ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 3 - 2026

A LSH-Based Detection Algorithm for the Adversarial Samples of Early-Stage

Jia Zheng[1], *Hua Zhang*[1,*], *Ming Lv*[2], *Wanjin Hou*[1], *Huiting Hou*[2] *and Lili Zhang*[3]

[1] School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100044, China
[2] Network Business Unit, China Mobile Communications Group Corporation, Beijing 100032, China
[3] Operations and Services Department, China Satellite Network Application Co., Ltd., Beijing, 100071 China

SUMMARY: *Addressing the poor performance of detection on early-stage adversarial samples in speaker recognition tasks, we propose a detection algorithm based on locality sensitive hashing (LSH-GPD) for adversarial sample of early-stage during the generation process in this paper. Firstly, a locality-sensitive hashing algorithm tailored to adversarial audio samples is designed, achieving rapid retrieval of similar speech samples and addressing the challenge of similarity search in high-dimensional speech data. Subsequently, the FastDTW algorithm is utilized to further group similar speech samples, eliminating hash collisions. Finally, adversarial samples are identified by determining whether the number of samples within each group exceeds a threshold. Adversarial samples generated on public datasets, the TIMIT, Common Voice, and Voxceleb2, are used to evaluate the LSH-GPD algorithm. Experimental results demonstrate that the LSH-GPD algorithm outperforms both the Voting and WaveGuard algorithms, the accuracy of detection for adversarial samples achieves 99.9%, and an early detection rate achieves 99.0%.*

KEYWORDS: *Adversarial Sample Detection; Locality-Sensitive Hashing; Speaker Recognition*

1 Introduction

Speaker Recognition systems, utilized for verifying the identity of speakers, play a pivotal role in security authentication [1]. However, they also confront numerous challenges, notably the threat posed by adversarial samples [2]. Adversarial samples, generated by introducing subtle perturbations into the original speech, can induce erroneous judgments in Speaker Recognition systems, thereby exposing the systems to the risk of unauthorized access [3]. To address the threats of adversarial samples, detection algorithms based on adversarial training and based on network verification were proposed [4].

On the one hand, methods based on adversarial training [5, 6] train speaker recognition models by mixing normal samples with adversarial samples. The trained models can detect adversarial samples with high accuracy and demonstrate robust performance. However, these algorithms rely on the characteristics of adversarial samples in the training dataset, resulting in poor performance when detecting unknown attacks [7]. Additionally, each update of a model

*zhanghua_288@bupt.edu.cn
https://doi.org/10.65102/is20261088
Published: 10 June, 2026

requires re-training, leading to instability of model [4]. On the other hand, methods based on network verification [8-11] detect adversarial samples by comparing the output differences of speech samples before and after processing or the output differences of a speech sample across multiple models. They can achieve high accuracy and good generalization ability against adversarial samples generated by various attack methods, effectively detecting unknown attacks without affecting the original model. However, these algorithms rely on the targeted model, resulting in high computational costs and reduced the efficiency of detection. The above methods typically detect adversarial samples when they are nearly complete, failing to provide early-stage warnings during the early-stage of an attack.

To address the aforementioned issues, an algorithm based on locality-sensitive hashing is proposed for detecting adversarial sample of early stages during the generation process in this paper, named the LSH-GPD algorithm. Adversarial sample generation methods based on optimization produce a large number of similar speech samples during the attack process, so this algorithm can identify the adversarial sample of early stages by comparing these similar speech samples. Tailored to the characteristics of adversarial samples, an adversarial locality-sensitive hashing algorithm is designed, named Ad-LSH. It takes the FBANK features of speech samples as input. Firstly, it normalizes the frequency intensity to mitigate the impact of perturbations on the frequency strength. Secondly, the mean frequency intensity is calculated along the time dimension to capture the overall characteristics of the audio signal and reduce sensitivity to local variations. Finally, the mean frequency intensity is processed through a unit step function to obtain a feature string consisting only of 0s and 1s, which serves as the hash value of the speech sample. The hash values generated by the Ad-LSH algorithm encapsulate the global feature representation of the audio signal in the frequency domain, enabling effective differentiation between different speakers while reducing sensitivity to minute perturbations. Furthermore, by mapping similar speech samples to the same hash value, the algorithm facilitates rapid search for similar speech samples within vast speech databases.

2 Related research

2.1 Adversarial Sample Defense Methods

(1) Adversarial Training-based Methods

These methods enhances model robustnesss by incorporating adversarial samples into training sets, which notably improves defense capabilities against adversarial samples.

