resolution results in analysis over larger, more averaged time periods. This component is capable of handling audio files of various lengths. We used the base model, which has a feature dimension of 768 [16].

B. Max Pooling

Max pooling is a downsampling technique widely used in Convolutional Neural Networks (CNNs) and other deep learning models. It operates by sliding a fixed-size window over the input data and selecting the maximum value within each region. This process reduces the spatial dimensions (height and width) of the input while preserving the most important features. As a result, it decreases the computational load and memory requirements for processing.

Additionally, max pooling introduces a level of invariance to small translations in the input data, which can enhance the model's robustness. It enables the model to concentrate on more significant patterns and makes it less sensitive to minor changes or noise in the data.

In this model, max pooling is applied after feature extraction to reduce the sequence length, thereby reducing computational complexity and increasing efficiency. The pooling operation is governed by a hyperparameter known as the 'max pooling factor', which defines the size of the pooling window and the stride, or step size, of the window. By utilizing max pooling, the model decreases the feature dimensions, allowing for more efficient data processing without sacrificing critical information. This pooling factor ensures that the model's representation becomes more compact, capturing the most relevant features.

Overall, max pooling enables the model to focus on essential features while reducing the input size. This approach leads to fewer parameters, lower memory consumption, and faster computations. It also contributes to the model's ability to generalize better by making it more invariant to small translations in the input data.

Output: The 768 feature dimensions obtained from the wav2vec block are reduced in the max pooling block to 256. The output of the max pooling layer contains the batch size, time resolution, and feature dimensions with a size of 256.

C. Conformer Block

After the audio is processed by wav2vec 2.0 and max pooling, the resulting feature vectors are sent through a Conformer block. The Conformer block, which stands for Convolution-augmented Transformer, is highly effective for PF detection. It successfully combines the strengths of both convolutional neural networks (CNNs) and transformers, making it especially suitable for audio data. This type of data is sequential and often contains both local (short-term) and global (long-term) dependencies.

Here are the reasons why the Conformer block is particularly effective in PF audio detection:

A. Capturing Local and Global Features:

- Convolutional Layers: The convolutional component of the Conformer block is responsible for identifying local features in the audio signal. These features represent short-term localized patterns in the waveform, such as transitions between phonemes. In the context of speech, this involves capturing details about how sounds or words evolve over time.
- Transformer Layers: The transformer component, which utilizes a self-attention mechanism, allows the model to capture long range dependencies across time. This is crucial for PF detection tasks, where understanding the global context, like sentence structure or emotional tone, is important. For example, recognizing how a segment of a sentence connects to the overall structure can enhance the accuracy of speech classification.

B. Handling Sequential Data:

- Audio signals are sequential, meaning that each part of the signal depends heavily on the previous one, such as phonemes in speech or notes in music.
- The self-attention mechanism of the Conformer block enables the model to process sequential dependencies by considering all time steps at once. This capability helps it capture long-term patterns that may cover significant portions of the audio. For instance, the relationship between distant syllables in a spoken sentence can be crucial, and the transformer's attention mechanism effectively captures these connections.

Output: The output of the conformer block consists of batch size, time resolution and feature dimension 256.

D. Attention Based Pooling

To reduce the dimensionality of the sequence while preserving key feature representations, we apply Self-Weighted Pooling. This global pooling layer reduces the sequence length, while the self-weighting mechanism allows the model to focus on the most informative parts of the sequence. The SAP pooling layer employs a mean pooling strategy that calculates the mean of features across different time resolutions. This ensures a compact feature representation highlighting the critical aspects of the audio necessary for PF detection.

Output: The output of the attention-based pooling block consists of batch size and feature dimension 256.

E. Fully Connected Layers

The feature vector is processed through a fully connected refinement block, which further refines the features before detection. This refinement block consists of the following:

- A linear layer followed by a GELU activation and layer normalization.
- A dropout layer is used for regularization, which helps prevent overfitting in the training data.

