Audio Deepfake Detection using Spectrograms

[1]Surbhit Pratik, [2]Hriday Eluru

Student
Department of Computational Intelligence
SRM Institute of Science & Technology
Kattankulathur, India

***Abstract*- High-quality fake videos and audio generated by AI algorithms (the deep fakes) have started challenging the status of videos and audio as definitive evidence of events. These sounds and clippings, if used as evidence, could even become a threat to someone's private life or even national security. In this project, we have developed a system that can accurately determine whether a given piece of evidence is real or artificially mutated using a Keras model with ConformerEncoder architecture as its backbone. The ASVspoof 2019 dataset has been used to perceive the accuracy of the model built.**

***Index Terms*- audio, deepfake, Keras, TensorFlow, Conformer, ASVspoof, spectrogram**

I. INTRODUCTION

The problem of recognizing fake or computer-generated audio requires identifying where it differs from real audio. Audio files contain subtle differences which are imperceptible to the human ear. Also, advancements in the field of speech synthesis make it impractical to try picking up on differences between these audio classes without first transforming the audio file. In this research, I used a dataset of pre-recorded audio files containing both fake audio and real audio. Transformed the audio into spectrograms to gather information on the frequency spectrum of an audio file as it varies with time. Spectrograms are visual representations of audio files which capture intensity and frequency of speech. These are used as input to a Conformer (Convolutional Augmented Transformer) to train a model on classifying the data.

The term "audio deepfakes" refers to manipulated audio files that are generated using advanced techniques for machine learning. Deep fakes like these can be utilized to fabricate information, disseminate propaganda, or even commit crimes. As a consequence of this, the development of methods that can detect and identify these deepfakes has become increasingly important. The use of spectrograms, which are graphical representations of sound waves that demonstrate how the frequency of a sound changes over the course of time, is one method that has a lot of potential. It is possible to identify patterns and characteristics that are exclusive to either real or fake audio files by performing an analysis of the spectrogram.

For the purpose of this project, we will make use of spectrograms in conjunction with a deep learning model that will be based on the Keras framework and will have a ConformerEncoder architecture. The ConformerEncoder architecture is a model that has been proven to be effective in speech recognition tasks. It is considered to be a state-of-the-art model. This project aims to develop a model that is capable of accurately detecting audio deepfakes, even when those deepfakes are extremely realistic and difficult to differentiate from genuine audio files. We hope that by incorporating spectrograms and the ConformerEncoder architecture into our model, we will be able to achieve high accuracy and robustness, which will make it an effective instrument for identifying audio deepfakes and halting their proliferation.

II. MOTIVATION

Computer Generated voices which are identical to a targeted person's voice, this feature has been widely exploited and used in unethical ways to manipulate people for gaining private information with advancement in technology nowadays it has become really difficult to differentiate between real and fake audios and currently in the market there are various ways to generate fake audios like voice conversion system text-to-speech system and replay attacks therefore a system which can accurately differentiate between real and fake audios is globally demanded.

III. DATASET

Numerous research groups around the globe are attempting to identify media manipulations, such as audio deepfakes as well as image and video deepfakes. These initiatives are typically supported by public or private funding and maintain close ties with academic institutions. Numerous competitions have been arranged over the past few years to further the field of audio deepfake research.

ASVspoof, also known as the Automatic Speaker Verification Spoofing and Countermeasures Challenge, is the most well-known challenge in the world. This challenge is a biannual community-led initiative designed to encourage the contemplation of spoofing and the creation of countermeasures.

The ASVspoof 2019 dataset contains speech signals collected from several different databases. The dataset contains both genuine and spoofed speech signals, with a total of over 31000 utterances. We divided the dataset into training, validation, and testing sets.

We performed several preprocessing steps on the dataset, including normalization and feature extraction. We extracted Mel Frequency Cepstral Coefficients (MFCCs) and used them as features for our model. These extracted features have been translated into a spectrogram which is nothing but an image of a graph representing the frequency of sound over a certain time interval. Some of these spectrograms have been depicted below in Fig 1 and Fig 2.

