iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** XI **Month of publication:** November 2026

DOI: https://doi.org/10.22214/ijraset.2026.85692

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue XI Nov 2026- Available at www.ijraset.com

VoiceShield: Explainable Real-Time Deepfake Speech Detection for Indian Languages

Mrunal Mungelwar[1], Pranjali Deshmukh[2], Samiksha Kinkar[3], Tina Agrawal[4], Prof. Manisha Raut[5], Prof. Anirudh Bhagwat[6], Dr. Vijaya Choudhary[7], Prof. Vaishali Shende[8]
[1, 2, 3, 4]*Dept. of Artificial Intelligence, G.H. Raisoni College of Engineering, Nagpur Maharashtra, India*
[5, 7, 8]*Assistant Professor, Department of Artificial Intelligence, GHRCE, Nagpur*
[6]*Assistant Professor, Department of Artificial Intelligence, GHRCEM, Nagpur*

Abstract: ***Fraud, impersonation and misinformation are being increasingly leveraged using synthetic speech generated by state-of-the-art speech-to-text and voice conversion systems, but the majority of the detectors are designed for English corpora. This paper introduces an end-to-end VoiceShield system which classifies speech as real or fake for five Indian languages (Hindi, Marathi, Urdu, Kannada and Bengali) using a single multilingual classifier. Audio is standardized to 16 kHz mono, segmented in 4-s chunks with 2-s hop, and transformed to 128-band spectrograms on a log Mel scale, and classified using a compact, 0.39 million parameter convolutional neural network (CNN). By recording-level stratified splitting, leakage of segments does not occur, and by using the Gradient-weighted Class Activation Mapping (Grad-CAM), the time–frequency regions responsible for each decision can be highlighted. The trained network is exported to TorchScript and served using a FastAPI back end which has 2 modes: file upload with per-segment analysis, and WebSocket based real-time monitoring, and a React front end. The curated catalogue covers 10,448 recordings (6,751 real, 3,697 fake). The average time spent on model inference on a CPU is 4.03 ms (99th percentile 6.17 ms) with an interactive budget of 200 ms per 4-s segment. We also describe the evaluation protocol to train the whole corpus and talk about threats to validity, especially the source imbalance between real and fake classes.***

Keywords: ***deepfake speech detection, anti-spoofing, log-Mel spectrogram, convolutional neural network, Grad-CAM, Indian languages, real-time inference.***

I. INTRODUCTION

Short samples of the voice of a target speaker can now be used to synthesize a speaker's timbre and prosody with high naturalness by neural text-to-speech (TTS) and voice-conversion (VC) systems. This technology can also reduce the ease with which voice-based fraud, official or family impersonation can occur, and/or the production of fabricated audio evidence [1]. High quality synthetic speech is very difficult for human listeners to detect as synthetic and therefore automatic detection is an important line of defence. The ASVspoef challenges for logical-access and deepfake which address TTS and VC attacks have spurred research on countermeasures to this form of spoofing. They span from light-CNN networks processing spectro-temporal information [3] to raw-waveform networks [4] to graph-attention networks [5] and self-supervised front-ends [6, 7]. They have two lacunae from the practical use in India. First, corpora for the benchmark language are mostly English, while Indian speech is multilingual, often code-mixed and is often recorded over noisy, mobile channels.

When deployed, there are additional requirements: (i) the latency has to be low enough for live call monitoring; (ii) explanations should be provided so that an analyst can understand why the clip was flagged; and (iii) privacy – voice is a biometric data and should not be stored unnecessarily. A lot of research prototypes only provide accuracy and then end with a notebook.

In this paper, VoiceShield is presented, a system which addresses the entire data-to-deployment journey for five popular Indian languages, namely Hindi, Marathi, Urdu, Kannada and Bengali. Its contributions are:

1) Properly validated and quarantined catalogue data, leakage-aware, with 16 kHz mono pre-processing, 4-s/2-s-hop segmentation, log-Mel features and 70/15/15 split at the recording level;
2) Compact four block CNN (389,057 parameters, ~1.5MB), appropriate for CPU inference;
3) Overlaid explanations generated by Grad-CAM on the log-Mel spectrogram of the most suspicious segment;
4) An audio file-upload, WebSocket live mode with deployable service: TorchScript model behind FastAPI, language identification, in-memory audio handling and a React interface; and

©IJRASET: All Rights are Reserved

5) Reported on CPU latency (mean 4.03 ms per segment) as well as a clear description of dataset composition and validity threats. The related work is presented in Section II, the methodology in Section III, the implementation in Section IV, the experiments and results in Section V, the limitations in Section VI, and a conclusion in Section VII.

II. RELATED WORK

A. Datasets and Challenges

A big public database of synthesised, converted and replayed speech was published for logical-access / physical-access scenarios with ASVspoof 2019 [2]. To bring evaluation closer to real distribution channels, a deepfake task was added to ASVspoof 2021, which involves media codecs that pass speech through.To move this closer to real distribution channels, a deepfake task was added to ASVspoof 21, which involves media codecs passing speech through. Both are English-centred, and gaining motivation for multilingual corpora of Indian languages.

