INTERNSHIP REPORT ON :- IMPROVED AUDIO DEEPFAKE DETECTION USING WHISPER FEATURES

Report by : Amarpreet Kaur Lotte (Summer Intern)

TABLE OF CONTENTS :

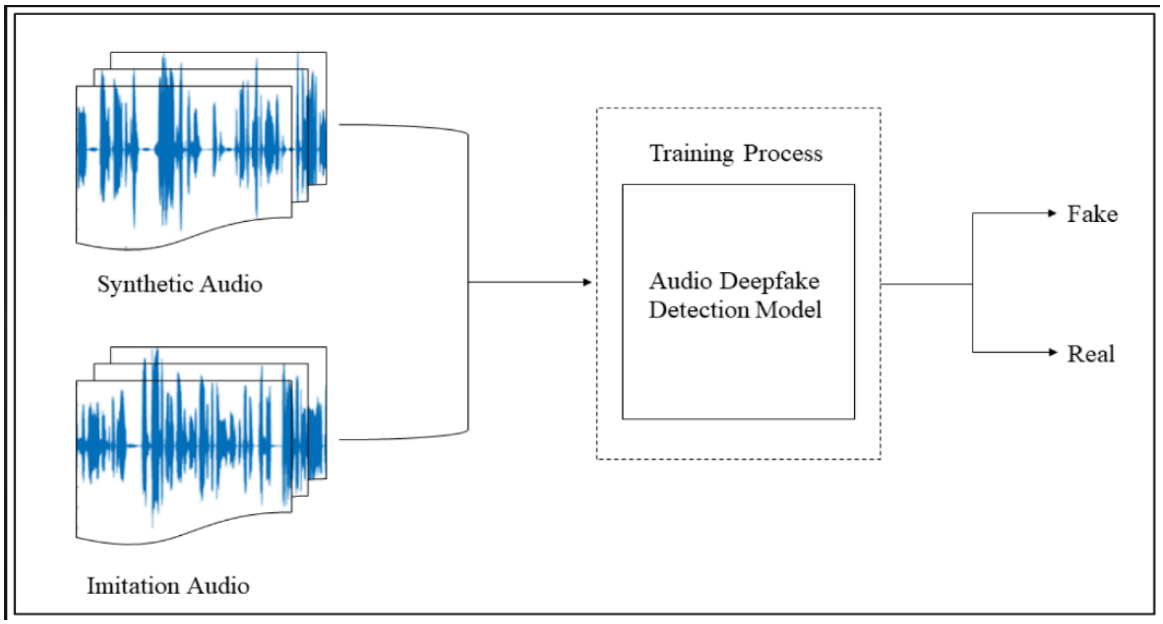
**shows that the contents pertain to information concerning the research paper "Improved DeepFake Detection Using Whisper Features." Piotr Kawa 1, Marcin Plata 1, Michał Czuba 1 , Piotr Szymanski' 1, and Piotr Syga 1, Wrocław University of Science and Technology, Wrocław, Poland.*

1.) ABSTRACT :

With the recent inflow of speech synthesis tools, the threat posed by audio DeepFake (DF) is expanding. Several distinct detecting methods have been proposed as a countermeasure. Many methods rely on so-called front-ends, which modify raw audio and highlight aspects important for determining the authenticity of the audio sample. This contribution analyzes the impact of the cutting-edge Whisper automatic speech recognition model as a DF detection front-end. The comparison of various combinations of Whisper and well-established front-ends is done by training three detection models (LCNN, SpecRNet, and MesoNet) on a widely used ASVspoof 2021 DF dataset and then evaluating them on the DF In-TheWild dataset.We demonstrate that utilizing Whisper-based features improves detection for each model and beats previous results on the In-The-Wild dataset, lowering Equal Error Rate by 21%.

2.) INTRODUCTION :

Audio DeepFakes (DF) is a collection of deep learning algorithms for producing artificial speech. These methods may include creating entirely new sentences using Text-To-Speech or Voice-Cloning (with the goal of mimicking a specific person's speech patterns or sounding natural to a human listener) or transferring the characteristics of the victim's voice to the attacker's speech, also known as Voice-Conversion. With the advancement of deep learning techniques, it has become very simple to construct audio DeepFakes that are difficult to discern from authentic (genuine or unforged) recordings. Such malevolent behaviors can inflict serious harm, such as undermining the security of speaker recognition-protected systems, propagating bogus news, or defaming an individual's reputation.



As a result, creating effective DF detection techniques has become increasingly important in maintaining the integrity and trustworthiness of audio-based communication systems. It is important to remember that spoof and deepfake audio are not the same thing; while they may look to be, they are not.

BASIC DIFFERENCE BETWEEN DEEPFAKE AND SPOOF :

AUDIO DEEPFAKE	SPOOFING
In simple terms, it's a fake audio recording that sounds just like someone else, even though they didn't actually say those words.	Audio spoofing refers to the act of manipulating or forging audio recordings to deceive or trick listeners.

Many existing DeepFake detection algorithms rely on extracted features rather than raw waveforms, with the goal of identifying and emphasizing the salient qualities of an audio signal.

