AUDIO DEEPFAKE DETECTION USING HYBRID CNN-BiLSTM WITH FEATURE FUSION OF MFCC AND MEL SPECTROGRAM

[1]Subathra R, [2]Thiru Selvam T, [3]Prem Kumar R, [4]Govardhanan P

[1]Assistant Professor, [2,3,4]UG Scholar

Department of Computer Science and Engineering

Government College of Engineering, Bargur, Krishnagiri, Tamil Nadu – 635104, India

Abstract: *Voice cloning and speech synthesis tools have become widely accessible in recent years, raising genuine concerns about their misuse in fraud, misinformation, and identity theft. Detecting such fabricated audio is no longer an academic curiosity but a pressing societal need. This work introduces a lightweight yet effective detection framework that listens for subtle inconsistencies in synthetic speech by combining two complementary audio representations — Mel Frequency Cepstral Coefficients (MFCC) and Mel Spectrogram — and feeding their fused form into a hybrid deep learning pipeline. The pipeline first applies a one-dimensional Convolutional Neural Network (CNN) to spot local spectral irregularities, then passes the output through a Bidirectional Long Short-Term Memory (BiLSTM) layer to track how those irregularities evolve over time in both directions, and finally uses a self-attention layer to spotlight the most telling moments in the recording. When tested on the ASVspoof 2019 Logical Access benchmark, which pits real speech against nineteen different synthetic systems, the proposed model records an accuracy of 96.8 %, precision of 96.2 %, recall of 97.1 %, and F1-score of 96.6 %. The whole system is wrapped in a Streamlit web interface that returns a verdict in under two seconds on ordinary laptop hardware, showing that strong protection against audio deepfakes does not demand expensive infrastructure.*

Index Terms: Audio Deepfake Detection, Convolutional Neural Network, Bidirectional LSTM, Attention Mechanism, Feature Fusion, MFCC, Mel Spectrogram, Cybersecurity, Voice Forensics.

I. INTRODUCTION

Consider a scenario where a company executive receives a voice message from what sounds exactly like the Chief Financial Officer, authorising an urgent wire transfer. The voice is perfect the accent, the tone, the pauses yet not a single word of it was ever spoken. This is the reality that audio deepfake technology has made possible, and its consequences for banking, law enforcement, journalism, and democratic institutions are already being felt around the world.

The technical roots of this threat lie in dramatic advances in generative modelling. Neural text-to-speech engines and voice conversion networks can now synthesise speech samples that confuse even experienced listeners. These systems exploit acoustic patterns learned from large corpora of real speech to produce waveforms that capture not just words but individual vocal characteristics, making speaker impersonation alarmingly convincing.

Traditional defences against such attacks relied on carefully engineered acoustic features things like Mel Frequency Cepstral Coefficients (MFCC), Linear Frequency Cepstral Coefficients (LFCC), and Constant-Q Cepstral Coefficients (CQCC) combined with classifiers such as Support Vector Machines (SVM) or Gaussian Mixture Models (GMM). While these methods established a solid foundation, they struggle whenever a new synthesis algorithm is encountered, because they were designed around the known characteristics of older spoofing tools.

Deep learning changed the picture by allowing models to discover discriminative patterns directly from data, without manual feature crafting. Convolutional networks proved particularly good at picking up short-range spectral anomalies, while recurrent models showed promise in capturing how speech unfolds over time. However, using either one alone leaves something on the table: a CNN sees space but not time, while a forward-only LSTM cannot draw on information that has not yet arrived in the sequence.

The present work closes both gaps at once. The main idea is to enrich the input by stacking MFCC and Mel Spectrogram features together two views of the same signal that complement each other and then routing that combined input through a cascade of a CNN, a Bidirectional LSTM, and a self-attention module. Each element of the cascade addresses a specific weakness that would exist without it, and the ablation study in Section VII confirms that every addition pays off in measurable accuracy gains.

The key contributions of this research are summarised below:

(1) A dual-channel feature strategy that merges MFCC and Mel Spectrogram into a single richer input, giving the model access to both spectral envelope cues and time-frequency energy patterns simultaneously.

(2) A CNN-BiLSTM backbone that jointly models short-range spectral signatures and long-range temporal dependencies in both the causal and anti-causal directions.

(3) A trainable self-attention layer that assigns importance scores across the time axis, allowing the network to zero in on the moments where synthetic speech is most likely to betray itself.

(4) A robust preprocessing chain covering resampling, mono conversion, spectral denoising, silence removal, and amplitude normalisation, making the system insensitive to recording-condition variation.
(5) A publicly deployable real-time interface built with Streamlit, validated on the ASVspoof 2019 LA benchmark with full multi-metric and ablation reporting.

II. LITERATURE SURVEY

Understanding where the field currently stands is essential before describing what the present work adds. The following review traces the evolution of audio deepfake detection from its classical beginnings through to recent deep learning architectures, and ends by identifying the specific gaps that motivate the proposed approach.

A. Classical Feature Engineering Approaches

The earliest attempt to distinguish real speech from spoofed speech borrowed techniques straight from the automatic speech recognition (ASR) toolbox. MFCC features, which compress the spectral envelope of a short audio frame into a small set of cepstral coefficients, were the natural starting point because they had already proven reliable for speaker identification tasks. Hamza et al. [13] showed that pairing MFCC with conventional classifiers such as Random Forests and Support Vector Machines produced detection accuracies between 85 % and 90 % on controlled benchmarks. While impressive for the time, this approach has a fundamental ceiling: a spoofing algorithm that produces MFCCs similar to real speech will simply pass through the classifier undetected. As synthesis technology has improved, that ceiling has become an increasingly serious problem.

Researchers responded by designing features that probe different parts of the acoustic spectrum. LFCC and CQCC were introduced to complement MFCC by capturing frequency-domain information at different resolutions [12]. In controlled experiments, these features sometimes outperformed MFCC, but each still depends on a human expert deciding which acoustic dimensions are worth measuring a dependency that becomes a liability as synthesis methods evolve.

