KEC Conference 2025

ISSN: 3021-940X (Print) DOI: https://doi.org/10.3126/injet.v2i2.78619

Deep Fake Audio Detection Using a Hybrid CNN-BiLSTM Model with Attention Mechanism

Shubham Chapagain[1], Er. Bishal Thapa[2], Shubham Man Singh Baidhya[3, *], Smriti B.K[4], Shrawan Thapa[5]

[1]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, shubhamchapagain01@gmail.com*

[2]*Senior Lecturer, Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, bishalthapa@kec.edu.np*

[3]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, shubhambaidhya7@gmail.com*

[4]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, smriti.bk201@gmail.com*

[5]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, thapashrawan2@gmail.com*

Abstract

The increasing sophistication of deepfake technologies has raised significant concerns regarding the authenticity of audio files. To address this challenge, we propose a deepfake audio detection system that employs a CNN-BiLSTM hybrid model for identifying synthetic speech. The system processes audio files by converting them into Mel-Spectrograms, which effectively capture the unique spectral features distinguishing real voices from fake ones. The processed data is then fed into the CNN-BiLSTM model, which leverages the strengths of Convolutional Neural Networks (CNNs) for spatial pattern recognition and Bidirectional Long Short-Term Memory Networks (BiLSTMs) for capturing long-term dependencies in the temporal sequence of the audio data. The model, trained on dataset, achieves an accuracy of 95%, effectively detecting subtle irregularities indicative of deepfake audio. The system provides users with a comprehensive analysis, including a confidence score and detailed insights into the authenticity of the audio, offering an effective tool for distinguishing real from fake audio. Our system combines cutting-edge machine learning technology with a user-friendly interface, making it both highly effective and accessible for practical applications in deepfake detection. Unlike existing systems that rely solely on either CNN or RNN architectures, our approach integrates both to enhance detection accuracy, particularly for complex and subtle manipulations. Additionally, we introduce a confidence scoring mechanism and insight analysis that provides users with transparent reasoning behind each detection.

Keywords: Deepfake Audio Detection, CNN-BiLSTM Hybrid Model, Synthetic Speech, Audio Authenticity.

1. Introduction

Deepfakes involve the generation of audio in which authentic human features are replaced by synthetic ones, posing significant risks of misinformation and digital deception. Misinformation has long existed on the internet, but the rise of AI-generated deepfakes has made it significantly more alarming(Khanjani, Watson, & Janeja, 2023). Deepfakes primarily consist of three types of contents: images, videos, and audio; in some cases, they also include text-based artifacts. Among these types, our research primarily focuses on the creation and detection methodology of deepfake audio. With the recent advancement in artificial intelligence, there has been research on a wide range of spectra for detecting these deepfake contents. We observe that deepfake research has primarily focused on images and videos, leading to the development of various state-of-the-art deepfake detection algorithms. This leads us to the fact that not much has been investigated in the sector of deepfake audios (Iqbal, Javed, Jalil, & Al-Karaki, 2022). Detecting such manipulated audio requires extracting both spatial and temporal features from speech signals to capture the intricate patterns of authenticity and deception. To address this gap, the CNN-BiLSTM hybrid model has emerged as a robust deep learning architecture for detecting audio deepfakes. This model leverages the strengths of Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (BiLSTM) networks to achieve robust

**Corresponding Author*

and accurate classification. CNNs are adept at extracting local spatial features from audio spectrograms, such as frequency patterns and transitions. At the same time, BiLSTMs process these features over time, capturing the contextual temporal dependencies in both forward and backward directions. By integrating these architectures, the model learns both short-term signal irregularities and long-range voice modulations characteristic of synthetic audio. The hybrid structure enhances the system's capability to differentiate between real and fake audio with high precision. The CNN-BiLSTM framework not only outperforms traditional classifiers but also offers a scalable and reliable solution for detecting deepfake audio in real-world applications. To the best of our knowledge, our work is among the few to implement and evaluate a CNN-BiLSTM hybrid architecture specifically for deepfake audio detection. While previous studies have primarily focused on either CNN or RNN-based models, our integration of both allows for richer feature learning by combining spatial and temporal cues. Furthermore, our system introduces an interpretable output layer that includes a confidence score and analysis insights for end-users, making it both technically robust and practically useful.

