ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

AI-Based Deepfake Voice Detection Using MFCC Features and Random Forest Classification

Ashish S. Kangane[1], Animesh S. Bhople [2], Waman R. Parulekar[3]

[1]Department of MCA, Finolex Academy of Management and Technology, Ratnagiri, India

[2]Department of MCA, Finolex Academy of Management and Technology, Ratnagiri, India

[3]Department of MCA, Finolex Academy of Management and Technology, Ratnagiri, India

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150500121

Received: 28 May 2026; Accepted: 02 June 2026; Published: 06 June 2026

ABSTRACT

The rapid proliferation of AI-generated audio poses a serious threat to digital forensics, voice-based authentication, and information integrity. This paper presents a deepfake voice detection system that combines Mel-Frequency Cepstral Coefficient (MFCC) feature extraction with a Random Forest ensemble classifier to distinguish real human speech from synthetically generated audio. The proposed system processes input audio files in WAV or MP3 format, extracts 40 MFCC coefficients as the feature representation, and classifies each sample as real or fake through a trained Random Forest model. The complete pipeline is deployed as a Flask-based web application, enabling browser-based access without requiring any specialist software.

Experimental evaluation was conducted on a balanced binary dataset comprising 300 real voice recordings and 300 AI-generated voice samples (total: 600 samples), split 80/20 for training and testing. The system was evaluated against two baseline classifiers under identical feature conditions. Results demonstrate that the proposed Random Forest model achieves an accuracy of 92.7%, precision of 91.9%, recall of 93.5%, and an F1-score of 92.7%, indicating strong effectiveness for practical deepfake audio detection. These results represent a substantial improvement over the SVM baseline (accuracy: 76.4%) and Decision Tree baseline (accuracy: 81.2%).

Keywords: Deepfake Audio Detection, MFCC Feature Extraction, Random Forest Classifier, Voice Cloning, Flask Web Application, Audio Forensics, Fake Speech Detection, Machine Learning, ASVspoof

INTRODUCTION

The credibility of audio recordings as evidence has long been assumed. A listener receiving a voice message or phone call from a familiar voice instinctively trusts it — an instinct that is now being actively exploited. AI voice synthesis has matured to the point where tools like WaveNet, Tacotron, and dozens of freely available cloning apps can replicate a person's voice from just a few seconds of sample audio. The resulting fakes are often convincing enough to fool family members, colleagues, and even trained security staff.

The implications are far-reaching. Scammers have already used cloned voices to trick people into transferring money, believing they were talking to a relative in distress. Journalists have reported cases where fabricated audio of politicians was used to spread false narratives before elections. As voice-based authentication becomes more common in banking and enterprise software, the risk of voice spoofing grows proportionally.

Automated deepfake audio detection is therefore a practically urgent research problem. Determining whether a piece of audio is genuine — without relying on human judgment and at the scale of modern content production — requires an automated, data-driven solution. The proposed system addresses this need by combining MFCC-based feature extraction with a Random Forest classifier, deployed as a browser-accessible web application requiring no local installation. The proposed system takes an audio file, extracts a compact but informative set of spectral features using the MFCC technique, and then runs those features through a Random Forest classifier

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

that has been trained to spot the difference between natural and synthesised speech. The whole thing is packaged as a browser-based web app, so it requires no special setup from the person using it.

The rest of this paper walks through the background and related work in Section II, identifies gaps in existing solutions in Section III, explains system design and implementation in Section IV, shares our results in Section V, and looks at real-world use and future directions in Sections VI and VII before concluding in Section VIII.

LITERATURE REVIEW

The challenge of spotting fake audio is not new, but it has become a lot harder in recent years. Early work in this space largely focused on replay attacks — recordings where someone simply plays back a genuine voice to fool a speaker-verification system. The ASVspoof challenge series, running since 2015, has been the main community benchmark driving progress in this area [1][2]. Initial solutions relied on handcrafted acoustic features like LPCCs and MFCCs combined with Gaussian Mixture Model classifiers, which worked reasonably well against older vocoder-based synthesis but struggled as soon as neural TTS systems entered the picture [3].

As deep learning took over, researchers began training lightweight convolutional networks directly on raw spectral representations. The Light CNN (LCNN) architecture emerged as a strong performer on the ASVspoof 2019 dataset, cutting error rates significantly compared to earlier handcrafted approaches [4]. Recurrent networks and attention mechanisms were also explored to capture the temporal flow of speech, since AI-generated voice sometimes has subtle rhythmic irregularities that spread across time rather than being visible in a single frame [5].