Mel-frequency cepstral coefficients (MFCC) and linear-frequency cepstral coefficients (LFCC) are two of the most popular techniques. In this study, we leverage the pre-trained Whisper encoder's feature extraction capabilities to explore its performance in DF detection rather than capturing speech features for eventual use in ASR. We employ it with three detection algorithms to see if neural-network characteristics provided can aid in DF identification. We chose Whisper for the study because of its effectiveness in voice recognition, which stems from the enormous and diverse speech corpus on which it was trained. As a result, we conclude that Whisper's characteristics will disregard the majority of naturally occurring artefacts while assisting in the detection of artificially changed voice samples. It would especially help with the problem of generalization, which refers to the models' poor efficacy on data outside of the training set's distribution.

3.) DETAILS ON MODELS , DATASETS AND FRONTENDS :

a.) FRONTENDS :

MFCC (Mel Frequency Cepstral Coefficients) :

Steps to Extract MFCCs:

- **Pre-Emphasis:** Enhance high frequencies by applying a filter to the audio signal.
- **Framing:** Divide the audio signal into small overlapping frames to analyze short segments of the signal.
- **Windowing:** Apply a window function (like Hamming window) to each frame to reduce signal discontinuities at the frame edges.

- **FFT (Fast Fourier Transform):** Convert each frame from the time domain to the frequency domain.
- **Mel Filter Bank:** Apply a set of filters spaced according to the Mel scale to mimic human ear perception, emphasizing frequencies that humans are more sensitive to.
- **Logarithm:** Take the logarithm of the filterbank energies to convert the scale from linear to logarithmic.
- **DCT (Discrete Cosine Transform):** Apply DCT to the log filter bank energies to obtain the MFCCs, which represent the short-term power spectrum of a sound.

***** MFCCs are based on the human auditory system's non-linear frequency response

LFCC (Linear Frequency Cepstral Coefficients) :

Steps to Extract LFCCs:

- **Pre-Emphasis:** Enhance high frequencies by applying a filter to the audio signal.
- **Framing:** Divide the audio signal into small overlapping frames for analysis.
- **Windowing:** Apply a window function (like Hamming window) to each frame to minimize signal discontinuities at the frame edges.
- **FFT (Fast Fourier Transform):** Convert each frame from the time domain to the frequency domain.
- **Linear Filter Bank:** Apply a set of filters that are spaced linearly across the frequency range, unlike the Mel scale used in MFCC.
- **Logarithm:** Take the logarithm of the filterbank energies to switch from a linear to a logarithmic scale.
- **DCT (Discrete Cosine Transform):** Apply DCT to the log filter bank energies to obtain the LFCCs, which represent the short-term power spectrum of a sound.

* LFCCs are designed to address some of the limitations of MFCCs, such as their insensitivity to low-frequency information and lack of robustness to noise.

WHISPER :

Whisper is an advanced automatic speech recognition (ASR) system. It was trained with 680,000 hours of information. Because of its diversity and volume, the data has proven to be robust against a wide range of background interferences, accents, and languages. Its name refers to a family of models that varies, for example, in terms of layer breadth and size. Whisper uses an off-the-shelf encoder-decoder Transformer design. Its encoder consists of two convolutional layers, each with a GeLU activation function. The data is then updated by adding position embeddings. The encoder concludes with a sequence of pre-activation residual attention blocks, followed by the normalization layer.

DETECTION MODELS :

LCNN (Light Convolutional Neural Network) :

In the case of LCNN, the input features and hidden features of two bi-LSTM layers, as well as the input features of the final Linear layer, are increased from 160 to 768 in size.

THE ARCHITECTURES OF THE LIGHT CNN-4 MODEL

Type	Filter Size /Stride	Output Size	#Params
Conv1	$9 \times 9/1$	$120 \times 120 \times 96$	7.7K
MFM1	-	$120 \times 120 \times 48$	-
Pool1	$2 \times 2/2$	$60 \times 60 \times 48$	-
Conv2	$5 \times 5/1$	$56 \times 56 \times 192$	230.4K
MFM2	-	$56 \times 56 \times 96$	-
Pool2	$2 \times 2/2$	$28 \times 28 \times 96$	-
Conv3	$5 \times 5/1$	$24 \times 24 \times 256$	614K
MFM3	-	$24 \times 24 \times 128$	-
Pool3	$2 \times 2/2$	$12 \times 12 \times 128$	-
Conv4	$4 \times 4/1$	$9 \times 9 \times 384$	786K
MFM4	-	$9 \times 9 \times 192$	-
Pool4	$2 \times 2/2$	$5 \times 5 \times 192$	-
fc1	-	512	2,457K
MFM_fc1	-	256	-
Total	-	-	4,095K

SPECRNET (Spectrogram Network) :

SpecRNet used in our comparison differs from its original version — to facilitate processing of the higher-dimensions front-ends we include an adaptive 2D average pooling after the last SeLU layer .