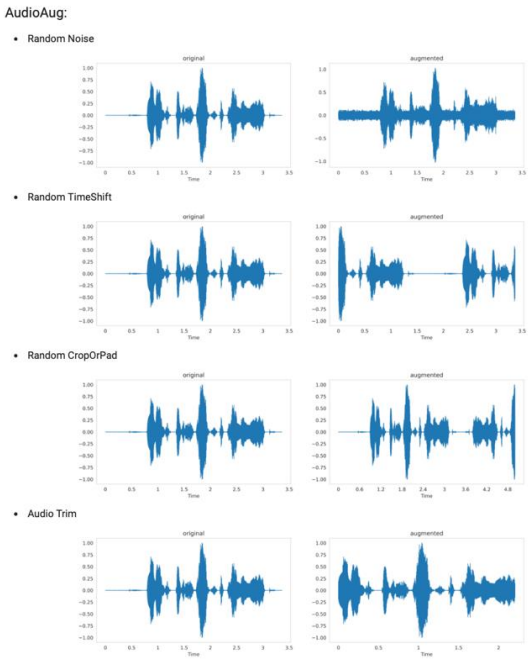

Fig 1. Augmented audio

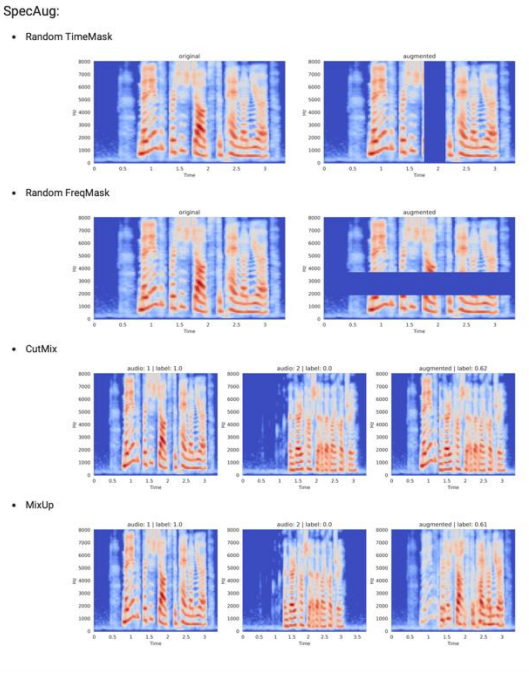

Fig 2. Augmented spectrograms

IV. TYPES OF AUDIO DEEPFAKES

A. *Replay - based*

Malicious works known as replay-based deepfakes have the intention of reproducing an audio recording of the voice of the interlocutor. These can be broken down into two categories: cut-and-paste detection and far-field detection. A test segment consisting of a microphone recording of the victim is played as part of the far-field detection procedure on a hands-free phone. On the other hand, using cut-and-paste necessitates fabricating the requested sentence using a mechanism that is dependent on text. Verification of the speaker is reliant on the text can be used as a defense mechanism against replay-based attacks. Deep convolutional neural networks are currently being utilized as a method for the detection of end-to-end replay attacks in the industry.

B. *Synthetic – based*

Speech synthesis uses system programs to simulate human speech. Text-To-Speech uses linguistic rules to convert text into natural-sounding speech in real time. Text analysis, acoustic, and vocoder modules make up a classical system. Generation requires two steps. Clean, well-structured raw audio with the transcripted voice audio sentence is needed. Second, this data must train the Text-To-Speech model to generate synthetic sounds.

The generating model receives the target speaker's voice-transcribed text. Text analysis turns input into linguistic features. Based on the text analysis module's linguistic properties, the acoustic module extracts target speaker parameters from audio data. Finally, the vocoder creates vocal waveforms using acoustic feature parameters. The finished audio file includes synthetic simulation audio in waveform format, providing speech audio in the voice of many speakers, including those not in training.
WaveNet, a neural network that generates raw audio waves that mimic numerous speakers, was the first breakthrough in this field. Over time, other systems have surpassed this network in synthesizing lifelike artificial voices at affordable prices. Text-To-Speech relies on the quality of the voice corpus, which is expensive to create. Speech synthesis systems cannot detect periods or special letters, another drawback. Since two words in the same manner might signify different things, ambiguity persists.