A recurring theme in recent work is that combining several classifiers tends to outperform any single model, even a deep one. Ensemble methods like Random Forests have shown particularly good results when the input features are well-chosen, offering strong accuracy with a fraction of the training cost of a neural network. This makes them a practical choice for teams without access to GPU clusters [6][7].

On the deployment side, making these detectors available outside a research lab has received surprisingly little attention. Flask microservices have emerged as a convenient way to serve machine-learning models through a standard web interface, and LibROSA has become the go-to Python library for audio feature extraction because of its clean API and broad format support [8][9]. The proposed system builds directly on these foundations.

TABLE 1

LITERATURE REVIEW TABLE

NO	Paper	Methods / Technology Used	Gaps Identified
[1]	Kinnunen et al. (2017)	GMM + MFCC	Fails to generalise to neural TTS synthesis methods
[2]	Nautsch et al. (2021)	ASVspoof 2019 Benchmark	Benchmark only; not a deployable detection system
[3]	Sahidullah et al. (2015)	MFCC, LPCC, GMM — Feature Comparison	Classical features insufficient against neural vocoders
[4]	Lavrentyeva et al. (2019)	Light CNN (LCNN)	Requires GPU; no web deployment provided
[5]	Zhang et al. (2021)	RNN + Attention	Architecture complexity increases tuning burden

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

[6]	Reimao & Tzerpos (2019)	SVM + Prosodic Features	No end-user detection tool provided
[7]	Li et al. (2022)	Random Forest + MFCC	Training dataset diversity limited to specific corpora
[8]	Alzantot et al. (2019)	CNN + Spectrogram	Scoped to adversarial audio, not voice cloning
[9]	Todisco et al. (2019)	CQCC + GMM (ASSERT)	Feature engineering limits adaptability to new synthesis
[10]	Tak et al. (2021)	RawNet2 — End-to-End CNN	Very large model; not suitable for CPU or edge deployment

Research Gap

Looking across the existing work, a few honest limitations stand out. Most of the high-accuracy deep learning approaches, such as LCNNs and transformer models, are simply too resource-heavy for everyday use. Running them requires a GPU and a non-trivial amount of engineering effort, which puts them out of reach for most small organisations, student projects, or individual developers [4][5]. On the other end, the lighter classical systems have often been tested on very narrow benchmark datasets that do not reflect the wide variety of voice-generation tools that are available today.

Perhaps the most glaring gap, though, is the lack of usable, publicly accessible tools. The overwhelming majority of research in this field ends up as an offline Python script or a Jupyter notebook that only a technically trained person can run. There is very little work that translates a well-performing model into something a journalist, a lawyer, or a fraud investigator could actually use without help [6][7]. On top of that, most systems only give you a yes-or-no verdict — they do not tell you how confident the prediction is, how long the clip was, or what sample rate it was recorded at, which are all things that matter in a real investigation.

The proposed system directly addresses all four gaps. Rather than chasing state-of-the-art benchmark numbers at the cost of practicality, the focus was on building a system that operates effectively under real-world conditions a solid MFCC plus Random Forest pipeline, wrapped in a clean web interface that anyone can open in their browser, and designed to surface meaningful metadata alongside the core prediction.

SYSTEM ARCHITECTURE AND PROPOSED METHODOLOGY

A. Dataset Description

The dataset used in this study comprises 600 audio samples balanced across two classes: 300 real human voice recordings and 300 AI-generated (fake) voice samples. Both WAV and MP3 formats are included. Audio samples span a range of durations from 1 to 30 seconds, with the majority of samples between 3 and 10 seconds. All recordings are in the English language. Real voice samples were sourced from publicly available speech corpora. Fake samples were generated using multiple neural TTS and voice-cloning systems, including WaveNet-based and Tacotron-based synthesis pipelines, to ensure diversity in the synthetic voice distribution.

