Deepfake Audio Detection With Time-Delay Neural Network

Chun-Hua Lin
Columbia university
cl4335@columbia.edu

***Abstract*—Deepfake audios have emerged as a significant concern in recent years due to their potential use for fraud, impersonation, and spreading disinformation. Convolutional neural networks (CNN) and long short-term memory networks (LSTM) have been widely used in Deepfake audio detection research, however in this study, we focus on applying time-delay neural network (TDNN) architecture to the problem. TDNN uses a fixed-length sliding window to capture local dependencies between nearby elements in the sequence, producing a compressed representation of the input sequence. While TDNN still takes into account the time dimension of the audio data, it has the advantage of being faster and easier to train compared to other methods such as LSTMs, making it an attractive option for Deepfake audio detection. Our approach computes MFCCs for input audio files utilizing Kaldi and passes them into a TDNN to capture local features and generate feature maps for further inference. We compare the results of TDNN to CNN-based and LSTM methods using the ASVspoof 2019 database of bona fide and spoofed speech signals. Our experiments reveal that TDNN is an efficient and effective method for Deepfake audio detection.**

***Index Terms*—TDNN, CNN, LSTM, DL, deepfake, MFCC, kaldi, ASVspoof 2019**

I. Introduction

Deepfake audio has become a significant concern in recent years due to its potential for fraud, impersonation, defamation, spreading disinformation, manipulation of public opinion for propaganda, or even terrorism [1]. With the advancement of AI-synthesized tools developed with the ability to generate high-quality synthetic audio. Therefore, the development of reliable and efficient Deepfake audio detection methods has become crucial. Many methods of detection have been developed to distinguish fake audio files from real speech Deepfakes. Different deep learning models have been built that utilize different approaches to identify fake audio.

A. Deepfake Audio Attacks

Deepfake audio refers to the use of technology to create fake audio. This type of audio can be created using different techniques, including imitation-based, synthetic-based, and replay-based methods.

Imitation-based Deepfakes are a method for converting speech to resemble another speech. Voices may be imitated using a variety of techniques, such as finding humans who possess similar voices and can imitate the original speaker. However, masking algorithms have also been created to imitate audio and produce Deepfake speech. Employed an imitation generation method, the original audio signal is converted to produce the target audio speech, resulting in a new fake speech. As a consequence, distinguishing between genuine and fake audio created using this technique is challenging for humans.

Synthetic-based or Text-To-Speech (TTS) aims to convert text into natural speech. To generate synthetic Deepfake audio, first, feed the transcript text with the target speaker's voice into the generation model. Then, the text analysis module processes the incoming text and converts it into linguistic characteristics. Next, the acoustic module extracts the target speaker's parameters from the dataset, based on the linguistic features generated by the text analysis module. Finally, the vocoder learns to produce speech waveforms using the acoustic feature parameters. Well-known models include FastSpeech [2], Tactoran [3], and DeepVoice [4].

Replay-based Deepfakes aim to replay a recording of the target speaker's voice. There are two types of replay-based Deepfakes: far-field detection and cut-and-paste detection. Far-field detection involves playing a microphone recording of the victim's voice on a telephone handset with a loudspeaker. On the other hand, cut-and-paste detection involves fabricating the required sentence for a text-dependent system.

B. Deepfake Audio Detections

Convolutional neural networks (CNN) and long short-term memory networks (LSTM) have been widely used in deepfake audio detection research due to their ability to extract relevant features from audio signals and have achieved promising results in detecting deepfake audio.

CNNs are particularly effective in analyzing and processing high-dimensional data, such as spectrograms, which are commonly used to represent audio signals, it is also better in capturing spurious correlations. However, the main limitation of CNN is that they can only handle images as input, and thus the audio needs to be preprocessed and transformed to a spectrogram or 2D figure to be able to feed it as input to the network.

LSTMs are a well-liked option for studying sequential data due to their proficiency in capturing long-term dependencies. Audio data sometimes consists of lengthy sequences of sounds that are closely related to one another. However, the lengthy training process of LSTMs, which can be time-consuming and costly computationally is one of its key shortcomings when working with big datasets or intricate models. Bidirectional LSTMs (BiLSTMs) are a type of LSTM that can capture dependencies not only in the past but also in the future by processing the data in both directions. BiLSTMs are particularly useful in scenarios where both the past and future contexts are important for making accurate predictions. However, the training process for BiLSTMs can also be time-consuming due to the large number of parameters involved. More models can be seen in [1].

C. Audio Features

CQCC, LFCC, and MFCC are three important audio features that have been widely used in audio signal processing for various applications, including speech recognition, music analysis, and audio classification, including Deepfake audio detection.