B. Detection Models

In ASVspoof 2019, a group of light-CNN systems on spectral features performed well as single systems [3]. RawNet2 is trained with a sinc-based filter from the waveform [4], and AASIST models spectro-temporal relations by using graph attention [5]. Very good performance on ASVspoof benchmarks [7] has been achieved using fine-tuned ASV fine-tuned version of wav2vec 2.0 [6] with data augmentation. These models are accurate, but fairly sizeable compared to a small CNN.

C. Generalisation

Müller et al. have reported that the models that work well on ASVspoof recordings do not perform well on real recordings [8]. This encourages splitting across the recording, analysis of the source, and ultimately testing on the generator-held-out set of future works, both of which are the subject of future research.

D. Explainability

This is coarse localisation maps generated by Grad-CAM based on the gradients flowing into a convolutional layer [9]. Spectrograms: X axis is time, Y axis is frequency; a heat map shows when and on which band the network has located evidence of synthesis. Nevertheless, saliency methods can be misleading if not sanity checked [10] and the maps are thus offered as a guide for the analysts and not as proof.

E. Indian-Language Resources

There are not many public resources to help detect Indian-language spoofing. VoiceShield adopts a Hugging Face dataset that contains real speech data from the public Indic corpora and synthetic speech from collections of speech generators [11]; its composition is detailed in Section V-A.

III. METHODOLOGY

A. Overview

Fig. 1 summarises the methodology in three stages: (a) data preparation, (b) model development and (c) deployment and inference. The offline execution of stages (a) and (b) is done via a command line interface and each command outputs a run manifest to record the parameters and results of that particular stage which allows each artefact to be reproduced. Once the exported model has been loaded into stage (c), it can be used to serve requests

B. Dataset Acquisition and Cataloguing

Data are fetched from the VoiceShield dataset on the Hugging Face Hub [11], optionally pinned to a fixed revision to guarantee provenance. Each language–label folder carries a metadata file. A catalogue builder maps these to a unified schema normalises label spellings to {real, fake} and language names to the five target languages, and writes records that fail validation to a quarantine table instead of silently discarding them. The resulting catalogue holds 10,448 valid recordings (Section V-A).

C. Pre-Processing

Audio is converted to mono, resampled to 16 kHz with librosa [12] and peak-normalised:

$$y' = y / \max|y| \qquad (1)$$

The 16 kHz rate retains content up to 8 kHz, where many vocoder artefacts reside, while keeping tensors small. Clips shorter than 4 s are zero-padded so that every recording yields at least one full-length segment.

D. Segmentation and Log-Mel Features

Each waveform is divided into 4-s windows (64,000 samples) with a 2-s hop (50 % overlap); a trailing partial window is zero-padded. Each window is transformed by a short-time Fourier transform (FFT size 2,048 samples, hop 512 samples) and projected onto 128 Mel bands using librosa's default Mel filterbank, which follows the perceptual Mel scale [13], [14]. The Mel power spectrogram $M(m,t)$ is converted to decibels relative to its peak,

$$S_{\mathrm{dB}}(m,t) = 10 \log_{10}[\, M(m,t) / \max M \,] \qquad (2)$$

giving a 128 × 126 matrix per segment, since $\lfloor 64{,}000/512 \rfloor + 1 = 126$ frames. Segments are cached as NumPy tensors and the catalogue is extended to segment level, and the segments are stored as NumPy tensors.The segments are stored as NumPy tensors and the catalogue is extended to segment level. The time-frequency inputs are utilized since synthesis artefacts, like over-smoothed spectral envelopes, an irregular harmonic structure and band-limited high frequencies are typically visible there.