2. Related Works

In a paper (Iqbal, Javed, Jalil, & Al-Karaki, 2022), the authors tackled the critical issue of distinguishing real audio from synthetic deepfake audio using advanced machine learning techniques. The study leveraged the comprehensive Fake or Real (FoR) dataset, which comprises over 195,000 audio samples produced through sophisticated text-to-speech (TTS) models. The dataset is categorized into three subsets: ‘for-norm', 'for-2sec', and 'for-rerec', each designed to assess different aspects of audio authenticity. The 'for-norm' set includes 53,868 unique audio clips post-duplication removal, while 'for-2sec' and 'for-rerec' contain time-trimmed samples designed to highlight key spectral characteristics at a 44,100 Hz sampling rate. The proposed approach focuses on effective feature engineering and model selection for optimal detection performance. Pre-processing involves cleaning, normalizing, and transforming audio from the time domain to frequency representations, then extracting key features like Mel power spectrograms. After dividing the dataset into an 80:20 train-test split, six machine-learning models are trained to classify audio as real or fake. The results reveal a notable 26% performance improvement over baseline techniques, with the model achieving a testing accuracy of 65.617%, underscoring the effectiveness of combining meticulous feature selection with robust machine learning algorithms to enhance the reliability of deepfake audio detection across various real-world applications.

In a paper (Kawa & Syga, 2022), the authors introduce the Attack Agnostic Dataset to enhance the generalization and stability of deepfake audio detection models. By combining FakeAVCeleb, WaveFake, and ASVspoof 2019 LA, the dataset includes 31,083 actual and 222,035 fake audio samples from 27 deepfake generation techniques. Various machine learning models, including LCNN, XceptionNet, and RawNet2, are evaluated across three dataset folds to assess their robustness. LCNN, using LFCC and Mel-spectrogram features, outperforms others, reducing error rates by up to 5%. The study highlights Mel-based feature limitations and applies pre-processing techniques for consistency. Results show that models trained on this dataset perform better against unseen attacks, making it a valuable resource for improving deepfake audio detection in real-world applications.

The Fake-or-Real (FoR) database contains over 195,000 human and synthetic speech samples from sources like Deep Voice 3 and Google Wavenet TTS (Hamza, et al., 2022). It is divided into four subsets: For-original (raw recordings), For-norm (standardized versions with modified sampling rate or volume), For-2sec (shortened For-norm samples), and For-rerec (re-recorded For-2sec samples simulating voice transmission). The study applies pre-processing, normalization, and feature extraction using MFCC and other acoustic features. Multiple machine learning models are tested, with SVM achieving the highest accuracy of 97.57% for For-2sec and 98.83% for For-rerec. Different models, including MLP and Random Forest, also performed well, while Gradient Boosting led the For-norm dataset with 92.63% accuracy. The results highlight SVM’s effectiveness in deepfake audio detection across various dataset formats.

In a paper (Lim, Chae, & Lee, 2022), the authors utilize two core datasets: ASVspoof 2019 Logical Access and LJSpeech. ASVspoof contains 2,580 actual speech samples from 107 speakers and 22,800 synthetic

samples generated by 19 synthesizers, categorized as 'human voice' and 'deepfake voice.' LJSpeech comprises 13,100 audio files with transcripts, with deepfake versions created using Tacotron, an attention-based voice synthesizer. The research focuses on making deepfake audio detection interpretable through explainable AI (XAI) techniques. Spectrogram-based feature extraction ensures interpretability with minimal pre-processing. Three CNN-based models: CNN, CNN-LSTM, and CNN-LSTM-permuted—are evaluated using XAI methods like Deep Taylor decomposition, integrated gradients, and layer-wise relevance propagation (LRP). The ASVspoof dataset is further divided based on synthesis methods, including single and multiple TTS or voice conversion models. While performance varies under different conditions, the goal is interpretability rather than optimization. Among the models, CNN-LSTM-permuted performs best, achieving up to 99.97% accuracy, while CNN-LSTM also performs well, particularly in ASV Single TTS (98.91%) and LJSpeech (99.92%). The basic CNN model is effective but shows lower accuracy, ranging from 77.78% to 98.88%.