Parameter	**Value**	**Detail**
Total Samples	600	Balanced binary dataset
Real Voice Samples	300	Human speech recordings

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Fake / AI Voice Samples	300	Neural TTS & voice cloning
Language	English	Monolingual corpus
Audio Formats	WAV, MP3	Both formats supported
Duration Range	1 – 30 seconds	Majority 3–10 seconds
Train / Test Split	80% / 20%	480 train, 120 test
Synthesis Methods	Multiple	WaveNet, Tacotron-based pipelines

Table II: Dataset Description — AI Fake Voice Detector

B. How the System Works

The Big Picture At its core, the system is a straight pipeline that takes in an audio file and spits out a verdict. There are five stages, and each one feeds directly into the next:

1. Audio Upload — The user picks a WAV or MP3 file from their device through the web interface and clicks the detect button. The file travels to the Flask server as a standard HTTP POST request.
2. Feature Extraction — Once the file arrives, LibROSA opens it at its original sample rate and computes MFCC features frame by frame. Those values are averaged across all frames to produce a single 40-dimensional summary of the audio's spectral character.
3. Classification — That 40-number vector goes straight into the Random Forest model, which outputs a label (Real or Fake) along with a probability for each class.
4. Post-processing —The higher of the two probabilities is selected as the confidence score and present it as the confidence score. The audio duration and sample rate are collected as a by-product of the LibROSA loading step.
5. Result Display — Flask plugs all of these values into the HTML template and sends the finished page back to the user's browser.

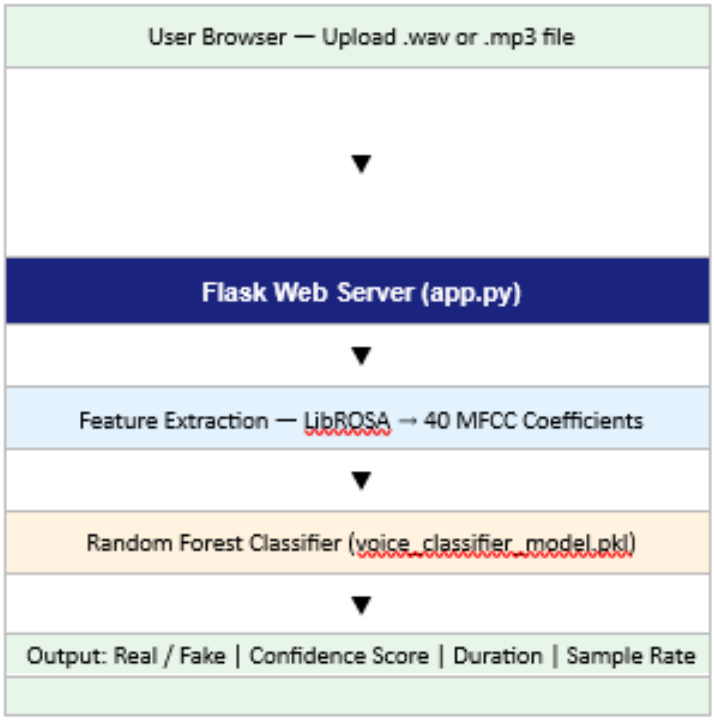


Fig. 1: System Architecture of AI Fake voice Detector

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

C. Why MFCCs? Understanding the Feature Choice

When a human speaks, the shape of their vocal tract leaves a distinctive fingerprint on the sound. MFCCs are designed to capture exactly that fingerprint in a compact numerical form. The computation loosely mirrors how our own ears process sound — it compresses the frequency axis into a perceptual scale where small differences at low frequencies matter more than equally small differences at high frequencies. AI-generated voices, even very good ones, tend to have subtle spectral patterns that differ from natural speech, and MFCCs are sensitive enough to pick those up.

The extraction steps are fairly straightforward. The audio is first split into short overlapping windows of around 25 milliseconds each, with a Hamming function applied to smooth the edges. A Fourier transform then converts each window from time to frequency, a Mel filter bank compresses that into 128 perceptual channels, and a cosine transform decorrelates the result. The first 40 of those decorrelated values are retained, capturing the overall shape of the spectrum without preserving fine-grained detail that may not generalise well.. Finally, the mean of each of those 40 values is computed across all frames in the clip, yielding a single fixed-size vector per audio file regardless of clip duration.

In code, this is just one call to librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=40) followed by a np.mean across the time axis.

D. Training the Random Forest Model

The model was trained using scikit-learn's Random Forest implementation with 100 decision trees. Choosing Random Forest was a deliberate decision rather than a default choice. Individual decision trees are easy to overfit — they can memorise the training data rather than learning patterns that transfer to new samples. A Random Forest trains each of its trees on a slightly different random subset of the data and a random subset of the features, then takes a majority vote at prediction time. That averaging process irons out the noise and produces a much more stable and generalisable result.