Samizade et al. [12] approached adversarial sample detection as a classification problem, developing a CNN-based neural network as a standalone module to classify normal and adversarial samples. While achieving nearly 100% accuracy in detecting attacks from Carlini et al. [13] and Alzantot et al. [14], this method did not evaluate against more advanced attack methods. Sun et al. [5] integrated adversarial samples generated by the Fast Gradient Sign Method (FGSM) algorithm [15] into speaker recognition training sets, enabling models to differentiate normal and adversarial samples. Although maintaining normal sample classification accuracy, the method showed limited effectiveness against unknown attacks [4]. To enhance the defensive capability of adversarial training, Pal et al. [16] proposed a defense mechanism based on mixed adversarial training, with multi-task objectives including cross-entropy, feature scattering, and margin loss. This method can defend against some black-box attacks while improving adversarial robustness. However, these methods require retraining models with new adversarial samples, leading to computational resource waste and model instability [17].

(2)Network Verification-based Methods

These methods detect adversarial samples by comparing outputs differences of speaker recognition models. They do not rely on specific model architectures, achieving good generalization capabilities and can be widely applied to various types of speaker recognition systems. Two strategies are implemented: The first involves inputting the original speech samples into multiple speaker recognition models, then detect adversarial samples by comparing the differences in their outputs. The second involves transforming the original speech samples, then detect adversarial samples by comparing the differences in the outputs of the same model before and after the transformation.

Tamura et al. [9] leveraged the weak transferability of adversarial samples across multiple models, proposing an adversarial sample detection method with multiple sub-detectors. These sub-detectors include those based on dynamic sampling, denoising, and temporal dependency. Each sub-detector independently outputs a detection result. The system conclusively identifies an input speech as an adversarial sample when over 50% of the sub-detectors classify it as such. Experimental evaluation using false negative rates (FNR) and false positive rates (FPR) metrics demonstrated 0.78% FNR and 0.18% FPR under optimal thresholds, though this trade-off between FNR and FPR needs to be considered based on the application scenario. Inspired by N-version programming principle, Zeng et al. [18] detected adversarial samples through differential analysis of the same input speech's transcription results across distinct speech recognition systems. This method achieved 98.6% detection accuracy against attacks proposed by Carlini et al. [19] and Taori et al. [20].

Rajaratnam et al. [8] preprocessed speech using audio compression, audio encoding, filtering, and audio shifting to disrupt perturbations in adversarial samples. By comparing the differences in speaker recognition model outputs before and after preprocessing, they distinguished between normal samples and adversarial samples, achieving a precision of 93.5% and a recall rate of 91.2% in adversarial sample detection. Hussain et al. [7] detected adversarial samples by analyzing output differences between original and transformed audio, achieving 100% detection accuracy under optimal hyperparameters. Wu et al. [10] identified adversarial samples using a voting method. Firstly, they generated multiple neighbor samples by adding Gaussian noise to the input samples. Secondly, if the input is adversarial samples, the superposition of Gaussian noise and adversarial noise may neutralize adversarial perturbations in the neighbor samples. Thirdly, they used the decision results of the input sample and neighbor samples for voting. Finally, they determined whether the input sample was an adversarial sample based on the voting results. On the Common Voice dataset, with one neighbor, the method achieved a false acceptance rate (FAR) of 2.25% and a false rejection rate (FRR) of 2.27%. As the number of neighbors increased, the FAR gradually decreased, while the FRR gradually increased.

The aforementioned methods demonstrates weak detection capabilities during early-stage adversarial sample generation, typically detecting adversarial samples only near generation process completion. This detection delay causes defense mechanisms to lag behind attacks, failing to provide early-stage warnings in the early stages and leaving insufficient time for formulating countermeasures, thereby reducing system security.

2.2 Locality-Sensitive Hashing

Locality-Sensitive Hashing (LSH) is a hashing technique for processing high-dimensional data, mapping similar data points to the same hash value during hashing to enable rapid similar speech search in high-dimensional space. Common LSH algorithm in the audio domain include Shazam, Waveprint, AudioHash, and Spectral Hashing. The Shazam algorithm firstly samples and filters the audio, then converts it from time to frequency domain via Fast Fourier Transform (FFT), finally identifying prominent spectral peaks as audio fingerprint. The Waveprint

algorithm combines audio and image processing techniques. Firstly, it converts the audio signals to image format. Secondly, it leverages image processing techniques to extract image features, which are used to generate binary hash codes as fingerprints. The AudioHash algorithm focuses on identifying key moments and frequencies in the audio to generate speech hash value. The Spectral Hashing algorithm firstly extracts spectral features from the audio and constructs similarity matrices representing audio segments. Finally, it utilizes spectral decomposition to extract principal spectral components and defines a hash function to map the spectral features to compact binary codes as hash values.

These LSH algorithms are rimarily used for music recognition or audio identification. Their features focus differs from speaker recognition requirements, failing to accurately capture the complex acoustic characteristics of speakers. This may cause misidentification of some highly similar non-adversarial speech samples as adversarial ones, such as same speaker speech in different environments, emotional variations, or minor vocal changes.

