Deepfake Speech Detection

Yashraj Kadam (22BDS066) Ayush Singh (22BDS012) Nachiket Apte (22BDS041)
Harsh Raj (22BDS027)

Indian Institute of Information Technology Dharwad,
Dharwad, India, 580009
{22bds012, 22bds027, 22bds041, 22bds066}@iiitdwd.ac.in

Abstract— **Deepfake speech detection is a crucial task in ensuring the authenticity of digital communication by identifying manipulated audio that mimics human voices. This work presents a robust binary classification framework designed to distinguish between genuine and synthetic speech, leveraging advanced deep learning techniques to capture subtle spectral and temporal features. Pre-trained models such as Wav2Vec2, WavLM, and HuBERT serve as foundational backbones, enabling effective representation learning through fine-tuning on labeled datasets of real and synthetic audio. The proposed system demonstrates high precision, recall, and generalization across diverse speech characteristics, underscoring its effectiveness in classifying deepfake audio. This work addresses the growing misuse of audio manipulation technologies in misinformation, fraud, and security breaches, contributing to the fields of digital forensics, media verification, and online security by providing a practical and scalable solution for detecting synthetic speech.**

I. Introduction

The rise of deepfake speech, synthetic audio designed to mimic human voices, poses significant challenges to the authenticity of digital communication, with potential misuse in misinformation, fraud, and security breaches. Detecting such manipulated audio is crucial for safeguarding trust in digital interactions. This work addresses the challenge of distinguishing between genuine (bonafide) and synthetic (spoofed) speech by leveraging advanced pre-trained models like Wav2Vec2 with Bi-LSTM, WavLM, HuBERT and ResNet, which capture rich spectral and temporal audio features. Fine-tuned on the ASVspoof 2019 dataset, the proposed binary classification framework effectively detects deepfake audio, demonstrating high accuracy, robustness, and generalization across diverse spoofing techniques.

II. Dataset

This is a database used for the Third Automatic Speaker Verification Spoofing and Countermeasures Challenge, for short, ASVspoof 2019 (http://www.asvspoof.org) organized by Junichi Yamagishi, Massimiliano Todisco, Md Sahidullah, Héctor Delgado, Xin Wang, Nicholas Evans, Tomi Kinnunen, Kong Aik Lee, Ville Vestman, and Andreas Nautsch in 2019 [1]. The ASVspoof 2019 database is designed to reflect two different use case scenarios, namely logical and physical access control. For this experiment, only the logical access control part is used. Logical access (LA) control implies a scenario in which a remote user seeks access to a system or service protected by ASV. In the LA scenario, it is assumed that spoofing attacks are presented to the ASV system in a worst-case, post-sensor scenario. Attacks then take the form of synthetic speech or converted voice, which are presented to the ASV system without convolutive acoustic propagation or microphone effects. ASV/CM training data comprises 2,580 bonafide utterances and 22,800 spoofed utterances generated by using four TTS and two VC algorithms. The ASV/CM development partition contains 1,484 bona fide target utterances, 1,064 bona fide non-target utterances, and 22,296 spoofed utterances generated with the same TTS and VC algorithms. The setup for the ASV/CM evaluation set is similar to that for the ASV/CM development set. There are 5,370 bona-fide target utterances and 1,985 bona fide non-target utterances. Spoofed data comprises 63,828 utterances.

III. Methodology and ApproAch

The task here is to perform CM (Counter Measure protocol) or binary classification of the audio files as bonafide or spoofed. Four different models are used for this: HuBERT, WavLM, Bi-LSTM and ResNet.

Wav2vec2 Architecture: The complete architecture of the framework can be divided into 3 components:

Feature encoder: This is the encoder part of the model. It takes the raw audio data as input and outputs feature vectors. Input size is limited to 400 samples which is 20ms for 16kHz sample rate. The raw audio is first standardized to have zero mean and unit variance. Then it is passed to 1D convolutional neural network (temporal convolution) followed by layer normalization and GELU activation function. There could be 7 such convolution blocks with constant channel size (512), decreasing kernel width (10, 3x4, 2x2) and stride (5, 2x6). The output is a list of feature vectors each with 512 dimensions.