3. Methodology

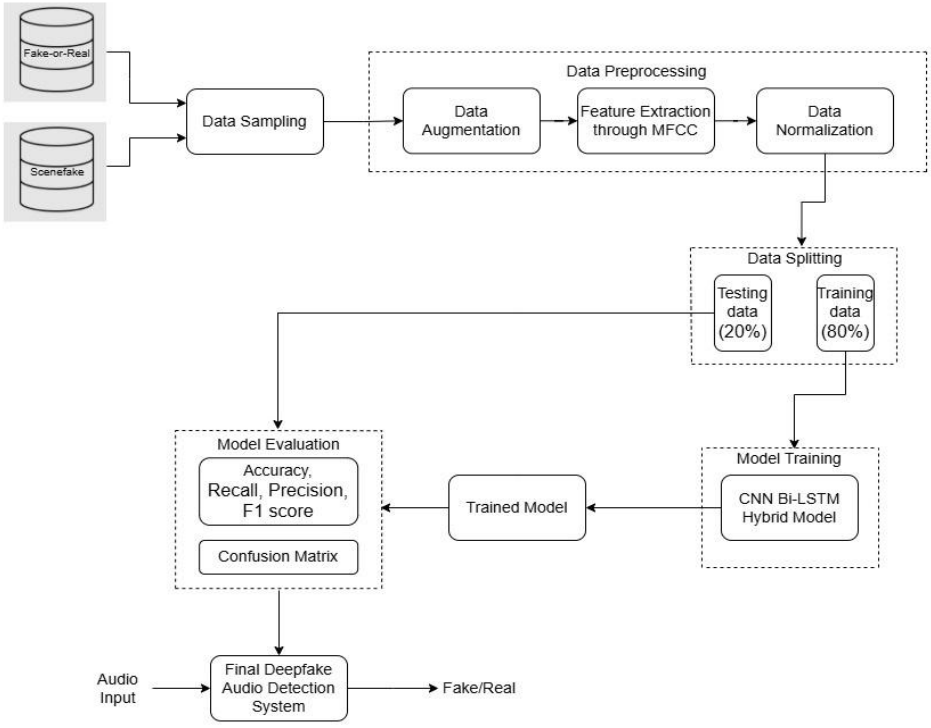

Figure 1. Working Mechanism of Deepfake Audio Detection using CNN BiLSTM

3.1. Dataset Overview

The dataset used in the study includes two primary sources: the SceneFake dataset and the Fake-or-Real (FoR) dataset, both from Kaggle (Abdeldayem M. , n.d.). SceneFake aids deepfake audio detection by offering a mix of actual human speech and manipulated audio generated through techniques like text-to-speech synthesis and voice cloning, comprising 12,848 files for development, 32,734 for evaluation, and 13,225 for training. The FoR dataset is structured into four subsets: For-2sec (1,088 testing, 13,956 training, 2,826 validation samples), For-norm (4,634 testing, 53,800 training, 10,798 validation samples), For-original (4,634 testing, 53,800 training, 10,800 validation samples), and For-rerec (816 testing, 10,208 training, 2,244 validation samples). Each subset serves a distinct purpose in evaluating model performance under varied conditions. For-2sec contains short, 2-second audio clips, challenging models to make accurate predictions with limited contextual information. For-norm consists of normalized audio samples, providing a clean and consistent baseline to train models without the interference of extraneous noise. For-original includes raw,

unprocessed audio clips, enabling assessment of model generalizability to natural, real-world audio. For-rerec features re-recorded samples, created by playing and re-capturing audio to simulate real-world distortions such as room acoustics and microphone artifacts. By combining these datasets, the study creates a diverse and balanced collection of authentic and synthetic speech samples, enhancing deep learning model's ability to distinguish between real and fake audio for more effective deepfake detection.

3.2. Dataset Pre-processing

3.2.1. Data Sampling

After merging the SceneFake and FoR datasets, the audios were categorized into fake and real, totaling 132,363 fake and 96,165 actual samples. Undersampling was applied to balance the dataset, using Scikit-learn's resample function, which reduced both classes to 96,165 samples without duplication. The data was then shuffled and saved as a CSV file for training and evaluation, ensuring equal class distribution and preventing model bias.

3.2.2. Data Augmentation

To enhance the robustness of our deepfake Audio Detection system, we applied data augmentation techniques to improve model generalization and prevent overfitting. These techniques included noise addition, where random noise (0.002–0.01 Gaussian distribution) was added to simulate real-world conditions; time stretching, which adjusted speaking speed (0.8–1.2x) without altering pitch; and pitch shifting, which modified the audio by ±3 semitones to mimic voice modulation. These transformations created varied audio samples, helping the model detect deepfake audio across different noise levels, speech speeds, and pitch variations, ultimately improving its accuracy and reliability.

3.2.3. Noise Reduction

In our deepfake Audio Detection system, we have applied noise reduction to enhance the quality of the audio samples and reduce unwanted background noise, which could have interfered with the model's ability to detect deepfake audio. By using the nr. reduce noise function, we have ensured that the audio samples fed into the model are of higher quality, enabling more accurate feature extraction and improving the overall performance of our Deepfake Audio Detection System.

3.2.4. Feature Extraction

After applying data augmentation and noise reduction, we extracted key audio features to enhance deepfake detection. These include MFCCs, MFCCs are a way to represent the short-term power spectrum of sound in a form that mimics how humans perceive audio. They are especially useful for tasks like speech recognition and deepfake audio detection, which capture spectral characteristics crucial for distinguishing natural and synthetic speech, and spectral centroid, which identifies differences in timbral texture. Spectral bandwidth helped differentiate genuine and fake voices based on frequency distribution, while the zero-crossing rate detected abrupt signal changes indicative of synthetic speech. Chroma features analyzed pitch variations, revealing inconsistencies in manipulated audio. In total, 44 features were extracted and combined into a single feature vector, providing a comprehensive representation of temporal and spectral properties. This rich feature set improved the model's ability to classify real and Deepfake audio accurately.