Output: The final output layer consists of a linear layer that generates the ultimate detection score. This score is then processed through a sigmoid activation function to yield a value between 0 and 1, indicating whether the input audio is real or fake.

F. Hyperparameters Tuning Techniques

- *Dropout Scheduler.* Dropout is a regularization technique used in neural networks to prevent overfitting. The core idea is to randomly 'drop' (set to zero) a fraction of the input units (neurons) during training. This process compels the network to learn redundant representations and prevents it from becoming overly reliant on any specific feature. The dropout rate - the fraction of units to drop - is a hyperparameter that typically remains fixed throughout training.

In our model, the dropout probability is dynamically adjusted using a cosine annealing scheduler. Initially, the dropout probability is set to its maximum value and gradually decreases as the number of epochs increases. This method allows the model to start with strong regularization (high dropout) and then to progressively reduce the dropout rate as training progresses. This approach can enhance the model's ability to learn better representations in the later stages of training. The adjustment formula for the dropout probability is based on the cosine function, ensuring a smooth transition from the maximum to the minimum value over the course of training. This helps prevent overfitting early on, while allowing the model to refine its representations without excessive regularization later.

The following is the cosine-based formula for dropout probability p over training epochs:

$$p_t = p_{\max} \cdot \frac{1 + \cos\left(\frac{t}{T}\pi\right)}{2}$$

Where:

- p_t is the dropout probability at epoch t.
- $p_{\max}$ is the maximum dropout probability (usually set at the start of training).
- T is the total number of training epochs.
- $\cos\left(\frac{t}{T}\pi\right)$ smoothly decreases from 1 to -1 as t increases.

This scheduling strikes a balance between exploration and exploitation. In the early phase, a higher dropout rate encourages the model to explore more diverse features, whereas reduced dropout later in training permits the model to better exploit the features it has learned. This dynamic adjustment enhances generalization without compromising performance.

- *Learning Rate Scheduler.* The learning rate is a critical hyperparameter that determines the size of the steps taken

during each iteration when updating model weights. In deep learning, it is common to adjust the learning rate throughout training to ensure efficient convergence. If the learning rate is set too high, the model may overshoot the optimal weights. Conversely, if it is too low, the training process may become excessively slow and could get stuck in suboptimal solutions.

In our model, we adjust the learning rate dynamically using the ReduceLROnPlateau scheduler from PyTorch. This scheduler reduces the learning rate when a specified performance metric, such as validation loss, stops improving. It is designed to monitor the model's performance during training and lower the learning rate when progress stagnates, helping the model to converge more accurately in the later training stages.

The scheduler has several parameters that control when and how the learning rate is adjusted:

- Factor: This refers to the rate at which the learning rate is decreased when the model stops showing improvement. For example, a factor of 4/5 indicates that the learning rate is reduced by 20% each time the model's performance plateaus.
- Patience: This is the number of epochs without any improvement before the learning rate is reduced. It helps prevent unnecessary adjustments if the model is still experiencing gradual progress. We used 5 epochs in our model.
- Threshold: This represents the minimum change in the monitored metric that is considered an improvement. If the change is smaller than the threshold, the learning rate will be decreased.
- Min LR: This is the minimum learning rate that the scheduler can reach, ensuring that the learning rate does not become so small that it loses its effectiveness.

Impact on Training. The learning rate scheduler assists the model in avoiding overshooting or stagnation during training. By decreasing the learning rate when needed, the model is more likely to converge on an optimal solution. This adaptive learning rate enables more precise updates, especially in the later stages of training when the model is close to convergence.

Model Parameters: Our model consists of only 647,617 parameters, making it significantly more parameter-efficient than alternative models like gMLP [17], which has 23,496,283 parameters. The compact design of PFDM enables faster training and inference, making it a practical choice for real-time spoofing detection without sacrificing performance.

IV. Experiments

To assess the effectiveness of our proposed model, we conduct two experiments: The first experiment (Section IV-B) evaluates the effectiveness of the model when detecting partial fake (PF) speech using the RFP dataset. Additionally, we tested the model's ability to detect entirely fake audio using the ASVspoof 2019 dataset. These two datasets vary in audio length, and our model can handle all of them effectively, as discussed earlier in the wav2vec component Section III. The second experiment (Section IV-C) examines how the length of fake segments affects the detection process.