Transformers: The output of the feature encoder is passed on to a transformer layer. One differentiator is use of relative positional embedding by using convolution layers, rather than using fixed positional encoding as done in the original Transformers paper. The block size differs, as 12 transformer blocks with model dimension of 768 are used in BASE model but 24 blocks with 1024 dimension in LARGE version.

Quantization module: For self-supervised learning, we need to work with discrete outputs. For this, there is a quantization module that converts the continuous vector output to discrete representations, and on top of it, it automatically learns the discrete speech units. This is done by maintaining multiple codebooks/groups (320 in size) and the units sampled from each codebook are later concatenated (320x320=102400 possible speech units). The sampling is done using Gumbel-Softmax which is like argmax but differentiable.

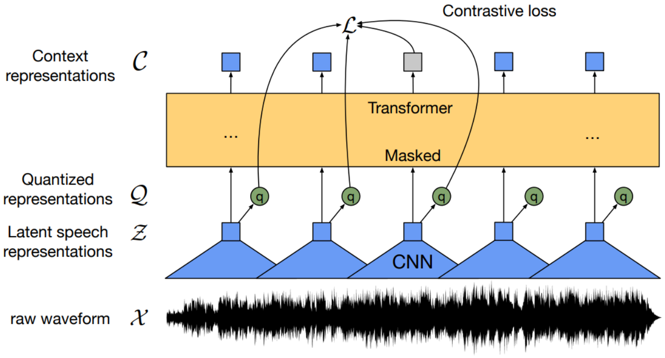


Fig 1. Illustration of the Wav2vec2 framework

A. HuBERT Model

The audio signals are preprocessed and feature-extracted using the Wav2Vec2Processor. This ensures a robust representation of the audio data, preserving critical temporal and spectral characteristics necessary for effective spoof detection.

Model Architecture

Transformer Encoder: The HuBERT model employs a transformer-based encoder to process the extracted audio features. By leveraging its multi-head self-attention mechanism, the encoder captures complex temporal dependencies and patterns in sequential audio data, enabling the model to distinguish subtle variations between spoofed and genuine speech.

Classification Layer: The encoder's output embeddings are passed through a fully connected classification layer with a softmax activation function. This layer outputs the probability scores for the two target classes: *spoof* and *bonafide*.

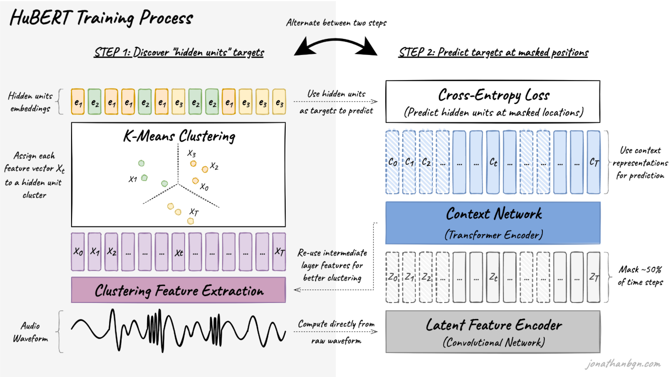

HuBERT Training Process
Alternate between two steps
STEP 1: Discover "hidden units" targets
STEP 2: Predict targets at masked positions
Hidden units embeddings
e_1 e_2 e_1 e_1 e_2 e_1 e_3 e_2 e_2 e_1 e_3 e_3 e_3
Use hidden units as targets to predict
Cross-Entropy Loss
(Predict hidden units at masked locations)
K-Means Clustering
Assign each feature vector X_t to a hidden unit cluster
X_3
X_2
X_1
X_0
X_T
C_0 C_1 C_2 ... ... C_t ... ... C_T
Use context representations for prediction
Context Network
(Transformer Encoder)
X_0 X_1 X_2 ... ... Xt ... ... X_T
Re-use intermediate layer features for better clustering
Clustering Feature Extraction
Z_0 Z_1 Z_2 ... ... Z_t ... ... Z_T
Mask ~50% of time steps
Audio Waveform
Compute directly from raw waveform
Latent Feature Encoder
(Convolutional Network)
jonathanbgn.com

Fig 2. HuBERT Model Architecture

Training and Optimization

The model is fine-tuned using Cross-Entropy Loss to minimize prediction errors and the Adam optimizer with a learning rate of 1e-5 for stable and efficient convergence. Regularization strategies include using a small batch size for better generalization and early stopping to mitigate overfitting.