C. *Imitation – based*

Imitation-based audio deepfake converts original speech from one speaker to another. An imitation-based algorithm changes style, intonation, and prosody to replicate the target voice without affecting the language information. This technique is also called voice conversion.

There is no apparent generating process distinction between this method and the Synthetic-based method. Both methods change speech audio signal acoustic-spectral and style features, although the imitation-based method usually preserves input and output text. Changes the sentence's pronunciation to match the voice of the target speaker.

There are many ways of imitating someone's voice, for example, humans with similar voices can imitate voices. Due to their versatility and high-quality outcomes, Generative Adversarial Networks (GAN) have become the most preferred method. An imitation generation approach develops a new speech, exhibited in the fake one, from the original audio signal.

V. LITERATURE SURVEY

In this paper [1], the authors implemented and analysed a CNN-based replaying detection system (M2) adapted from the state-of-the-art LCNNFFT using the updated version 2.0 ASVspoof database. System M2 shows comparable performance to LCNNFFT. They used the SLIME algorithm to generate class explanations from spectral and temporal perspectives. Their analysis is that M2 uses the first few milliseconds of the audio signal to make class prediction. They further demonstrated the significance of analysis and findings by pre-processing the test signals which led to a predictable change in the EER on both the development and evaluation subsets. In the paper [2], This is the first paper to use a convolutional neural network to detect audio splicing. The model can extract high-level features from the spectrogram of audio, which acts as an image to the input of the convolutional neural network, and classify on its own, so the proposed method can be effectively used for forensic authentication, which directly detects tampering. We can demonstrate audio splicing detection with increased accuracy and resistance to noise attack and compression. Though training requires a large database and more computing power, this method is useful for audio forensics. In this paper [3], the authors presented a novel database containing 620 deepfake videos for 16 pairs of subjects from VidTIMIT database. They generated two versions of the videos for each subject: one of low and other of high quality. They also demonstrated that the current VGG and Facenet based face recognition algorithms are vulnerable to the DeepFake videos and fail to distinguish such videos from the original ones with up to 95% equal error rate. They also evaluated several baseline face swap detection algorithms and found that lip-sync based approach fails to detect mismatches between lip movement and speech.

In this paper [4], The authors propose a robust end-to-end deep learning framework for voice spoofing detection in this paper, which can detect spoofed audio generated by a wide range of unknown TTS and VC systems with high accuracy. They have successfully demonstrated that adding a FreqAugment layer and a large-margin cosine loss, as well as data augmentation, can improve generalisation ability. On the ASVspoof 2019 evaluation set, the experimental results show an EER of 1.26%, a significant improvement over the state-of-the-art. In paper [5], The authors predict that several future technological developments will further improve the visual quality and generation efficiency of the fake videos. Firstly, one critical disadvantage of the current Deepfake generation methods is that they cannot produce good details such as skin and facial hairs. This is due to the loss of information in the encoding step of generation. However, this can be improved by incorporating GAN models which have demonstrated performance in recovering facial details in recent works. Secondly, the synthesized videos can be more realistic if they are accompanied with realistic voices, which combines video and audio synthesis together in one tool. In this paper [6], the author describes a summary of indicators that can be used to decide, whether a video or photo was changed with usage of face swapping AI-based algorithms. Their choice of building model is face warping artifacts detection that is one of the best indicators of fake video/photo right now. They predict that in the coming future it will become more difficult to differentiate deepfakes with the original content. In this study [7], the authors investigate the deepfake detection challenge as a fine-grained classification issue, which is a fresh approach to the field. They suggest a paradigm for multiple attentional deepfake detection. In order to capture more subtle artefacts, the proposed framework improves texture information from shallow layers and explores discriminative local regions by means of numerous attention maps. The attention maps are then used to aggregate the low-level textural features and high-level semantic features.