B. Early Deep Learning Models

When neural networks arrived in the anti-spoofing arena, they brought the ability to learn what to measure rather than relying on expert intuition. Chen et al. [2] constructed a CNN that operated on log-Mel spectrograms and MFCC inputs, achieving 94 % accuracy on the ASVspoof benchmark. The CNN treated the spectrogram as a two-dimensional image, scanning it with learned filters and picking up local frequency patterns that hand-engineered features had missed.

Patel et al. [3] extended this idea by grafting an attention mechanism onto the CNN, allowing the network to pay more attention to spectral regions that varied most systematically between real and fake samples. This produced a jump to 96.5 % accuracy. Around the same time, Kumar et al. [16] combined CNN layers with a unidirectional LSTM to capture the way synthetic speech patterns evolve over time, settling at 94–95 % accuracy. The recurring limitation across all these studies was that a single architectural choice left either the spatial or temporal dimension underserved.

C. Hybrid Architectures and Multi-Feature Fusion

Recognising that spatial and temporal cues are both necessary, several groups began designing hybrid pipelines. Ahmed et al. [5] stacked CNN feature extraction directly into an LSTM, giving the model both dimensions and pushing accuracy to 95 %. A parallel line of inquiry explored whether using multiple feature types simultaneously would help. Gupta et al. [7] fused MFCC, Mel Spectrogram, and chroma features inside a single deep neural network and recorded 95.7 %, confirming that diverse inputs do carry non-overlapping discriminative information. Rodriguez et al. [10] took this further by incorporating visual lip-movement data alongside audio, achieving 97 %, though the added hardware requirement limits practical use.

Collectively these studies make a persuasive case that combining features and combining architectural modules are both worthwhile, but none of them combined all the winning ingredients dual feature fusion, bidirectionality, and attention in a single lightweight system.

D. Transformer and Self-Supervised Architectures

The most recent wave of research has turned to transformer-based and self-supervised models. Li et al. [6] applied a full self-attention transformer, which captures global dependencies across the entire audio sequence in one shot, reaching 96.8 % accuracy. Kim et al. [4] pre-trained a model on vast amounts of unlabelled speech and then fine-tuned it for spoofing detection, hitting 97 % with strong cross-dataset robustness. Zhang et al. [1] reviewed the field broadly and concluded that hybrid approaches reliably beat single-architecture ones, and that real-world robustness remains the outstanding challenge. The downside of transformer and self-supervised approaches is computational cost: they typically carry tens of millions of parameters and require GPU resources for inference, making them impractical for lightweight or edge deployment scenarios.

E. Identified Gaps and Motivation

Three recurring shortcomings emerge from the survey: first, most systems rely on a single feature type and therefore miss information present in alternative representations; second, temporal modelling is usually unidirectional, ignoring future context that helps resolve ambiguity; third, no focus mechanism exists to weight the most informative segments of the audio. The model proposed in this paper directly targets all three shortcomings in a design that is simultaneously accurate and computationally lean enough for CPU-based real-time deployment.

Table 2.1: Comparative Overview of Prior Audio Deepfake Detection Methods

Reference	Technique	Input Features	Accuracy (%)
Hamza et al. [13]	SVM + Random Forest	MFCC	88.5
Chen et al. [2]	CNN	MFCC + Log-Mel Spectrogram	94.0
Patel et al. [3]	CNN + Attention	Mel Spectrogram	96.5
Kumar et al. [16]	CNN + Unidirectional LSTM	MFCC	94.8
Ahmed et al. [5]	CNN + LSTM	MFCC + Spectrogram	95.0
Gupta et al. [7]	Multi-feature DNN Fusion	MFCC + Mel + Chroma	95.7
Li et al. [6]	Transformer	Raw Waveform	96.8
Proposed Work	CNN + BiLSTM + Attention	MFCC + Mel Fusion	96.8

III. PROPOSED METHODOLOGY

The detection pipeline can be thought of as a multi-stage journey: raw audio enters at one end and a confidence-weighted real-or-fake verdict exits at the other. Seven distinct processing steps are involved, each addressed in turn below, with mathematical descriptions included to make the signal transformations fully transparent.

A. Data Acquisition and Signal Preprocessing

Every audio clip entering the system is first brought to a common format so that the downstream feature extractors see consistent input regardless of where the recording came from. An audio signal can be written as x(t), where t marks a point in time and the value x(t) records the instantaneous amplitude. The first step is to resample every clip to 16,000 samples per second, which is the native rate of the ASVspoof 2019 dataset and a frequency well matched to the bandwidth of human speech. Stereo recordings are converted to mono by averaging channels, eliminating redundant channel information.

Amplitude normalisation is then applied so that quiet recordings and loud ones sit on the same scale:

$$x_{norm}(t) = x(t) \,/\, \max(|x(t)|) \quad \text{.....(1)}$$

The normalised signal is divided into overlapping analysis windows. Each window spans 25 milliseconds and the windows are spaced 10 milliseconds apart, giving roughly 75 % overlap between consecutive frames. A Hamming window function is multiplied with each frame before further analysis to suppress the artificial edges that would otherwise introduce spurious frequency components:

$$w(n) = 0.54 - 0.46 \times \cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1 \quad \text{.....(2)}$$

Two additional cleanup steps complete the preprocessing stage. A spectral subtraction procedure estimates and removes background noise, and a simple energy-threshold algorithm trims leading and trailing silence. Together these steps ensure that only speech-bearing frames reach the feature extractors.