3 Detection Algorithm for the Adversarial Samples of Early-Stage

3.1 Locality-Sensitive Hashing Algorithm for the Adversarial Samples

Optimization-based adversarial sample generation methods produce adversarial samples via multiple iterations. During iterations, generated speech samples remain extremely close in the high-dimensional feature space. These adversarial samples can be detected by identifying similar samples within this high-dimensional space. However, the high-dimensional fnature and large-scale characteristics of speech data impose excessive computational and storage costs on traditional search methods, making real-time detection challenging.

Tailored to the characteristics of adversarial samples, an adversarial locality-sensitive hashing algorithm is designed, named Ad-LSH. By integrating temporal and frequency information from speech samples to extract global features, Ad-LSH maps proximate high-dimensional data points to the same hash value, thereby enabling rapid search for similar speeches. The Ad-LSH algorithm takes FBANK features of raw speech samples as input and, through three steps—frequency normalization, calculating the mean window frequency, and unit step function processing—generates a 32-character string. The detailed workflow is shown in Algorithm 1.

Algorithm 1 Ad-LSH algorithm
Input:FABNK feature *fbank* of speech samples
Output:hash values *hash*
1)$M \leftarrow len(fbank)$
2)$N \leftarrow len(fbank[0])$
3)// Step1: Frequency normalization
4)**for** i = 1 to M **do**
5) **for** j = 1 to N **do**
6) $fbank[i][j] \leftarrow 2 * \left(\frac{fbank[i][j] - min\_j}{max\_j - min\_j}\right) - 1$
7) **end for**
8)**end for**
9)// Step2: Calculate the mean window frequency
10)**for** j = 1 to N **do**
11) $frequency\_mean[j] \leftarrow \frac{1}{M}\sum_{i=1}^{M} fbank[i][j]$
12)**end for**

```
13)// Step3: Unit step function processing
14)for j = 1 to N do
15)     if frequency_mean[j] ≥ 0 then
16)         hash_array[j] ← 1
17)     else
18)         hash_array[j] ← 0
19)     end if
20)end for
21)hash ← concat(hash_array)
22)return hash
```

(1)Frequency normalization

The input to the Ad-LSH algorithm is the FBANK feature, denoted as *fbank*, of speech samples. *Fbank*, a two-dimensional array, has its dimensions denoted as time and frequency. The length of the time dimension, denoted as *M*, and the length of the frequency dimension, denoted as *N*. The time dimension corresponds to speech frames, each denoting a specific time interval. The frequency dimension denotes the frequency components obtained through Fourier transform for each frame, with each component corresponding to a specific frequency.

The Ad-LSH algorithm performs frequency normalization by constraining the frequency intensity corresponding to each frequency component within the range of -1 to 1, thereby reducing perturbation impacts. For the *j*-th frequency component, the Ad-LSH algorithm calculates the maximum value *max_j* and the minimum value *min_j* of the frequency intensity across all frames. The normalization process is shown in Equation 1.

$$fbank[i][j] \leftarrow 2 * \left(\frac{fbank[i][j] - min\_j}{max\_j - min\_j} \right) - 1 \tag{1}$$

(2)Calculate the mean window frequency

The Ad-LSH algorithm simplifies the two-dimensional array into a one-dimensional array by calculating the mean of each frequency component across the time dimension. This one-dimensional array provides global feature information of the speech sample. Each frequency component's mean reflects its mean intensity throughout the entire speech signal, encapsulating the overall characteristics of the speech signal and reducing local variation sensitivity. The processing is shown in Equation 2, where *frequency_mean* denotes the one-dimensional array generated after calculating the mean window frequency, and *M* denotes the length of the time dimension in the *fbank*.

$$frequency\_mean[j] \leftarrow \frac{1}{M} \sum_{i=1}^{M} fbank[i][j] \tag{2}$$

(3) Unit step function processing

The function performs a simple binarization process on the input values. If $\text{input} \geq 0$, it outputs 1; otherwise, it outputs 0, as shown in Equation 3. The Ad-LSH algorithm leverages the unit step function to convert the *frequency_mean* into discrete values of 0 and 1, thereby reducing the data complexity and facilitating the analysis of the positive and negative characteristics of the frequency mean. The *hash_array* denotes the one-dimensional array generated after processing with the unit step function.

$$\text{hash\_array}[j]=\begin{cases}1 & \text{frequency\_mean}[j] \geq 0\\ 0 & \text{frequency\_mean}[j] < 0\end{cases} \tag{3}$$

Finally, the Ad-LSH algorithm concatenates each frequency mean's binary data into a string *hash*, serving as the hash value of the speech sample. The length of the hash value is identical to that of the *fbank* frequency dimension. This hash value captures mean frequency intensity variations across different speech frames succinctly describing the frequency characteristics while reducing local variations sensitivity, thus making it suitable for similarity searches in adversarial sample.

3.2 LSH-GPD Detection Algorithm for the Adversarial Sample

We propose a Ad-LSH-Based detection algorithm for adversarial samples of early-stage during the generation process in the gray-box setting, named the LSH-GPD algorithm. The detailed workflow of the LSH-GPD algorithm, as shown in Figure 1, consists of three steps: extracting FBANK features, calculating hash values, and identifying adversarial samples.