In this paper [8], they have presented a temporal-aware system to automatically detect deepfake videos. Their experimental results using a large collection of manipulated videos have shown that using a simple convolutional LSTM structure we can accurately predict if a video has been subject to manipulation or not with as few as 2 seconds of video data. Their research, in our opinion, provides a strong first line of protection against fake media produced with the methods discussed in the study. They demonstrate how our system can complete this work with competitive results while utilising a straightforward pipeline architecture. In this study [9], they jointly model the audio and visual modalities to develop a novel challenge for detecting deep fakes. This duty is crucial since, in reality, we are unable to tell beforehand whether the audio or the video has been altered. They demonstrate how full sequence prediction and video and audio-based deepfake detection both performed better when using learnt intrinsic synchronization between video and audio. The model can be generalized to previously unknown deepfake categories thanks to the learned synchronization patterns. Last but not least, they offer a method for producing high-quality audio-visual deepfake data that will be helpful for future study in this area.
In this study [10], They suggested pair-wise self-consistency learning (PCL) to locate the manipulated regions and detect face forgeries produced by stitching-based techniques based on a less-noticed cue: the inconsistent source characteristics inside the altered images. Only a few parameters make up PCL, which can be used as a plugin module on standard backbone networks. In order to effectively facilitate PCL training, they also created a brand-new, lightweight image synthesis technique dubbed the inconsistency image generator (I2G), which dynamically creates falsified images together with annotations of their altered regions.

VI. METHODOLOGY

We began by installing the necessary Tensorflow libraries and connecting our datasets to the Python notebook. After installing the libraries and downloading the datasets, we installed the TensorFlow utilities, our audio classification models, and the weights and biases. The third and final thing we needed to install were the libraries required for dealing with various aspects of Python, such as NumPy, pandas, and some visualization libraries such as matplotlib and seaborn, as well as a few computing libraries such as SciPy. Now we had to do the configuration, which included initializing a few model components, selecting our device as none

because in our case, it would be selected automatically, and setting our audio and spectrogram parameters. We had also chosen the batch size and epochs, as well as the loss, optimizer, and learning rate scheduler in this configuration. We chose loss as binary cross entropy, adam as the optimizer, and cosine as the learning rate scheduler. We used a random seed value when creating the training and test data sets. The goal was to compare the performance of different models using different hyperparameters or machine learning algorithms using the same training and validation data sets. As soon as the seeding was finished, we wrote the code for our device's automatic configuration, whether to use TPU, GPU, or CPU. We utilized the try except block in Python for this purpose, which made the configuration task very simple.

We preprocessed the data we collected after we had our libraries and configurations. Because a large dataset contains many anomalies, we had to remove the incomplete data rows. We divided the data into three parts after cleaning it: training data (TRAIN_FILENAMES), testing data (TEST_FILENAMES), and validation set (VALID_FILENAMES). Our model was trained on train data, and the checkpoint was saved using valid data. Finally, we evaluated the performance of our model on test data. Next, we cleaned up our audio files by using audio augmentation (AudioAug) to remove any silent parts at the beginning and end of the audio. We also cropped if the audio was too long and padded if it was too short, to make each audio length equal. We also included Gaussian Noise and Random Shift augmentations in the dataset to help create a strong classifier. Second, we used spectrogram augmentation (SpecAug) to improve the classifier by introducing random time masking and frequency masking in a 50:50 ratio. We also divided this set into two batches, using Spectrogram-MixUp augmentation on one and Spectrogram-CutMix augmentation on the other. We created a function called Audio2Spec after we created these functions for SpecAug and AudioAug to convert these audio files into their respective spectrograms. To decode our audio dataset and its labels, which were in.wav format, we created a TensorFlow function that decoded and normalized the files. We read, parsed, and augmented the TFRecord of the used data to improve the sequential read of our dataset. All the preceding tasks were completed in order to achieve the goal of converting spectrograms from the raw dataset. We checked some samples from a batch to ensure our pipeline was correctly generating the correct spectrogram and its associated label, as shown in Fig 3 and Fig 4.