1) CQCC: Constant Q Cepstral Coefficients, is a feature that is extracted from the audio spectrum using a constant-Q transform. This technique helps to capture the pitch variation and frequency resolution in a more human-like manner than other transforms. The CQCC feature extraction algorithm is based on a combination of log-spectral amplitude and cepstral analysis, making it robust to noise and distortions.

2) LFCC: Linear Frequency Cepstral Coefficients, is another audio feature that has been extensively used for speech recognition and audio classification. LFCC is similar to the MFCC (Mel Frequency Cepstral Coefficients) feature, but instead of using the Mel-scale, LFCC uses a linear-scale. The linear-scale makes it easier to capture the spectral details of the audio in the low-frequency region, which is important for some applications, such as speaker recognition.

3) MFCC: Mel Frequency Cepstral Coefficients, is a widely used feature for speech recognition and audio classification, including Deepfake audio detection. The MFCC feature extraction technique involves taking the log of the power spectrum of the audio signal, followed by a transformation to a mel-scale, which maps the frequency range to a perceptually more uniform scale. The resulting mel-spectrum is then transformed using the Discrete Cosine Transform (DCT) to extract the MFCC coefficients, which represent the spectral envelope of the audio signal.

In this study, we focus on applying the TDNN architecture to the problem of deepfake audio detection and compare its results to benchmark CNN-based and LSTM methods [1]. We use the ASVspoof 2019 dataset [5], which includes all three types of deepfake audio attacks. However, we focus on synthetic and imitation Deepfakes, and replay-based attacks will be considered out of scope. Furthermore, We compute the MFCCs for input audio files. We demonstrate that TDNN is an efficient and effective method for deepfake audio detection.

II. State Of The Art

The state-of-the-art [6] presents a novel audio features descriptor named extended local ternary pattern (ELTP) based on a Local Ternary Pattern to capture the vocal tract dynamically induced attributes of bonafide speech and algorithmic artifacts in synthetic and converted speeches. They fused LFCC with the proposed ELTP feature to further strengthen the capability of the features for capturing the traits of bonafide and spoofed signals. The proposed ELTP-LFCC features are then employed to train the deep bidirectional Long Short-Term Memory (DBiLSTM) network for the classification of the bonafide and spoof signal to increase the robustness of the model and to detect fake audio in diverse indoor and outdoor environmental conditions. The model was tested over the ASVspoof 2019 dataset [5].

TABLE I
Performance comparison of SOTA and existing studies

Method	**ERR(%)**	**t-DCF**
CQCC+GMM (ASVspoof baseline[1]) [7]	2.71	0.0663
LFCC+GMM (ASVspoof baseline[1]) [7]	0.43	0.0123
MFCC+CQCC+Spec+ResNet [8]	6.02	0.1569
ELTP-MFCC+DBiLSTM [6]	0.74	0.008

[1]Baseline results in [7] differ from [6], [8].

A. ELTP-LFCC

The proposed approach involves an automated threshold computation method that calculates the standard deviation locally for each audio frame. The method utilizes ELTP features to analyze audio patterns in the time domain and capture algorithmic artifacts in synthetic speech signals as well as vocal tract-induced variations in genuine signals. The approach also integrates LFCC to extract significant information from the low- and high-frequency bands of the audios.

B. DBiLSTM

The proposed DBiLSTM used 10 bidirectional LSTM layers, each with 64 hidden units. Extracted ELTP-LFCC features are fed to the first BiLSTM layer. The outputs of one BiLSTM layer are concatenated and passed to the next BiLSTM layer. Feature vector from the 10th BiLSTM layer is passed into a fully connected (FC) layer. The output of FC layer is propagated to a softmax layer and finally to a classification layer that assigns each input to one of the mutually exclusive classes.

Table I presents the t-DCF and EER results of the proposed method and several comparative methods. Note that the proposed method outperforms existing methods, including the ASVspoof baseline methods. However, also note that the reported baseline results in other studies [6], [8] differ from those in the official evaluation plan [7]. The baseline results cited in other studies are much worse than the official ones.

III. Method

In this study, we propose applying the TDNN to perform deepfake audio detection. We use the ASVspoof 2019 dataset [5], and compute the MFCCs for input audio files. Then, we use tandem detection cost function (t-DCF) and equal error rate (EER) as our evaluation metrics as the evaluation plan of ASVspoof 2019 dataset [7].