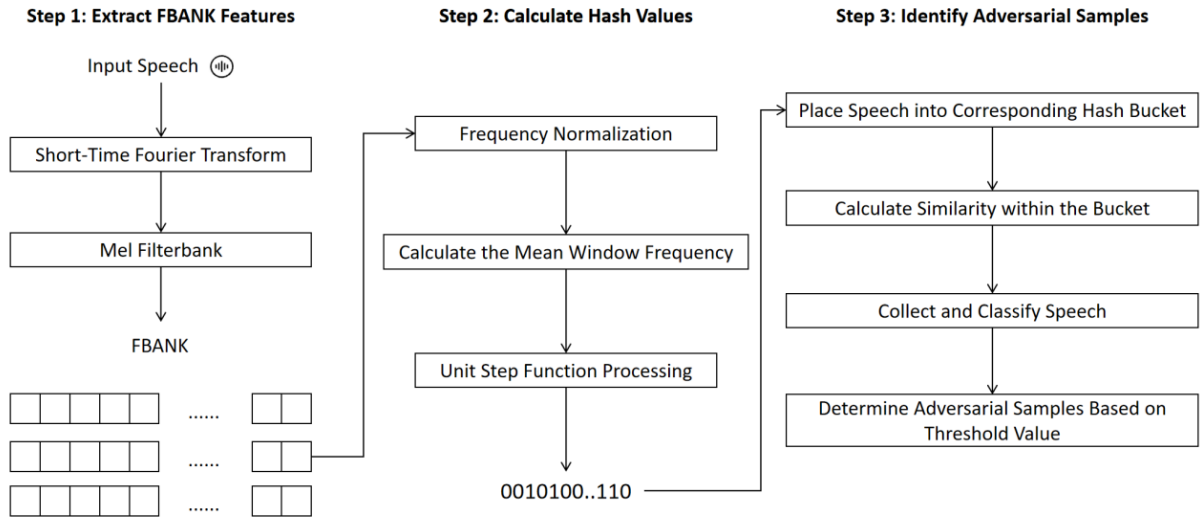

Figure 1: LSH-GPD Algorithm Process

Step 1: Extract FBANK Features

FBANK features are standard feature representation in speech signal processing, widely applied in various speech-related tasks. Firstly, the LSH-GPD algorithm segments the input speech sample into frames, with partial overlap between adjacent frames. Secondly, each frame is processed using a Hanning window to maintain signal continuity and smoothness, preventing edge information loss. Thirdly, Fourier Transform is applied to each pre-processed speech frame, converting signals from time-domain to frequency-domain energy distribution. Lastly, the frequency-domain information is processed through Mel filterbanks, capturing the speech signal's features by simulating the human ear's perception of frequency. The resulting FBANK features form a two-dimensional matrix, reflecting energy distribution across different frequencies and times in the speech signal. During the framing process, the LSH-GPD algorithm utilizes frame parameters include 2048-sample length and 512-sample shift.

Step 2: Calculate Hash Values

The LSH-GPD algorithm leverages the Ad-LSH algorithm to calculate hash values for speech samples, where these hash values denote the global features of the speech samples.

Step 3: Identify Adversarial Samples

The LSH-GPD algorithm maps the hash values obtained in Step 2 into hash buckets, aggregating speech samples with similar features for rapid search. When the number of speech samples in a hash bucket exceeds a predefined threshold, the LSH-GPD algorithm considers that there may be adversarial samples in that bucket. However, since Ad-LSH algorithm may produce hash collisions causing dissimilar speech samples sharing identical hash bucket, it is necessary to conduct further comparison of the speech samples within the bucket.

The LSH-GPD algorithm firstly utilizes the Fast Dynamic Time Warping (FastDTW) algorithm to calculate the pairwise similarity among speech samples within each bucket, thereby precisely identifying feature-level similar samples. Secondly, speech samples with high similarity are grouped into the same cluster. Finally, the algorithm inspects the number of speech samples in each cluster. If the number of speech samples in a cluster exceeds a predefined threshold, the speech samples within it are classified as adversarial samples. Specifically, the LSH-GPD algorithm sets this threshold to 10; that is, when the number of speech samples in a cluster exceeds 10, those samples are classified as adversarial samples. The detailed workflow is shown in Figure 2.