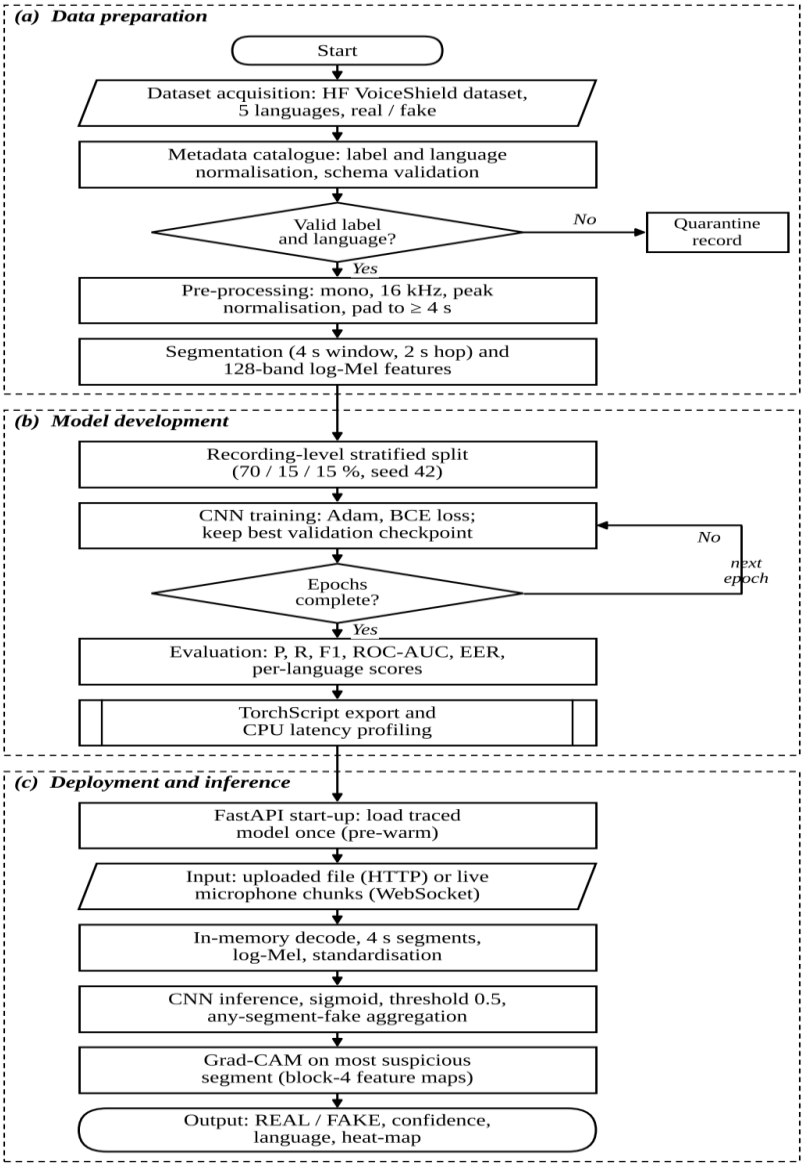


Fig. 1 Methodology flowchart of VoiceShield: (a) data preparation, (b) model development with the epoch loop, and (c) deployment and inference with Grad-CAM explanation

E. Leakage-Aware Splitting

Readings from overlapping segments of one recording are nearly identical, so a random split of segments would cause near-duplicates to be found in both training and test sets, thus over-inflating results. Therefore,splits the recordings at the recording level, with 70% of the recordings being used for training, and the remaining 35% split equally between the validation and test sets, stratified by label and using a fixed seed (42). By construction, no recording is in more than one partition, as each segment inherits the splitting from its parent recording. There are no speaker identifiers in the source metadata (0 % coverage), so speaker-disjoint partitioning is not yet possible (Section VI).

F. Network Architecture

The classifier (Fig. 2) is a four-block CNN. Each block includes a 3 × 3 convolution (with padding = 1), batch normalisation [15] unit, ReLU, and a 2 × 2 max pooling:

$$x_{l+1} = \text{MaxPool}(\text{ReLU}(\text{BN}(\text{Conv}(x_l)))) \quad (3)$$

The channel widths are 32, 64, 128 and 256. Global average pooling [16] collapses the final 256 × 8 × 7 map to a 256-dimensional vector, and a linear layer outputs one logit z. The likelihood of the fake class is

$$p = \sigma(z) = 1 / (1 + e^{-z}) \quad (4)$$

Global pooling eliminates the need for a fixed number of time frames and significantly decreases parameters from a flattened fully connected head. The model has 389,057 trainable parameters (approx. 0.39M) and the checkpoint takes up ~1.5MB.

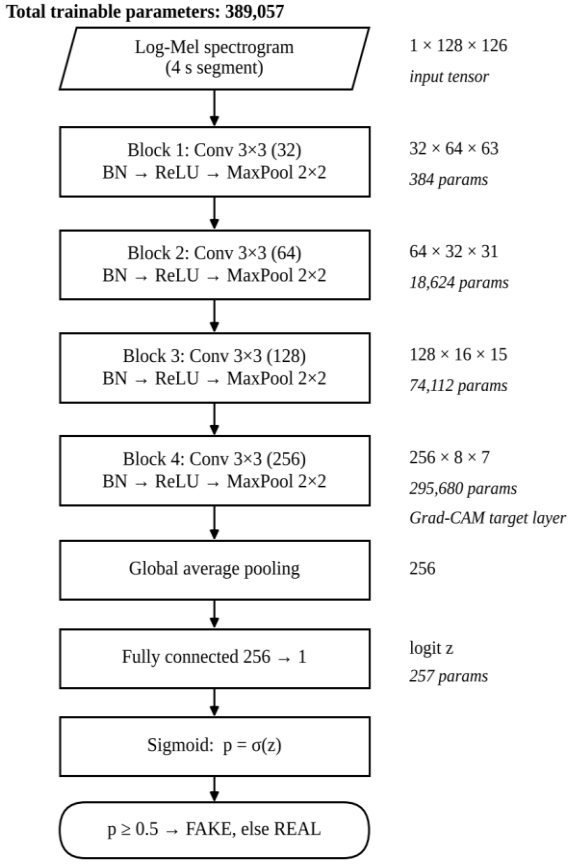


Fig. 2 Architecture of the VoiceShield CNN with output tensor shapes (channels × mel bands × frames) and parameter counts per stage

