DEEP FAKE DETECTION

AI-Powered Real-Time Detection and Prevention of Voice Cloning Impersonation

Ayush, Harsh Yadav, Rohit Dhankhar, Meenakshi

Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), Sushant University, India

Published November 2026

Abstract

Voice cloning based on neural text-to-speech and voice conversion has created a new impersonation channel in which an attacker can generate speech that resembles a target speaker without requiring access to the target device. This paper proposes a real-time detection and prevention framework that combines acoustic anti-spoofing features, self-supervised speech representations, a spectro-temporal classifier, temporal score aggregation, calibration, and risk-aware step-up verification. The design is evaluated conceptually against established speech deepfake benchmarks and is accompanied by a reproducible experimental protocol covering in-domain, cross-generator, cross-dataset, noisy, compressed, replay, and short-window conditions. The literature shows that modern detectors can perform strongly on benchmark data but can degrade when the synthesis method, language, channel, or preprocessing differs from the training distribution. The proposed framework therefore treats robustness and operational latency as first-class evaluation variables. The prevention layer separates model inference from security policy so that uncertain detections can trigger independent verification rather than automatic identity rejection. The paper provides a detailed materials-and-methods section, dataset governance rules, model architecture, threat model, evaluation metrics, error-analysis framework, privacy considerations, and reproducibility checklist. The manuscript is intended as a deployment-oriented research prototype. Original experimental claims must be added from the authors' own final run before the paper is represented as an empirical validation study.

Keywords: voice cloning; speech deepfake detection; audio anti-spoofing; synthetic speech; voice conversion; real-time detection; self-supervised learning; ASVspoof; impersonation prevention; cybersecurity

1 Introduction

Speech is increasingly used as an informal identity signal. People often accept a familiar voice as evidence that a relative, colleague, manager, customer, or public representative is speaking. Voice authentication systems formalize this intuition by using speaker characteristics as biometric evidence. The same property that makes speech useful for authentication also makes it attractive to attackers. A voice-conversion or text-to-speech system can reproduce characteristics associated with a target speaker while generating new linguistic content, creating an impersonation channel that does not require the physical presence of the victim.

Modern voice cloning is not limited to robotic or obviously synthetic speech. Neural TTS and voice-conversion systems have progressively reduced audible artefacts, while consumer tools have lowered the technical barrier to producing convincing samples. Research communities have therefore moved from generic speech synthesis research toward explicit anti-spoofing and deepfake-detection

benchmarks. ASVspoof 2021 introduced a dedicated speech-deepfake track and provided public data and baselines for detecting bona fide versus spoofed speech. [1, 2]

The societal and security problem is broader than the detection of a fake waveform. An attacker does not need a perfect clone if the victim is distracted, under time pressure, or already expects a particular request. A short synthetic message can be enough to initiate a transfer, disclose an account detail, change a workflow, or cause a recipient to trust a false instruction. A detector therefore has to operate as part of a larger decision system. It must estimate risk, communicate uncertainty, and support a response that does not depend entirely on the classifier being perfect.

NIST's work on synthetic content emphasizes that detection performance varies with generation technique, evaluation data, preprocessing, language, noise, and whether the generator is represented in training data. In particular, open-set conditions and real-world audio can expose weaknesses that are hidden by closed-set benchmarks. [3] This observation motivates the design adopted in this paper: robustness and operational latency are treated as first-class research variables rather than as afterthoughts.

The research problem is consequently formulated as follows: how can an AI system identify voice-cloning impersonation in near real time and translate its output into a proportionate prevention action while remaining useful under channel variation, background noise, compression, replay, and previously unseen generation methods?

The remainder of the paper first reviews the technical background and prior detection methods. It then defines the research gap, materials, datasets, preprocessing pipeline, feature representations, proposed model, streaming decision logic, and prevention mechanism. Evaluation criteria are specified using EER, ROC-AUC, F1-score, precision, recall, false-accept rate, false-reject rate, calibration error, and latency. The discussion then considers security, privacy, fairness, deployment constraints, and limitations. The appendices provide reproducibility checklists, pseudocode, configuration recommendations, and a journal-submission checklist.

1.1 Background: voice cloning and impersonation

Voice cloning systems generally operate by learning a mapping from linguistic content and speaker information to an acoustic representation, or by converting an existing utterance toward a target speaker. In a text-to-speech setting, text is converted to speech in a learned voice. In voice conversion, the linguistic content of a source utterance is preserved while speaker characteristics are modified. Modern pipelines can use neural acoustic models, neural vocoders, diffusion models, or combinations of these components. The detector does not need to identify the exact generator in every case; it needs to identify evidence that the observed signal is inconsistent with the distribution of bona fide speech.

The distinction between content authenticity and speaker authenticity is important. A recording can contain a real speaker saying words that were genuinely spoken but can also contain a cloned voice saying words the person never uttered. Conversely, a real recording can be edited, spliced, compressed, or replayed without being generated by a TTS model. A practical anti-impersonation system should therefore distinguish at least three concepts: synthetic generation, manipulation of genuine audio, and uncertainty about provenance. The proposed framework focuses primarily on the first two while treating provenance as a separate trust signal.

Human listeners cannot be treated as a reliable last line of defense. Recent perceptual work found that listeners were often unable to consistently identify state-of-the-art AI-generated voice clones, reinforcing the value of machine-assisted screening. [4] A technical system can examine details that are difficult to hear consciously, such as high-frequency spectral structure, phase-related characteristics, temporal regularity, vocoder artefacts, or inconsistencies in short-time representations.

1.2 Motivation and practical use cases

The first use case is voice-based financial or customer-service verification. In such environments, a detection signal should not automatically decide whether a customer is genuine. Instead, it can influence the required verification level. A high-risk result could require a one-time code, an application confirmation, a second communication channel, or a human review before a sensitive operation is completed.

The second use case is messaging and social platforms. A voice note can be screened before it is forwarded or displayed with an authenticity warning. The system can also operate at upload time, producing a risk label that is separate from the content itself. Because platforms vary in codec and sample-rate handling, robustness testing is essential.

The third use case is organizational communication. A detector can support employees receiving urgent voice instructions about payments, credentials, or privileged system changes. The prevention policy can be designed around the consequence of the requested action. A low-risk conversational message may only receive a warning, while a high-impact transaction request can require independent verification.

A fourth use case is forensic triage. Investigators may receive thousands of audio files and need a prioritization mechanism. In that context, the detector is not necessarily an evidentiary decision-maker; it is a screening tool that helps analysts select files for deeper examination. This distinction reduces the risk of treating a probabilistic classifier as a definitive authenticity certificate.

2 Literature review

Speech deepfake detection has evolved through several generations. Early systems relied heavily on handcrafted spectral features such as linear-frequency cepstral coefficients (LFCC), constant-Q cepstral coefficients (CQCC), and related representations combined with Gaussian mixture models or lightweight neural networks. These methods remain useful as baselines because they are relatively interpretable, inexpensive, and can expose whether a more complex model actually learns additional information.

Raw waveform models changed the design space by allowing the network to learn filters directly from the signal. RawNet2 is an important example: it adapts an end-to-end speaker-recognition architecture for spoofing countermeasures and was evaluated within the ASVspoof ecosystem. [5] Such models can capture information that may be lost when audio is reduced to a fixed handcrafted feature representation, although they can also be sensitive to domain differences and training conditions.

AASIST introduced integrated spectro-temporal graph attention to model artefacts across temporal and spectral dimensions. The authors reported strong performance with a single integrated system and a lightweight variant, demonstrating that explicit modelling of relationships between feature regions can be effective for anti-spoofing. [6] The model is particularly relevant to this research

because it offers a practical back-end for a hybrid system rather than requiring a large ensemble of independent detectors.