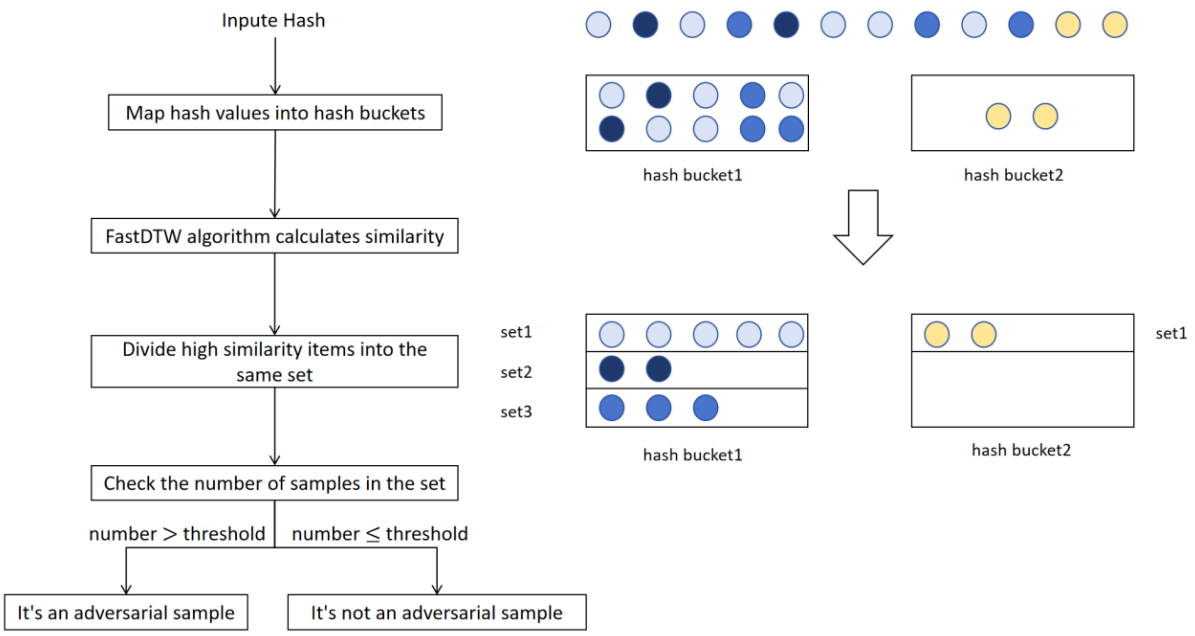

Figure 2: The The detailed workflow of identifying adversarial samples

The workflow of LSH-GPD algorithm is shown in Algorithm 2.

Algorithm2 LSH-GPD algorithm
Input: Speech sample set *speech*,Frame length *frame_length*,Frame shift *frame_shift*,threshold value *threshold*
Output: Adversarial samples *adversarial_samples*
1）// Initialize hash bucket and adversarial sample set
2)*buckets* ← null
3)*adversarial_samples* ← null
4)// Traverse set *speech*
5)**for** each *s* in *speech* **do**
6) // Step 1: Extract FBANK Features
7) *frames* ← null

```
    fbank ← null

    // Frame and apply Hanning window
    for i = 0 to (len(s) - frame_length) / frame_shift do
        frame ← s[i * frame_shift : i * frame_shift + frame_length]
        frame ← Hanning_window(frame)
        frames.append(frame)
    end for

    // Perform Fourier transform and Mel filtering
    for frame in frames do
        fft_result ← FFT(frame)
        power_spectrum ← abs(fft_result) ^ 2
        mel_filter_output ← Mel_Filterbank(power_spectrum)
        fbank.append(mel_filter_output)
    end for

    // Step 2: Calculate Hash Values
    hash ← Ad_LSH(fbank)

    // Step 3: Identify Adversarial Samples
    // Map hash values to hash buckets
    if hash not in buckets then
        buckets[hash] ← null
    end if
    buckets[hash].append(s)
end for
for each bucket in buckets.values() do
    if len(bucket) > threshold then
        // Initialize similarity matrix
        similarities ← null(len(bucket), len(bucket))

        // Calculate the similarity between pairs of speech samples in the bucket
        for i = 1 to len(bucket)-1 do
            for j = i+1 to len(bucket) do
                similarity ← FastDTW(bucket[i], bucket[j])
                similarities[i][j] ← similarity
                similarities[j][i] ← similarity
            end for
        end for

        // Divide the set based on similarity
        clusters ← ClusterBySimilarity(similarities, threshold)

        // Check the samples in each set
        for each cluster in clusters do
            if len(cluster) > threshold then
                // Add adversarial samples to the set adversarial_samples
                adversarial_samples.extend(cluster)
```

```
            end if
        end for
    end if
end for
return adversarial_samples
```

4 Experiments and Results

4.1 Experimental Setup

(1) Experimental Environment and Datasets

In our experiments, the operating system is Ubuntu 20.04.6 LTS. The primary software environment utilizes Python 3.9, PyTorch 2.1.0, transformers 4.35.2, speechbrain 0.5.14, and nemo-toolkit 1.18.1. The hardware configuration utilizes a GeForce GTX 3060 12G GPU and a 13th Gen Intel(R) Core(TM) i5-13600K CPU at 3.50GHz with 32GB RAM.