B. WavLM Model

Audio features are extracted using the Wav2Vec2 feature extractor, as the WavLM model does not include a native feature extraction module. These features effectively capture temporal and spectral properties of the audio signal, forming the input to the WavLM model.

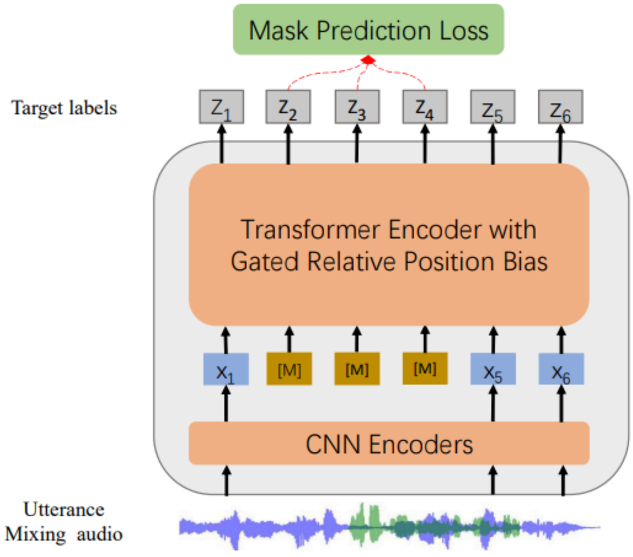


Fig 2. WavLM Model Architecture

Model Architecture

Pre-Trained WavLM Model: The extracted features are passed through the last hidden layer of the pre-trained WavLM model. This layer encodes the input features into high-dimensional representations, effectively capturing both local and global patterns in the audio data.

Feature Aggregation: To reduce the dimensionality and aggregate the learned representations, an adaptive average pooling layer with an output size of 128 is applied. This ensures that the feature size is consistent, regardless of input audio length.

Classification Head: The pooled features are passed through a linear layer, which maps the aggregated features to a lower-dimensional space. A sigmoid activation function, producing a binary output for classification into *spoof* or *bonafide* categories.

Training and Optimization

The model is fine-tuned using Binary Cross-Entropy Loss to minimize classification error, with the Adam optimizer ensuring efficient and stable gradient updates. Adaptive learning rate scheduling facilitates optimal convergence, and early stopping prevents overfitting while promoting generalization.

C. *Wav2vec with Bi-LSTM*

The input audio is preprocessed, and features are extracted using the Wav2Vec2Processor. These features capture both temporal and spectral properties of the audio signal, providing a robust representation for downstream processing.

Model Architecture

Feature Extraction with Wav2Vec2: Wav2Vec2 extracts contextualized audio embeddings directly from raw signals, providing rich representations that are fed into the Bi-LSTM network.

Bi-LSTM Network: A Bidirectional LSTM layer models sequential dependencies in the extracted features by processing them in both forward and backward directions.

Classification Head: The Bi-LSTM outputs are passed through: Adaptive Average Pooling to reduce the dimensionality and aggregate key features across the sequence.A fully connected linear layer for further dimensionality reduction.A sigmoid activation function to produce a binary classification output (*spoof* or *bonafide*).

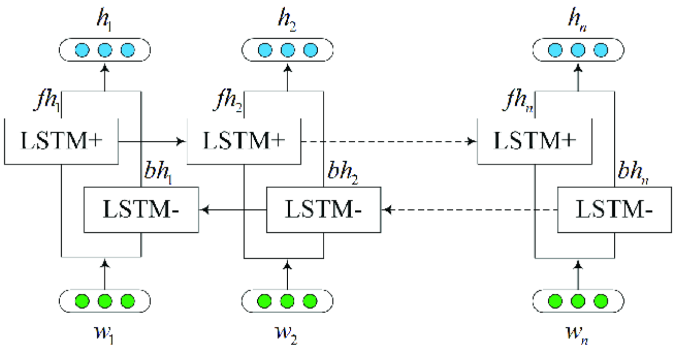


Fig 3. BiLSTM Model Architecture

Training and Optimization

The model is trained using Binary Cross-Entropy Loss for binary classification, with the Adam optimizer (learning rate=1e-5) to ensure efficient and stable convergence. Regularization techniques include dropout layers within the Bi-LSTM to prevent overfitting and early stopping to terminate training when validation performance plateaus.