Layer	Input	Output shape
LFCC	64600	$1 \times 80 \times N$
preliminary normalization	BN2D SELU	$1 \times 80 \times N$
ResBlock	Conv2D(3, 1, 20) BN2D LeakyReLU(0.3) Conv2D(3, 20, 20) ..... Conv2D(1, 1, 20)	$20 \times 80 \times N$
FMS Attention	Maxpool(2) FMS Maxpool(2)	$20 \times 20 \times \frac{N}{4}$
ResBlock	BN2D LeakyReLU(0.3) Conv2D(3, 20, 64) BN2D LeakyReLU(0.3) Conv2D(3, 64, 64) ..... Conv2D(1, 20, 64)	$64 \times 20 \times \frac{N}{4}$
FMS Attention	Maxpool(2) FMS Maxpool(2)	$64 \times 5 \times \frac{N}{16}$
ResBlock	BN2D LeakyReLU(0.3) Conv2D(3, 64, 64) BN2D LeakyReLU(0.3) Conv2D(3, 64, 64) ..... -	$64 \times 5 \times \frac{N}{16}$
FMS Attention	Maxpool(2) FMS Maxpool(2)	$64 \times 1 \times \frac{N}{64}$
pre-recurrent normalization	BN2D SELU	$64 \times 1 \times \frac{N}{64}$
GRU	GRU(64, bi)	128
GRU	GRU(128, bi)	128
FC	128	128
FC	128	1

Fig : The scheme of SpecRNet architecture. The convolution layers are defined according to the convention: Conv2D(kernel size, input channels, output channels), the layers under dotted lines in residual blocks refer to operations performed on identity tensors before adding.

MESONET :

In MesoNet, we introduce adaptive 1D average pooling just before the penultimate fully connected layer.

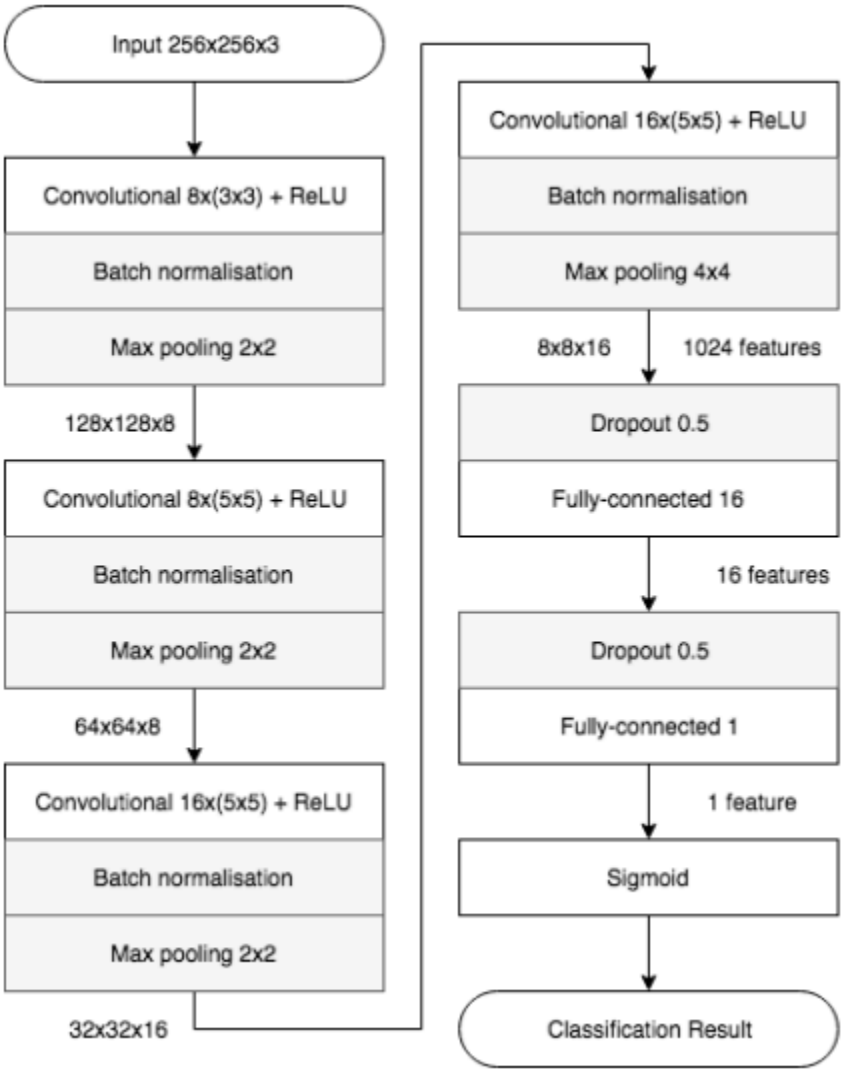


Fig: The network architecture of Meso-4. Layers and parameters are displayed in the boxes, output sizes next to the arrows.

How are frontends concatenated ?

For the spectrogram-based models, we consider three frontends: LFCC, MFCC, and the Whisper ASR encoder outputs. We also analyze the concatenated front-ends of cepstral coefficients using Whisper characteristics. The intuition behind it is as follows: concatenating diverse front-ends may produce superior outcomes. We utilize LFCC and MFCC with window and hop lengths of 400 and 160, respectively; each has 128 coefficients. We concatenate front-ends in the second dimension using delta and double delta features. This produces the shape data (128 * 3, 3000). In our trials, we employ the Whisper model's tiny.en variation. Its encoder has 7,632,384 parameters and outputs shape data (376, 1500). To match the size of the other front-ends , in the concatenated front-end setup, we replicate one of the dimensions, resulting in a tensor with size (376, 3000).

DATASETS :

The dataset utilized in the article includes all 31,779 DeepFakes In-The-Wild samples plus 125,000 randomly chosen samples from ASVspoof 2021 (DF). The lack of DF datasets in general, among which ASVspoof is one of the biggest and most well-liked, is the driving force behind the choice. The samples in the latter collection, on the other hand, are taken from the Internet and represent real-world situations.