We select three public speech datasets, TIMIT[21], Common Voice[22], and VoxCeleb2[23]. For each dataset, the speech is first classified according to speaker labels. Subsequently, 1000 pairs of speaker labels are randomly selected, and corresponding speech samples are randomly chosen for each pair. Finally, the speech samples corresponding to each pair of labels serve as the input and the targeted speech samples for the attack algorithm.

(2)Models

In our experiments, two speaker recognition models are selected, one based on the ECAPA-TDNN [16] architecture and the other based on the TitaNet [17] architecture. The two architectures achieve low equal error rates(EER), demonstrating strong recognition performance.

The model based on the ECAPA-TDNN architecture [24] leverages a Time-Delay Neural Network (TDNN) as its core. By introducing an attention mechanism in key layers, it enhances the ability to capture speaker-specific features, effectively identifying speakers. Its EER on the VoxCeleb2 dataset is 0.87%. The model based on the TitaNet [25] architecture leverages one-dimensional depthwise separable convolutions and a squeeze-and-excitation layer that incorporates global context. This architecture map variable-length speech segments into fixed-length vector representations, achieving an EER of 0.68% on the VoxCeleb1 dataset.

NVIDIA NeMo, an open-source deep learning framework developed by NVIDIA, focuses on accelerating research and application development in AI domains including speech recognition, natural language processing, and speech synthesis. This framework employs a modular design philosophy, simplifying the construction and training processes of complex AI models through pre-built reusable modules. In our experiments, we leverage the pre-trained speaker recognition models based on the ECAPA-TDNN and TitaNet architectures provided by the NVIDIA NeMo framework as the targeted attack models for adversarial sample generation.

(3)Baseline

In our experiments, we leverage the Voting and WaveGuard algorithm as baselines to compare with the LSH-Based detection algorithm for the adversarial samples.

The Voting algorithm proposed by Wu et al. [10] generates multiple neighbor samples by adding Gaussian noise to the input sample. If the input is an adversarial sample, the superposition of Gaussian noise and adversarial noise may neutralize adversarial perturbations in the neighbor samples. Subsequently, the decision results of the input sample and neighbor samples are used for voting to identify adversarial samples.

The WaveGuard algorithm proposed by Hussain et al. [7] processes input speech using an

audio transformation algorithm and identifies adversarial samples by comparing the decision results between original and transformed speech. In our experiment, the audio transformation algorithm leverages a filtering algorithm, which can reduce frequency amplitudes beyond the corresponding threshold, thereby eliminating potential adversarial perturbations.

(4)Attack Algorithms

In our experiment, we leverage the gradient-based FakeBob attack algorithm and the gradient-free genetic algorithm to attack the above models for adversarial samples generation.

Chen et al. [26] proposed the FakeBob attack algorithm, which leverages gradient information to generate adversarial samples. Inspired by Carlini et al. [27], the generation process of adversarial samples is formulated as an optimization problem, balancing the strength and imperceptibility of adversarial samples by combining loss function of confidence scores and maximum distortion. They utilize a natural evolution strategy to estimate gradients and optimize the samples, iteratively generating adversarial samples through iterative optimization.

Alzantot et al. [28] leveraged a genetic algorithm to generate adversarial samples targeting intelligent speech software. They utilized the scores on the targeted labels as fitness scores, where high-scoring members undergo crossover and mutation to produce the next generation of adversarial samples, iterating continuously until the attack succeeded. In our experiment, the methodology from Alzantot et al. [20] is applied to speaker recognition, where the confidence scores of input speech samples are used as fitness scores to iteratively generate adversarial samples.

(5)Metrics

This experiment employs five metrics, - Accuracy, False Negative Rate(FNR), False Positive Rate(FPR) [18], Average Detection Stride(ADS), and Early Detection Rate(EDR) - to quantitatively analyze the LSH-GPD, Voting, and WaveGuard algorithm. Among them, ADS and EDR are two metrics proposed in this paper, specifically designed to assess the detection capability of algorithms for adversarial samples in the early-stages of their generation process.

The Accuracy [18] denotes the rate of correctly classified samples by the LSH-GPD algorithm out of the total samples, which is leveraged to measure the performance of the detection algorithm in correctly classifying both normal and adversarial samples. The FNR [18] denotes the rate of adversarial samples that are incorrectly identified as normal samples. The FPR [18] denotes the rate of normal samples that are incorrectly recognized as adversarial samples.

The ADS denotes the average number of intermediate speech segments required for the model to successfully detect adversarial samples, as shown in Equation 4.

$$\mathrm{ADS} = \frac{1}{n}\sum_{i=1}^{n} S_i \tag{4}$$

where n denotes the total number of adversarial samples, S_i denotes the number of intermediate speech segments required to detect an adversarial sample in the i-th speech sequence. A smaller ADS value indicates successful detection algorithm of adversarial samples within fewer steps, indicating higher detection efficiency.