D. Residual Networks Model (ResNet101)

The input audio signals are converted into Mel spectrograms.The resulting spectrograms are normalized and resized to ensure compatibility with the ResNet model. A fixed number of samples (6×16,000) is used for uniform input length.

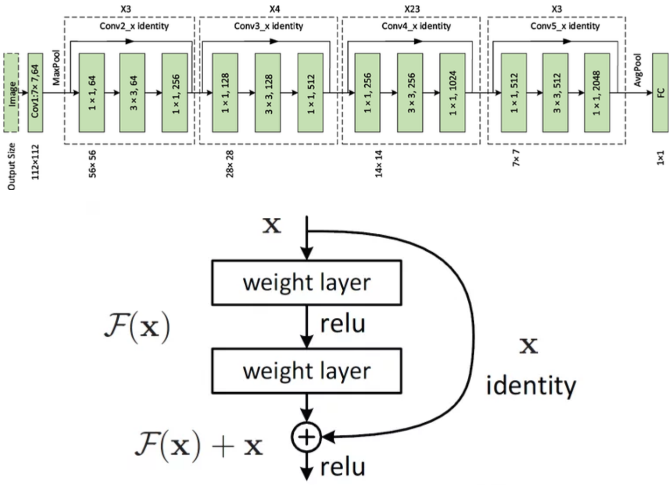


Fig 4. ResNet(101) Model Architecture

Model Architecture

Pre-Trained ResNet101: The model leverages a pretrained ResNet101 architecture, with modifications to adapt it for single-channel input: The first convolutional layer is modified to accept one input channel. The initial 39 layers of the model are frozen to retain pre-trained features while reducing computational load.

Feature Extraction: The ResNet's feature extractor outputs are passed through a custom head comprising Adaptive Average Pooling (aggregates features), Flattening (prepares for dense layers), a Fully Connected Layer (maps to a single output), and a Sigmoid Activation (produces binary classification probabilities for spoof or bonafide).

Training and Optimization

The model is trained with Binary Cross-Entropy Loss for classification, using the Adam optimizer for efficient convergence. A learning rate scheduler dynamically adjusts the learning rate during training, and dropout in the custom layers helps prevent overfitting.

IV. Spectrogram

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal.

Spectral Smoothing: Deepfake audio may show an over-smooth appearance in the spectrogram due to the synthetic nature of the audio generation process.

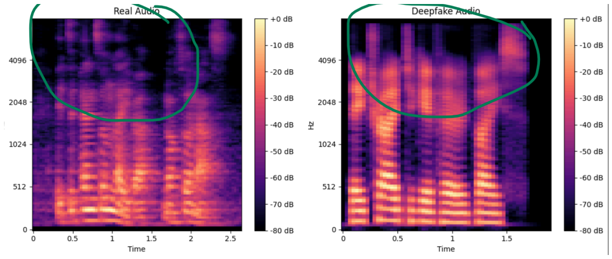


Fig 5. Spectral Smoothing in Spectrogram

Unusual or consistent background noise that doesn't align with human speech patterns can be a sign of synthetic audio.

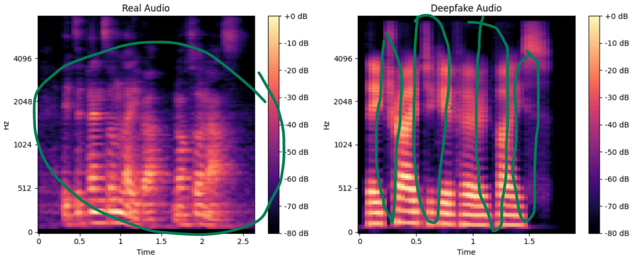
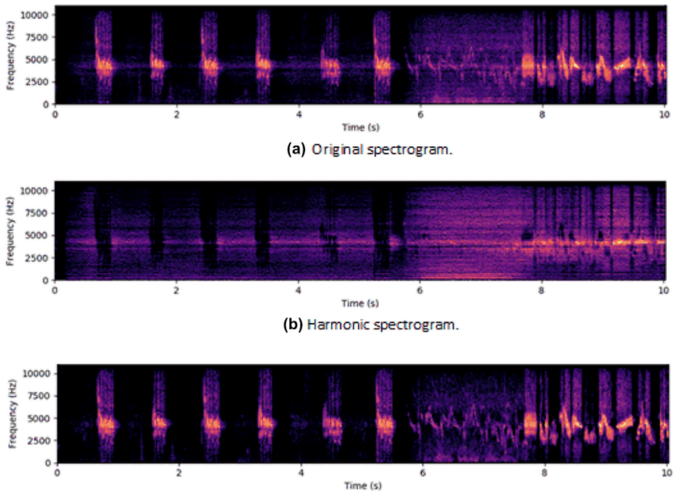