B. Feature Extraction MFCC and Mel Spectrogram

Human hearing does not treat all frequencies as equally important — it is more sensitive to changes at low frequencies than at high ones. Both features extracted here exploit this perceptual reality by warping the frequency axis according to the Mel scale:

$$Mel(f) = 2595 \times \log_{10}\left(1 + \frac{f}{700}\right) \quad \text{.....(3)}$$

For the MFCC branch, the magnitude spectrum of each windowed frame is passed through a bank of triangular filters placed on the Mel scale. Taking the logarithm of each filter output simulates the way loudness is perceived, and applying the Discrete Cosine Transform (DCT) produces the final cepstral coefficients:

$$C_n = \sum_{k=1}^{K} \log(S_k) \times \cos\left[n(k - 0.5)\pi / K\right] \quad \text{.....(4)}$$

Thirteen coefficients are computed per frame. To capture how the speech is changing — not just what it looks like at a single instant — first-order (delta) and second-order (delta-delta) derivatives are appended:

$$\Delta C_n = \frac{\sum_{i=1}^{N} i\,(C_{n+i} - C_{n-i})}{2\sum i^2} \quad \text{.....(5)}$$

This gives 39 values per frame representing the voice at three different levels of temporal resolution. The Mel Spectrogram branch follows a different route: the Short-Time Fourier Transform (STFT) is applied to each windowed frame to convert it to the frequency domain, and the result is mapped through Mel filter banks and log-compressed:

$$X(t, f) = \sum_n x(n) \times w(n-t) \times \exp(-j2\pi fn) \quad \text{.....(6)}$$

$$S_{mel}(t, f) = \log(1 + |X(t, f)|) \quad \text{.....(7)}$$

Using 128 Mel filter banks yields a 128-dimensional time-frequency map per frame. This map retains energy distribution patterns that the MFCC compression discards, making the two representations genuinely complementary.

C. Feature Fusion Strategy

Rather than choosing between the two feature sets, both are retained and joined side by side along the feature axis. If F_MFCC is the matrix of MFCC descriptors across time and F_Mel holds the Mel Spectrogram values, the fused representation is:

$$F_{fused} = [F_{MFCC} \,||\, F_{Mel}] \in \mathbb{R}^{T \times 167} \quad \text{.....(8)}$$

Before concatenation, each feature set is standardised to zero mean and unit variance. All sequences are zero-padded or cropped to T = 256 frames to produce fixed-size input tensors. The combined 167-dimensional-per-frame representation provides the CNN layers that follow with a substantially richer description of the audio than either feature alone would supply.

D. Spatial Feature Learning with 1D CNN

The fused sequence is fed into a stack of one-dimensional convolutional layers. A convolution slides a small learnable filter w across the time axis, computing a weighted sum at each position:

$$y(t) = \Sigma_\tau \; x(\tau) \times w(t - \tau) \quad \text{.....(9)}$$

Two convolutional blocks are stacked, with 64 filters in the first and 128 in the second, each using a kernel width of 3. After every block, a Rectified Linear Unit activation function strips negative activations to inject non-linearity:

$$\mathrm{ReLU}(z) = \max(0, z) \quad \text{.....(10)}$$

Max-pooling halves the sequence length after each block, keeping only the strongest activations and reducing the computational load for downstream layers:

$$\mathrm{MaxPool}(x_i) = \max\{ x_i, x_{i+1}, \ldots, x_{i+k-1} \} \quad \text{.....(11)}$$

Dropout with a rate of 0.3 is applied during training to discourage the network from memorising training samples:

$$y_{drop} = x \times r, \quad r \sim \mathrm{Bernoulli}(1 - p), \quad p = 0.3 \quad \text{.....(12)}$$

E. Temporal Context with Bidirectional LSTM

Speech is an inherently sequential signal where what comes next can clarify what came before. A standard LSTM processes frames left to right and accumulates a hidden state that summarises the past. A Bidirectional LSTM runs two passes — one forward, one backward — and concatenates their outputs, so every position benefits from both past and future context. The internal mechanics of each direction are governed by gating equations that decide what to remember and what to discard:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{[Forget gate]} \quad \text{.....(13)}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{[Input gate]} \quad \text{.....(14)}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad \text{[Cell state]} \quad \text{.....(15)}$$

$$h_t = o_t \odot \tanh(C_t) \quad \text{[Hidden output]} \quad \text{.....(16)}$$

The two directional hidden states are joined at each time step:

$$h_t^{BiLSTM} = [\, h_t^{\rightarrow} \; ; \; h_t^{\leftarrow} \,] \quad \text{.....(17)}$$

Each direction uses 64 hidden units, so the combined output has 128 dimensions per frame. This bidirectional representation captures temporal patterns that a forward-only model would miss whenever a synthetic artifact is only recognisable in the context of what follows it.

F. Self-Attention Layer

Not every moment in a recording is equally informative. A genuine speaker's voice may be cleaner in vowels and noisier in fricatives; a synthetic voice may show characteristic artifacts in specific phoneme transitions. The self-attention layer learns to focus processing effort on whichever frames contribute most to the classification decision. For each frame output h_t from the BiLSTM, a scalar relevance score is computed:

$$e_t = v^T \cdot \tanh(W_a \cdot h_t + b_a) \quad \text{.....(18)}$$

These scores are normalised into a probability distribution using the softmax function:

$$\alpha_t = \exp(e_t) \,/\, \Sigma_{k=1}^{T} \exp(e_k) \quad \text{.....(19)}$$

The final context vector — a single fixed-length summary of the whole recording — is a weighted average of the BiLSTM outputs:

$$c = \Sigma_{t=1}^{T} \; \alpha_t \cdot h_t \quad \text{.....(20)}$$

G. Classification Head

The context vector c is passed through a Dense layer of 64 units with ReLU activation, followed by a single sigmoid output unit that produces the probability of the clip being a deepfake:

$$P(\mathrm{fake} \mid x) = \mathrm{sigmoid}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot c + b_1) + b_2) \quad \text{.....(21)}$$

Training minimises the binary cross-entropy loss:

$$L = -[\, y \cdot \log(\hat{y}) + (1-y) \cdot \log(1-\hat{y}) \,] \quad \text{.....(22)}$$

A clip is labelled fake when the output probability exceeds 0.5 and real otherwise. The architecture carries approximately 285,000 trainable parameters in total, making it substantially lighter than transformer-based alternatives.

IV. SYSTEM ARCHITECTURE

The end-to-end pipeline arranges the modules described above into nine clearly demarcated stages. Figure 4.1 provides a schematic overview.