A. Metrics

The Equal Error Rate (EER) is a commonly used metric for assessing the performance of fake speech detection models. It represents the point at which the False Acceptance Rate (FAR) and False Rejection Rate (FRR) are equal. Here is how we calculate EER for audio classification:

For each audio sample, the system produces a score that indicates the likelihood of the sample belonging to the target class (e.g., a specific speaker). A threshold (t) must then be defined, which serves as a decision boundary used to classify audio samples.

A lower threshold may accept more fake samples, thereby increasing the FAR, while a higher threshold may reject more real samples, which increases the FRR. The threshold determines how the scores generated by the model are converted into class labels. When a model predicts the probability of an instance belonging to a specific class, the classification threshold specifies the point at which this probability is considered sufficient to assign the instance to that class.

Next, we calculate the False Acceptance Rate (FAR), which is the percentage of fake (negative) samples incorrectly classified as real (positive), and the False Rejection Rate (FRR), which is the percentage of real (positive) samples incorrectly classified as fake (negative). The EER value is the threshold value at which FAR equals FRR.

$$\text{FAR}(\tau) = \frac{\text{False Acceptances (FA)}}{\text{Total Fake Samples (TN + FA)}}$$

$$\text{FRR}(\tau) = \frac{\text{False Rejections (FR)}}{\text{Total Real Samples (TP + FR)}}$$

$$\text{EER} = \text{FAR}(\tau) = \text{FRR}(\tau)$$

B. Experiment 1: Partial Fake and Entirely Fake detection performance

In this experiment, we study the performance of our model when detecting partial fake audio and when detecting entirely fake audio.

Datasets The RFP dataset [14] contains Real, Fake, and Partial Fake speech data. In this experiment, we only use the Partial Fake subset of RFP. This subset has 58742 audio files in total, which we split into Training, Validation, and Evaluation datasets in a ratio of approximately 70:20:10. We also use the ASVSpoof2019 LA dataset [15], which contains 98299 audio files in total, all of which are entirely fake audio. This dataset is already split into Training, Validation, and Evaluation datasets in a ratio of approximately 21:20:59. Table I shows the exact number of files in the split for each of the two datasets. No other processing was done on the datasets.

Results We use the RFP dataset to compare the EER performance of our detection method with three publicly available SOTA methods: PartialSpoof [9], M2S-ADD [18], and SSL_Anti-spoofing [19]. Note that in this case we are detecting partial fake audio.

TABLE I
NUMBER OF FILES FOR TRAINING, VALIDATION, AND EVALUATION FOR THE TWO DATASETS IN EXPERIMENT 1

Subset	PF subset of RFP	ASVSpoof2019 LA
Training (# Files)	40,938	25,380
Validation (# Files)	11,662	986
Evaluation (# Files)	6,142	71,933
Total (# Files)	58742	98299

TABLE II
PARTIAL FAKE AUDIO DETECTION: EER PERFORMANCE IN THE RFP DATASET

Detection Method	Evaluation EER
Ours	0%
PartialSpoof [9]	3.70%
M2S-ADD [18]	18.69%
SSL_Anti-spoofing [19]	50.16%

The results in Table II show that our model outperforms all three SOTA models in detecting PF utterances in the RFP dataset.

We also evaluated our model when detecting entirely fake audio, using the ASVspoof 2019 LA dataset. Our model achieved a 2.99% EER, outperforming the ASVspoof 2019 challenge baselines as well as many models that were specifically designed to detect entirely fake audio. Table III illustrates a comparison between the results of our model and other detection models.

C. Experiment 2: Varying the Length of the Fake Segments

In this experiment, we study how varying lengths of fake segments impact our detection model.