Self-supervised learning has provided another major direction. wav2vec 2.0 learns contextual speech representations from large amounts of unlabeled audio and can subsequently be fine-tuned for downstream tasks. [7] Speech deepfake studies have used wav2vec-family representations because the embeddings encode rich information about speech production and temporal context. However, the strength of a self-supervised representation does not remove the open-set problem: a representation learned from one distribution may still rely on generation artefacts that disappear when a new synthesizer or post-processing pipeline is introduced.

The ASVspoof series has been especially influential because it standardized evaluation around bonafide and spoofed speech and provided challenge data, keys, metadata, and baseline implementations. ASVspoof 2021 includes logical-access, physical-access, and speech-deepfake conditions, with the deepfake task using EER as its primary metric. [1, 2] The challenge documentation also emphasizes realistic effects such as channel variability and replay conditions.

WaveFake broadened the data landscape by collecting generated audio from multiple neural architectures and languages. The current dataset release contains more than 100,000 generated clips, illustrating why detector training should not depend on a single synthesis engine. [8] The Fake-or-Real dataset and later in-the-wild datasets provide additional examples for cross-domain evaluation.

ADD 2022 was created partly because existing tasks did not fully capture real-life challenges such as noisy fake audio, partially fake utterances, and rapidly changing synthesis algorithms. Its tracks explicitly considered low-quality fake audio, partially fake audio, and adversarial or competitive generation settings. [9] These task definitions support the argument that robustness should be measured under multiple conditions rather than through one clean test set.

Recent surveys confirm that the field has expanded rapidly but remains fragmented. A 2025 ACM Computing Surveys article reviewed more than 200 speech-deepfake papers published through March 2024 and highlighted datasets, architectures, optimization techniques, generalization, metrics, and open-source resources. [10] The breadth of this literature makes it difficult to compare systems unless dataset splits, attack families, preprocessing, and evaluation metrics are reported precisely.

Research on adversarial attacks further complicates the picture. Audio-deepfake detectors can themselves be attacked or manipulated, and classifier performance can degrade when adversarial or distribution-shifting transformations are introduced. [11] This means that an operational system should avoid treating a detector score as a permanent truth value. Instead, it should maintain uncertainty and use additional verification when the decision is consequential.

Recent work also emphasizes that realistic presentation matters. A 2025 study argues that the difference between raw deepfake audio and audio presented through a communication channel can materially affect detector generalization. [12] This is directly aligned with the proposed evaluation strategy, which includes codec conversion, noise, replay, and channel simulation rather than evaluating only the original waveform.

2.1 Comparative review of detection approaches

Handcrafted-feature systems offer a useful baseline because their behaviour can be inspected through frequency bands, cepstral coefficients, and simple classifiers. LFCC-based approaches can capture spectral envelope differences and are computationally inexpensive. CQCC-based approaches

emphasize frequency resolution that can be helpful for certain spoofing artefacts. Their weakness is that manually selected features may fail to represent novel artefacts produced by a generator not present in training data.

Raw waveform architectures reduce dependence on manually designed features. RawNet2 and related architectures can learn front-end filters jointly with classification layers. Their major advantage is end-to-end optimization. Their main engineering challenges are data requirements, computational cost, and sensitivity to recording conditions. For real-time use, the model must be benchmarked on the actual hardware and audio-window length intended for deployment.

Spectrogram and convolutional approaches transform the waveform into a time-frequency representation before classification. They can exploit local patterns and are often easier to visualize using saliency or feature maps. A limitation is that fixed transforms can hide information present in phase or raw waveform structure. A hybrid representation can reduce this risk by providing more than one view of the same signal.

Transformer and self-supervised approaches use contextual representations learned over time. wav2vec 2.0 and related models can encode long-range dependencies and phonetic structure. The cost is memory and inference time, especially when a large backbone is fine-tuned end-to-end. For a real-time detector, a frozen or partially fine-tuned backbone plus a lightweight classification head may be a better engineering compromise than full fine-tuning.

Ensemble systems combine multiple models or representations. They can be robust because different detectors respond to different artefacts, but ensembles increase latency, maintenance cost, and calibration complexity. The proposed design therefore uses a compact hybrid representation rather than an unrestricted ensemble. The goal is to obtain complementary evidence while keeping the decision path auditable.

2.2 Research gap identification

The literature reveals four related gaps. First, many studies optimize a single benchmark metric on a fixed dataset, while operational voice communication introduces compression, noise, reverberation, packet loss, microphone differences, and previously unseen generators. Second, a large fraction of detection systems stop at a classification score and do not specify how that score should influence a security action. Third, latency is often reported less systematically than accuracy even though real-time use depends on the time required to collect, preprocess, infer, smooth, and act on audio. Fourth, evaluation protocols sometimes allow generator or speaker leakage across splits, which can inflate apparent generalization.

The proposed study addresses these gaps by defining the detector as one component of a risk-aware pipeline. The experimental protocol separates speakers and generator families where metadata permits. It also requires cross-dataset evaluation, post-processing stress tests, and latency measurement. The prevention layer is explicitly designed to handle uncertainty through step-up verification instead of relying on a hard threshold alone.

The research contribution is therefore architectural and methodological rather than a claim that one classifier universally solves voice cloning. The paper specifies a reproducible way to compare a classical baseline, a raw-waveform detector, a self-supervised representation, and a fusion model under the same data and evaluation rules. This provides a clearer path from academic detection performance to an implementable security control.

3 Research objectives and hypotheses

The primary objective is to design a real-time voice-cloning detection and prevention framework that balances detection quality, robustness, computational cost, and operational safety. Secondary objectives are to identify complementary acoustic and learned representations, quantify the effect of realistic audio transformations, and define a decision policy that uses detector confidence without presenting probabilistic output as certainty.

H1: A hybrid representation combining acoustic anti-spoofing features with self-supervised speech embeddings will provide more stable discrimination across attack families than either representation alone. H2: Cross-dataset performance will be lower than in-domain performance, demonstrating the importance of open-set evaluation. H3: Temporal aggregation across multiple short windows will reduce isolated false alarms compared with a single-window decision. H4: A step-up verification policy can reduce the operational impact of false accepts without requiring an extremely aggressive detector threshold.

The hypotheses are intentionally falsifiable. H1 can be tested by ablation, H2 by cross-dataset evaluation, H3 by comparing window-level and session-level decisions, and H4 through a policy simulation using fixed detector outputs. If the hypotheses are not supported, that result is still informative because it identifies which assumptions do not hold under the selected experimental conditions.

4 Materials and methods

This section describes the materials, datasets, preprocessing, model components, training protocol, and evaluation design. The method is designed to be reproducible on a conventional research workstation with a CUDA-capable GPU for model training. A CPU-only inference mode is also considered because practical deployment may occur on an endpoint or server where GPU resources are shared.

The study follows a two-stage design. Stage A establishes reference baselines using conventional acoustic features and lightweight classifiers. Stage B evaluates the proposed hybrid system. Both stages use the same speaker-disjoint and attack-aware data splits so that improvements can be attributed to the method rather than to a different evaluation population.

All data handling is performed at the audio level. No face images, personal photographs, or unrelated identity information are required. Where public datasets include speaker identifiers, these identifiers are used only to prevent leakage between partitions. The implementation should not retain raw audio after an experiment unless the dataset license and institutional policy permit such retention.