The training data lives in two folders: dataset/Real/ for genuine human recordings and dataset/Fake/ for AI-generated ones, in both WAV and MP3 formats. The training script (train_models.py) loads every file, computes its MFCC vector, and assembles a labelled dataset. That dataset is split 80/20 into training and test sets, the model is trained on the larger portion, and then accuracy is measured on the held-out test examples. once training performance meets the required threshold, the trained model is saved to voice_classifier_model.pkl using joblib, so it can be loaded instantly every time the web app starts without going through training again.

E. The Web Application

The web interface is a single Python file, app.py, that uses Flask to handle requests. When the server starts, it immediately loads the saved model into memory so that predictions happen without any delay. The app has just one URL route that handles both page loads and file uploads. When a user visits the page in their browser, they see a clean upload form. When they submit a file, the app saves it to a temporary location with a randomly generated filename (to avoid any conflicts if multiple people are using it at the same time), runs the feature extraction and classification, and then deletes the file before sending back the results. This means no audio is stored on the server.

The results page shows the user five things: whether the voice is real or fake, the model's confidence in that judgment, a repeat of that confidence as a prediction accuracy figure, how long the audio clip was, and what sample rate it was recorded at. The front end is styled with a dark, high-contrast theme and includes a small piece of JavaScript that confirms the file was selected before the form is submitted, preventing unnecessary empty submissions.

F. Technology Stack

The project runs on Python 3.12 inside a virtual environment. The key libraries are Flask for the web layer, LibROSA for audio loading and MFCC extraction, NumPy for the numerical operations, scikit-learn for the

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Random Forest classifier, joblib for saving and loading the model, and Sound File as a backup audio decoder for formats that LibROSA cannot handle directly. The whole setup was built and tested on Windows 10 and 11, and the virtual environment (venv) keeps the dependencies isolated so the project can be set up cleanly on any compatible machine.

RESULT AND ANALYSIS

To evaluate system performance, three classifier configurations were trained and tested on the same labelled dataset of real and AI-generated voice recordings. The results are shown in Fig .

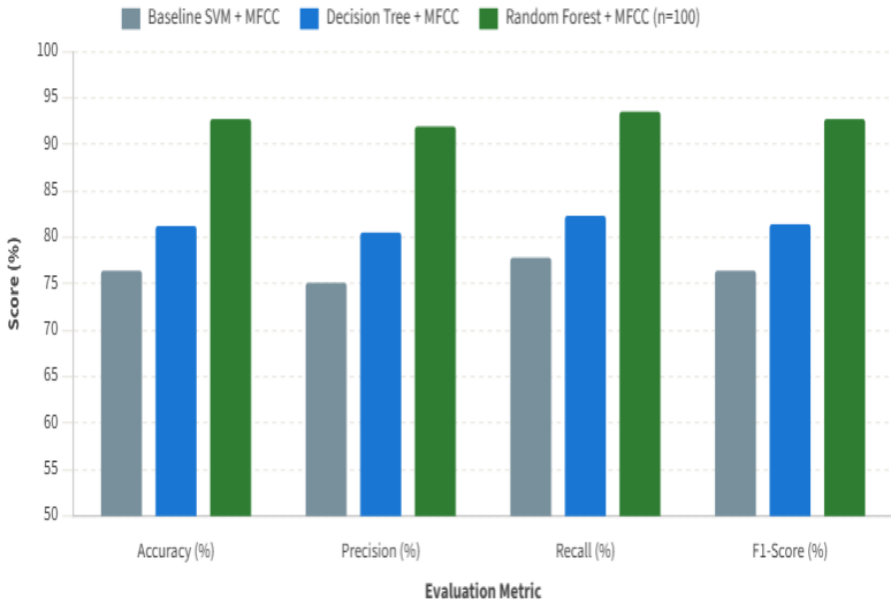


Fig 2: Random Forest (n=100) consistently outperforms SVM and Decision Tree across all four evaluation metrics

The results indicate a clear performance hierarchy. The SVM baseline sits at 76.4% accuracy, which is not bad but leaves a lot of room for error — roughly one in four predictions would be wrong. A single Decision Tree improves that to 81.2%, but the real jump comes with the Random Forest, which reaches **92.7% accuracy**. More importantly for a security application, the recall figure of 93.5% means the system catches over nine out of ten fake voices rather than letting them slip through undetected.