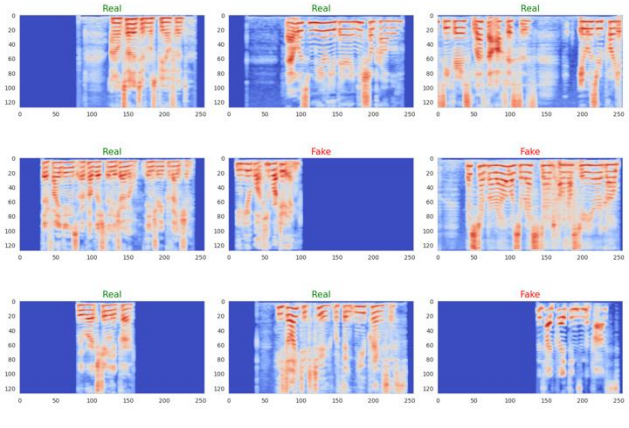

Fig 3. Spectrograms without augmentation

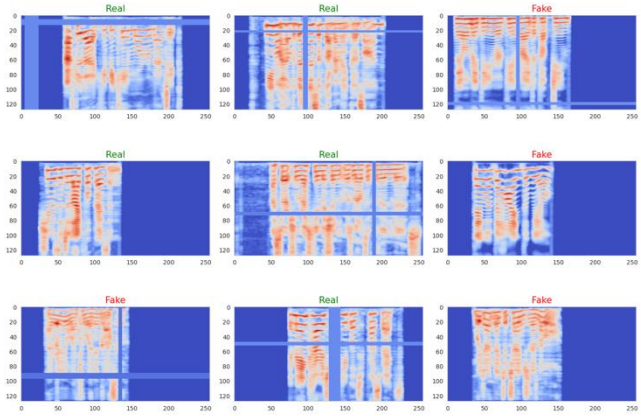

Fig 4. Spectrograms with augmentation

We defined our learning rate scheduler in the following section, which governs how quickly an algorithm updates or learns the values of a parameter estimate. In other words, the learning rate controls the weights of our neural network in relation to the loss gradient. We increased the learning rate from 0 to the maximum learning rate, 0.0005, in the first four epochs and then gradually decreased it back to 0 in the next 25 epochs. This learning rate was visualized, as shown in Fig 5. We created a Keras model that employs the Conformer encoder architecture for our audio classification model. The audio_classification_models module's ConformerEncoder serves as the model's foundation. The output of the backbone is routed through a GlobalAveragePooling1D layer, then a Dense layer with num_classes units and activation via final_activation. The activation function had been set to adam during the configuration phase. After we finished building the model, we were ready to train it on our preprocessed dataset, which consisted of spectrograms of audio clips. All the images had a dimension of 256 x 128, a batch size of 32 and the Conformer model ran for 12 epochs.

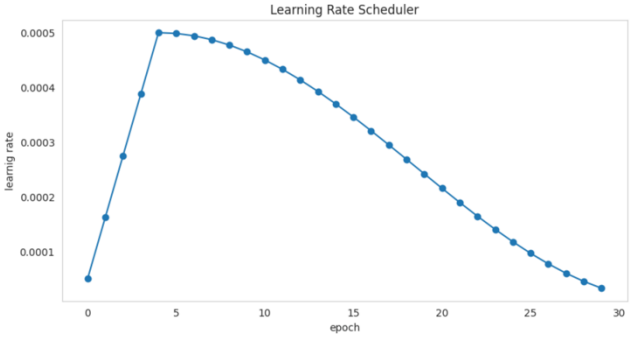

Fig 5. Visualization of the used Learning Rate Scheduler

VII. RESULTS

For this study, we used the ASVspoof 2019 dataset, which contains over 31000 utterances of both real and spoofed voice. To train on this dataset, we used a Keras model built with TensorFlow and based on the ConformerEncoder architecture. The audios were converted into spectrograms before being used as input data. Spectrograms are basically graphs of the frequency of the audio versus time. Our model was successfully trained on a 5000-spectrogram training dataset and a 4000-spectrogram validation dataset. When we tested the remaining dataset, which was the testing data, we discovered that the loss was approximately 2.08, the binary accuracy was 0.74, and the recall was approximately 0.48. The model's accuracy, or f1 score, was found to be 0.6512, or 65.12%. For the above stated results, we have visualized a graph of f1 score of the model per epoch for training and validation data, as seen in Fig 6, and also depicted a confusion matrix as seen in Fig 7.