The EDR provides a more detailed evaluation of the detection algorithm's performance in identifying adversarial samples in early-stage. EDR denotes the rate of adversarial samples successfully detected by the algorithm within a predefined number of steps, as shown in Equation 5.

$$\mathrm{EDR} = \frac{\sum_{i=1}^{n} I(S_i \le Q)}{n} \tag{5}$$

In this formula, Q denotes the predefined threshold for the early stride, and $I(.)$ denotes an indicator function that returns 1 when the condition inside the parentheses is met, and 0 otherwise. A higher EDR value indicates successful detection algorithm of adversarial samples samples in early-stage, indicating higher detection performance.

4.2 Experimental Design

For convenience of presentation, we define the full sequence and the early-stage sequence as follows. When generating adversarial samples leveraging the optimization-based generation algorithm, a series of intermediate speech samples are produced to query the targeted model. These chronologically ordered intermediate speech samples denote as the full sequence. The early-stage sequence denotes first quarter of the full sequence for evaluating the detection algorithm's performance in the early-stage of adversarial sample generation. The position of each sample in the sequence is denoted by its positional index.

In our experiments, we evaluate the LSH-GPD, Voting, and WaveGuard algorithm in three dimensions: full sequence detection performance, early-stage sequence detection performance, and detection efficiency, aiming to validate the effectiveness of the LSH-GPD algorithm. Firstly, adversarial samples are generated using the FakeBob method and genetic algorithm respectively on the TIMIT, Common Voice, and VoxCeleb2 dataset. The targeted models attacked include speaker recognition systems based on the ECAPA-TDNN and the TitaNet architecture. Subsequently, during the process of generating these adversarial samples, the intermediate speech samples interacting with the targeted models are sequentially numbered and stored.

For the full sequence detection performance, this experiment mixes normal with adversarial samples and leverages the LSH-GPD, Voting, and WaveGuard algorithm for detection respectively. The detection results and the position indices of successfully detected samples are recorded, where these position indices denote as detection stride. The experiment ensures that the same speech sequence is input sequentially to simulate a realistic attack setting. This experiment calculate Accuracy, FNR, FPR, and ADS for the three adversarial sample detection algorithms based on compiled results. For the early-stage sequence detection performance, this experiment defines EDR as the rate of samples with detection strides within the first quarter of the full sequence. For detection efficiency, given the large sample size of the full sequence, it provides a better assessment of algorithm efficiency. Therefore, this experiment records and compiles statistics on the runtime, CPU usage, GPU usage, memory consumption, and storage consumption of the detection algorithms across the full sequence to evaluate the detection efficiency of the three algorithms.

4.3 Experimental Results and Analysis

(1) Full sequence detection performance

The average Accuracy, FNR, and FPR of the LSH-GPD, Voting, and WaveGuard algorithm are shown in Table 1. The experimental results demonstrate that the LSH-GPD algorithm achieves the highest average Accuracy of 99.9%, outperforming Voting algorithm by 1.3% and WaveGuard algorithm by 2.1%. The LSH-GPD algorithm focuses on the generation process of adversarial samples and detects them based on the similarity of speech sequences produced during this process. This algorithm effectively leveraging the correlations among multiple speech samples rather than individual samples, achieving a higher Accuracy.

The LSH-GPD algorithm achieves the lowest FNR of 0.2%, while the WaveGuard algorithm shows the highest FNR of 1.8%. This is because the LSH-GPD algorithm detect adversarial samples using partial intermediate speech during the adversarial sample generation, enabling multiple detection windows throughout the entire process. In contrast, the Voting and WaveGuard algorithm can only detect adversarial samples when they achieve corresponding features. Therefore, the LSH-GPD algorithm achieves a lower FNR.

Table 1: Comparison of the Accuracy, FNR, and FPR for LSH-GPD, Voting, and WaveGuard Algorithms

Detection algorithm	Average Accuracy	Average FNR	Average FPR
LSH-GPD	99.9%	0.2%	0.0%
Voting	98.6%	1.1%	1.6%
WaveGuard	97.8%	1.8%	2.5%

The LSH-GPD algorithm achieves the lowest FPR of 0.0%, while the WaveGuard algorithm shows the highest FPR of 2.5%. This is because that the LSH-GPD algorithm leverages Locality-Sensitive Hashing (LSH) to process speech samples, which maps similar speeches to the same hash value and dissimilar speeches to different hash values. In contrast, it is almost impossible for two speech samples to be extremely similar in the setting of speaker recognition. Even if the same person utters the same sentence twice, the computer will recognize them as two distinct speeches. Therefore, normally recorded speech samples will be mapped to different hash values by the LSH algorithm and will not be misidentified as adversarial samples.

The ADS for the LSH-GPD, Voting, and WaveGuard algorithm are shown in Table 2. Among them, the LSH-GPD algorithm achieves the least ADS of 16.8 adversarial samples across three datasets. In contrast, the Voting algorithm achieves the highest ADS of 2209.6 adversarial samples across three datasets. The LSH-GPD algorithm detects adversarial samples based on the similarity of their sequences. Even if adversarial samples do not possess corresponding features in the early-stage of generation process, it can still detect them, thus requiring fewer adversarial samples for detection.