3.2.5. Data Scaling

Standard Scaling (Z-score normalization) was applied to rescale features to a mean of 0 and a standard deviation of 1, preventing features with more extensive ranges from dominating.

$$z = \frac{x - \mu}{\sigma} \qquad \text{(Equation 1)}$$

where x= original features, μ = mean of the feature, and σ = standard deviation.

The pre-processing pipeline involved loading data from .npy files in Google Colab, padding variable-length sequences to a uniform shape, and applying Scikit-learn's StandardScaler to standardize the features. The

scaler was saved using Python's pickle module, and the processed data was stored as .npy files. This approach ensured normalized input data, improving model performance and convergence speed.

3.3. Data Splitting

After scaling the dataset, we split it into training and testing sets to evaluate the model's performance. Typically, 80% of the data has been used for training, while 20% has been reserved for testing.

3.4. Model Description

3.4.1. CNN Bi-LSTM Hybrid Model

The CNN-Bi-LSTM hybrid model used for Deepfake Audio Detection is designed to extract key features from audio signals and classify them as real or fake. It combines convolutional layers for local feature extraction with Bidirectional LSTMs to capture temporal dependencies, enhancing the model's ability to analyze speech characteristics effectively. Below is a detailed explanation of each layer and its function:

1. **Input Layer (Input Layer)** – This layer receives the pre-processed audio features, with each sample represented by 44 features. It ensures the model can handle variable-length sequences using an input shape of (None, 44), serving as the starting point for feature extraction and classification.
2. **Reshape Layer (Reshape)** – Converts the input into a 3D array of shapes (None, 44, 1), where each of the 44 features is treated as a separate channel. This transformation prepares the data for the convolutional layer. It ensures compatibility with the subsequent processing layers.
3. **1D Convolutional Layer (Conv1D)** – Applies 64 filters that slide over the feature sequence, detecting important patterns. Each filter learns specific characteristics of the audio, such as frequency variations, helping in capturing local feature representations effectively.

$$y[t] = \sum_{j=0}^{k-1} x[t-j] \cdot w[j] \qquad \text{(Equation 2)}$$

   where $x[t-j]$ = input sequence at time *t,* w[j] =filter weights, k = kernel size, y[t]=output of the convolution layer.
4. **Batch Normalization Layer (Batch Normalization)** – Normalizes the output of Conv1D to maintain a stable distribution of activations. These speeds up training and reduces the risk of overfitting. It ensures that data remains well-scaled before moving to the following layers.
5. **MaxPooling Layer (MaxPooling1D)** – Reduces the feature dimensions by taking the maximum value from each pool of size 2. This operation retains the most important features while decreasing computational load. It helps down-sample the feature maps while preserving essential information.
6. **Bidirectional LSTM Layer (Bidirectional LSTM)** – Processes the audio sequence in both forward and backward directions, improving the model's understanding of temporal dependencies, which is crucial for speech analysis since context plays a significant role in differentiating real and fake speech. It outputs a feature map of size (None, 22, 128), capturing time-dependent characteristics.
7. **Attention Weights (Dense)** – Enhances the model's focus on essential sequence parts by assigning higher weights to relevant time steps, which ensures that the model learns from audio portions rather than treating all parts equally. It allows the model to prioritize significant speech features.
8. **Layer Normalization (Layer Normalization)** – Normalizes the output of the attention layer to stabilize training and improve convergence, preventing data distribution variations from affecting model learning. It ensures that the network maintains consistency in feature scales.
9. **Multiply Layer (Multiply)** – Performs element-wise multiplication between the Bidirectional LSTM output and attention weights, combining sequence features with learned attention, emphasizing the most crucial speech characteristics. It ensures that essential time steps contribute more to the final decision.
10. **Global MaxPooling Layer (GlobalMaxPooling1D)** – Reduces the sequence dimension to a single 128-dimensional feature vector by selecting the maximum value across all time steps, helping in compressing the learned features into a compact form. It extracts the most significant aspects of the audio features.

11. **Dense Layer (Dense, 128 neurons)** – Processes the pooled features to create a refined representation before classification. It helps in capturing higher-level patterns in the extracted features. This layer enhances feature discrimination between real and fake audio.
12. **Dropout Layer (Dropout)** – Randomly deactivates some neurons during training to prevent overfitting, ensuring better generalization by making the model less reliant on specific neurons. It reduces the risk of memorizing training data and improves performance on unseen samples.
13. **Final Dense Layer (Dense, 1 neuron)** – Outputs a single probability score between 0 and 1, where 0 represents real audio and 1 represents fake audio, which serves as the final classification output of the model. The model predicts Deepfake likelihood based on learned representations.