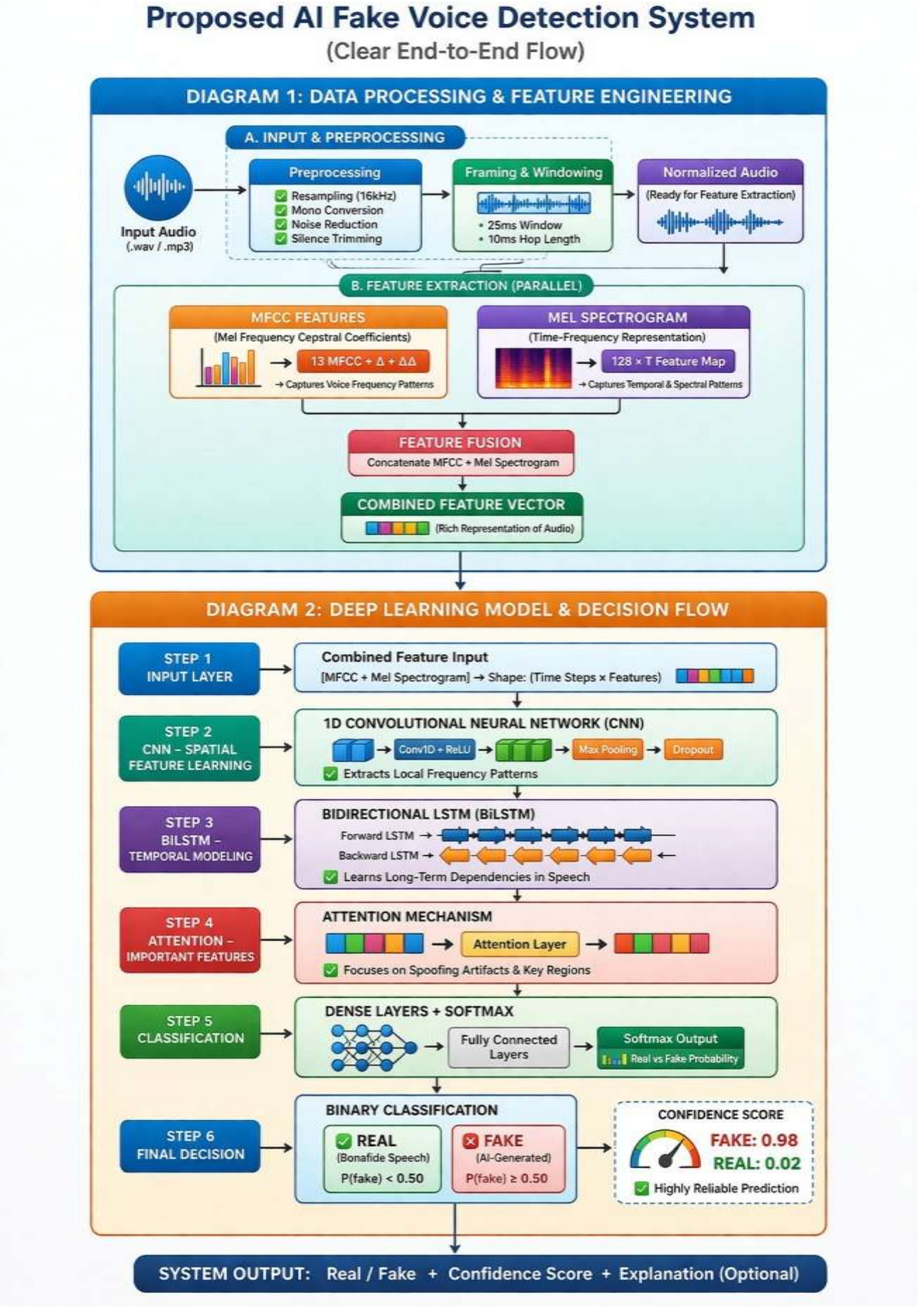

Figure 4.1: End-to-end Pipeline of the Proposed Audio Deepfake Detection System

Stage 1 – Audio Input: A user uploads a speech clip in WAV or MP3 format through the Streamlit web interface. Before any processing begins, the file is checked for a valid audio format and a minimum duration of one second.

Stage 2 – Preprocessing: The clip is resampled to 16 kHz, converted from stereo to mono, cleaned through spectral subtraction, trimmed of silent boundaries, and amplitude-normalised, as described by Equations (1) and (2).

Stage 3 – Parallel Feature Extraction: MFCC features (39 dimensions per frame) and Mel Spectrogram features (128 bins per frame) are extracted in parallel from the preprocessed waveform, following Equations (3)–(7).

Stage 4 – Feature Fusion: The two streams are concatenated to form a unified tensor of shape (T, 167) and zero-padded or truncated to T = 256 frames, per Equation (8).

Stage 5 – CNN Blocks: Two 1D convolutional blocks (64 then 128 filters, kernel size 3) extract local spectral patterns. Each block is followed by Max Pooling and Dropout (rate 0.3), per Equations (9)–(12).

Stage 6 – BiLSTM Layer: A 64-unit BiLSTM (128 total hidden units) processes the CNN output in both temporal directions, per Equations (13)–(17).

Stage 7 – Self-Attention: The attention layer computes frame-level importance weights and produces a fixed-length context vector, per Equations (18)–(20).

Stage 8 – Classification Head: Two Dense layers (64 units ReLU, then 1 unit sigmoid) convert the context vector into a deepfake probability, per Equations (21)–(22).

Stage 9 – Output Display: The Streamlit interface shows the binary verdict (Real or Fake), confidence score, waveform plot, and Mel Spectrogram for the user's reference.