Datasets We created 10-second PF files organized into 18 groups, each with different fake segment lengths. The files in Group `10f90r` begin with 1 second of a fake segment (f) followed by 9 seconds of a real segment (r). The files in each subsequent group `20f80r` to `90f10r` begin with a longer fake segment followed by a correspondingly shorter real segment, so the total length is always 10 seconds. Then, groups `10r90f` to `90r10f` begin with a real segment (1 second to 9 seconds long), followed by the fake segment (9 seconds to 1 second long). The real and fake segments are taken from the RFP dataset.

TABLE III
ENTIRELY FAKE AUDIO DETECTION: EER PERFORMANCE IN THE ASVSPOOF 2019 LA DATASET

Method	Validation	Evaluation EER
Ours	1.23	**2.99**
LCNN+CE [20]	0.86	3.13
Resnet18-AM-Softmax [21]	0.43	3.26
GMM [22]	-	3.50
LCNN-4CBAM [23]	-	3.67
ResNet [24]	1.52	3.72
End-to-End Inc-TSSDNet [25]	1.09	4.04
8 Features+MLP [26]	0.00	4.13
ResNet18-GAT-S [27]	-	4.48
PC_DARTS [28]	0.00	4.96
ASVspoof2019 baseline LFCC+GMM [29]	0.43	9.57
ASVspoof2019 baseline CQCC+GMM [29]	2.71	8.09

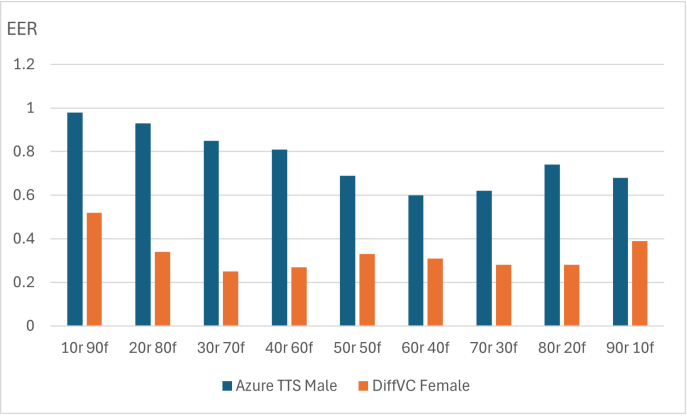

Fig. 2. The EER using 10 s PF files starting with real segment against our proposed model

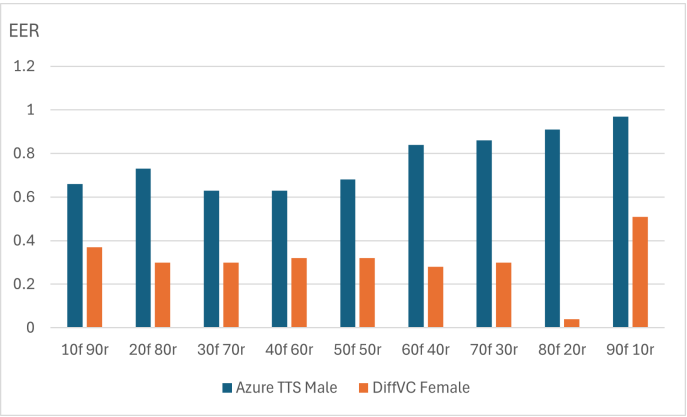

Fig. 3. The EER using 10 s PF files starting with fake segment against our proposed model

Results Figures 2 and 3 illustrate the results of Experiment 2. The detection performance for PF audios generated by VC was superior to that of audios generated by TTS methods. The best detection performance occurred when the audio contained 8 seconds of fake segments followed by 2 seconds of real segments, achieving an EER of 0.04%. In contrast, the optimum detection result for TTS was observed when the audio started with 6 real segments followed by 4 seconds of fake segments, yielding an EER of 0.60%.

From our observations, we noted that as the percentage of fake audio increases, detection becomes easier, particularly with VC. However, this trend was less consistent with TTS.