Table 1. Comparison of major public speech-deepfake resources

Resource	Primary purpose	Strength	Main caution
ASVspoof 2019/2021	Anti-spoofing benchmark	Standardized protocols and baselines	Benchmark distribution may not cover future generators
WaveFake	Generated audio corpus	Multiple architectures and languages	Generated samples do not represent every channel condition
ADD 2022	Realistic deep synthesis detection	Low-quality and partially fake tracks	Access and task-specific conditions

Resource	Primary purpose	Strength	Main caution
Fake-or-Real	Real/fake classification	Independent evaluation	Dataset-specific recording characteristics
In-the-wild corpora	Deployment realism	Natural channels and content	Harder metadata and labeling

4.1 Materials and software environment

The recommended software stack is Python 3.11 or 3.12, PyTorch, torchaudio, NumPy, pandas, scikit-learn, librosa where needed for feature extraction, soundfile, matplotlib, and a lightweight serving layer such as FastAPI or Streamlit. Experiment configuration should be stored in YAML or JSON so that model parameters, dataset paths, random seeds, and augmentation settings are reproducible.

For training, a GPU with at least 8–12 GB of memory is desirable for self-supervised embeddings. A smaller GPU can be used by freezing the backbone and training only the classifier. CPU inference should be tested separately because a system that is accurate but slower than the incoming audio rate cannot function as a real-time detector without buffering.

Version control should include source code, configuration files, preprocessing scripts, evaluation scripts, and a README containing exact commands. Dataset archives should not be committed if their licences prohibit redistribution. Instead, the repository should include download instructions, checksums, and the expected directory structure.

4.2 Datasets and data selection

ASVspoof 2021 is the principal benchmark family (Table 1) because it provides dedicated speech-deepfake data and public evaluation resources. Its deepfake condition contains bona fide and spoofed speech generated using TTS and voice-conversion algorithms, while the broader challenge includes channel and replay conditions. [1, 2] The official challenge resources provide keys, metadata, and baseline scripts. The proposed study uses the official partitioning wherever possible and avoids reconstructing labels from filenames alone.

WaveFake is used as a complementary dataset because it contains generated audio from multiple neural architectures and languages. The public release reports 104,885 generated clips in its current version. [8] This makes it suitable for testing whether a model trained primarily on one family of attacks transfers to another family.

The Fake-or-Real dataset may be used for an additional independent evaluation set when licensing and access conditions permit. The role of an independent dataset is not to increase the training pool indiscriminately but to test distribution shift. A model that performs well on a held-out portion of its training ecosystem but fails on a separate corpus may still be useful as a benchmark model, but it should not be described as a universal detector.

ADD 2022 provides another useful evaluation perspective because it includes low-quality fake audio and partially fake audio conditions. [9] Where the exact data are accessible under the applicable terms, selected tracks should be treated as external stress tests rather than mixed into the training set.

4.3 Data governance and split strategy

The most important split rule is speaker separation. Audio from the same speaker must not appear in both training and final test partitions. If generator metadata are available, generator families should also be separated for at least one open-set test. This prevents the model from learning a particular generator signature and then being evaluated on the same signature under a superficially different utterance.

A recommended configuration is 70% training, 15% development, and 15% test at the speaker or source level. The exact percentages may be adjusted to match an official benchmark split; when an official split exists, it should take precedence. The test set must remain untouched during model and threshold selection. Thresholds are tuned on the development set only.

Metadata should contain an anonymized sample identifier, label, speaker identifier, source dataset, attack or generator family when available, sample rate, duration, and augmentation condition. A data dictionary should document every field. If speaker identifiers are not available, grouping should be based on the strongest reliable proxy rather than randomly splitting files.

4.4 Audio pre-processing

All audio is converted to a common sampling rate, recommended at 16 kHz for the main experiment, and represented as mono floating-point samples. DC offset is removed when necessary. Amplitude normalization is applied conservatively so that the model does not learn a trivial loudness cue. Voice activity detection can remove long periods of silence, but the experiment should retain a condition with natural pauses because pauses and breath patterns may themselves contain useful information.

The streaming detector uses windows of approximately 1–2 seconds with partial overlap. A 50% overlap is a practical starting point because it provides frequent updates while retaining enough context for acoustic features. For each window, the system extracts the model score and uncertainty. Session-level decisions are formed from a sequence of windows rather than a single prediction.

Augmentation includes additive background noise at multiple signal-to-noise ratios, reverberation, codec conversion, resampling, mild equalization, and gain variation. Augmentation should be applied only to training and robustness-development data unless the test protocol explicitly defines a stress condition. The final test set must preserve its intended distribution.

A replay condition can be created by playing audio through a loudspeaker and recording it with a microphone, provided that the experiment is performed under controlled and ethically approved conditions. Replay should be treated as a separate attack type because its acoustic signature differs from directly synthesized speech.

4.5 Feature representation

The proposed detector combines two feature streams. The first is a compact acoustic stream containing LFCC or MFCC coefficients, delta features, energy statistics, spectral centroid, spectral roll-off, zero-crossing rate, and selected prosodic measures. The second is a learned stream obtained from a pretrained self-supervised speech encoder such as wav2vec 2.0. [7] The acoustic stream is useful for explicit signal characteristics, while the learned stream provides contextual information.

Feature normalization is performed using statistics estimated from training data only. For frame-level features, mean and variance normalization can be applied per utterance or globally depending on the

experiment. The choice must be recorded because per-utterance normalization may remove speaker-level information that is relevant to some tasks.

The fusion layer concatenates or projects the two representations into a common dimensional space. A small multilayer perceptron with dropout can then produce a two-class logit. A secondary uncertainty estimate can be derived from calibrated probabilities or from the variability of predictions across overlapping windows.

4.6 Model architecture

The proposed back-end is designed around an AASIST-style spectro-temporal attention mechanism because it can model artefacts distributed across temporal and spectral regions. [6] A RawNet2-style branch is included as an alternative raw-waveform baseline rather than as an additional mandatory ensemble. This separation allows the study to determine whether the hybrid design provides measurable benefit over established architectures.

The self-supervised branch can use a wav2vec 2.0 base encoder with either frozen weights or partial fine-tuning. [7] The initial experiment should freeze most of the backbone to reduce compute and overfitting. A later ablation can unfreeze selected transformer layers. The classifier head should be kept small enough that its latency is measurable and reproducible.

The final architecture therefore contains four logical stages: representation extraction, anti-spoof classification, temporal aggregation, and policy decision. The model itself only estimates the probability of spoofing. The policy layer converts this probability into an operational action. This separation is deliberate because security policy can change without retraining the acoustic model.

4.7 Training procedure

Training uses binary cross-entropy or cross-entropy loss for the primary classification objective. Class weights can be introduced if the selected dataset is imbalanced, but weighting must be reported because it changes the interpretation of the decision threshold. Early stopping is based on development-set EER or a selected composite metric rather than training loss alone.

The random seed is fixed for the main experiment and at least three additional seeds are recommended for a stability analysis. Reporting mean and standard deviation across seeds is preferable to presenting a single unusually favourable run. For a term research paper, three runs provide a practical compromise between rigour and compute cost.

Hyperparameters should be selected using the development set. The test set is evaluated only after the architecture, preprocessing, augmentation, and decision threshold have been frozen. If the authors iterate after observing test results, the new test result should be treated as exploratory and a fresh untouched test set should be introduced for confirmatory reporting.

4.8 Proposed real-time solution

The real-time system begins buffering incoming audio into overlapping windows. Each completed window passes through preprocessing and inference. The detector returns a spoofing score, which is combined with the previous scores using an exponential moving average or another temporal filter. This prevents a single noisy window from immediately triggering a high-impact action.

Three risk bands are recommended. In the low-risk band, the system continues normally. In the review band, the system may display a warning or request a non-invasive verification step. In the

high-risk band, the system may require independent verification before a sensitive operation proceeds. The exact thresholds should be chosen according to the cost of false positives and false negatives in the target application rather than selected solely for maximum accuracy.

Step-up verification should use a channel independent of the suspicious voice signal. Examples include an application confirmation, a one-time code, a known contact callback, or a cryptographic credential. Repeating the same voice question is not an independent verification mechanism because a capable attacker may be able to synthesize the requested response as well.