TABLE II
COMPONENT OF ASVSPOOF 2019 DATASET

	Scenario			Speaker			Spoofing Algorithm			Sample rate	Spoofing system
	Logical Access			Total: 107			VC	TTS	VC + TTS		
	Bonafide	Spoofed	Total	Male	Female	Total					
Train	2580	22800	25380	8	12	20	2	8	-	16 kHz	Known: 6 Unknown: 11
Development	2548	22296	24844	8	12	20					
Evaluation	7355	63822	71177	30	37	67	5	4	3		

TABLE III
CONFIGURATION OF MFCC FEATURE EXTRACTION

Option	
use-energy	false
num-mel-bins	40
num-ceps	40
sample-frequency	16000
frame-length	25
frame-shift	10

A. Dataset

The ASVspoof 2019 dataset [7] is a benchmark dataset for the development and evaluation of anti-spoofing countermeasures against automatic speaker verification (ASV) systems. It contains bona fide and spoofed speech signals generated with state-of-the-art technologies, including Tacotron [3] and WaveNet. The dataset is based on the Voice Cloning Toolkit (VCTK) corpus, which is a multi-speaker English speech database recorded in a hemi-anechoic chamber at a sampling rate of 96 kHz. The utterances from 107 speakers (46 male, 61 female) in the VCTK corpus are downsampled to 16 kHz at 16 bits-per-sample, and two different spoofing protocols are defined according to two use case scenarios: logical (LA) and physical access (PA) control. In this study, we focus on the LA scenario.

The LA scenario involves synthetic speech attacks where speech synthesis, text-to-speech (TTS), and voice conversion (VC) algorithms are presented to the ASV system without convolutive acoustic propagation or microphone effects. Spoofed audios are generated using different VC and TTS algorithms, processors, conversion, and waveform generators to eliminate or minimize the neural network from learning generator-dependent features. The ASVspoof 2019 LA dataset is divided into three subsets: training, development, and evaluation subsets, each containing bona fide and spoofed samples generated using several spoofing algorithms. The training subset contains 25,380 samples, the development subset contains 24,844 samples, and the evaluation subset contains 71,117 audio samples. The duration of each utterance ranges from one to two seconds, and all audio files in the three subsets are stored in flac format.

B. Feature Extraction

In our study, we compute the MFCC for our input audio files utilizing Kaldi. The command takes as input a list of audio files in wav format and outputs a binary file containing the extracted features. Therefore, we first convert the flac format audio files to into wav format using the command line tool sox.

Then we use the make-mfcc.sh tool in Kaldi to compute the MFCCs, the options can be seen in Table III. Specifically:

- sample-frequency=16000: This option specifies the sampling rate of the audio signal.
- frame-length=25: This option specifies the length of each frame in milliseconds.
- frame-shift=10: This option specifies the shift between consecutive frames in milliseconds. The frames overlap allows for temporal smoothing of the feature representation.
- use-energy=false: This option tells Kaldi to ignore the energy of each frame when computing the MFCC features. The energy of a speech signal refers to the amount of power contained in each frame, and it can be useful for distinguishing between speech and silence. However, it can also introduce noise into the feature representation. Therefore, we disregard the feature.
- num-mel-bins=40: This option specifies the number of Mel-frequency filterbank bins to use. The Mel scale is a non-linear transformation of frequency. The Mel-frequency filterbank is a set of triangular filters that are evenly spaced on the Mel-frequency scale. Each filter represents a band of frequencies, and the output of the filterbank is the energy in each band. We used a higher number of Mel bins to capture higher frequency resolution.
- num-ceps=40: This option specifies the number of cepstral coefficients to compute. Cepstral coefficients are a set of coefficients derived from the Fourier transform of the log power spectrum of a signal. We used a higher number to capture more information about the spectral envelope of the speech signal.

First, the audio signal is pre-emphasized using a high-pass filter to amplify the higher frequencies. Then, the audio signal is then divided into frames of 25ms with a frameshift of 10ms. The hamming window function is applied to each frame to reduce spectral leakage. Next, the power spectrum of each frame is computed using the discrete Fourier transform (DFT) Finally, the Mel filterbank is applied to the power spectrum, which groups the spectral energy into 40 Mel frequency bins. The logarithm of the filterbank energies is taken to convert the power values into a logarithmic scale, which is more similar to the human perception of sound.

The discrete cosine transform (DCT) is applied to the Mel

TABLE IV
t-DCF COST FUNCTION PARAMETERS ASSUMED IN ASVSPOOF 2019 [7]

Priors			**ASV costs**		**CM costs**	
π_{tar}	π_{non}	π_{spoof}	C^{asv}_{miss}	C^{asv}_{fa}	C^{cm}_{miss}	C^{cm}_{fa}
0.9405	0.0095	0.05	1	10	1	10

filterbank coefficients to decorrelate them and reduce the dimensionality of the feature space. We then obtained the resulting 40-dimensional MFCC feature vector.