Table 1. CNN Bi-LSTM Architecture representing 13 layers

Layer (Type)	Output Shape	Param#	Connected to
input_layer_2 (Input-Layer)	(None,44)	-	-
reshape_2(Reshape)	(None,44,1)	0	input_layer_2[0][0]
conv1d_2(Conv1D)	(None,44,64)	256	reshape_2[0][0]
batch_normalization_2 (Batch Normalization)	(None,22,64)	256	conv1d_2[0][0]
max_pooling1d_2 (MaxPooling1D)	(None,22,64)	0	batch_normalization_2[0][0]
bidirectional_2 (Bidirectional)	(None,22,128)	66,048	max_pooling1d_2[0][0]
attention_weights (Dense)	(None,22,1)	16,512	bidirectional_2[0][0]
layer_normalizatin_1 (LayerNormalization)	(None,22,1)	256	attention_weights [0][0]
multiply_2(Multiply)	(None,22,128)	0	bidirectional2[0][0], layernormalization1[0][0]
globalmaxpooling1d1 (GlobalMaxPool ing1D)	(None,128)	0	multiply2[0][0]
dense2(Dense)	(None,128)	16,512	globalmaxpooling1d1[0][0]
dropout1(Dropout)	(None,128)	0	dense2[0][0]
dense3(Dense)	(None,1)	129	dropout1[0][0]

3.5. Gradient Boosting

In a Deepfake Audio Detection System, Gradient Boosting is a machine learning method used to tell apart real and fake audio recordings. The process starts by turning audio signals into useful features, such as spectrograms and MFCCs, which help describe how the sound changes over time and frequency. These features are like fingerprints of the audio and can help the model spot strange patterns that are common in fake audio. Once these features are collected, they are given to the Gradient Boosting model, which works by combining many small decision trees. Each tree tries to fix the mistakes made by the one before it, slowly improving the overall prediction.

As more trees are added, the model becomes better at catching the tricky signs of Deepfake audio. In the end, the system gives a result showing whether it thinks the audio is real or fake. This method is useful because it can handle complex patterns in the data, focuses on examples that are hard to classify, and avoids overfitting if set up correctly. Gradient Boosting also allows us to see which features were most important in making the decision, which helps in understanding how the model works. Overall, it's a strong choice for detecting fake audio because it's accurate and smart at learning from the data.

3.6. XGBoost

In a Deepfake Audio Detection System, XGBoost is a popular and efficient machine learning method used to classify audio as real or fake. Just like Gradient Boosting, the process starts by extracting features from the

audio, such as spectrograms, MFCCs, pitch, and other voice-related signals. These features help describe how the audio behaves and are useful in spotting unnatural patterns. XGBoost takes these features and builds many small decision trees, one at a time. Each new tree focuses on fixing the errors made by the trees before it, which helps the model learn from its mistakes and improve over time.

What makes XGBoost special is that it's faster and more accurate than many other boosting methods because of its advanced techniques like regularization (to prevent overfitting), parallel processing, and handling missing values. It also uses a smart way of choosing which tree to grow and how, making it very efficient with large or complex datasets. At the end of training, the XGBoost model can predict whether a new audio file is real or fake based on the patterns it has learned. It also provides feature importance, helping researchers understand which audio characteristics were most useful in detecting Deepfakes.

3.7. Model Interpretation and X-AI

3.7.1. SHAP Summary Plot: Feature Importance

Deep learning models like CNN-BiLSTM have been effective in Deepfake audio detection but often function as black boxes, making their decisions difficult to interpret. Explainable AI (XAI) techniques, such as SHAP and attention visualization, have been applied to better understand the model's decision-making process. Before applying explainability methods, key audio features like MFCCs, spectral centroid, spectral bandwidth, zero-crossing rate, and chroma features were extracted using Librosa. SHAP assigns importance scores to these features, showing their influence on predictions. A SHAP summary plot highlights key contributors, such as chroma1 and specific MFCC values, which significantly impacting classification. Positive SHAP values increase the likelihood of a Deepfake classification, while negative values suggest real audio.

Additionally, feature value representation using color scales shows that high values (red) of chroma1 are linked to Deepfake predictions, while low values (blue) push the model towards actual speech. Features like mfcc7 and mfcc5 have a broad range of SHAP values, indicating varied influence, whereas spectral bandwidth and zero-crossing rate have a more consistent effect across samples. This analysis helps us understand how feature intensity and variability contribute to the model's predictions as shown in figure 3.