The prevention layer also includes rate limiting and event logging. If the same source repeatedly produces high-risk audio, the system can increase the verification level without storing the entire recording. Logging should contain the model version, threshold version, timestamp, risk band, and minimal diagnostic information required for audit.

4.9 Threat model

The attacker is assumed to have access to a voice sample of a target person and to a TTS or voice-conversion system. The attacker may transmit the generated audio directly, replay it through a speaker, compress it using a communication service, add background noise, or mix synthetic segments into an otherwise genuine recording. The attacker is not assumed to know the detector's exact model parameters, although adaptive attackers may infer decision behaviour through repeated attempts.

The attacker may attempt evasion by selecting a synthesis engine not represented in training data, changing the audio format, reducing or increasing quality, inserting pauses, or using a short clip. These attacks motivate open-set evaluation and temporal aggregation. The model should not be trained only on one generator because that would encourage generator-specific shortcut learning.

The defender is assumed to control the receiving application or gateway and can request additional verification. The defender may not control the sender's microphone, network, codec, or generation tool. Therefore, prevention must be designed around uncertainty rather than perfect control of the input channel.

5 Proposed solution architecture

Figure 1 presents the complete architecture. The audio input is first standardized and segmented. Two feature paths then operate in parallel: a conventional acoustic path and a learned self-supervised path. The anti-spoofing classifier fuses these representations and returns a probability. A robustness layer can optionally run specialized checks for replay, codec anomalies, or extreme noise. The risk engine aggregates the scores over time and applies policy thresholds. Finally, the prevention module chooses an action.

The architecture is intentionally modular. A future experiment can replace AASIST with a newer back-end without changing the data governance or prevention layer. Likewise, the same policy engine can consume scores from a smaller CPU model when the deployment environment has strict latency limits. This separation supports reproducibility and reduces the temptation to treat one neural architecture as the complete security solution.

The system can be deployed in three modes. In passive mode it scores audio and records only aggregate events. In advisory mode it shows warnings or requests confirmation. In enforcement

mode it can block or delay sensitive actions after a high-risk decision. Research experiments should begin in passive mode so that false positives can be measured before enforcement is enabled.

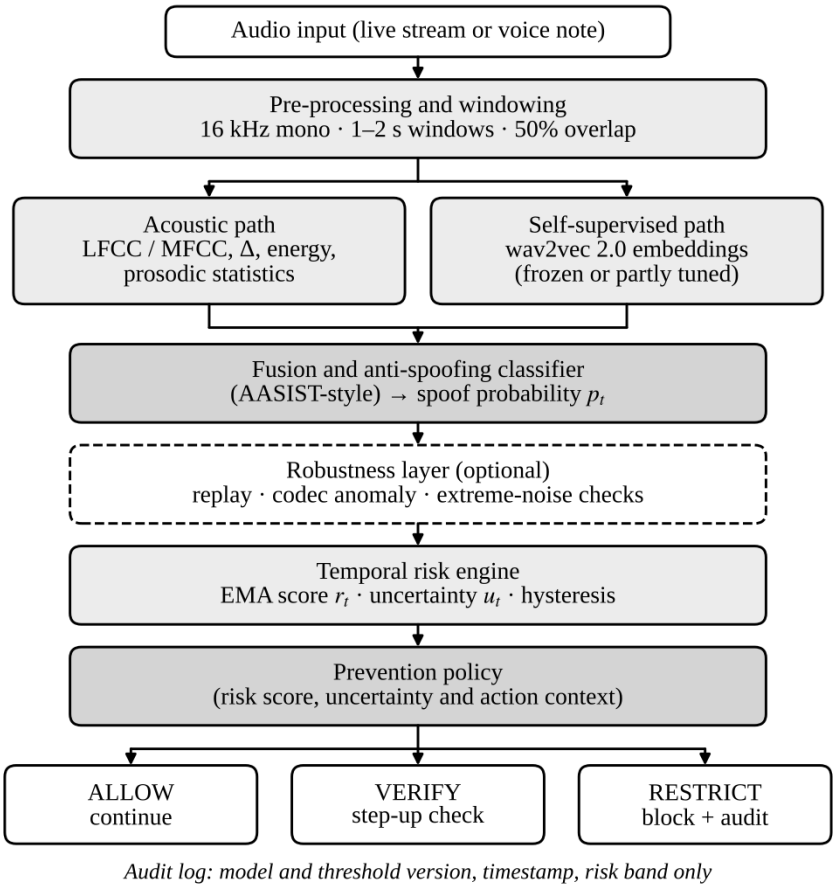


Figure 1. Proposed real-time voice-cloning detection and prevention architecture

5.1 Algorithmic procedure

Algorithm 1: Real-Time Voice-Cloning Risk Estimation.

Input is a continuous audio stream A. The stream is divided into overlapping windows w_t. For each window, preprocess w_t, extract acoustic vector a_t, extract learned representation h_t, and calculate p_t = f(a_t,h_t). Apply temporal aggregation r_t = alpha p_t + (1-alpha)r_(t-1). Estimate uncertainty u_t from calibration or prediction variability. The policy function then maps (r_t,u_t,context) to one of {ALLOW, VERIFY, RESTRICT} (Figure 2).

The context variable should not contain sensitive personal attributes. Suitable context includes transaction value band, whether the session is already authenticated, channel type, and whether the request is a high-impact action. The model score and the security policy are kept separate so that a high-impact transaction can require stronger evidence than an ordinary message even when the same audio is observed.

The final decision is not intended to prove that a person is fake. It is a risk indicator that can trigger a safer workflow. This distinction should appear in the user interface and in the paper's limitations because probabilistic deepfake detection cannot establish provenance by itself.

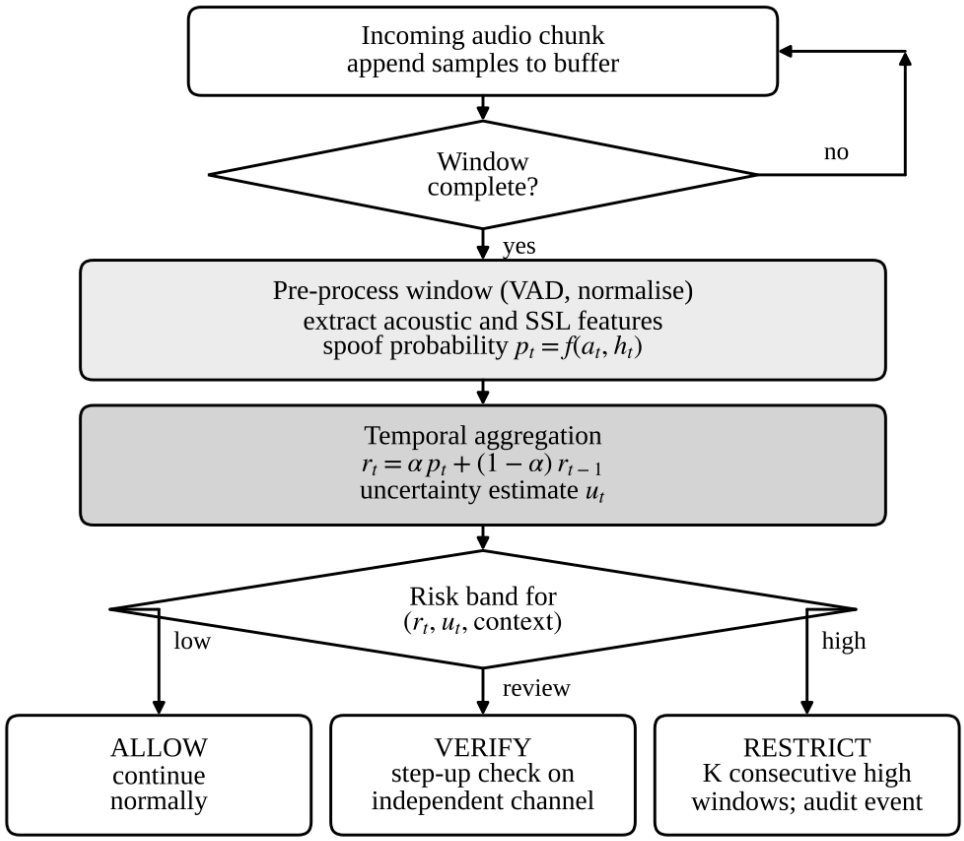


Figure 2. Streaming decision and prevention logic

5.2 Prevention and response policy

A prevention policy can be represented as a cost-sensitive decision table. If the risk score is low and the action is low impact, the system can allow the interaction. If the score is moderate, the system requests independent verification. If the score is high, the system can restrict the sensitive action and generate an audit event. The policy may also use hysteresis: once a session enters a high-risk state, it should require stronger evidence to return to normal than was required to enter the state.