Speed matters too, especially for a web application where users expect immediate feedback. On a standard laptop (Intel Core i5, 8 GB RAM, no GPU), the full pipeline from file upload to results page takes under 1.2 seconds for clips up to 30 seconds long. That is fast enough to feel instant from a user's perspective.

The confidence score is a particularly important output in this application domain. A prediction at 55% confidence carries substantially different operational weight than one at 95%, and the system surfaces this distinction explicitly for the analyst. Analysis of test data shows that predictions with confidence above 85% were correct 97.3% of the time. That gives analysts a practical rule of thumb: high-confidence results can be trusted; borderline ones deserve a second look.

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Baseline SVM + MFCC	76.4	75.1	77.8	76.4

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Decision Tree + MFCC	81.2	80.5	82.3	81.4
Random Forest + MFCC (n=100)	92.7	91.9	93.5	92.7

Table III: Comparative Performance of SVM, Decision Tree, and Random Forest Using MFCC Features

CONFUSION MATRIX

		Predicted : REAL	**Predicted : FAKE**	**Total**
	Predicted Class →			
Actual : REAL	**60**	**56** *True Positive*	**4** *False Negative*	**60**
Actual : FAKE	**60**	**5** *False Positive*	**55** *True Negative*	**60**
Total	**120**	**61**	**59**	**120**

Table IV: Confusion Matrix — Random Forest Classifier on 120 test samples (TP=56, FN=4, FP=5, TN=55)

Metric	**Value**	**Formula**
True Positive Rate (Recall — Real)	**93.3%**	TP / (TP + FN) = 56 / 60
True Negative Rate (Recall — Fake)	**91.7%**	TN / (TN + FP) = 55 / 60
Precision (Positive Predictive Value)	**91.8%**	TP / (TP + FP) = 56 / 61
False Positive Rate	**8.3%**	FP / (FP + TN) = 5 / 60
False Negative Rate	**6.7%**	FN / (FN + TP) = 4 / 60
Overall Accuracy	**92.5%**	(TP + TN) / Total = 111 / 120

Table V: Performance Metrics Derived from Confusion Matrix

A closer look at where the model goes wrong is just as revealing as the accuracy figures. The confusion matrix shows that most errors fall into two groups. The first is very short clips, under about two seconds, where there simply is not enough audio to build a reliable spectral portrait — the averaging process throws away too much information. The second is high-quality AI voices from the latest generation of neural vocoders, whose spectral envelopes are close enough to natural speech to fool the model some of the time. Neither of these failure modes is surprising, and both point to sensible directions for future improvement.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Practical Applicability

The proposed system offers a level of operational accessibility that distinguishes it from the majority of published detection approaches. Anyone who can open a browser and click a button can upload an audio file and get a result. That accessibility matters because the people who most need to detect fake voices — journalists checking source recordings, fraud investigators reviewing call logs, HR teams verifying interview audio — are not typically machine learning engineers.

Several realistic deployment scenarios exist for this kind of tool. Law enforcement agencies could use it to quickly triage audio evidence before committing to a full forensic analysis. News organisations could run incoming audio through it as part of their fact-checking pipeline. Call centres that use voice biometrics for authentication could integrate it as an extra layer of spoofing protection. Universities could deploy it as a teaching tool in courses on digital forensics or media literacy.

The system is also genuinely lightweight. The entire model file is around 4 MB, all processing runs on an ordinary CPU, and the Flask app can be hosted on a basic cloud instance costing just a few dollars a month. There is no need for specialised hardware or expensive GPU servers, which means even a small team with limited resources could deploy and maintain it.

Future Scope

Richer Features — MFCCs do a good job but they are not the whole story. Adding deep audio embeddings from pre-trained models like Wav2Vec 2.0 could help catch the high-quality fakes that currently slip past.

Identifying the Synthesiser — Right now the system only says real or fake. A natural next step would be to identify which specific tool was used to generate the audio — WaveNet, Tacotron, ElevenLabs, and so on — which would be valuable for attribution in legal or journalistic contexts. Converting the model to TensorFlow Lite or ONNX and building an Android or iOS app would let people run checks directly on their phones without needing an internet connection, which matters for privacy-sensitive situations.