4.) EXPERIMENTAL SETUP :

Every sample was put through a consistent preprocessing process. It covered reducing samples to 30 s of content, padding (by repetition), and resampling to 16 kHz mono-channel. Silences greater than 0.2 s were also removed. We chose a 30-second input time rather than the standard 4-second length for two reasons. It demonstrates that longer utterances provide more insightful analysis. Second, Whisper accepts 30 seconds of information as input, which is used to clip or pad samples with zeros. Rather than

adding zeros to it, we chose to fill the input tensor with voice. A random subset of 100,000 training and 25,000 validation samples from the ASVspoof 2021 DF dataset were used to train the models. A portion of this dataset was utilized forFor two reasons: first, we intended our method to be able to train in roughly 24 hours on a single GPU; second, we didn't think the quantity of data would yield a meaningful improvement for an architecture like Whisper tiny .Using oversampling, we addressed the disparity between real and phony classes. For every spectrogram-based model, we employed a weight decay of 10^−4 and a learning rate of 10^−4. A learning rate of 10^−3 and a weight decay of 5 · 10^−5 were employed by RawNet3. Using an 8-batch batch size, we trained the models for 10 epochs using a binary cross-entropy function. RawNet3 was trained using SGDR scheduling, which restarted after every epoch. For further running tests on the entire In-The-Wild dataset, the checkpoint with the highest validation accuracy was chosen. We use the Equal Error Rate (EER) metric as a fraction to show our results. EER is frequently applied to spoofing and DF issues.

WHAT IS EER ?

Equal Error Rate (EER) is a critical metric for evaluating the performance of audio deepfake detection systems. It is the rate at which the proportion of false acceptances (fake audio clips identified as real) equals the proportion of false rejections (real audio clips identified as fake). A lower EER indicates a more accurate system.

Detailed Explanation for Audio Deepfake Detection

1. **False Acceptance Rate (FAR):** The percentage of fake audio clips that are incorrectly classified as genuine.
2. **False Rejection Rate (FRR):** The percentage of genuine audio clips that are incorrectly classified as fake.

3. **Equal Error Rate (EER):** The point at which FAR and FRR are equal. It is a single value that represents the trade-off between FAR and FRR.

EER Calculation and Visualization

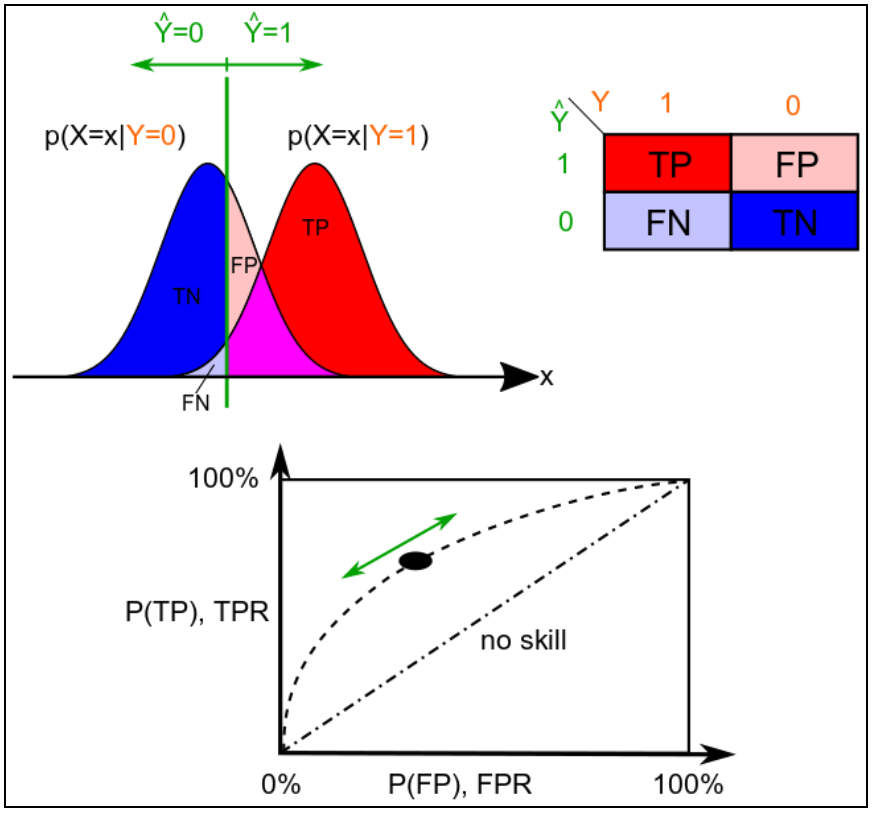
1. **Receiver Operating Characteristic (ROC) Curve:** This curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The EER can be derived from this curve.
2. **Detection Error Trade-off (DET) Curve:** This curve plots the FRR against the FAR, making it easier to identify the EER visually.