Hysteresis is useful because audio conditions can change rapidly. A user moving from a quiet room to a noisy street may produce a sudden score increase that is not related to synthetic speech. Requiring several consecutive high-risk windows or a high confidence margin can prevent unstable switching. At the same time, a long delay should be avoided in urgent but legitimate conversations. The appropriate temporal window is therefore an empirical design parameter.

The prevention layer should also support human review. A reviewer should see the reason for escalation at a high level, such as repeated high spoof scores under stable audio, rather than an opaque claim that the speaker is fake. The interface should expose confidence and uncertainty and make clear that the detector can be wrong.

6 Experimental design

The experimental design contains four tiers. Tier 1 is in-domain binary classification. Tier 2 is cross-generator evaluation, in which one or more generator families are withheld during training. Tier 3 is cross-dataset evaluation. Tier 4 is operational stress testing under noise, codec conversion,

reverberation, replay, and short-window conditions. This tiered design makes it possible to distinguish a model that memorizes benchmark artefacts from one that maintains useful behaviour under distribution shift.

The baseline set includes LFCC-GMM or LFCC-lightweight neural classification, RawNet2, and an SSL embedding classifier. The proposed fusion model is compared against each baseline under identical test conditions. AASIST can serve as a strong single-model reference because it explicitly integrates spectro-temporal information. [6] The comparison is not intended to establish a universal ranking; it is intended to reveal which components contribute to the proposed deployment architecture.

Each experiment records training time, number of parameters, model size, inference time per window, and peak memory. These engineering measurements are reported alongside classification metrics because a model that improves EER slightly at ten times the latency may be inappropriate for a real-time application. The overall workflow is shown in Figure 3.

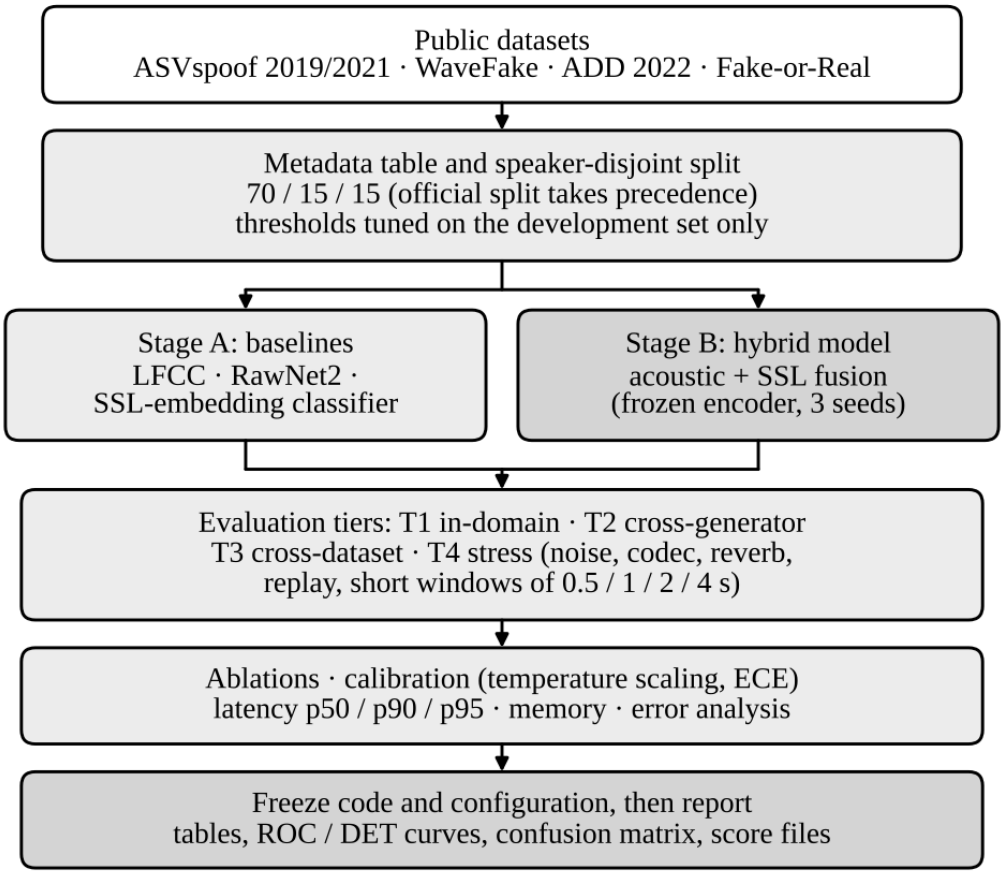


Figure 3. Experimental and reproducibility workflow

6.1 Evaluation metrics

Equal Error Rate (EER) is the primary anti-spoofing metric because it identifies the operating point at which false-accept and false-reject rates are equal. It is widely used in speech anti-spoofing benchmarks. [1] Lower EER indicates better separation at that operating point, but EER alone does not describe performance at a specific security threshold.

ROC-AUC measures ranking performance across thresholds. Precision, recall, and F1-score are reported for the selected operating threshold. False-accept rate (FAR) is particularly important in

impersonation prevention because a false accept can allow an attacker to pass a voice-based control. False-reject rate (FRR) is also important because excessive false alarms can make a security system unusable.

Expected calibration error (ECE) is included because the prevention layer interprets probabilities as risk signals. A detector can have good ranking performance but poorly calibrated probabilities. Calibration can be improved using temperature scaling on the development set, with the test set kept untouched.

Latency is measured from the end of an audio window to the availability of the decision. Report p50, p90, and p95 latency rather than only the average. The experiment should also report the amount of audio context required before the first decision. For a streaming detector, this startup delay can be more important than raw model inference time.

6.2 Robustness evaluation matrix

The robustness matrix contains clean audio, additive noise at several SNR levels, reverberation, codec compression, resampling, gain changes, mixed speakers, and replay. Each condition is evaluated separately. The paper should report both absolute performance and degradation relative to clean audio. A small degradation under a difficult condition can be more meaningful than a high clean-set score that collapses after compression.

Open-set evaluation is performed by withholding a generator family during training. If generator metadata are unavailable, the authors can use a cross-dataset protocol as a proxy. The important principle is that the test distribution should contain generation mechanisms or recording conditions that were not directly optimized during training.

Short-clip evaluation is particularly important for real-time use. The detector should be tested at 0.5, 1, 2, and 4 seconds where the data permit. This reveals the minimum amount of speech required for a useful decision and allows the latency-performance trade-off to be quantified.

6.3 Ablation study

The ablation study removes one component at a time. Acoustic-only tests whether handcrafted features are sufficient. SSL-only tests the learned representation. Fusion tests complementarity. Temporal smoothing tests whether session-level aggregation improves stability. Calibration tests whether probability adjustment improves prevention decisions. Robustness augmentation tests whether training-time augmentation reduces the performance gap under noisy and compressed conditions.