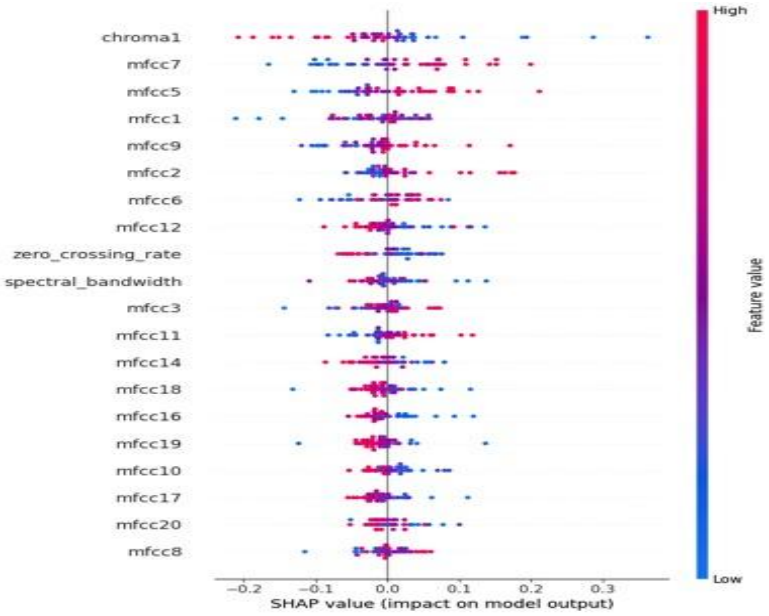

Figure 2. Visualizing SHAP summary plot

4. SHAP Waterfall Plot: Single Sample Analysis

The SHAP Waterfall Plot provides a detailed breakdown of how each feature influences the final prediction for a single test sample as shown in figure 4. It starts with the baseline prediction, representing the model's

average output across all data points (0.453 in this case), serving as the reference point. From there, individual feature contributions are added or subtracted. Features with negative SHAP values (shown in blue) reduce the prediction, pushing it closer to 0, indicating "real audio" (non-deepfake). For example, features like mfcc1 (-0.08), mfcc37 (-0.06), and mfcc39 (-0.06) strongly support the actual audio classification. Conversely, features with positive SHAP values (shown in red) increase the prediction, making the model more inclined to predict "deepfake." Examples include mfcc20 (+0.05) and mfcc19 (+0.04), slightly raising the prediction score. The most significant impact comes from 36 other features, which decrease the prediction by -0.25, making them the most influential in classifying this instance as real audio. After considering all contributions, the final prediction score is 0.0, classifying the instance as real audio.

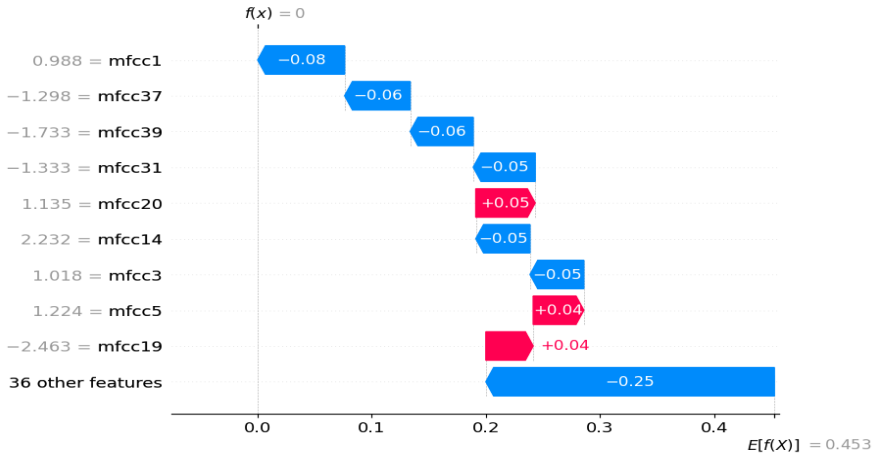

Figure 3. Visualizing SHAP for a Single Prediction

4.1. Experimental Results

4.1.1. CNN Bi-LSTM Hybrid

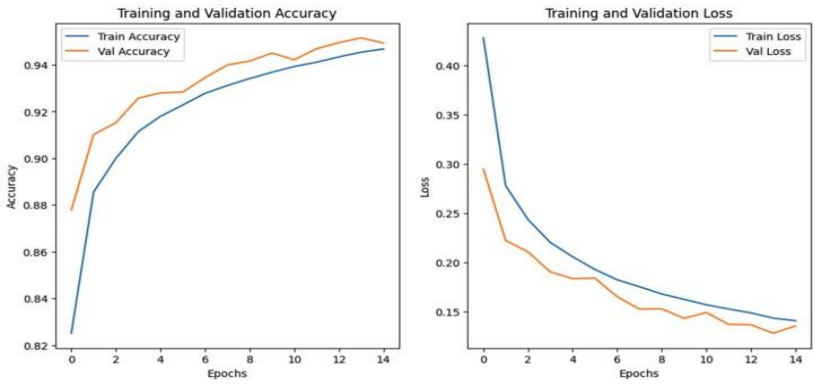

Figure 4. Loss and Accuracy (CNN Bi-LSTM)