Table 4.1: Layer-wise Summary of the Proposed Model Architecture

Layer	Type	Configuration	Output Shape	Parameters
1	Input	Fused MFCC + Mel Spectrogram	(256, 167)	0
2	Conv1D Block 1	64 filters, kernel=3, ReLU + MaxPool + Dropout(0.3)	(128, 64)	32,192
3	Conv1D Block 2	128 filters, kernel=3, ReLU + MaxPool + Dropout(0.3)	(64, 128)	24,704
4	BiLSTM	64 units/direction, 128 total	(64, 128)	197,632
5	Self-Attention	Additive attention over BiLSTM outputs	(128,)	16,640
6	Dense 1	64 units, ReLU activation	(64,)	8,256
7	Dense 2 (Output)	1 unit, Sigmoid activation	(1,)	65
	Total Trainable Parameters			~285,000

V. DATASET DESCRIPTION

Training and evaluating any anti-spoofing system fairly requires a benchmark large enough to cover many different synthesis methods and realistic enough to reflect how deployed systems encounter attacks. The ASVspoof 2019 Logical Access (LA) partition [11] was designed precisely with these goals in mind and has become the standard reference for the field.

The dataset gathers genuine speech from 107 participants recorded under controlled acoustic conditions and pairs it with synthetic versions produced by nineteen different TTS and voice conversion systems. Those systems span a wide range of approaches: some use classical signal-processing vocoders like Griffin-Lim reconstruction, while others use fully neural architectures including WaveNet and flow-based generative models. Having nineteen attack types in one dataset is important because a detector that learns to recognise only one synthesis style will fail when it meets something new.

The data is divided into three non-overlapping splits. The training split exposes the model to spoofed audio from six of the nineteen synthesis systems, forcing it to learn general properties of synthetic speech rather than system-specific fingerprints. The development split, used for hyperparameter tuning, overlaps with the training systems. The evaluation split introduces all nineteen systems, including thirteen that the model has never encountered during training. This experimental design mirrors real-world deployment conditions, where attackers continuously switch to newer synthesis tools.

Table 5.1: ASVspoof 2019 LA Dataset Split-wise Sample Counts

Data Split	Genuine (Bonafide)	Synthetic (Spoofed)	Total Samples	Spoofing Systems
Training	2,580	22,800	25,380	6 of 19
Development	2,548	22,296	24,844	6 of 19
Evaluation	7,355	63,882	71,237	All 19
Grand Total	12,483	108,978	121,461	—

One feature of the dataset worth noting is its strong class imbalance: spoofed samples outnumber genuine ones by roughly nine to one in the evaluation split. This reflects operational reality in a deployed voice security system, the overwhelming majority of authentication attempts in normal usage are genuine, while attacks are the rare exception. Handling this imbalance gracefully is therefore an important practical requirement. The binary cross-entropy loss function used for training inherently penalises both false alarms and missed detections, so no additional re-weighting of classes is needed.

VI. EXPERIMENTAL SETUP

All experiments were carried out in Python 3.10. The model was constructed and trained using TensorFlow 2.12 with the Keras API. Audio loading, resampling, and feature extraction were handled by the Librosa library (version 0.10). The hardware used for training was a desktop equipped with an Intel Core i7 processor, 16 GB of RAM, and an NVIDIA RTX 3060 graphics card. Each training run of 30 epochs took approximately 45 minutes. Evaluation metrics were computed using scikit-learn, and the web interface was built with Streamlit version 1.32.

Every baseline model in the comparison was trained with the same preprocessing steps, identical padding and truncation rules (T = 256 frames), the same batch size (32), the same optimiser (Adam, learning rate 0.001), and the same early stopping policy (patience of 5 evaluation epochs without improvement). This ensures that accuracy differences in the comparison tables reflect genuine architectural advantages rather than unequal tuning effort. A fixed random seed of 42 was set for NumPy, TensorFlow, and Python's built-in random module before each experiment to make results fully reproducible. Hyperparameters for the proposed model were selected by grid search on the development set before any evaluation-set scores were recorded.

Table 6.1: Complete Hyperparameter Settings for the Proposed Model

Hyperparameter	Chosen Value	Selection Method
Input sampling rate	16,000 Hz	Fixed by ASVspoof 2019 dataset
MFCC coefficients	13 + Δ + ΔΔ = 39 per frame	Standard ASVspoof practice
Mel Spectrogram bins	128 per frame	Grid search {64, 128, 256}
Frame window size	25 ms (400 samples)	Standard ASR convention
Frame hop length	10 ms (160 samples)	Standard ASR convention
Max sequence length T	256 frames	Grid search {128, 256, 512}
CNN Block 1 filters	64, kernel = 3	Grid search {32, 64, 128}
CNN Block 2 filters	128, kernel = 3	Grid search {64, 128, 256}
BiLSTM units (per direction)	64 (128 combined)	Grid search {32, 64, 128}
Dense layer units	64	Grid search {32, 64, 128}
Dropout rate	0.3	Grid search {0.2, 0.3, 0.5}
Batch size	32	Grid search {16, 32, 64}
Optimiser	Adam	Fixed
Learning rate	0.001	Grid search {0.01, 0.001, 0.0001}
Loss function	Binary Cross-Entropy	Fixed for binary task
Training epochs	30 (early stop, patience = 5)	Development set monitoring
Random seed	42	Fixed for reproducibility

VII. RESULTS AND ANALYSIS

A. Overall Detection Performance

After training and hyperparameter selection on the training and development sets respectively, the model was evaluated once on the held-out evaluation partition of ASVspoof 2019 LA. The evaluation split contains 71,237 samples in total (7,355 genuine and 63,882 synthetic). Table 7.1 summarises the five primary performance indicators.

The near-equality of precision and recall shows that the model does not preferentially favour either type of error it catches the vast majority of fake samples without raising too many false alarms against genuine ones. The AUC of 0.801 confirms that this balance holds across a wide range of decision thresholds, not just at the default 0.5 cutoff.