V. DISCUSSION

In this paper, we demonstrate the need to create an efficient model for detecting PF audio files. This model should be capable of identifying PF audio regardless of the methods used to generate fake segments, the length of those fake segments, the overall length of the audio. Building on our previous experiments [8], we developed a model and evaluated

it using the RFP dataset, which includes all the mentioned audio types. Additionally, we expanded our experiments by generating new PF fake files that are 10 seconds long, as discussed in Experiment 2.

Our proposed model, which focuses solely on fake and PF audio, achieves an EER of 0% outperforming all existing detection models. We also tested our model using one of the most recognized datasets, ASVspoof 2019 LA, where it achieved a result of 2.99%. This result surpassed all ASVspoof challenge baselines, as well as many other state-of-the-art detection tools.

VI. Conclusion

We proposed a new detection model for PF speech audio. The model was evaluated using three datasets and achieved outstanding results compared to other existing detection tools. We conducted two experiments to assess the model, both of which demonstrated high detection performance. We believe that this model will serve as a foundation for future work in various tasks, such as PF speech localization, which involves identifying the start and end of fake segments within the audio, and PF speech diarization, the process of clustering spoofed segments based on different spoofing methods.

References

[1] Sefik Emre Eskimez, Xiaofei Wang, Manthan Thakker, Canrun Li, Chung-Hsien Tsai, Zhen Xiao, Hemin Yang, Zirun Zhu, Min Tang, Xu Tan, et al. E2 tts: Embarrassingly easy fully non-autoregressive zero-shot tts. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pages 682–689. IEEE, 2024.

[2] Edresson Casanova, Julian Weber, Christopher D Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A Ponti. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International Conference on Machine Learning*, pages 2709–2720. PMLR, 2022.

[3] Edresson Casanova, Kelly Davis, Eren Gölge, Görkem Göknar, Iulian Gulea, Logan Hart, Aya Aljafari, Joshua Meyer, Reuben Morais, Samuel Olayemi, et al. Xtts: a massively multilingual zero-shot text-to-speech model. *arXiv preprint arXiv:2406.04904*, 2024.

[4] Sungwon Kim, Kevin Shih, Joao Felipe Santos, Evelina Bakhturina, Mikyas Desta, Rafael Valle, Sungroh Yoon, Bryan Catanzaro, et al. P-flow: A fast and data-efficient zero-shot tts through speech prompting. *Advances in Neural Information Processing Systems*, 36:74213–74228, 2023.

[5] Zhichao Wang, Yuanzhe Chen, Lei Xie, Qiao Tian, and Yuping Wang. Lm-vc: Zero-shot voice conversion via speech generation based on language models. *IEEE Signal Processing Letters*, 30:1157–1161, 2023.

[6] Zhiyuan Tan, Jianguo Wei, Junhai Xu, Yuqing He, and Wenhuan Lu. Zero-shot voice conversion with adjusted speaker embeddings and simple acoustic features. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5964–5968. IEEE, 2021.

[7] Kaizhi Qian, Yang Zhang, Shiyu Chang, Xuesong Yang, and Mark Hasegawa-Johnson. Autovc: Zero-shot voice style transfer with only autoencoder loss. In *International Conference on Machine Learning*, pages 5210–5219. PMLR, 2019.

[8] Abdulazeez Alali and George Theodorakopoulos. Partial fake speech attacks in the real world using deepfake audio. *Journal of Cybersecurity and Privacy*, 5(1):6, 2025.

[9] Lin Zhang, Xin Wang, Erica Cooper, Nicholas Evans, and Junichi Yamagishi. The partialspoof database and countermeasures for the detection of short fake speech segments embedded in an utterance. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:813–825, 2022.

[10] Lam Pham, Phat Lam, Dat Tran, Hieu Tang, Tin Nguyen, Alexander Schindler, Florian Skopik, Alexander Polonsky, and Canh Vu. A comprehensive survey with critical analysis for deepfake speech detection. *arXiv preprint arXiv:2409.15180*, 2024.

[11] Juan Manuel Martín-Doñas and Aitor Álvarez. The Vicomtech partial deepfake detection and location system for the 2023 add challenge. In *DADA@ IJCAI*, pages 37–42, 2023.