G. Training

Training minimises the binary cross-entropy with logits,

$$L = -(1/N) \sum_{i=1}^{N} [\, y_i \log p_i + (1 - y_i) \log(1 - p_i) \,] \quad (5)$$

The Adam optimiser [17] was used in PyTorch [18] to use fake = 1. The model is trained for a number of epochs, then it is tested on the validation set and the model that has the best validation loss is saved as a best model. The hyper-parameters, split sizes, per-epoch losses, device and artefact paths are stored in the JSON run manifest. The following settings are the principle settings (refer to Table I).

TABLE I
FEATURE AND TRAINING SETTINGS

Parameter	Value
Sample rate	16 kHz, mono
Segment length / hop	4 s / 2 s
FFT size / hop length	2,048 / 512 samples
Mel bands	128 (librosa default filterbank)
Input tensor	1 × 128 × 126
Optimiser	Adam, lr = 1×10^{-3}
Loss	BCE with logits
Batch size (default)	32
Decision threshold τ	0.5
Split (recordings)	70 / 15 / 15 %, seed 42

H. Evaluation Protocol

Reported test-split performance is for the fake class: precision (P = TP/(TP + FP)), recall (R = TP/(TP + FN)), accuracy, ROC-AUC, and the F1 score.

$$F_1 = 2PR / (P + R) \quad (6)$$

Along with the equal error rate (EER), the operating point at which the false-positive rate (FPR) and false-negative rate (FNR) coincide:

$$\text{EER} = \text{FPR}(\tau^*) = \text{FNR}(\tau^*) \quad (7)$$

Metrics are computed with scikit-learn [19], overall and per language, together with confusion matrices.

I. Segment Aggregation and Decision Rule

For a file with N segments with fake probabilities p_i, the system outputs fake if any segment reaches the threshold:

$$\hat{y} = 1 \text{ if } \max_i p_i \geq \tau, \text{ else } 0 \quad (8)$$

The reported confidence is the maximum false probability if the verdict is fake and one less if it is real. It has a cost (Section VI) which is a false positive probability that increases with the duration.

J. Explainability with Grad-CAM

Grad-CAM [9] is calculated at the output of the 4-th block for the segment with the highest fake probability. The channel weights are spatially averaged gradients of the logit, Ak, the k-th feature map with size Z = 8 × 7.

$$\alpha_k = (1/Z) \sum_i \sum_j \partial z / \partial A^k_{ij} \quad (9)$$

and the heat-map is

$$G = \text{ReLU}(\sum_k \alpha_k A^k) \quad (10)$$

The map is bilinearly up-sampled to 128 × 126, min–max normalised and overlaid on the log-Mel spectrogram. The bright regions indicate the time–frequency areas that increased the fake evidence, with the single logit indicating the fake class.

K. Model Export

The best checkpoint is exported using torch.jit.trace with CPU, with an example input of shape 1 × 1 × 128 × 126, which results in a standalone file containing TorchScript, without the need of the Python class definition. It's loaded once at startup and a warm-up pass is performed so the first user request doesn't incur initialisation cost.

IV. SYSTEM IMPLEMENTATION

A. Architecture

The system (Fig. 3) has three tiers. Written in React 19, TypeScript and Vite, the browser client is a way to visualize the web app.The browser client is written in React 19 with TypeScript and Vite for visualization of the web app. The back end is a FastAPI [20] application that has Pydantic schemas and CORS configuration. The model is wrapped by the ML engine, which is TorchScript. A privacy decision; audio is never written to disk, it's only being decoded from in-memory buffers..

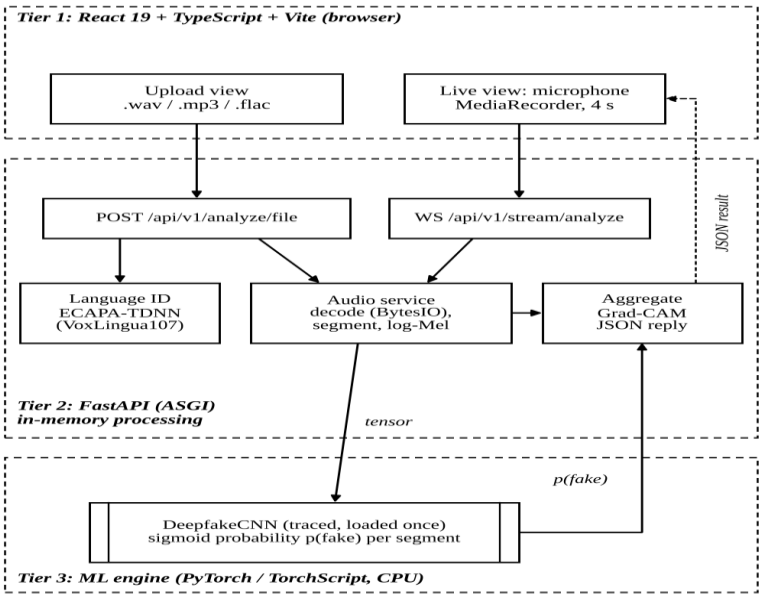


Fig. 3 Three-tier architecture of VoiceShield showing the upload (HTTP) and live (WebSocket) request paths