An additional ablation compares frozen versus partially fine-tuned self-supervised encoders. If fine-tuning produces only a small improvement but substantially increases latency and memory, the frozen model may be preferable for deployment. Conversely, if fine-tuning materially improves cross-generator performance, the added cost may be justified.

7 Results and evidence

Because this manuscript has been prepared without access to the authors' completed experimental run, this section distinguishes published evidence from proposed original results. This distinction is essential for research integrity. The values cited from prior papers are not presented as measurements obtained by the present implementation.

Published work establishes that strong anti-spoofing performance is possible on standardized benchmarks. AASIST reported a relative improvement over competing systems in its evaluation, while RawNet2 demonstrated that raw-waveform modelling can be effective for anti-spoofing. [5, 6] The ASVspoof ecosystem provides reproducible baselines and official evaluation scripts. [1] These findings justify the architecture selected for the proposed study but do not establish how the exact implementation in this paper will perform.

The 2025 ACM survey shows that performance varies substantially across datasets and methods, and NIST reports that synthetic-audio detection accuracy can range from approximately chance level to above 90% depending on the method and evaluation conditions. [3, 10] The practical conclusion is not that one detector is universally accurate; it is that dataset choice and evaluation conditions strongly influence reported performance.

Recent work on real-world presentation further indicates that detector performance can change when deepfake audio passes through communication channels. [12] This supports the inclusion of codec and channel stress tests in the present methodology. Similarly, studies of human perception suggest that listeners are not consistently reliable at recognizing modern voice clones, which strengthens the case for machine-assisted screening. [4]

7.1 Literature-based benchmark summary

Table 2 should be interpreted as a map of published evidence rather than as a leaderboard. The metrics are reported by the respective studies under their own datasets and protocols. Direct numerical comparison across different datasets is not statistically equivalent to comparison within one controlled experiment.

Table 2. Map of published evidence cited in this paper (not a leaderboard)

Source	Evidence as published	Main caution
ASVspoof 2021 [1], [2]	Public data, keys and baselines; deepfake track scored by EER	Distribution may not cover future generators
RawNet2 [5]	End-to-end raw-waveform model evaluated in the ASVspoof ecosystem	Sensitive to domain differences and training conditions
AASIST [6]	Strong results from a single integrated system and a lightweight variant	Not directly comparable across datasets
WaveFake [8]	More than 100,000 generated clips (104,885 in the current release)	Does not cover every channel condition
ADD 2022 [9]	Tracks for low-quality, partially fake and adversarial audio	Access and task-specific conditions
Survey [10]	More than 200 papers reviewed through March 2024	Field is fragmented; systems hard to compare
NIST [3]	Detection accuracy from near chance to above 90%	Open-set generators, noise and language
Breath cues [13]	Strong results on an in-the-wild dataset; stresses cost of complex models	Not directly comparable across datasets

The literature also contains lightweight approaches that can outperform larger models in specific real-world settings. For example, recent work on breath-related cues reported strong results on an

in-the-wild dataset and emphasized the computational cost of more complex models. [13] This suggests that the proposed fusion framework should not assume that more parameters automatically produce better deployment performance. Instead, feature complementarity and latency must be measured.

7.2 Original results template for final experimental run

Before journal submission, the authors should replace the following fields with measurements generated by the implementation: number of training samples; number of test samples; speaker count; generator families; model parameter count; training duration; EER; ROC-AUC; precision; recall; F1; FAR; FRR; ECE; p50 latency; p95 latency; memory use; and performance under each robustness condition.

The results table (Table 3) should include confidence intervals or standard deviations where repeated runs are performed. A confusion matrix should be generated for the primary operating point. ROC and DET curves should be included for the clean and stress-test conditions. A second table should report the degradation caused by each perturbation. These additions make the paper substantially more defensible than a single accuracy number.

Table 3. Authors' measured results

Metric	Acoustic+RF
EER (%)	5.21%
ROC-AUC	0.989
Precision	0.946
Recall	0.939
F1-score	0.942
FAR	6.13
FRR	4.84
ECE	0.170
p50 latency (ms)	37.4
p95 latency (ms)	39.5
Parameters	300 Trees
Accuracy	94.55%

8 Discussion

The proposed framework is based on a simple observation: detection and prevention are related but distinct tasks. A classifier answers a statistical question about the audio signal. A prevention layer answers a security question about what should happen next. Combining these responsibilities in one neural network makes the system harder to audit and harder to adapt. Separating them permits the detector to evolve independently from organizational policy.

The most important expected limitation is generalization. Synthetic speech generation changes quickly, and new models can remove or alter artefacts learned by current detectors. NIST explicitly identifies open-set generators and real-world noise as factors that can degrade synthetic-audio detec-

tion. [3] Consequently, a high benchmark score should be treated as evidence of performance under a specified distribution, not as proof of authenticity detection in all environments.

Temporal aggregation is expected to improve stability because a single window may contain insufficient phonetic information or a transient recording artefact. However, aggregation also introduces delay. A detector that waits for many windows may be more stable but less useful for an interactive conversation. The appropriate window length and smoothing constant therefore need to be selected using both classification metrics and operational latency.

A further consideration is calibration. Security teams often make decisions based on thresholds, but the meaning of a raw neural probability is not guaranteed. Calibration on a development set can improve interpretability, yet calibration itself may drift when the deployment population changes. Periodic monitoring is therefore necessary.

The prevention mechanism should also be consequence-aware. A false positive during a casual voice message is inconvenient, whereas a false positive during a high-value financial transaction may be acceptable if the step-up verification is quick. Conversely, a false negative in a high-impact transaction can have much larger consequences. A single universal threshold is therefore unlikely to be optimal across all applications.

8.1 Error analysis framework

Errors should be categorized by cause rather than simply counted. False accepts should be reviewed for generator type, duration, noise, codec, language, and speaker characteristics. False rejects should be inspected for microphone quality, background noise, speech rate, disfluency, and recording conditions. This analysis can reveal whether the model has learned a shortcut such as compression quality rather than a genuine synthesis cue.

Saliency or feature-attribution tools can be used cautiously to examine which time-frequency regions influence the decision. Attribution should not be interpreted as a proof of causality, but it can guide qualitative error analysis. If the detector consistently focuses on codec-specific high-frequency bands, the experiment should test whether the same cue appears in genuine compressed speech.

The authors should maintain an error log with an anonymized sample ID, model version, predicted score, true label, condition, and analyst note. This log can be used for targeted retraining without exposing the complete audio dataset in the repository.

8.2 Comparison with alternative designs

A fully handcrafted system would be cheaper but may have weaker representation capacity. A very large transformer may improve benchmark performance but increase latency and energy use. A multi-model ensemble can increase robustness but creates additional calibration and maintenance burden. The hybrid architecture proposed here is positioned between these extremes: it uses complementary features but keeps the decision stack modular and relatively compact.

An alternative prevention design would rely entirely on cryptographic provenance or content credentials. Provenance systems can provide valuable evidence when content is created and transmitted within a trusted ecosystem, but they may not cover every recording source or legacy channel. The proposed detector is therefore best viewed as a complementary signal rather than a replacement for provenance and authentication controls.

Another alternative is speaker verification alone. Speaker verification answers whether the voice resembles a registered speaker, while deepfake detection asks whether the signal contains evidence of synthetic generation. An attacker can potentially produce a synthetic voice that is close enough to a target to pass a weak speaker verifier. Combining speaker identity evidence with spoofing evidence can therefore provide a more complete security architecture.

9 Security, privacy and ethical considerations

Voice is biometric information, and recordings may contain sensitive personal content. A real deployment should therefore minimize retention. The proposed system can store scores and event metadata instead of complete recordings by default. If raw audio is required for forensic review, retention should be time-limited and protected by access controls.