Steps to Calculate EER :

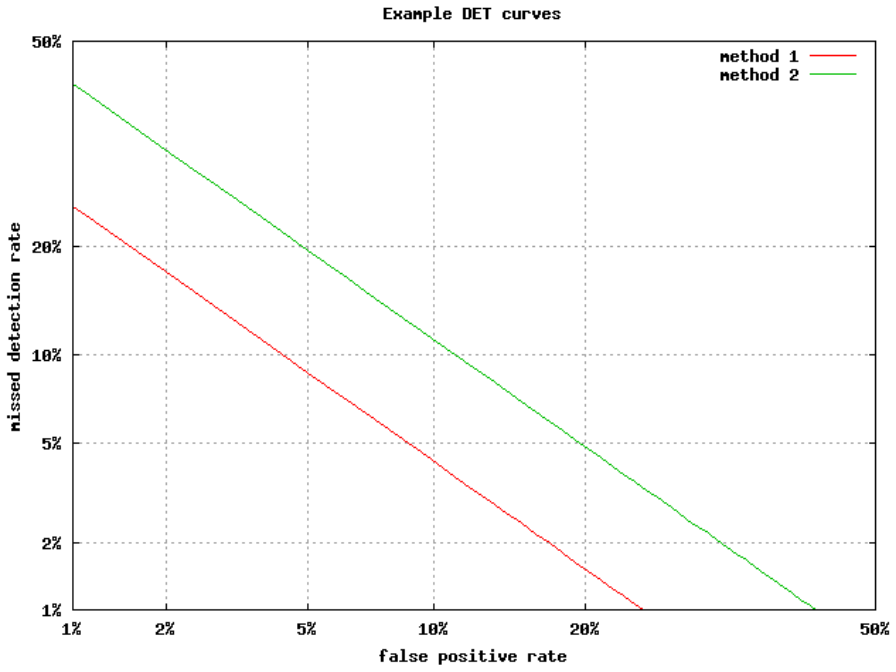
1. **Generate Predictions:** Obtain the model's prediction scores for both real and fake audio samples.
2. **Calculate FAR and FRR:** For various threshold values, compute the FAR and FRR.
3. **Find the EER:** Identify the threshold where FAR equals FRR.

Here are visual examples to illustrate these concepts:

1. **ROC Curve: This shows the TPR vs. FPR for different thresholds.**
    - **True Positive Rate (TPR):** The rate at which real audio clips are correctly identified.
    - **False Positive Rate (FPR):** The rate at which fake audio clips are incorrectly identified as real.



2. **DET Curve: This plots FRR vs. FAR to highlight the EER point.**



- The EER is the point where the DET curve intersects the line where FRR equals FAR.

3. **EER Point:** The intersection point on the DET curve where FAR equals FRR is the EER.

5.) BENCHMARKS :

a.) BASELINE COMPARISON

The baseline comparison included LCNN, MesoNet, SpecRNet, and RawNet3 models. The LFCC and MFCC front-ends were tested, as well as the Whisper encoder. The encoder was not optimized (the weights were frozen), thus we only utilized it as a feature extractor using pre-trained features. It should be noted that some of the models trained on the LA subset of ASVspoof 2019 perform worse than random guessing (EER=0.5). These findings do not call into question the established models. In fact, during validation on ASVspoof 2021, LCNN with LFCC front-end and SpecRNet with MFCC, two architectures that produced the lowest performance on the In-The-Wild dataset, scored satisfying EERs of 0.0149 and 0.0218, respectively. Notably, in the case of all detection models, LFCC and MFCC front-ends tend to produce characteristics that were well-suited in the case of ASVspoof, and when trained and verified on data from the same source, had high efficacy, but do not occur often in the DeepFakes from the In-The-Wild dataset.

Model	Front-end	EER
SpecRNet	LFCC	0.5184
SpecRNet	MFCC	0.6897
SpecRNet	Whisper	0.3644
LCNN	LFCC	0.7756
LCNN	MFCC	0.6762
LCNN	Whisper	0.3567
MesoNet	LFCC	0.5451
MesoNet	MFCC	0.3132
MesoNet	Whisper	0.3856
RawNet3	-	0.5199

Table 1: The comparison of EER scores on In-The-Wild dataset. Using Whisper's encoder as a front-end contributes to the significant enhancement in case of SpecRNet and LCNN networks

b.) CONCATENATING FRONTENDS :

Concatenating Whisper's encoder with LFCC and MFCC under the mindset that since this improves the representation of audio Spectrogram , it may lead to improvement in Performance.

Model	Front-end	EER
SpecRNet	Whisper + LFCC	0.3485
SpecRNet	Whisper + MFCC	0.4116
LCNN	Whisper + LFCC	0.6270
LCNN	Whisper + MFCC	0.6117
MesoNet	Whisper + LFCC	0.8029
MesoNet	Whisper + MFCC	0.3822

Table 2: The EER values of the models using concatenated front-end features. The only enhancement is visible in case of two models, whereas for other models it had a negative impact.