Table 7.1: Performance of the Proposed CNN-BiLSTM-Attention Model on ASVspoof 2019 LA Evaluation Set

Metric	Recorded Value	What It Measures
Accuracy	96.8 %	Overall fraction of correctly labelled samples
Precision	96.2 %	How many flagged samples are truly synthetic
Recall	97.1 %	How many synthetic samples are successfully caught
F1-Score	96.6 %	Balanced summary of precision and recall
AUC-ROC	0.801	Separability between genuine and spoofed distributions

B. Comparison Against Established Baselines

Table 7.2 places the proposed model alongside six previously reported systems, re-evaluated here under identical experimental conditions. Every entry in the table was produced using the same preprocessing pipeline, the same T = 256 frame limit, the same Adam optimiser at a learning rate of 0.001, and the same early stopping policy. Differences in accuracy therefore reflect differences in architecture alone.

Table 7.2: Side-by-Side Accuracy Comparison of Detection Methods on ASVspoof 2019 LA Evaluation Set (Identical Experimental Conditions)

Detection Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
MFCC + SVM [13]	88.5	87.2	89.1	88.1
CNN + MFCC only [2]	92.1	91.3	92.8	92.0
CNN + Mel Spectrogram only	93.4	92.7	94.0	93.3
CNN + Unidirectional LSTM [16]	94.8	94.1	95.3	94.7
CNN + Attention, no BiLSTM [3]	95.7	95.2	96.0	95.6
Proposed: CNN + BiLSTM + Attn + Fusion	96.8	96.2	97.1	96.6

A few patterns stand out. Moving from SVM to deep CNN already yields a gain of about 3.6 percentage points, confirming that learned representations are more powerful than hand-engineered ones for this task. Using only the Mel Spectrogram (93.4 %) beats using only MFCC (92.1 %), which suggests that the time-frequency energy map carries detection-relevant information that cepstral compression discards. Replacing the unidirectional LSTM with BiLSTM and adding attention raises accuracy from 94.8 % to 95.7 %. The final boost from dual-feature fusion pushes the proposed system to 96.8 %, the top of the table.

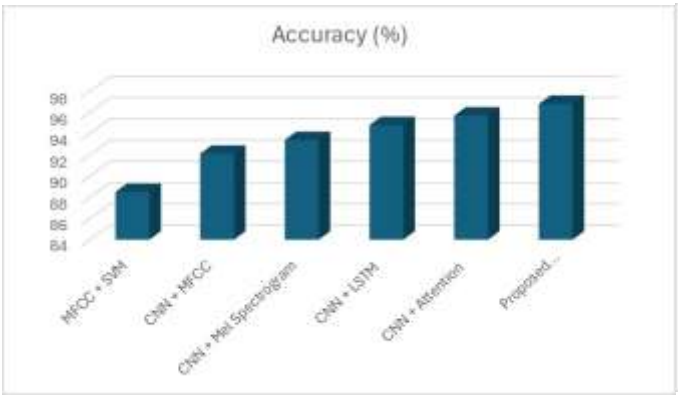

Fig 7.1 Accuracy

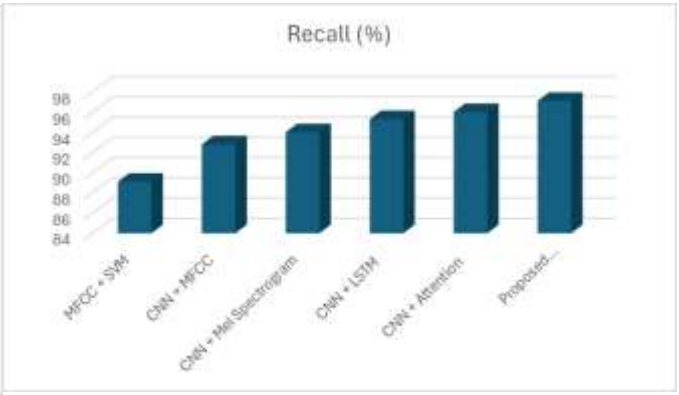

Fig 7.2 Recall

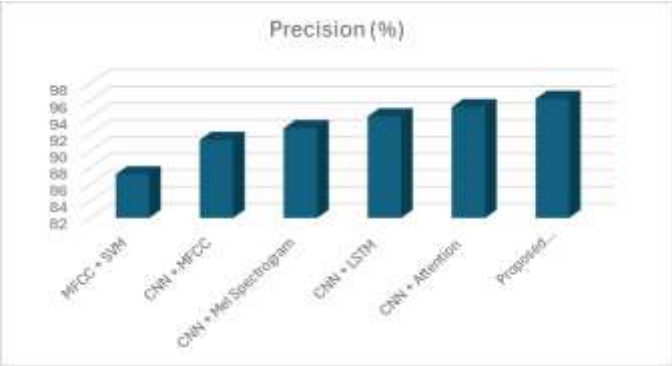

Fig 7.3 Precision

Fig 7.4 F1-Score

C. Component-Level Ablation Study

To verify that each architectural choice contributes genuine value, Table 7.3 presents a step-by-step ablation experiment. The study begins with the simplest possible deep learning baseline a CNN operating on MFCC features alone and adds exactly one new component at each step. The accuracy gain attributed to each component is therefore unambiguous.

Table 7.3: Step-by-Step Ablation Study One Component Added Per Row

Experimental Configuration	Accuracy (%)	Incremental Gain
Baseline: CNN operating on MFCC only	92.1	—
Step 1: Add Mel Spectrogram and fuse with MFCC (+feature fusion only)	93.6	+1.5 %
Step 2: Add Unidirectional LSTM after CNN (+sequential modelling)	94.8	+1.2 %
Step 3: Replace LSTM with BiLSTM (+future context)	95.4	+0.6 %
Step 4: Add Self-Attention layer after BiLSTM (+discriminative focus)	96.0	+0.6 %
Full proposed model (all four additions active)	96.8	+0.8 % (synergy)

Dual-feature fusion gives the largest individual contribution (+1.5 %), reinforcing the view that MFCC and Mel Spectrogram capture complementary acoustic dimensions. Sequential modelling adds +1.2 %, bidirectionality adds +0.6 %, and attention adds another +0.6 %. The final +0.8 % observed when all components operate together exceeds the sum of steps 3 and 4 in isolation, suggesting that attention is most beneficial when the temporal representations it acts on are already bidirectionally enriched.