Testing Accuracy: 0.95

4.2. Model Comparison

Table 2. Model Comparison of CNN BiLSTM with Gradient Boosting and XGBoost

Model	accuracy	precision	recall	f1-score	Support
CNN Bi-LSTM	0.95	0.95	0.94	0.95	38,466
Gradient Boosting	0.87	0.89	0.84	0.87	38,466
XGBoost	0.95	0.94	0.94	0.94	38466

The performance evaluation table compares three machine learning models- CNN Bi-LSTM, Gradient Boosting, and XGBoost- based on key classification metrics: accuracy, precision, recall, F1-score, and support, evaluated on a test dataset of 38,466 samples. The CNN Bi-LSTM model stands out as the best-performing model with an accuracy of 95%, meaning it correctly classifies 95% of the total samples. Its precision of 0.95 indicates that out of all the instances predicted as positive (e.g., fake audio, if applied in deepfake detection), 95% were correct. The recall of 0.94 further shows that it can correctly detect 94% of all actual positive cases in the dataset. As a result, it achieves a strong F1-score of 0.95, which is the harmonic mean of precision and recall, indicating an excellent balance between false positives and false negatives, which demonstrates that CNN Bi-LSTM is not only highly accurate but also robust in detecting the target class effectively as shown in table 1.

Compared to other models, Gradient Boosting performed worse across all metrics, achieving an accuracy of 87%, a precision of 0.89, a recall of 0.84, and an F1-score of 0.87. Although the model still performs adequately, the lower recall indicates that it misses a significant number of actual positive samples, leading to more false negatives. As a result, the model's reliability is reduced in scenarios where accurately identifying all positive samples is critical. The most concerning results come from the XGBoost model. While it shows high precision (0.94) and accuracy (93%), it suffers from an extremely low recall of 0.02, which means it only correctly identifies 2% of all actual positive instances. This suggests that the model is biased towards predicting negative classes, resulting in it missing nearly all actual positives. Despite having a reported F1-score of 0.94, which appears misleading due to the high precision, the poor recall makes the model ineffective in real-world applications where detecting all positive instances is critical. This discrepancy could arise due to class imbalance, incorrect threshold settings, or inadequate model tuning.

5. Discussion

The CNN Bi-LSTM model significantly outperforms both XGBoost and Gradient Boosting models in all significant evaluation metrics, establishing itself as the most robust and effective architecture for deepfake audio detection. With an accuracy of 95%, precision of 0.95, recall of 0.94, and F1-score of 0.95, the CNN Bi-LSTM demonstrates a highly balanced and reliable classification performance. These metrics show that the model not only accurately predicts most classes but also strikes an optimal balance between false positives and false negatives, which is crucial in applications where detecting fake audio is essential.

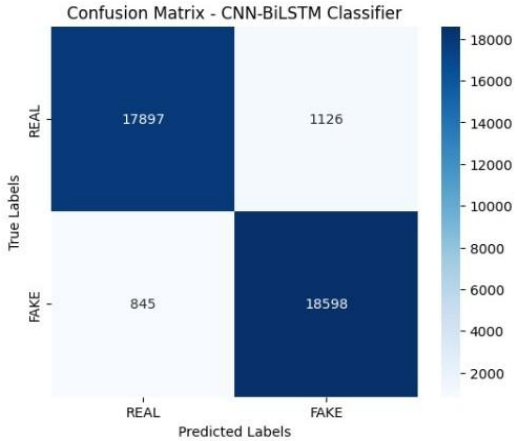

Figure 2. Confusion Matrix

The confusion matrix for our CNN-BiLSTM classifier in deepfake audio detection illustrates the model's performance in distinguishing actual and fake audio samples. The matrix shows that out of 19,023 real audio samples, 17,897 were correctly classified as accurate, while 1,126 were misclassified as fake. Similarly, out of 19,443 fake audio samples, 18,598 were correctly classified as fake, while 845 were incorrectly labelled as accurate. The high number of correctly classified samples in both categories indicates that the model performs well, with relatively low misclassification rates. The small number of false positives (real audio

misclassified as fake) and false negatives (fake audio misclassified as actual) suggests strong generalization and robustness in detecting deepfake audio.