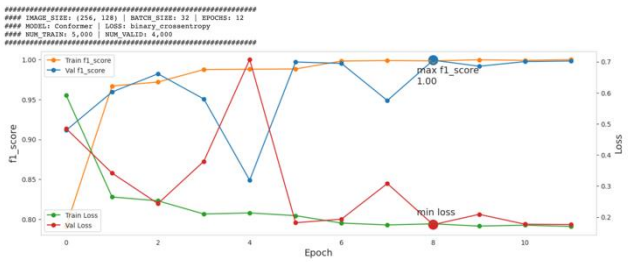

Fig 6. Graph of f1 score and loss vs epoch.

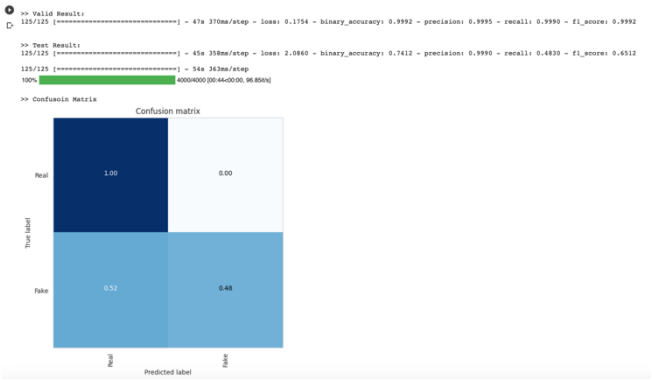

Fig 7. Confusion Matrix

REFERENCES:

[1] D. A. Reynolds, “Speaker identification and verification using gaussian mixture speaker models,” Speech Communication, vol. 17, no. 1, pp. 91–108, 1995.

[2] K. A. Lee, B. Ma, and H. Li, “Speaker verification makes its debut in smartphone,” IEEE Signal Processing Society SLTC Newsletter, 2013.

[3] Z. Wu, S. Gao, E. S. Cling, and H. Li, “A study on replay attack and anti-spoofing for text-dependent speaker verification,” in Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific, Dec 2014, pp. 1– 5.

[4] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, “Spoofing and countermeasures for speaker verification: A survey,” Speech Communication, vol. 66, pp. 130 – 153, 2015.

[5] Z. Wu, J. Yamagishi, T. Kinnunen, C. Hanili, M. Sahidullah, A. Sizov, N. Evans, M. Todisco, and H. Delgado, “ASVspoof: The Automatic Speaker Verification Spoofing and Counter- measures Challenge,” IEEE Journal of Selected Topics in Sig- nal Processing, vol. 11, no. 4, pp. 588–604, June 2017.

[6] Y. W. Lau, M. Wagner, and D. Tran, “Vulnerability of speaker verification to voice mimicking,” in Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004., Oct 2004, pp. 145–148.

[7] J. Gaka, M. Grzywacz, and R.Samborski, “Playback attack detection for text-dependent speaker verification over telephone channels,” Speech Communication, vol. 67, pp. 143 – 153, 2015.

[8] J. Antonio, V. López, A. Miguel, A. Ortega, and E. Lleida, “Spoofing detection with DNN and one-class SVM for the ASVspoof 2015 challenge,” in Interspeech, 2015.

[9] C. Zhang, C. Yu, and J. H. L. Hansen, “An investigation of deep-learning frameworks for speaker verification antispoofing,” IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 4, pp. 684–694, June 2017.

[10] P. Nagarsheth, E. Khoury, K. Patil, and M. Garland, “Replay Attack Detection using DNN for Channel Discrimination,” in Interspeech, 2017, pp. 97–101.