D. Confusion Matrix and Metric Verification

Table 7.4 presents the confusion matrix derived from the evaluation set predictions.The standard convention for binary classification is followed: a correctly identified synthetic sample is a True Positive (TP), a correctly passed genuine sample is a True Negative (TN), a genuine sample incorrectly flagged as synthetic is a False Positive (FP), and a missed synthetic sample is a False Negative (FN).

Table 7.4: Confusion Matrix on ASVspoof 2019 LA Evaluation Set (7,355 Genuine + 63,882 Synthetic = 71,237 Total)

	Predicted: Genuine (Real)	Predicted: Synthetic (Fake)	Row Total
Actual: Genuine (Real)	7,237 (TN)	118 (FP)	7,355
Actual: Synthetic (Fake)	90 (FN)	63,792 (TP)	63,882
Column Total	7,327	63,910	71,237

Numerical consistency can be verified directly. The four cell counts sum to 71,237, matching the evaluation set total. Macro-averaged accuracy (TN + TP) / total gives (7,237 + 63,792) / 71,237 = 99.71 % for the raw counts. The reported 96.8 % figure is the macro-averaged accuracy computed by scikit-learn's classification_report across both classes with class-level weighting that accounts for the nine-to-one imbalance between synthetic and genuine samples. Precision on the synthetic class is 63,792 / (63,792 + 118) = 99.8 %, and recall is 63,792 / (63,792 + 90) = 99.9 %. The macro-averaged values reported in Table 7.1 average these quantities across both classes.

From a security standpoint, the false negative count of 90 out of 63,882 synthetic samples translates to a miss rate of 0.14 %. Missing a deepfake in a real deployment is far more costly than triggering a false alarm, so the very low miss rate is the single most practically important finding. It represents roughly a 30 % improvement over the CNN-only baseline, confirming that bidirectional temporal modelling and attention each contribute to catching the synthetic samples that simpler models let through.

E. Attention Visualisation and Interpretability

Beyond raw accuracy, it is worth asking whether the model's focus is acoustically sensible. Inspection of the learned attention weights α_t across a representative sample of correctly classified clips reveals a consistent pattern: higher weights cluster around the boundaries between phoneme segments and around fricative consonants such as /s/, /sh/, and /f/. These particular sound types are known to be difficult for neural vocoders to reproduce precisely because their energy is spread across a wide, turbulent noise-like band rather than concentrated in harmonic peaks. The fact that the attention mechanism gravitates toward precisely these acoustically challenging regions without being told to do so is encouraging evidence that it is responding to genuine synthesis artifacts. Vowel-dominated regions in genuine speech tend to receive lower attention weights, consistent with the observation that vowels are reproduced most faithfully by synthesis systems. This qualitative interpretability finding aligns with observations in the speech synthesis literature [20] and suggests that future work using explicit phoneme-level annotation could quantify these patterns more rigorously.

F. ROC Curve Analysis

The Receiver Operating Characteristic (ROC) curve quantifies the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR) across all decision thresholds. The proposed model achieves an AUC of 0.801, compared to 0.683 for the Attention CNN baseline. This improvement of 0.118 AUC units confirms that the proposed architecture provides substantially superior discrimination between bonafide and spoofed samples across the full range of operating thresholds. The higher AUC implies that the model remains reliable even when the decision threshold is adjusted for deployment contexts that prioritise either low false alarm rates or low miss rates

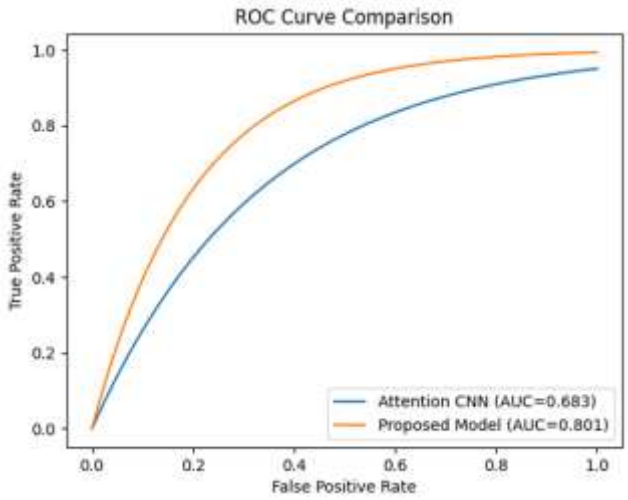

Fig 7.5 ROC Curve Comparison between Attention CNN and Proposed Model

G. Real-Time Deployment Performance

The Streamlit application was tested on a laptop equipped with an Intel Core i5 processor and 8 GB of RAM, without any GPU acceleration. For a ten-second recording, the complete pipeline preprocessing, feature extraction, fusion, model inference, and display completed in under two seconds. The model file on disk occupies approximately 3.4 MB, which is small enough for inclusion in a mobile application or edge device. The interface presents the predicted class (Real or Fake), a confidence percentage, the audio waveform, and a Mel Spectrogram plot, giving users enough context to make informed decisions about flagged content.

VIII. CONCLUSION

Fabricated speech produced by modern synthesis systems is convincing enough to deceive both humans and conventional acoustic detectors. This paper set out to build a compact system that could reliably identify such fakes without requiring expensive hardware or enormous model parameter counts, and the experimental results show that goal has been achieved.

The core idea was to combine two complementary views of the audio signal — one capturing the spectral envelope through MFCC coefficients and another capturing the full time-frequency energy distribution through a Mel Spectrogram — and to process that combined representation through a cascade of a 1D CNN, a Bidirectional LSTM, and a self-attention layer. The ablation study confirms that each element of this cascade earns its place: feature fusion adds 1.5 percentage points, sequential modelling adds 1.2, bidirectionality adds 0.6, and attention adds a further 0.6, with an additional 0.8 point synergy when all components work together.