C. Model

In this study, we propose a novel approach for deepfake audio detection using a simple yet effective Time Delay Neural Network (TDNN) structure. Our goal is to investigate the potential of TDNNs in detecting deepfake audios while keeping the number of parameters manageable. Unlike previous approaches that utilize complex neural network architectures with a large number of parameters, our proposed TDNN structure is lightweight.

Time-delay neural networks (TDNNs) are a type of neural network architecture designed for processing sequential data. TDNNs are an extension of the feedforward neural network model, but they incorporate time-delayed connections to allow the network to capture temporal patterns in the data. The time-delayed connections are achieved by introducing a set of fixed filters that slide over the input sequence in a way that allows the network to capture information from different time steps.

D. Metric

This study uses two metrics to evaluate the performance of the proposed TDNN-based deepfake audio detection system: equal error rate (EER) and tandem detection cost function (t-DCF) [10].

1) t-DCF: t-DCF is a cost function that takes into account both the errors in the decision-making process and the operational costs of the system. It is defined as the weighted sum of false acceptance cost (FAC), false rejection cost (FRC), and decision cost (DC). The lower the t-DCF, the better the performance of the system. The basic form of t-DCF is as follows:

$$t-DCF(s) = C_1 P^{cm}_{miss}(s) + C_2 P^{cm}_{fa}(s) \tag{1}$$

where $P^{cm}_{miss}(s)$ is the miss rate and $P^{cm}_{fa}(s)$ is the false alarm of the CM system at threshold s.

$$P^{cm}_{miss}(s) = \frac{\#\{\text{bonafide trials with CM score} \leq s\}}{\#\{\text{bonafide trials}\}} \tag{2}$$

$$P^{cm}_{fa}(s) = \frac{\#\{\text{spoof trials with CM score} > s\}}{\#\{\text{spoof trials}\}} \tag{3}$$

and C_1 and C_2 are dictated by the t-DCF costs, priors, and the ASV system detection errors, where the ASV systems is developed by the challenge organizers, therefore is given.

$$\begin{cases} C_1 = \pi_{tar}(C^{cm}_{miss} - C^{asv}_{miss} P^{asv}_{miss}) - \pi_{non} C^{asv}_{fa} C^{asv}_{fa} \\ C_2 = C^{cm}_{fa} \pi_{spoof}(1 - P^{asv}_{miss,spoof}) \end{cases}$$

where C^{asv}_{miss} is costs of ASV system to miss a target speaker and C^{asv}_{fa} is the costs of ASV system to accept a non-target (zero-effort impostor). C^{cm}_{miss} is the costs of CM system to reject a human (bonafide) trial and C^{cm}_{fa} is the costs of CM system to accept a spoof trial. In addition, π_{tar},π_{non}, π_{spoof} are the priori probability of target, nontarget and spoof. Note that $\pi_{tar}+\pi_{non}+\pi_{spoof} = 1$. The costs and prior probabilities are fixed in advance to values shown in Table V.

Then the defined normalized t-DCF is as:

$$t-DCF_{norm}(s) = \frac{C_1}{C_2} P^{cm}_{miss}(s) + P^{cm}_{fa}(s) \tag{4}$$

Then the defined minimum normalized t-DCF is as:

$$t-DCF^{min}_{norm} = t-DCF_{norm}(s*) \tag{5}$$

where $s* = argmin_s t - DCF_{norm}(s)$, which is the primary metric we will be using in this study.

2) ERR: EER is used in the past two editions of ASVspoof. EER corresponds to CM threshold s_{EER} at which the miss and false alarm rates equal each other.

$$ERR = P^{cm}_{fa}(s_{ERR}) = P^{cm}_{miss}(s_{ERR}) \tag{6}$$

IV. EXPERIMENT AND RESULT

For the experiment, we followed a series of steps. Firstly, we converted the flac format audio files into wav format using the command line tool "sox". Next, we created an scp file containing the path to the data file for Kaldi feature extraction. Using the wav files and scp file, we extracted MFCC features with specific steps outlined in the previous section. To ensure the consistency of input size to the TDNN, we truncated the extracted MFCC features with a fixed size length window. Finally, we fed the truncated MFCC features into the TDNN, and the output of the TDNN is the probability of whether the input data belongs to the bonafide or spoofed class.