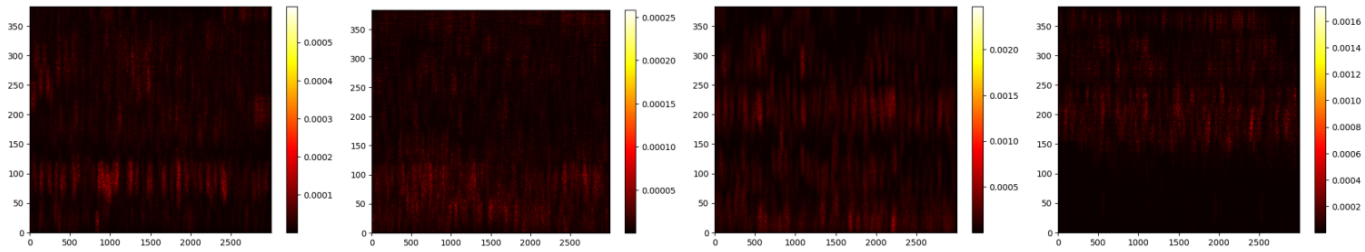
c.) FINETUNING WHISPER TO THE PROBLEM OF DF DETECTION

On this occasion, though, we adjusted the encoder to the DF detection issue rather than treating it as a front-end algorithm in its strict sense. The intuition was as follows: whereas Whisper's features often outperform those of the commonly used front-ends (Tab. 1), this model was trained on biased (DF) data and for a different purpose, ASR. We presume that even better outcomes could be obtained from an encoder that has been fine-tuned to a particular task, in our case DeepFake detection. We used the Whisper front-end to fine-tune the models for this purpose, and we used an improved feature extractor to assess the outcomes. Following the first training described in Sect (5a), we trained the models for five more epochs. On this occasion, though, we defrosted the Whisper layers and used a 10−6 learning rate to fine-tune.Notably, we were able to surpass even the prior best result thanks to unfrozen Whisper features. Our findings suggest that the critical problem of generalization might be resolved by unfreezing the model and utilizing Whisper derived characteristics to detect DeepFakes from a distribution that differs greatly from the set the model was trained on.

Model	Front-end	EER (frozen)	EER (tuned)
SpecRNet	Whisper + LFCC	0.3485	0.3795
SpecRNet	Whisper + MFCC	0.4116	0.3769
SpecRNet	Whisper	0.3644	0.3338
LCNN	Whisper + LFCC	0.6270	0.6270
LCNN	Whisper + MFCC	0.6117	0.5899
LCNN	Whisper	0.3567	0.3290
MesoNet	Whisper + LFCC	0.8029	0.5526
MesoNet	Whisper + MFCC	0.3822	0.2672
MesoNet	Whisper	0.3856	0.3362

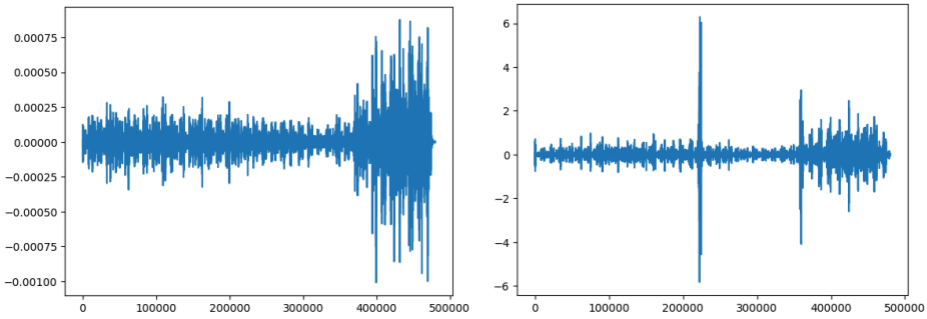
Table 3: Evaluation of the fine-tuned Whisper feature extraction (as a sole front-end and in concatenation), on DeepFake In-The-Wild dataset. Note that 'EER (frozen)' refers to the case, where Whisper's encoder was not fine-tuned (cf. Tab. 1, Tab. 2) and 'EER (tuned)' to the results of fine-tuned extractor.

6.) FEATURE COMPARISON : We examine which aspects of the input data have the greatest impact on the detection outcomes in order to determine whether the various front-ends do, in fact, produce distinct features. We also examine two designs that obtained the best results: MesoNet, which is mostly composed of convolutional and max-pooling layers, and SpecRNet, which has a recurrent layer (GRU). We found that while processing front-ends, the architecture of the model matters a lot. The final linear layers of the model get max-pooled input from spatially distributed blocks of size equal to 32 × 32 because the MesoNet comprises of 4 max-pooling layers (see Fig. 1) . In turn, information in SpecRNet is processed using the GRU, and the decision is mainly impacted by the end part of the signal (Fig. 2a). The models utilizing Whisper features often rely on some characteristics extracted from one or more narrow signal parts (see two peaks around 220k and 360k in Fig. 2b). Whisper is thought to function effectively with RecurrentNN because it extracts salient features that move across the recurrent sequence without frequently becoming concealed. Our results imply a detrimental effect of Whisper characteristics on other front-ends, since we have not evaluated any mechanism for geographically independent processing of the combination of front-ends. To be more precise, models that only use MFCC or LFCC to make decisions typically give the front-end's lower band more weight than its delta and double-delta features (Fig. 1a and 1b). On the other hand, with a clear center of emphasis, the 2D Whisper characteristics demonstrate the greatest influence of the entire band (Fig. 1c). Further speed gain can be achieved by fine-tuning Whisper features; unexpectedly, in this scenario, the significance of delta and double-delta features increases (Fig. 1d). We suppose that using the Whisper front-end effectively captures the speech features, and including deltas could enhance the results by providing additional coefficients to describe the spectrum dynamic.