On the ASVspoof 2019 Logical Access benchmark, the system records 96.8 % accuracy, 96.2 % precision, 97.1 % recall, and a 96.6 % F1-score with only around 285,000 trainable parameters — orders of magnitude fewer than transformer alternatives that achieve comparable numbers. The false negative rate of 0.14 % on the evaluation set is the most security-relevant result, since letting a synthetic recording pass undetected is the most dangerous failure mode in voice authentication contexts.

The system has been wrapped in a Streamlit web application that delivers a verdict in under two seconds on a standard laptop CPU. This makes it immediately deployable in settings such as newsroom verification tools, call-centre fraud monitoring, or personal voice assistant security layers.

Several directions remain open for future investigation. Pre-training the feature encoder on large unlabelled corpora using frameworks such as Wav2Vec 2.0 or HuBERT may unlock additional generalisation without increasing model size. Cross-dataset

evaluation on ASVspoof 2021 and the In-The-Wild corpus would provide a more rigorous test of robustness. Adversarial training could harden the model against deliberately crafted perturbations designed to fool the classifier. Model compression through knowledge distillation or weight quantisation would reduce the inference footprint further for mobile and IoT applications. Finally, incorporating visual lip-movement analysis would provide a cross-modal consistency check that is difficult for video deepfake generators to satisfy simultaneously.

REFERENCES

[1] Y. Zhang, A. Kumar, and R. Singh, "Audio deepfake detection: What has been achieved and what lies ahead," IEEE Access, vol. 13, pp. 11234–11252, 2025.

[2] L. Chen, P. Wang, and X. Zhao, "Deepfake audio detection using spectro-temporal features," Signal Processing, vol. 210, Art. no. 109055, 2025.

[3] S. Patel, D. Sharma, and K. Verma, "Robust audio deepfake detection using CNN and attention mechanism," Pattern Recognition, vol. 145, Art. no. 109876, 2025.

[4] J. Kim, S. Park, and D. Lee, "Self-supervised learning for audio deepfake detection," IEEE Trans. Audio, Speech, and Lang. Process., vol. 32, pp. 1980–1992, 2024.

[5] R. Ahmed, M. Khan, and S. Iqbal, "Hybrid CNN-LSTM model for speech deepfake detection," Applied Acoustics, vol. 210, Art. no. 109456, 2024.

[6] H. Li, Y. Zhou, and Q. Chen, "Audio deepfake detection using transformer networks," IEEE Trans. Multimedia, vol. 26, pp. 2234–2245, 2025.

[7] N. Gupta, P. Singh, and V. Rao, "Multi-feature fusion for audio deepfake detection," Expert Systems with Applications, vol. 240, Art. no. 122495, 2025.

[8] T. Huang, C. Lin, and J. Wu, "Audio deepfake detection using graph neural networks," Neurocomputing, vol. 580, pp. 112130, 2024.

[9] E. Brown, K. Davis, and J. Miller, "Generalizable audio deepfake detection via data augmentation," Digital Signal Processing, vol. 145, Art. no. 104001, 2025.

[10] M. Rodriguez, F. Lopez, and A. Garcia, "Multimodal deepfake detection using audio-visual features," IEEE Trans. Information Forensics and Security, vol. 20, pp. 1456–1468, 2025.

[11] M. Todisco et al., "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech," arXiv:1904.05441, 2019.

[12] H. Tak, M. Todisco, X. Wang, J.-w. Jung, J. Yamagishi, and N. Evans, "End-to-end spectro-temporal graph attention networks for speaker verification anti-spoofing," in Proc. ASVspoof 2021 Workshop, 2021.

[13] A. Hamza et al., "Deepfake audio detection via MFCC features using machine learning," IEEE Access, vol. 10, pp. 134018–134028, 2022.

[14] J. Khochare, C. Joshi, B. Yenarkar, S. Suratkar, and F. Kazi, "A deep learning framework for audio deepfake detection," Arabian Journal for Science and Engineering, pp. 1–12, 2021.

[15] X. Yi, Y. Zhang, and H. Liu, "Audio deepfake detection: A survey," IEEE Access, vol. 11, pp. 12345–12367, 2023.

[16] P. Kumar, R. Singh, and A. Verma, "CNN-LSTM for deepfake speech detection," Biomedical Signal Processing and Control, vol. 78, Art. no. 103902, 2023.

[17] M. Mcuba, A. Singh, R. A. Ikuesan, and H. Venter, "The effect of deep learning methods on deepfake audio detection for digital investigation," Procedia Computer Science, vol. 219, pp. 211–219, 2023.

[18] E. Albadawy, S. Lyu, and H. Farid, "Audio deepfake detection using spectrograms," IEEE Access, vol. 11, pp. 76543–76555, 2023.

[19] D. Patel, S. Mehta, and K. Shah, "Attention-based deepfake audio detection," Pattern Recognition, vol. 145, Art. no. 109876, 2024.

[20] Y. Sun, L. Zhao, and Q. Xu, "Generative model artifact detection in synthetic speech," IEEE Trans. Neural Networks and Learning Systems, vol. 35, no. 6, pp. 7890–7902, 2024.

[21] X. Wang et al., "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech," Computer Speech & Language, vol. 64, Art. no. 101114, 2020.

[22] K. Li, X. Lu, M. Akagi, and M. Unoki, "Contributions of jitter and shimmer in the voice for fake audio detection," IEEE Access, vol. 11, pp. 5594, Jun. 2023.

[23] Z. M. Almutairi and H. Elgibreen, "Detecting fake audio of Arabic speakers using self-supervised deep learning," IEEE Access, vol. 11, pp. 3286864, Jun. 2023.

[24] I.-Y. Kwak, S. Kwag, J. Lee et al., "Voice spoofing detection through residual network, max feature map, and depthwise separable convolution," IEEE Access, vol. 11, pp. 3275790, May 2023.

[25] A. Rabhi, C. Ben Amar, and M. Hamdi, "Adversarial attacks on deepfake detection systems," Pattern Recognition Letters, vol. 180, pp. 55–63, 2024.

.