The training and testing steps is outlined as follows:

1) Train: First, to ensure the consistency of input size for the TDNN, we truncated the extracted MFCC features using a fixed-size window. We also pad the segment where the window size exceeds the actual dimension of the feature. See Fig. 2 for the distribution of numbers of wav file duration in seconds and MFCC features per file. Most of the numbers of MFCC features per files lies in between 200 to 400, this is intuitive since the duration of each wav files in seconds is between 2 to 4 seconds. Therefore, we choose the window size of 100, 200, 400, and 600 to see how it affects the representation of the audio file. It is worth noting that in the training process, each segmentation of the audio file is treated as an independent input and was passed to the network for training. We experimented with different window sizes to determine how the length of the window affects the representation of an audio signal. Then, the training process consisted of two parts. Firstly, we trained the model for 10 epochs using the preprocessed training data set. Secondly, within each epoch, we evaluated the model's performance using the development set after every 100 minibatches. The weights associated with the lowest loss were saved as the best-performing model for

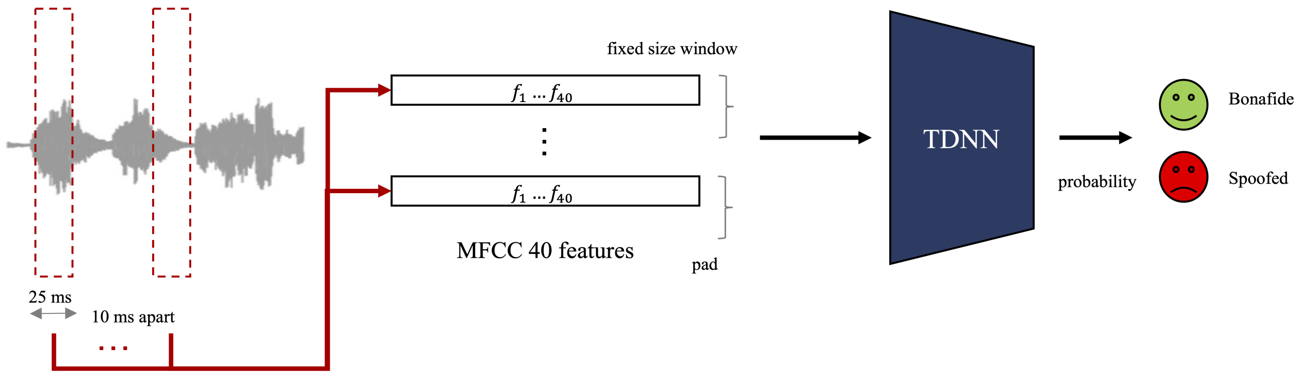

Fig. 1. An overview of out proposed architecture.

TABLE V
THE ARCHITECTURE OF THE TDNN

Layer	Channel	Kernel Size	Dilation	Stride
Conv1D	(40,32)	3	1	1
BatchNorm1D	-			
ReLU	-			
Conv1D	(32,64)	3	2	1
BatchNorm1D	-			
ReLU	-			
Conv1D	(64,128)	3	3	1
BatchNorm1D	-			
ReLU	-			
Conv1D	(128,256)	3	4	1
BatchNorm1D	-			
ReLU	-			
FC	$(c_in, 512)$	-		
ReLU	-			
FC	(512,1)	-		

[1]MFCC feature dimension. [2]Depends on the window size.

TABLE VI
PERFORMANCE COMPARISON OF SOTA, EXISTING STUDIES AND OUR STUDY

Method	ERR(%)	t-DCF
CQCC+GMM (ASVspoof baseline[1]) [7]	2.71	0.0663
LFCC+GMM (ASVspoof baseline[1]) [7]	0.43	0.0123
MFCC+CQCC+Spec+ResNet [8]	6.02	0.1569
ELTP-MFCC+DBiLSTM [6]	0.74	0.008
ELTP-MFCC+DBiLSTM [6]	0.74	0.008
MFCC+TDNN (Window size 100)	22.09	0.5951
MFCC+TDNN (Window size 200)	16.72	0.4399
MFCC+TDNN (Window size 400)	14.09	0.3198
MFCC+TDNN (Window size 600)	13.48	0.2723
MFCC+F-TDNN [12] (Window size 100)	18.76	0.5123
MFCC+F-TDNN [12] (Window size 600)	12.08	0.2732

that epoch. By monitoring the model's performance on the development set, we were able to avoid overfitting and select the best-performing model for evaluation.

2) Test: The testing process is similar to the training step, with one key difference. We introduce a *semi-voting mechanism* in which we group the truncated MFCC features belonging to the same audio into one testing trial. The probabilities for each truncated MFCC feature are averaged to produce a final probability.