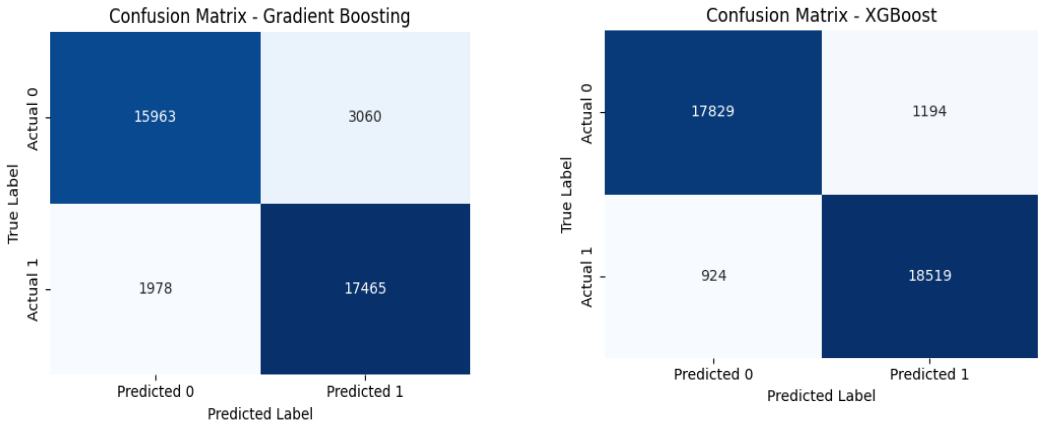

Figure 3. Confusion Matrix of Gradient Boosting and XGBoost

6. Conclusion and Future Enhancement

The rapid advancement of deepfake audio technologies poses a significant threat to various sectors, including media integrity, cybersecurity, and financial security. This study addresses these challenges by developing a robust deepfake audio detection system using a CNN-BiLSTM hybrid model, which effectively classifies audio as actual or synthetic with a high accuracy of 95%. By leveraging critical audio features such as MFCCs, spectral centroid, spectral bandwidth, zero-crossing rate, and chroma features, the model identifies subtle anomalies that distinguish deepfake audio from actual human speech. The performance evaluation, including precision, recall, F1-score, and confusion matrix analysis, demonstrates the system's strong generalization ability and effectiveness in real-world applications, with minimal misclassification rates. A key strength of this approach is its integration of Explainable AI (XAI) techniques, particularly SHAP (SHapley Additive exPlanations), which enhances the interpretability and transparency of the model's decision-making process. This is crucial in building trust and reliability for users, especially in sensitive fields such as journalism, law enforcement, and finance, where the consequences of misclassification can be significant.

Furthermore, rigorous dataset processing, including noise reduction, data augmentation, and feature extraction, enhances the model's resilience against evolving deepfake techniques. The systematic training process and balanced dataset ensure the model maintains high performance even when exposed to new and complex deepfake variations. Despite the impressive results achieved in this study, deepfake audio detection remains a continuously evolving challenge as adversarial techniques improve. Therefore, ongoing research, model refinement, and adaptation to emerging threats are essential to ensure the continued effectiveness of detection systems. By providing a strong foundation in deepfake detection, this study contributes significantly to the growing field of AI-driven security solutions, reinforcing the importance of technological advancements in 45 safeguarding digital integrity.

Acknowledgement

We would like to express our gratitude to everyone who helped us complete this project. First and foremost, we would like to acknowledge the crucial role of our teachers from the Department of Electronics and Computer Engineering for their guidance, support, and feedback throughout the project. Next, we would like to give our gratitude to our classmates for providing constructive feedback and engaging in thought-provoking discussions regarding our project. We also extend our thanks to all the lecturers in our department for guiding us from the beginning to the end of our project. A special thanks to our supervisor, Er. Bishal Thapa, for his invaluable mentorship and direction. Finally, we offer our deepest gratitude to our family and friends for their love, encouragement, and support throughout our academic journey.

References

Hamza, A., Javed, A. R., Iqbal, F., Kryvinska, N., Almadhor, A. S., Jalil, Z., & Borghol, R. (2022). Deepfake audio detection via MFCC features using machine learning. *IEEE Access*, 134018--134028. doi:https://doi.org/10.1109/ACCESS.2022.3231480

Iqbal, F. a., Javed, A. R., Jalil, Z., & Al-Karaki, J. N. (2022). Deepfake Audio Detection Via Feature Engineering And Machine Learning., (pp. 1--12). Retrieved March 10, 2025, from https://ceur-ws.org/Vol-3318/paper4.pdf

Kawa, P. a., & Syga, P. (2022). Attack agnostic dataset: Towards generalization and stabilization of audio deepfake detection. *arXiv preprint arXiv:2206.13979*. doi:https://doi.org/10.48550/arXiv.2206.13979

Khanjani, Z., Watson, G., & Janeja, V. P. (2023). Audio deepfakes: A survey. *Frontiers in Big Data*, *5*, 1001063. doi:https://doi.org/10.3389/fdata.2022.1001063

Lim, S.-Y., Chae, D.-K., & Lee, S.-C. (2022). Detecting deepfake voice using explainable deep learning techniques. *Applied Sciences*, 3926. doi:https://doi.org/10.3390/app12083926