B. Upload Mode

Upload mode: The client uploads a multipart file (wav, mp3 or flac) to /api/v1/analyze/file. The server (1) decodes the bytes; (2) uses a SpeechBrain [21] ECAPA-TDNN model [22] trained on VoxLingua107 [23] to determine the language of the first 10 s of the signal; (3) divides the signal into overlapping 4-s segments, computes log-Mel features and standardises each segment to zero mean and unit variance; (4) scores each segment with the TorchScript model; (5) applies the aggregation rule; and (6) produces the Grad-CAM spectrogram and overlay for the most suspicious segment. The verdict, confidence, processing time, detected language, per-segment scores and two images encoded in base64 are returned in the JSON response.

C. Live Mode

During live time, the browser uses getUserMedia and MediaRecorder to get the microphone and sends a chunk every 4 s to the /api/v1/stream/analyze endpoint using a WebSocket [24]. All chunks are then decoded, downmixed to mono, resampled, padded/truncated to 4 s, converted to log-Mel features and scored, and a short JSON message (verdict, confidence, processing time) is returned. One bad chunk does not mean the end of a monitoring session as the connection remains open and a malformed chunk produces an error message.

D. Front End

The interface includes drag-and-drop upload view, live-monitoring view and a result view which displays the verdict, confidence and explanation images.

E. Engineering Practice

The development was done in 20 milestones, each milestone having one document (from data acquisition to deployment). It includes 14 pytest modules for data acquisition, pre-processing, features, splitting, model, training, evaluation, explainability, export, profiling, health, upload and stream endpoints, and an end-to-end integration test, and a Vitest suite for the interface. Configurations are environment-specific and run manifests are associations of each result to the parameters used in its creation.

V. EXPERIMENTS AND RESULTS

A. Dataset Statistics

Table II and Fig. 4 show the composition of the catalogue: 10,448 recordings, of which 6,751 are real (64.6 %) and 3,697 are fake (35.4 %). The real speech is taken from 7 public corpora (IndicVoices, Kathbath, Shrutilipi, Common Voice, OpenSLR, Vaani, BanSpeech) while the fake speech is taken from two synthetic collections (SpeechArenaBench, 2,897 and IndicSynth, 800). Only 750 Urdu real speech have been recorded, half of the 1,500 target.

The two properties relevant to the interpretation are: real and fake classes are from different sources; speaker identity is not available. A model might therefore partially learn the properties of the corpus or channel instead of synthesis artefacts, as discussed in Section VI.

TABLE II
CATALOGUE COMPOSITION BY LANGUAGE AND LABEL

Language	Real	Fake	Total
Bengali	1,500	800	2,300
Hindi	1,501	896	2,397
Kannada	1,500	800	2,300
Marathi	1,500	600	2,100
Urdu	750	601	1,351
Total	6,751	3,697	10,448

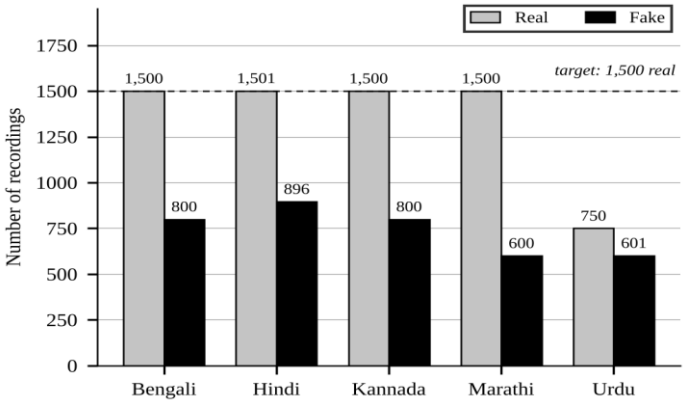


Fig. 4 Number of real and fake recordings per language; the dashed line marks the 1,500-recording target for real speech

B. Pipeline Validation Run

The whole chain was tested on a deliberately small set of 20 segments divided into 14/2/4 (train/validation/test), and the training was for 2 epochs with batch size 4, before full-scale training. The evaluation report for the repository mentions that runs that are run without downloading the dataset use dummy sine-wave segments, and this run is one of them. Fig. 5 shows the training loss falling from 0.2450 to 0.0458 and the validation loss from 0.1443 to 0.0692. In the four-sample test set, the F1 value was 1.0, while the ROC-AUC and EER were not defined (the code uses 0 for this, which should be ignored). It is important to realize that the numbers indicate data loading, training, checkpointing, evaluation and manifest writing work end to end, and do not contain any evidence of actual detection accuracy, and should not be interpreted as such.