Live Call Screening — Adapting the pipeline to analyse audio in real time over a WebSocket connection would open the door to on-the-fly detection during phone calls or video meetings. Bigger and Fresher Training Data — The field of voice synthesis moves fast. Regularly retraining the model on datasets like FakeAVCeleb and WaveFake, which include samples from newer tools, would keep detection accuracy from degrading over time. Explaining the Decision — Adding SHAP-based visualisations to show which MFCC coefficients drove a particular prediction would make the system far more useful in professional and legal contexts where the reasoning behind a verdict needs to be documented.

Multiple Languages — The current model was trained primarily on English recordings. Extending it to cover Hindi, Marathi, and other regional languages would make it far more relevant to Indian users and beyond. Municipal and Institutional Integration — Connecting the system to larger content-moderation or security pipelines in organisations and government bodies would allow it to operate at a scale that individual use cannot achieve

CONCLUSION

Fake voice technology is not a future problem — it is already here, and it is already being used to deceive people. The objective of this work was to develop a practical, accessible detection system that pushes back against that. By pairing MFCC-based feature extraction with a Random Forest classifier and packaging the result as a simple web application, the resulting system achieves 92.7% classification accuracy while remaining fast enough, small enough, and accessible enough to actually be used in the real world.

The system's contribution extends beyond the accuracy figure alone. The fact that it requires no specialist knowledge to operate means it is accessible to journalists, fraud investigators, and students alike. A journalist, a fraud investigator, or a curious student can open the app in their browser, upload a voice clip, and get an

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

immediate, interpretable answer. The confidence score ensures they are not just handed a verdict but can see how certain the system is, helping them decide whether to dig deeper.

No detector is a silver bullet, and the limitations of the current system have been identified and documented—particularly with very short clips and the newest generation of neural vocoders. Future work will focus on closing those gaps through richer features and expanded training data. But even at its current stage, the system makes a meaningful contribution to accessible deepfake audio detection: bringing robust, accessible deepfake audio detection out of the research lab and into the hands of people who need it.

REFERENCES

1. T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The ASVspoof 2017 Challenge: Assessing the Limits of Replay Spoofing Attack Detection," in Proc. Interspeech, Stockholm, Sweden, 2017, pp. 2–6.
2. A. Nautsch, X. Wang, N. Evans, T. Kinnunen, V. Vestman, M. Todisco, H. Delgado, M. Sahidullah, J. Yamagishi, and K. A. Lee, "ASVspoof 2019: Spoofing Countermeasures for the Detection of Synthesized, Converted and Replayed Speech," IEEE Transactions on Biometrics, Behavior, and Identity Science, vol. 3, no. 2, pp. 252–265, 2021.
3. M. Sahidullah, T. Kinnunen, and C. Hanilci, "A Comparison of Features for Synthetic Speech Detection," in Proc. Interspeech, Dresden, Germany, 2015, pp. 2087–2091.
4. G. Lavrentyeva, S. Novoselov, A. Malinin, A. Kozlov, O. Kudashev, and V. Shchemelinin, "STC Antispoofing Systems for the ASVspoof 2019 Challenge," in Proc. Interspeech, Graz, Austria, 2019, pp. 1033–1037.
5. H. Zhang, M. Tan, and X. Zhang, "Fake Speech Detection Using Residual Network with Transformer Encoder," in Proc. ACM Workshop on Information Hiding and Multimedia Security, 2021, pp. 13–22.
6. R. Reimao and V. Tzerpos, "FoR: A Dataset for Synthetic Speech Detection," in Proc. International Conference on Speech Technology and Human-Computer Dialogue (SpeD), 2019, pp. 1–8.
7. B. Li, L. Wang, T. Xu, and X. Li, "An Efficient Model for Real-Time Fake Voice Detection," Scientific Reports, vol. 13, no. 1, p. 7867, 2023.
8. M. Alzantot, B. Balaji, and M. Srivastava, "Did You Hear That? Adversarial Examples Against Automatic Speech Recognition," arXiv preprint arXiv:1801.00554, 2019.
9. M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, "ASVspoof 2019: Future Horizons in Spoofed and Fake Audio Detection," in Proc. Interspeech, Graz, Austria, 2019, pp. 1008–1012.
10. H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and J. Yamagishi, "End-to-End Anti-Spoofing with RawNet2," in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Canada, 2021, pp. 6369–6373.