Then, we calculate the Log-likelihood ratios (LLRs). LLRs are a powerful metric in speech processing and audio analysis, providing a means to compare the likelihood of input data under different hypotheses. LLRs offer the ability to capture variability in the input data and their robustness to changes in the acoustic environment. The LLRs are used as the score for t-DCF and ERR.

$$LLR = \log(\frac{P_{bonafide}}{1 - P_{bonafide}}) \tag{7}$$

The experimental results in Fig 3. and Fig 4. demonstrate that the TDNN architecture is capable of converging without exhibiting signs of overfitting. However, it is important to note that the TDNN still falls short when compared to other state-of-the-art models as presented in Table VI. One possible explanation for this discrepancy is that the TDNN lacks a ”memory component” and therefore struggles to handle long-term dependencies, which may be crucial in detecting deepfake audio. Moreover, the simple architecture of the TDNN may limit its ability to learn and represent the input MFCC feature effectively, as opposed to more complex and deeper models. Lastly, due to the limited number of epochs (10), there might have been a missed opportunity for the model to learn better parameters. By implementing an improved training strategy and increasing the number of epochs, the TDNN could potentially achieve better results.

Regarding the window size, we see that larger window sizes can lead to better performance, indicating the importance of capturing long-term dependencies in deepfake audio detection. However, it is important to consider the practical implications of using a longer window size for deployment in real-life scenarios. Depending on the specific task, a balance must be struck between the window size and the performance such that the input data would not be mostly padded. In this regard, selecting an appropriate window length that provides a good trade-off between capturing relevant information and minimizing unnecessary padding of the input data would be

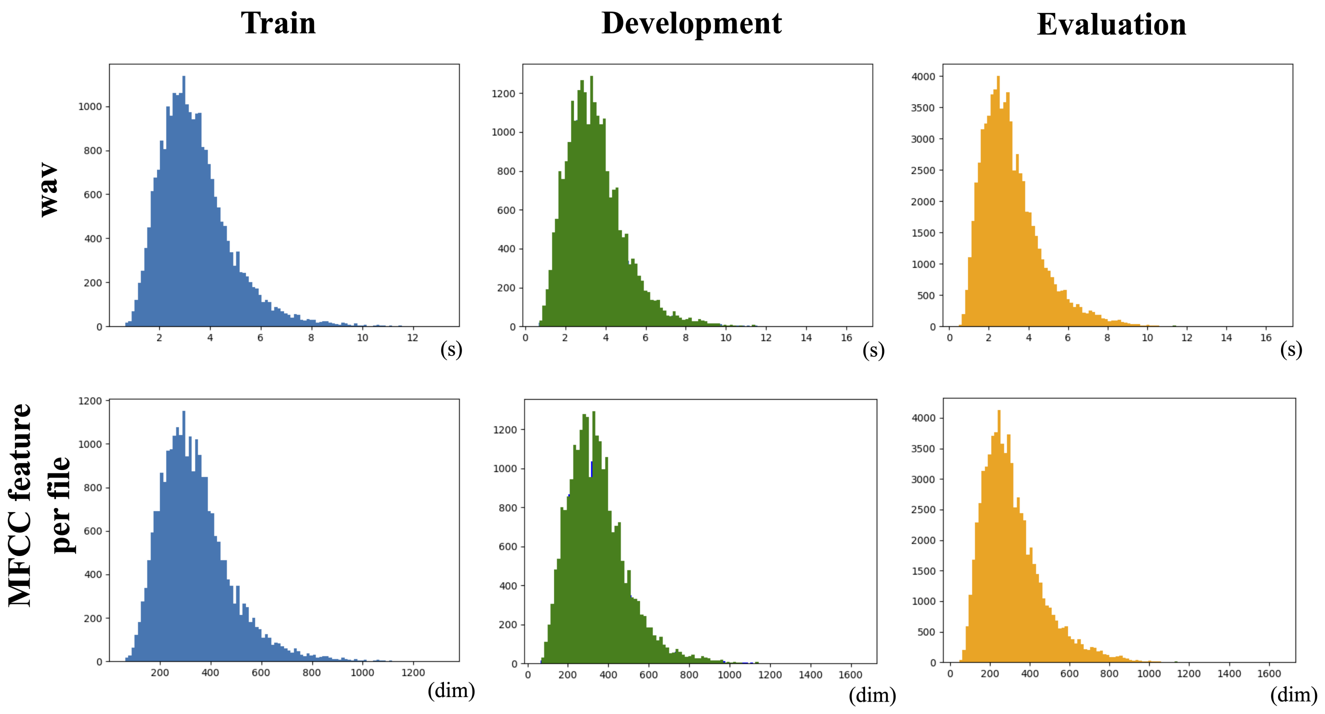

Fig. 2. The diftribution of numbers of duration of wav files and MFCC features per files.