(a) *LFCC* (b) *MFCC* (c) *Whisper (frozen)* (d) *MFCC + Whisper (tuned)*

Figure 1: *Saliency maps of front-ends of a bona fide sample for the MesoNet.*



(a) *LFCC* (b) *Whisper (frozen)*

Figure 2: *Gradient calculated on a spoofed signal for the SpecRNet. Note that the y-axis ranges differ.*

CODE : https://github.com/piotrkawa/deepfake-whisper-features

7.) REPRODUCED RESULTS :

MODEL	RESULTS FROM PAPER(EER)	RESULTS OBTAINED	APPROX TIME TAKEN
WHISPER_SPECRNET_LFCC	0.3485	0.3581	1 day 14 hrs
WHISPER FINETUNED+MFCC+ MESONET	0.2672	0.2586	1 day 23 hrs
WHISPER+LCNN+ LFCC	0.6270	0.6582	1 day 17 hrs
RAWNET3	0.5199	0.4824	7 days 7 hrs

OTHER METRICS CALCULATED :

MODEL	ACCURACY	PRECISION	RECALL	FI SCORE	AUC (AREA UNDER CURVE)
WHISPER_SPECRNET_LFCC	58.0249	0.8073	0.4358	0.5661	0.6877
WHISPER FINETUNED+MFCC+MESONET	69.9773	0.8823	0.6025	0.7160	0.8033
WHISPER+LCNN+LFCC	32.6315	0.3331	0.0723	0.1188	0.2930
RAWNET3	39.6652	0.7440	0.603	0.1115	0.5155

8.) FURTHER WORK :

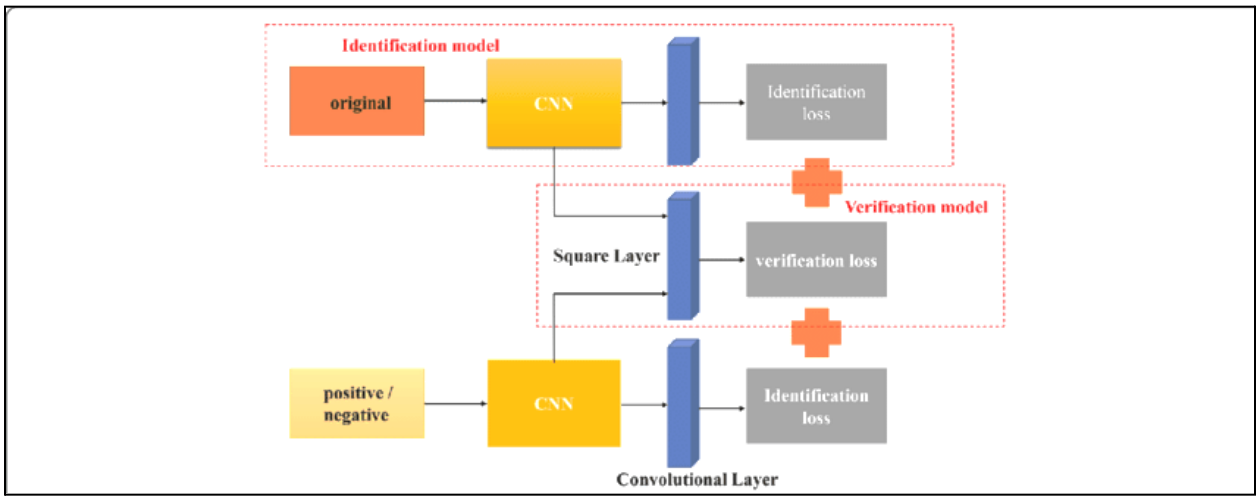
- Figuring out the EER by employing a different model called Siamese Convolutional Neural Network and observing the performance for deepfake detection.
- Since there were two Bi-LSTM layers in the LCNN model, instead of using these layers, we tried to incorporate the Transformer Layer into the LCNN model and identify the changes .

a.) WHAT IS SIAMESE CONVOLUTIONAL NEURAL NETWORK ?

A Siamese Convolutional Neural Network (Siamese CNN) is a type of neural network architecture that uses two or more identical subnetworks to process different inputs and learn to distinguish between them. Each subnetwork is typically a convolutional neural network (CNN) designed to extract features from the inputs. The key feature of a Siamese CNN is that both subnetworks share the same parameters, allowing them to produce comparable feature representations for different inputs. These representations are then compared using a distance metric to determine similarity or difference.

A Siamese Convolutional Neural Network (Siamese CNN) can be effectively used for audio deepfake detection by comparing the audio features of two audio clips and determining whether one is a manipulated version of the other. This approach leverages the ability of Siamese networks to learn discriminative features and measure similarities between inputs.

ARCHITECTURE :



1. **Input Layer**: Two audio clips are fed into the network.
2. **Preprocessing**: Convert audio signals to spectrograms or mel-frequency cepstral coefficients (MFCCs) or linear- frequency cepstral coefficients (LFCCs) to create image-like representations.
3. **Siamese CNN**: Each audio representation is passed through identical CNNs to extract feature vectors.
4. **Distance Metric**: A distance layer (e.g., Euclidean distance) compares the feature vectors.
5. **Output Layer**: The network outputs a similarity score indicating whether the two audio clips are similar (one is a deepfake) or different (one is authentic).