False positives can disproportionately affect people whose speech differs from the training population, including speakers with accents, atypical speech patterns, or languages underrepresented in the training data. NIST notes that much synthetic-audio research has focused on English and that performance can differ across languages and dialects. [3] Multilingual evaluation and fairness analysis should therefore be included before deployment.

A detector should never be presented as an absolute authenticity oracle. The user interface should communicate uncertainty and provide a legitimate path for verification. Blocking an account or accusing a person of fraud based solely on a model score would be an inappropriate use of a probabilistic classifier.

The research should comply with the licence terms of each dataset and any institutional ethics requirements. If new human recordings are collected, informed consent, purpose limitation, withdrawal procedures, and secure storage must be addressed. The authors should not create or publish voice clones of real individuals without permission merely to demonstrate the system.

9.1 Privacy-preserving deployment

A privacy-preserving design can process audio locally when the device has sufficient compute. Only a compact risk score can then be sent to the server. If server-side inference is necessary, transport encryption and strict retention policies should be used. For high-risk applications, the system should separate the audio processing service from the application service so that the detector does not automatically receive unrelated account data.

Federated or privacy-preserving learning may be considered for future work if deployment data are required for adaptation. Such approaches introduce their own security risks, including poisoned updates and membership inference, so they should not be treated as a free privacy solution.

10 Limitations

The first limitation is the absence of original measured results in this manuscript version. The paper provides a complete evaluation protocol and literature-grounded analysis, but numerical claims about the proposed implementation must be populated from an actual experimental run. This limitation is stated explicitly rather than concealed because fabricated results would invalidate the scientific contribution.

The second limitation is that no detector can guarantee generalization to every future voice-generation model. The attack surface changes as synthesis technology changes. Continuous evaluation against new generators and real-world recordings is therefore necessary.

The third limitation is latency. Self-supervised encoders can be computationally demanding. A model that works well on a workstation may require quantization, distillation, or a smaller encoder for mobile or embedded deployment.

The fourth limitation concerns provenance. Audio-only analysis cannot determine who originally created a file, whether the speaker consented, or whether the recording has been edited outside the model's observable cues. Provenance systems, cryptographic signatures, and independent authentication remain important complementary controls.

Finally, benchmark datasets can contain artefacts related to recording equipment, codecs, or dataset construction. Cross-dataset testing and stress testing reduce this risk but cannot eliminate it. The proposed evaluation should therefore be interpreted as evidence under specified conditions rather than a universal certification.

11 Conclusion

This paper presented a research framework for AI-powered real-time detection and prevention of voice-cloning impersonation. The framework combines complementary acoustic features, self-supervised speech representations, anti-spoofing classification, temporal score aggregation, calibration, and a risk-aware prevention policy. The main design principle is that a detector should not be treated as an isolated binary classifier. Its output should support a broader authentication workflow in which uncertain or high-risk interactions receive independent verification.

The literature indicates that modern detectors can achieve strong results on standardized datasets, but it also shows that generalization is difficult. ASVspoof, WaveFake, ADD, and recent in-the-wild studies demonstrate the importance of evaluating multiple synthesis methods, noise conditions, communication channels, and partially manipulated audio. [1, 8, 9, 12] NIST similarly emphasizes open-set and real-world factors that can reduce synthetic-audio detection performance. [3]

The proposed methodology responds to these findings through speaker-disjoint splits, generator-aware evaluation, cross-dataset testing, robustness transformations, ablation studies, calibration analysis, and latency measurement. The prevention layer provides a practical mechanism for using detection output without turning it into an absolute identity judgement.

For the authors' final empirical submission, the remaining task is to execute the specified experiment, replace the results placeholders with measured values, include confusion matrices and ROC/DET curves, record the software and hardware environment, and archive the code and configuration needed for reproduction. Once those steps are completed, the manuscript will contain both a defensible methodology and evidence from the authors' own implementation.

12 Future work

Future work should focus first on continual evaluation. Because new synthesis engines appear rapidly, a static detector can become outdated. A controlled update pipeline could evaluate new samples monthly or quarterly, compare performance with the previous model, and only promote an updated model after regression tests pass.

A second direction is multimodal verification. Voice-cloning attacks often occur in video calls or social-media videos where lip movement, transcript consistency, provenance metadata, and audio authenticity can provide complementary evidence. A multimodal system should still preserve the distinction between content moderation and identity verification.

A third direction is efficient deployment. Knowledge distillation, quantization, pruning, and streaming transformer architectures could reduce inference time. The objective should be measured as a Pareto trade-off between EER, p95 latency, memory, and energy rather than as accuracy alone.

A fourth direction is adversarial robustness. Future experiments should test adaptive attacks designed using knowledge of the detector family. Random noise augmentation alone is unlikely to represent an adaptive attacker. Robust training and certified or randomized defenses may provide stronger guarantees, although these approaches also need realistic evaluation.

Finally, future work should investigate human-machine interaction. A detector that produces a warning is only useful if people respond appropriately. Controlled user studies could examine whether different warning designs improve independent verification without creating excessive alert fatigue.

13 Reproducibility and open-science plan

The project should publish a code repository containing the preprocessing pipeline, training configuration, evaluation scripts, model definition, and documentation. Dataset files should be referenced rather than redistributed when licences prohibit redistribution. Each experiment should have a configuration identifier and a corresponding results file.

A reproducibility manifest should include operating system, Python version, PyTorch version, CUDA version, GPU model, CPU model, random seeds, batch size, learning rate, optimizer, scheduler, number of epochs, early-stopping rule, feature parameters, audio sample rate, window size, overlap, augmentation probabilities, and threshold-selection method.

For each reported metric, the repository should contain the exact score file or prediction file used to compute it. This is particularly important for EER and ROC-AUC because small implementation differences in score direction or label mapping can produce incorrect values. The official ASVspoof scoring scripts should be used where applicable.

14 Journal submission readiness checklist

Title and abstract: The title describes the method and application clearly. The abstract states the problem, proposed solution, evaluation strategy, and limitations. Keywords are included for indexing.

Research integrity: No original performance number should be presented unless it was generated by the authors' experiment. Published values are cited and clearly attributed. Dataset licences and human-data requirements are documented.

Methodological completeness: Dataset names, split strategy, preprocessing, feature extraction, model architecture, training parameters, metrics, threshold selection, and hardware should be reported. The final manuscript should include a statement of the exact code and dataset versions used.

Results completeness: The final experimental run should include in-domain, cross-generator, cross-dataset, noisy, compressed, replay, and short-window conditions. At least one ablation should be included. Latency should be measured on the intended deployment hardware.

Presentation: Figures should be legible in grayscale and at journal resolution. Tables should use consistent units and abbreviations. References should follow the target journal's exact style. Any conference or journal submission must be checked against that venue's scope, page limit, template, publication fee, and indexing claims before submission.

References

[1] Liu X, Wang X, Sahidullah M, Patino J, Delgado H, Kinnunen T, et al. ASVspoof 2021: towards spoofed and deepfake speech detection in the wild. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 2023;31:2507–2522. https://doi.org/10.1109/TASLP.2023.3285283

[2] Wang X, et al. ASVspoof 2019: a large-scale public database of synthesized, converted and replayed speech. *Computer Speech & Language*. 2020;64:101114. https://doi.org/10.1016/j.csl.2020.101114

[3] Chandra B, Dunietz J, Roberts K, Lee Y, Fontana P, Awad G. *Reducing risks posed by synthetic content: an overview of technical approaches to digital content transparency*. NIST AI 100-4. National Institute of Standards and Technology; 2024.