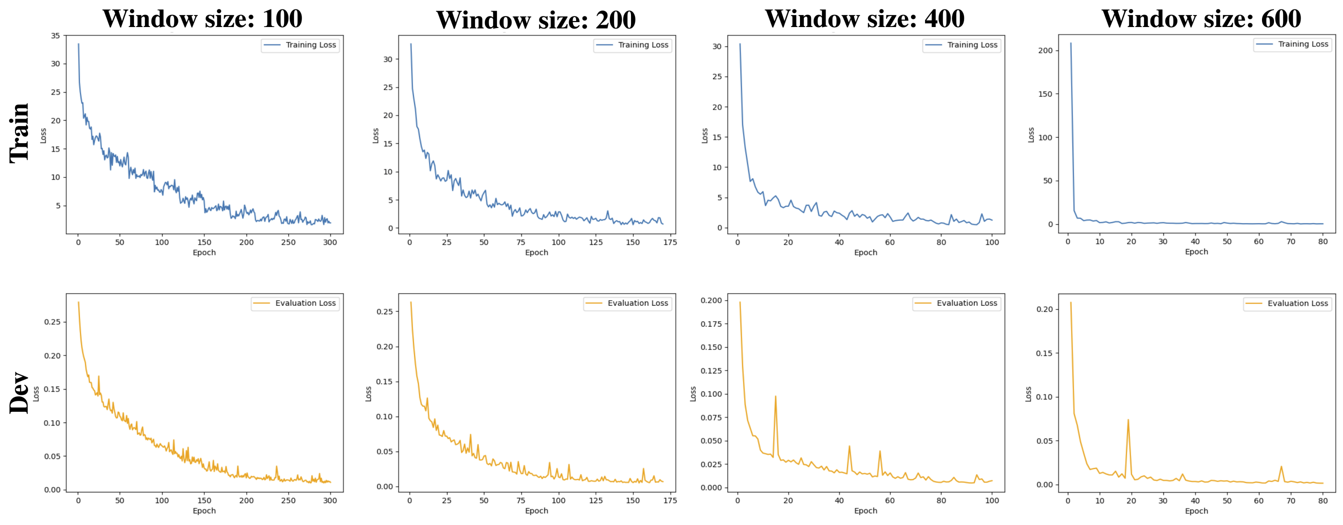

Fig. 3. The loss of TDNN with different window size.

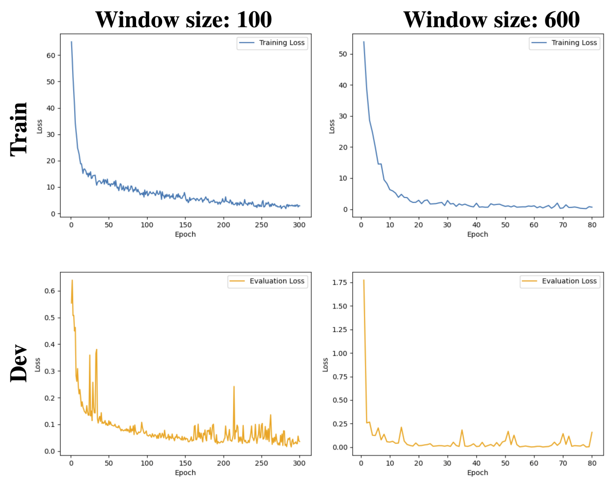

Fig. 4. The loss of F-TDNN with different window size.

beneficial.

V. Conclusion

TDNNs have shown promising results in deepfake audio detection and are able to achieve comparable performance to other neural network architectures such as CNNs and LSTMs. However, TDNNs have also been shown to be outperformed by more complex architectures such as DBiLSTM, ResNet.

Real-time deepfake audio detection is crucial for applications such as voice biometrics, fraud detection, and identity verification. By detecting and preventing fraudulent use of deepfake audio in real-time, these technologies can help to protect individuals and organizations from malicious actors seeking to deceive or impersonate others.

In general, TDNNs are faster and more computationally efficient than LSTMs and DBiLSTMs, which is an important consideration when deepfake audio detection needs to be performed in real-time or near real-time. However, for tasks that require more complex modeling of temporal dependencies, LSTMs and DBiLSTMs may be more appropriate.