[12] Liwei Liu, Huihui Wei, Dongya Liu, and Zhonghua Fu. Harmonet: Partial deepfake detection network based on multi-scale harmof0 feature fusion. In *Proc. Interspeech*, volume 2024, pages 2255–2259, 2024.

[13] Jiangyan Yi, Jianhua Tao, Ruibo Fu, Xinrui Yan, Chenglong Wang, Tao Wang, Chu Yuan Zhang, Xiaohui Zhang, Yan Zhao, Yong Ren, et al. Add 2023: the second audio deepfake detection challenge. *arXiv preprint arXiv:2305.13774*, 2023.

[14] Abdulazeez AlAli and George Theodorakopoulos. An RFP dataset for real, fake, and partially fake audio detection. In *International Conference on Cyber Security, Privacy in Communication Networks*, pages 1–15. Springer, 2023.

[15] Xin Wang, Junichi Yamagishi, Massimiliano Todisco, Héctor Delgado, Andreas Nautsch, Nicholas Evans, Md Sahidullah, Ville Vestman, Tomi Kinnunen, Kong Aik Lee, et al. ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech. *Computer Speech & Language*, 64:101114, 2020.

[16] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[17] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in neural information processing systems*, 34:9204–9215, 2021.

[18] Rui Liu, Jinhua Zhang, Guanglai Gao, and Haizhou Li. Betray oneself: A novel audio deepfake detection model via mono-to-stereo conversion. *arXiv preprint arXiv:2305.16353*, 2023.

[19] Hemlata Tak, Massimiliano Todisco, Xin Wang, Jee-weon Jung, Junichi Yamagishi, and Nicholas Evans. Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation. *arXiv preprint arXiv:2202.12233*, 2022.

[20] Rohan Kumar Das, Jichen Yang, and Haizhou Li. Data augmentation with signal companding for detection of logical access attacks. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6349–6353. IEEE, 2021.

[21] You Zhang, Fei Jiang, and Zhiyao Duan. One-class learning towards synthetic voice spoofing detection. *IEEE Signal Processing Letters*, 28:937–941, 2021.

[22] Hemlata Tak, Jose Patino, Andreas Nautsch, Nicholas Evans, and Massimiliano Todisco. Spoofing attack detection using the non-linear fusion of sub-band classifiers. *arXiv preprint arXiv:2005.10393*, 2020.

[23] Xinyue Ma, Tianyu Liang, Shanshan Zhang, Shen Huang, and Liang He. Improved lightcnn with attention modules for asv spoofing detection. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[24] Jichen Yang, Hongji Wang, Rohan Kumar Das, and Yanmin Qian. Modified magnitude-phase spectrum information for spoofing detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1065–1078, 2021.

[25] Guang Hua, Andrew Beng Jin Teoh, and Haijian Zhang. Towards end-to-end synthetic speech detection. *IEEE Signal Processing Letters*, 28:1265–1269, 2021.

[26] Rohan Kumar Das, Jichen Yang, and Haizhou Li. Long range acoustic features for spoofed speech detection. In *Interspeech*, pages 1058–1062, 2019.

[27] Hemlata Tak, Jee-weon Jung, Jose Patino, Massimiliano Todisco, and Nicholas Evans. Graph attention networks for anti-spoofing. *arXiv:2104.03654*, 2021.

[28] Wanying Ge, Michele Panariello, Jose Patino, Massimiliano Todisco, and Nicholas Evans. Partially-connected differentiable architecture search for deepfake and spoofing detection. *arXiv:2104.03123*, 2021.

[29] Massimiliano Todisco, Xin Wang, Ville Vestman, Md Sahidullah, Héctor Delgado, Andreas Nautsch, Junichi Yamagishi, Nicholas Evans, Tomi Kinnunen, and Kong Aik Lee. Asvspoof 2019: Future horizons in spoofed and fake audio detection. *arXiv:1904.05441*, 2019.