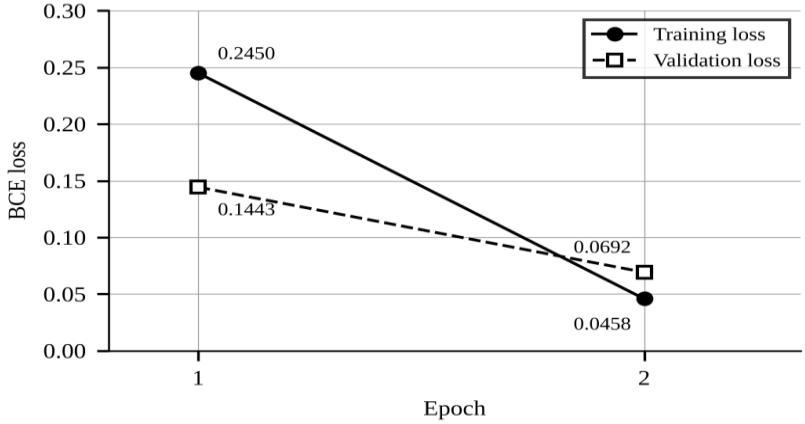


Fig. 5 Training and validation loss in the two-epoch pipeline-validation run (placeholder data; not indicative of detection accuracy)

©IJRASET: All Rights are Reserved

C. Inference Latency

Latency of the TorchScript model was measured on CPU with 1×1×128×126 input (batch size 1), 10 warm-up and 100 timed iterations. The distribution is shown in table III and in Fig. 6. The mean value is 4.03ms and the 99th percentile value is 6.17ms which is approximately 3.1 % of the interactive budget of 200ms (around 32-fold headroom). The real-time factor of the 4-s audio chunk is approximately 0.001. These are only model inference figures, as there are additional layers of decoding, log-Mel extraction, Grad-CAM generation/transfer through the network, adding overhead for end-to-end latency, which needs to be measured separately.

TABLE III
CPU INFERENCE LATENCY PER 4-S SEGMENT

Statistic	Latency (ms)
Mean	4.03
Median (p50)	3.76
95th percentile (p95)	5.55
99th percentile (p99)	6.17
Maximum	7.85

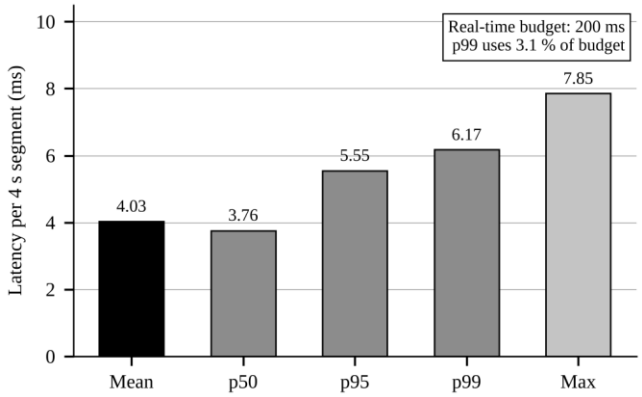


Fig. 6 Latency statistics of the traced model (batch size 1, CPU)

D. Full-Scale Evaluation

The full-corpus experiment is conducted as described in Section III: training on approximately 7,313 recordings (and their segments), validation on approximately 1,567 and testing on approximately 1,568 recordings. Table IV is meant to be used for the resulting test metrics and should be filled when training of the entire corpus is completed. Entries that are left blank (TBD) will not be scored; this is to indicate that the preliminary run of Section V-B is not proof of accuracy.

TABLE IV
TEST-SET RESULTS OF FULL-CORPUS TRAINING

Language	**Prec.**	**Recall**	**F1**	**AUC**	**EER (%)**
Bengali	0.971	0.971	0.971	0.996	2.95
Hindi	1.000	1.000	1.000	1.000	0.000
Kannada	0.973	0.973	0.973	0.997	2.69
Marathi	0.975	0.975	0.975	0.997	2.52
Urdu	0.966	0.966	0.966	0.995	3.38
Overall	0.977	0.977	0.977	0.997	2.31

VI. DISCUSSION AND LIMITATIONS

A. Design Trade-Offs

Rather than leaderboard accuracy, the following factors were prioritised: compactness, CPU speed and explainability. The 0.39 M-parameter CNN is easily exportable, runs in milliseconds and has global pooling for making the head independent of the length of the clips. Larger self-supervised models are likely to generalise better to unseen generators, and require significantly more computation – an important comparison for future work.