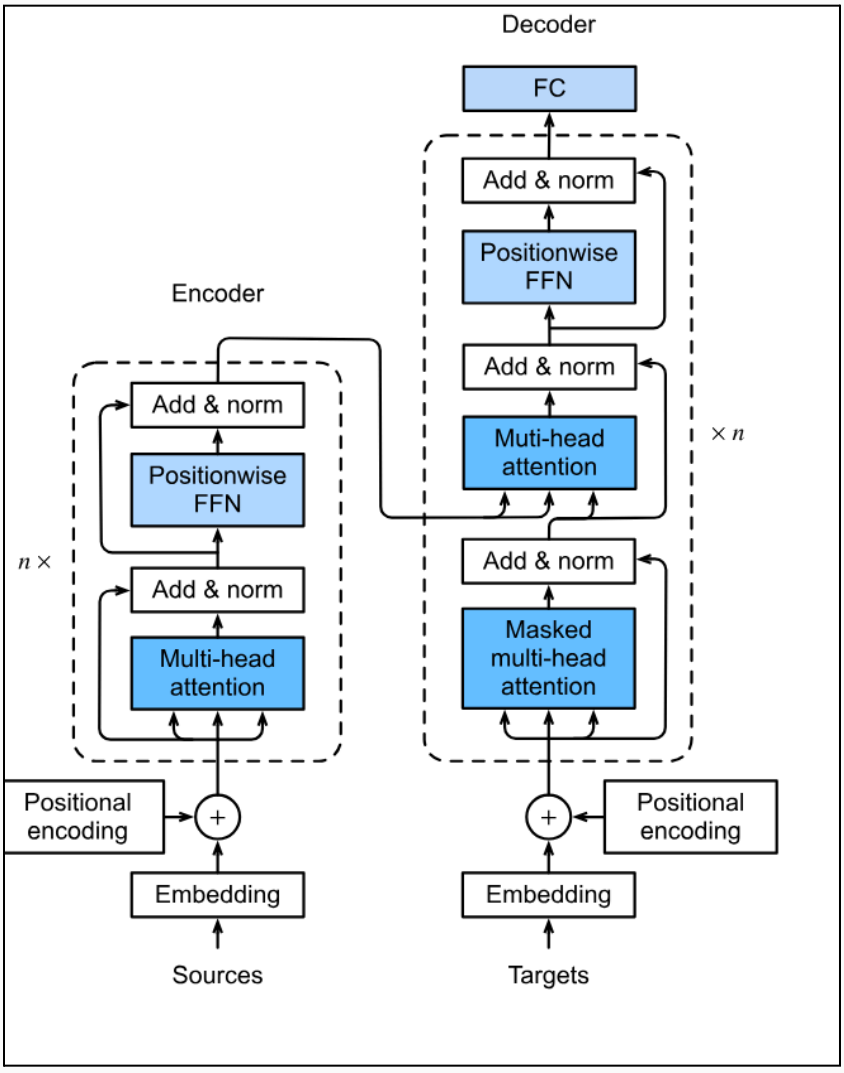
b.) RESULTS FROM INCORPORATING TRANSFORMER LAYER :

A Transformer layer is a key component of the Transformer model architecture, introduced by Vaswani et al. in their 2017 paper "Attention is All You Need." Unlike traditional recurrent neural networks (RNNs) like LSTMs, which process data sequentially, Transformers rely on self-attention mechanisms to process entire sequences of data in parallel. Here are some of the main features and advantages of using a Transformer layer:

1. **Self-Attention Mechanism**: The self-attention mechanism allows the model to weigh the importance of different tokens in a sequence relative to each other. This enables the model to capture long-range dependencies more effectively than RNNs, which may struggle with long-term dependencies due to their sequential nature.
2. **Positional Encoding**: Since Transformers do not process data sequentially, they use positional encodings to maintain the order of the input sequence. These encodings are added to the input embeddings to provide information about the relative or absolute position of the tokens.
3. **Parallelization**: Transformers can process all tokens in a sequence simultaneously, which makes them highly parallelizable and efficient on modern hardware. This is a significant advantage over RNNs, which require sequential processing and are more difficult to parallelize.
4. **Scalability**: Due to their parallel processing capability, Transformers can handle very large datasets and long sequences more effectively than RNNs. They have become the backbone of many state-of-the-art models in natural language processing (NLP) and other domains.
5. **Layer Structure**: A Transformer layer typically consists of two main sub-layers:

- **Multi-Head Self-Attention**: This sub-layer performs self-attention multiple times in parallel (with different sets of learned parameters), allowing the model to capture various aspects of the input sequence relationships.
- **Feed-Forward Neural Network**: This sub-layer is a fully connected feed-forward network applied independently to each position in the sequence. It provides additional non-linear transformations and feature extraction.

ARCHITECTURE :



RESULTS COMPARISON :

(EER COMPARISON)	FRONTEND MFCC	FRONTEND LFCC	FRONTEND WHSIPER
LCNN WITH BI-LSTM LAYERS	0.6693	0.7758	0.3697
LCNN WITH TRANSFORMER LAYER	0.6321	0.7239	0.3352

9.) CONCLUSION :

Based on our comprehensive study, it is evident that the Whisper encoder significantly enhances deepfake detection capabilities. Fine-tuning the Whisper encoder further amplifies its performance. Additionally, the concatenation of Whisper features with LFCC and MFCC frontends yields even better results. Furthermore, the integration of a Transformer Layer into the LCNN model results in a notable decrease in Equal Error Rate (EER), reinforcing the model's effectiveness. Collectively, these findings highlight the promising potential of this approach in advancing deepfake detection technologies.