Fig 6. Background noise in Spectrogram

Harmonic Distortion: In real audio, harmonics appear as regular, evenly spaced horizontal lines or bands in a Mel spectrogram. For voiced sounds, such as human speech, these lines are often clear and consistent. Deepfake audio might show irregularities in these patterns. You may notice uneven spacing, extra or missing harmonics, or irregularly shaped harmonic structures due to distortion introduced by the synthesis process.



(a) Original spectrogram.

(b) Harmonic spectrogram.

(c) Percussive spectrogram.

Fig 7. Spectrogram

Unusual Energy Concentration: Notice if certain frequency bands have unusually high or low energy compared to others. For example, energy might be concentrated in unusual frequency ranges or show non-standard distributions.

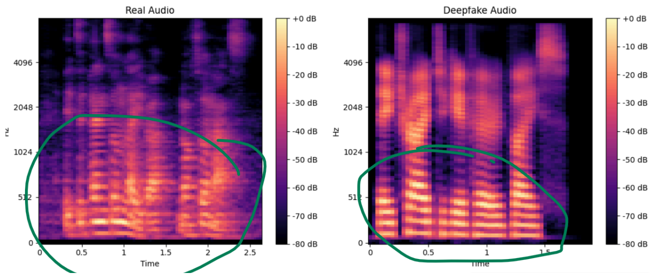


Fig 8. Energy Concentration in Spectrogram

V. Results

	Models			
	HuBERT	WavLM	Bi-LSTM	ResNet
EER	0.36	0.01	0.45	0.30

VI. Conclusions

This research evaluated various models for deepfake speech detection, analyzing their strengths and weaknesses. HuBERT and Bi-LSTM had higher Equal Error Rates (EERs) of 0.3683 and 0.45, respectively, highlighting challenges in capturing spoofing patterns effectively. ResNet101, leveraging Mel spectrograms, improved performance with an EER of 0.30, demonstrating the capability of convolutional networks in audio-based tasks. WavNet achieved a competitive EER of 0.074, showcasing its strength in processing raw audio and extracting hierarchical features. WavLM outperformed all models, achieving an EER of 0.01, emphasizing its superior ability to generalize and extract robust embeddings for deepfake detection.

VII. Acknowledgement

We thank Dr. Animesh Chaturvedi, our course instructor for allowing us to experiment our knowledge of Deep learning and leveraging the programming skills on this topic. We would also like to thank kaggle and ASVSpoof 2019 team for providing this robust dataset to fight the challenge of DeepFake. We would also like to thank fellow kaggle members who have helped in providing a starter code for properly accessing the labels and audio files stored in the directory.

VIII. References

[1] https://doi.org/10.48550/arXiv.1911.01601

[2] Nicholas Evans, Tomi Kinnunen and Junichi Yamagishi, "Spoofing and countermeasures for automatic speaker verification", Interspeech 2013, 925-929, August 2013

[3] Zhizheng Wu, Tomi Kinnunen, Nicholas Evans, Junichi Yamagishi, Cemal Hanilc, Md Sahidullah Aleksandr Sizov, "ASVspoof 2015: the First Automatic Speaker Verification Spoofing and Countermeasures Challenge", Proc. Interspeech 2015 2037-2041 September 2015

[4] Baevski, Alexei & Zhou, Henry & Mohamed, Abdelrahman & Auli, Michael. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. 10.48550/arXiv.2006.11477.

Github Repository