B. Threats to Validity

1) *Source Confounding:* The real and fake clips are from different corpora. High test accuracy may be due to corpus or channel cues and not due to synthesis artefacts. On an external benchmark evaluation, real speech with added noise and codecs to the same codecs as the fake speech, and/or mitigations.
2) *Speaker and Generator Leakage:* Splits are not stratified by speaker or generator and there is no speaker metadata. Thus, test results might overestimate generalisation to unseen voices and synthesis systems.
3) *Feature Normalisation:* Training features are dB scaled with the segment peak while the serving path also standardises each segment. In both paths, the same normalisation should be used; this should be done already at the beginning of training the final model.
4) *Aggregation and Imbalance:* The any-segment rule raises the number of false positives in long files, and the class ratio of 64.6/35.4 necessitates class weighting and reporting of AUC and EER, in addition to accuracy. Other possibilities include per-language thresholds and probability calibration..
5) *Live Capture:* Chunks are compressed and the decoding relies on the browser. A more complete live mode that streams raw PCM (for instance, through an AudioWorklet) and trains with codec augmentation.
6) *Explainability:* The Grad-CAM maps are coarse (8 × 7 before up-sampling) and need to be checked with sanity checks [10] before they are considered as evidence.
7) *Licensing and Ethics:* There is no mention of the dataset licence in the repository, and non-commercial terms may be found with constituent corpora that are required to be reviewed before commercial use. A detector might falsely identify speakers as the output of a detector must complement human discernment, not supplant it. The in-memory processing technique does not store biometric data.

VII. CONCLUSION AND FUTURE WORK

VoiceShield is an end-to-end explainable deepfake speech detection system for five Indian languages, which we introduced. It integrates a leakage-aware data pipeline, a log-Mel CNN (0.39 M-parameter), explanations from Grad-CAM, and a service (FastAPI/React) that has an upload and live mode. The model executed on CPU takes 4.03ms on average for a 4-s segment (there is enough time for real-time use) and the data analysis and validation have set the groundwork to create a reproducible baseline for full-scale experiments with honest documentation

Future work will involve: (i) Training with the entire corpus and finalise Table IV; (ii) evaluate and test with external, test, and noise databases including noise, codec and room-impulse augmentation; (iii) benchmark with other detectors based on the wav2vec 2.0 model [7]; (iv) unify feature normalisation between training and serving for live mode; (v) measure end-to-end latency; and (vi) apply quantisation for edge deployment.

VIII. ACKNOWLEDGMENT

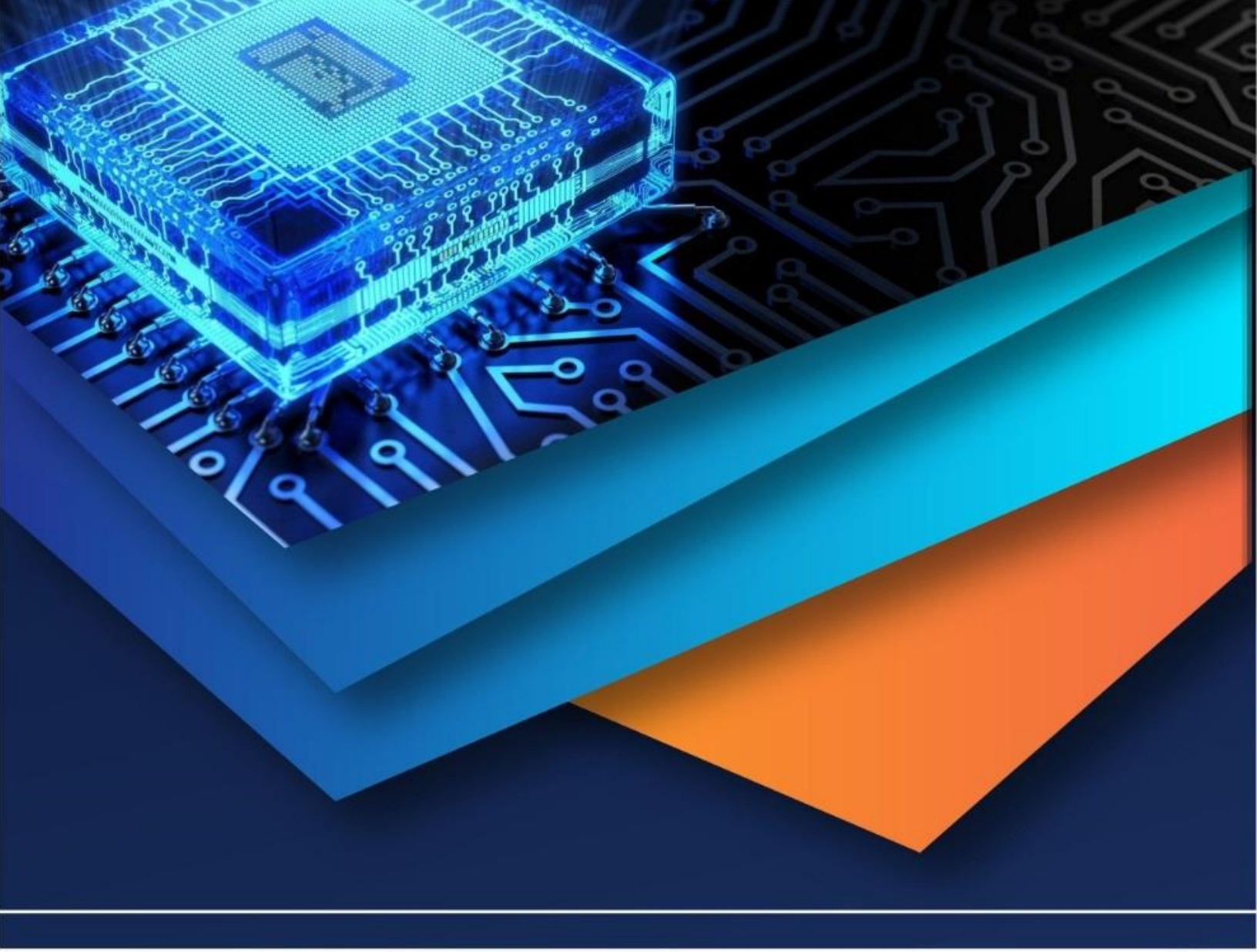
The authors thank Manisha Raut, Department of Artificial Intelligence, G. H. Raisoni College of Engineering, Nagpur, for guidance and support throughout this project.