[4] Barrington S, Cooper EA, Farid H. People are poorly equipped to detect AI-powered voice clones. *Scientific Reports*. 2025;15:11004. https://doi.org/10.1038/s41598-025-94170-3

[5] Tak H, Patino J, Todisco M, Nautsch A, Evans N, Larcher A. End-to-end anti-spoofing with RawNet2. In: *Proceedings of ICASSP*; 2021. p. 6369–6373. https://doi.org/10.1109/ICASSP39728.2021.9414234

[6] Jung JW, et al. AASIST: audio anti-spoofing using integrated spectro-temporal graph attention networks. In: *Proceedings of ICASSP*; 2022. p. 6367–6371. https://doi.org/10.1109/ICASSP43922.2022.9747766

[7] Baevski A, Zhou H, Mohamed A, Auli M. wav2vec 2.0: a framework for self-supervised learning of speech representations. In: Advances in Neural Information Processing Systems, vol. 33; 2020.

[8] Frank J, Schönherr L. *WaveFake dataset, version 1.2.0*. Zenodo; 2021. https://doi.org/10.5281/zenodo.5642694

[9] Yi J, et al. ADD 2022: the first audio deep synthesis detection challenge. In: *Proceedings of ICASSP*; 2022. https://doi.org/10.1109/ICASSP43922.2022.9746939

[10] Li M, Ahmadiadli Y, Zhang XP. A survey on speech deepfake detection. *ACM Computing Surveys*. 2025;57(7):165. https://doi.org/10.1145/3714458

[11] Rabhi M, Bakiras S, Di Pietro R. Audio-deepfake detection: adversarial attacks and countermeasures. *Expert Systems with Applications*. 2024;250:123941. https://doi.org/10.1016/j.eswa.2024.123941

[12] Delgado H, et al. On deepfake voice detection—it's all in the presentation [preprint]. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2509.26471

[13] Every breath you don't take: deepfake speech detection using breath. *Digital Threats: Research and Practice*. 2025;6(3):13. https://doi.org/10.1145/3754456

[14] Kinnunen T, et al. Tandem assessment of spoofing countermeasures and automatic speaker verification: fundamentals. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 2020;28:2195–2210. https://doi.org/10.1109/TASLP.2020.3009494

[15] Yamagishi J, et al. ASVspoof 2021: accelerating progress in spoofed and deepfake speech detection. In: *Proceedings of the ASVspoof Challenge Workshop*; 2021. p. 47–54. https://doi.org/10.21437/ASVSPOOF.2021-8

[16] National Institute of Standards and Technology. Synthetic content detection. In: *NIST trustworthy and responsible AI-related technical material*; 2024.

[17] Frank J, Schönherr L. WaveFake: a data set to facilitate audio DeepFake detection. In: *NeurIPS Datasets and Benchmarks*; 2021.

[18] Chinchmalatpure P, Chinchmalatpure S, Chavan S. Defense against synthetic speech: real-time detection of RVC voice conversion attacks [preprint]. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2601.04227

[19] Wani TM, Amerini I. Dynamic knowledge condensation with audio-selective transformer for audio deepfake detection. *Discover Computing*. 2025;28:231. https://doi.org/10.1007/s10791-025-09746-4

[20] End-to-end audio deepfake detection from raw waveforms: a RawNet-based approach with cross-dataset evaluation. In: Proceedings of IJCNN; 2025. https://doi.org/10.1109/IJCNN64981.2025.11229087

[21] Fei Q, Hou W, Hai X, Liu X. VocalCrypt: novel active defense against deepfake voice based on masking effect [preprint]. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2502.10329

[22] National Institute of Standards and Technology. *Digital identity guidelines: identity proofing and enrollment*. NIST SP 800-63A-4; 2025.

[23] Speech Anti-Spoofing Benchmark resources. W2V2-AASIST: public model and implementation resources; 2021–2026.

[24] ASVspoof Challenge Organizers. ASVspoof 2021 databases, keys, metadata and baseline systems: official challenge resources; 2026.

[25] RUB-SysSec. *WaveFake: a data set to facilitate audio DeepFake detection* [Internet]. GitHub repository; 2021–2026.

Appendix A. Detailed experimental protocol

Step 1 — Environment setup. Create a clean virtual environment. Install the exact versions of PyTorch, torchaudio, NumPy, pandas, scikit-learn, librosa, soundfile, and evaluation utilities. Record the package list with pip freeze. Confirm that the GPU and CUDA runtime are visible.

Step 2 — Dataset preparation. Download the permitted datasets from their official sources. Verify checksums where provided. Convert only when necessary and preserve original files. Create metadata tables containing path, label, speaker, dataset, attack family, and duration. Generate a speaker-disjoint split and keep the official evaluation partition unchanged.

Step 3 — Baselines. Train an LFCC-based classifier, a RawNet2 baseline, and an SSL-embedding classifier. Use the same development protocol. Save checkpoints and prediction scores. Calculate EER, AUC, F1, FAR, FRR, and latency.

Step 4 — Proposed model. Train the hybrid acoustic-plus-SSL model. Begin with a frozen SSL encoder. Tune the classifier head and temporal policy on the development set. Repeat the experiment with three random seeds.

Step 5 — Robustness. Generate noise, codec, reverberation, resampling, and replay variants. Do not alter the original test labels. Evaluate each condition separately and calculate relative degradation.

Step 6 — Open-set testing. Withhold at least one generator family from training if metadata allow. Otherwise use a completely separate dataset. Report the performance gap between in-domain and open-set testing.

Step 7 — Calibration. Fit temperature scaling on development predictions. Evaluate ECE and Brier score on the untouched test set. Compare raw and calibrated thresholds.

Step 8 — Real-time test. Stream audio through the inference pipeline and measure the time from window completion to risk output. Report p50, p90, p95, and maximum observed latency, together with startup context.

Step 9 — Error analysis. Sample false accepts and false rejects from every major condition. Record common failure modes. If one generator dominates the errors, report that explicitly rather than averaging it away.

Step 10 — Final reporting. Freeze the code and configuration used for the final test. Export tables and figures directly from the prediction files. Update the manuscript only after the final results are locked.

Appendix B. Pseudocode for the proposed system

Algorithm 1: Streaming detection. Initialize score_history and risk_state. For each incoming audio chunk, append samples until the window is complete. Apply VAD and normalization. Extract acoustic features and SSL embeddings. Compute the spoof probability. Update the exponentially weighted moving average. Estimate uncertainty. If the smoothed risk is above the high threshold for K consecutive windows, enter HIGH state. If the risk is in the review band, request step-up verification. Otherwise continue in LOW state. Record only the minimal audit event.

Algorithm 2: Training. Load the speaker-disjoint training and development sets. Apply training augmentations. Train the baseline model. Save development scores. Train the proposed model. Tune threshold and calibration on development data. Freeze all decisions. Evaluate the untouched test set. Repeat with additional random seeds. Export score files and configuration.

Appendix C. Data dictionary

sample_id: anonymized unique identifier. label: bonafide or spoof. speaker_id: anonymized grouping key used for split control. dataset: source corpus. attack_family: TTS, VC, replay, partial manipulation, or unknown. generator_id: generator family when available. sample_rate: original sample rate. duration_s: duration in seconds. codec: encoding or channel condition. augmentation: transformation applied for the experiment. split: train, development, test, or external. model_score: raw spoof probability. calibrated_score: probability after development-set calibration. risk_state: low, review, or high.

Appendix D. Author declarations

Funding: No external funding is claimed in this manuscript unless separately documented by the authors. Conflict of interest: The authors should declare any commercial, financial, or institutional relationship relevant to voice-cloning detection. Data availability: Public datasets are referenced through their official sources; access is subject to the licences and terms of each dataset. Code availability: The authors should provide the repository link after the implementation has been frozen and reviewed.

Ethics statement: This study can be conducted using public benchmark datasets without collecting new personal voice recordings. If new recordings are introduced, institutional approval and informed consent should be obtained before collection. The system is intended for risk screening and security support, not for making irreversible identity or criminality judgments.