Recently, deepfake audio has become a more grown concern on social media platforms, and one particularly troubling trend is the emergence of deepfake audio-generated song covers on TikTok. These audio forgeries use advanced machine learning algorithms to create convincing covers of popular songs, often featuring famous artists who never actually performed the song. The deepfake audio is so convincing that it can easily deceive listeners and even other artists, who may inadvertently promote the deepfake audio cover as if it were genuine. This highlights the need for more effective deepfake audio detection methods to combat the spread of this type of fraudulent content.

UMG said in a statement to MBW in the wake of today's news: ''The training of generative AI using our artists' music (which represents both a breach of our agreements and a violation of copyright law) as well as the availability of infringing content created with generative AI on DSPs, begs the question as to which side of history all stakeholders in the music ecosystem want to be on: the side of artists, fans and human creative expression, or on the side of deep fakes, fraud and denying artists their due compensation" [11]. The UMG statement highlights the need for action to prevent the misuse of generative AI and deepfake audio in the music industry. It's important for stakeholders to establish guidelines and regulations to protect the rights of artists while promoting innovation and creativity.

In future work, researchers can explore ways to improve the ability of TDNN to capture more time features of the audio data. The TDNN architecture could be modified to incorporate more complex time features, such as long-term temporal dependencies and context information, to improve its accuracy in detecting deepfake audio.

References

[1] Z. Almutairi and H. Elgibreen, "A Review of Modern Audio Deepfake Detection Methods: Challenges and Future Directions," Algorithms, vol. 15, no. 5, p. 155, May 2022, doi: 10.3390/a15050155.

[2] Ren, Y.; Hu, C.; Tan, X.; Qin, T.; Zhao, S.; Zhao, Z.; Liu, T.-Y. Fastspeech 2: Fast and High-Quality End-to-End Text to Speech. arXiv 2020, arXiv:2006.04558.

[3] Shen, J.; Pang, R.; Weiss, R.J.; Schuster, M.; Jaitly, N.; Yang, Z.; Chen, Z.; Zhang, Y.; Wang, Y.; Skerrv-Ryan, R. Natural Tts Synthesis by Conditioning Wavenet on Mel Spectrogram Predictions; IEEE: Piscataway, NJ, USA, 2018; pp. 4779–4783.

[4] Ping, W.; Peng, K.; Gibiansky, A.; Arik, S.O.; Kannan, A.; Narang, S.; Raiman, J.; Miller, J. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. arXiv 2017, arXiv:1710.07654.

[5] X. Wang, J. Yamagishi, M. Todisco, H. Delgado, A. Nautsch, N. Evans, M. Sahidullah, V. Vestman, T. Kin- nunen, K. Lee, et al., "The ASVspoof 2019 database," arXiv preprint arXiv:1911.01601, 2019.

[6] Arif, T.; Javed, A.; Alhameed, M.; Jeribi, F.; Tahir, A. Voice spoofing countermeasure for logical access attacks detection. IEEE Access 2021, 9, 162857–162868.

[7] Yamagishi, J., Todisco, M., Sahidullah, M., Delgado, H., Wang, X., Evans, N., ... & Nautsch, A. (2019). Asvspoof 2019: Automatic speaker verification spoofing and countermeasures challenge evaluation plan. ASV Spoof.

[8] Alzantot, M.; Wang, Z.; Srivastava, M.B. Deep residual neural networks for audio spoofing detection. arXiv CoRR 2019, arXiv:1907.00501.

[9] Aravind, P.R.; Nechiyil, U.; Paramparambath, N. Audio spoofing verification using deep convolutional neural networks by transfer learning. arXiv 2020, arXiv:2008.03464.

[10] T. Kinnunen, K. Lee, H. Delgado, N. Evans, M. Todisco, M. Sahidullah, J. Yamagishi, and D. A. Reynolds, "t-DCF: a detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification," in Proc. Odyssey, Les Sables d'Olonne, France, June 2018.

[11] T. Ingham, ''Universal Music Group responds to 'fake drake' ai track: Streaming platforms have 'a fundamental responsibility to prevent the use of their services in ways that harm artists'," Music Business Worldwide, 18-Apr-2023. [Online]. Available: https://www.musicbusinessworldwide.com/universal-music-group-responds-to-fake-drake-ai-track-streaming-platforms-have-a-fundamental-responsibility/. [Accessed: 07-May-2023].

[12] Povey, D., Cheng, G., Wang, Y., Li, K., Xu, H., Yarmohammadi, M., Khudanpur, S. (2018) Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks. Proc. Interspeech 2018, 3743-3747, doi: 10.21437/Interspeech.2018-1417