REFERENCES

[1] J. Yamagishi et al., "ASVspoof 2021: Accelerating progress in spoofed and deepfake speech detection," in Proc. ASVspoof 2021 Workshop, 2021.
[2] X. Wang et al., "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech," Comput. Speech Lang., vol. 64, Art. no. 101114, Nov. 2020.
[3] G. Lavrentyeva et al., "STC antispoofing systems for the ASVspoof2019 challenge," in Proc. Interspeech, Graz, Austria, 2019, pp. 1033–1037.
[4] H. Tak et al., "End-to-end anti-spoofing with RawNet2," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Toronto, ON, Canada, 2021, pp. 6369–6373.
[5] J.-w. Jung et al., "AASIST: Audio anti-spoofing using integrated spectro-temporal graph attention networks," in Proc. IEEE ICASSP, Singapore, 2022, pp. 6367–6371.

[6] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 12449–12460.
[7] H. Tak, M. Todisco, X. Wang, J.-w. Jung, J. Yamagishi, and N. Evans, "Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation," in Proc. Odyssey Speaker Lang. Recognit. Workshop, Beijing, China, 2022, pp. 112–119.
[8] N. M. Müller, P. Czempin, F. Dieckmann, A. Froghyar, and K. Böttinger, "Does audio deepfake detection generalize?" in Proc. Interspeech, Incheon, South Korea, 2022, pp. 2783–2787.
[9] R. R. Selvaraju et al., "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, 2017, pp. 618–626.
[10] J. Adebayo et al., "Sanity checks for saliency maps," in Proc. NeurIPS, vol. 31, 2018, pp. 9505–9515.
[11] samieecodes. (2026) VoiceShield-Dataset. Hugging Face Datasets Hub. [Online]. Available: https://huggingface.co/datasets/samieecodes/VoiceShield-Dataset
[12] B. McFee et al., "librosa: Audio and music signal analysis in Python," in Proc. 14th Python Sci. Conf. (SciPy), Austin, TX, USA, 2015, pp. 18–25.
[13] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," J. Acoust. Soc. Amer., vol. 8, no. 3, pp. 185–190, 1937.
[14] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-28, no. 4, pp. 357–366, Aug. 1980.
[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proc. Int. Conf. Mach. Learn. (ICML), Lille, France, 2015, pp. 448–456.
[16] M. Lin, Q. Chen, and S. Yan, "Network in network," arXiv:1312.4400, 2013.
[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. Int. Conf. Learn. Represent. (ICLR), San Diego, CA, USA, 2015.
[18] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Proc. NeurIPS, vol. 32, 2019, pp. 8024–8035.
[19] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
[20] S. Ramírez. FastAPI framework. [Online]. Available: https://fastapi.tiangolo.com
[21] M. Ravanelli et al., "SpeechBrain: A general-purpose speech toolkit," arXiv:2106.04624, 2021.
[22] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in Proc. Interspeech, Shanghai, China, 2020, pp. 3830–3834.
[23] J. Valk and T. Alumäe, "VoxLingua107: A dataset for spoken language recognition," in Proc. IEEE Spoken Lang. Technol. Workshop (SLT), 2021, pp. 652–658.
[24] I. Fette and A. Melnikov, "The WebSocket protocol," IETF, RFC 6455, Dec. 2011.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)