Table 2: Comparison of ADS for LSH-GPD, Voting, and WaveGuard Algorithms

Dataset	Model	Attack algorithm	ADS		
			LSH-GPD	Voting	WaveGuard
TIMIT	ECAPA-TDNN	FakeBob	20.7	308.0	116.2
		Genetic	15.6	643.6	129.6
	TitaNet	FakeBob	20.5	666.6	675.3
		Genetic	15.6	1320.9	1338.5
Common Voice	ECAPA-TDNN	FakeBob	18.2	1206.5	277.7
		Genetic	15.3	2569.0	413.0
	TitaNet	FakeBob	17.6	2561.4	2578.7
		Genetic	14.7	3901.7	3844.7
Voxceleb2	ECAPA-TDNN	FakeBob	18.5	2001.2	2173.7
		Genetic	14.2	5018.1	3904.8
	TitaNet	FakeBob	18.9	3022.8	3024.1
		Genetic	11.6	3295.1	3304.6
Average			16.8	2209.6	1815.1

(2) Early-stage sequence detection performance

In our experiment, we evaluate detection success by comparing whether their position indices fall within the early-stage sequence. The EDR denotes the rate of adversarial samples detected in the early-stage of adversarial sample generation.

The experimental results are shown in Table 3. The Voting algorithm achieves the lowest average EDR of 3.7%, while the WaveGuard algorithm achieves average EDR of 40.3%. The LSH-GPD algorithm achieves the highest average EDR of 99.0%. The EDR of the LSH-GPD algorithm increases with the complexity of samples in the dataset and the complexity of the speaker recognition model. This is because as speech samples and speaker recognition models become more complex, it becomes more challenging to generate adversarial samples, resulting in longer speech sequences during the generation process. The LSH-GPD algorithm detects adversarial samples based on the similarity of their speech sequences, and the length of the speech sequence required for detection is fixed. Therefore, as the speech sequence becomes longer, the rate of the length required for detection relative to the total length of the speech sequence decreases, leading to a higher EDR.

In our experiment, we evaluate the detection performance of the LSH-GPD, Voting, and WaveGuard algorithm on the early-stage sequence. The LSH-GPD algorithm achieves an EDR of 99.0%, effectively detecting the generation process of adversarial samples. In contrast, the Voting algorithm and WaveGuard algorithm achieve weaker detection capabilities for adversarial samples in the early-stage of their generation.

Table 3: Comparison of EDR of LSH-GPD, Voting, and WaveGuard Algorithms

Dataset	Model	Attack algorithm	EDR		
			LSH-GPD	Voting	WaveGuard
TIMIT	ECAPA-TDNN	FakeBob	94.4%	9.4%	47.1%
		Genetic	96.3%	1.2%	54.3%
	TitaNet	FakeBob	100%	1.1%	42.1%
		Genetic	98.7%	3.1%	44.6%
Common Voice	ECAPA-TDNN	FakeBob	98.9%	1.0%	33.3%
		Genetic	100%	2.3%	48.6%
	TitaNet	FakeBob	100%	4.3%	45.5%
		Genetic	100%	7.3%	31.4%
Voxceleb2	ECAPA-TDNN	FakeBob	100%	1.4%	34.1%
		Genetic	100%	6.6%	30.5%
	TitaNet	FakeBob	100%	3.4%	38.4%
		Genetic	100%	3.8%	33.3%
Average			99.0%	3.7%	40.3%

(3)Detection efficiency

For detection time, the Voting and WaveGuard algorithms heavily rely on the speaker recognition model. The detection process requires multiple model queries to obtain confidence scores, leading to longer average times required for detecting adversarial samples, which are 24.03 and 15.12 seconds respectively. In contrast, the LSH-GPD algorithm only rely on the characteristics of the adversarial sample's speech itself. The average time required for detecting adversarial samples is only 0.1 second, significantly less than that of the Voting and WaveGuard algorithm.

For CPU usage, the WaveGuard algorithm achieves the highest usage at 748%, due to speech transformation requirements. In contrast, the LSH-GPD algorithm only processes the inherent characteristics of speech and involves less computation, achieving the lowest CPU

usage of 148.3%. For GPU usage, the LSH-GPD algorithm does not interact with the targeted speaker recognition model, achieving zero GPU consumption. In contrast, both the Voting and WaveGuard algorithm necessitate interaction with the targeted model, leading to a GPU usage equivalent to that of the speaker recognition model, which is 404MB.

For memory usage, the Voting and WaveGuard algorithm require 11.34GB and 11.40GB of memory space respectively, indicating high memory consumption, but they do not necessitate additional storage space. The LSH-GPD algorithm achieves the lowest memory usage of 2.69GB, however, it requires 67.2KB of storage space for detecting adversarial samples. Considering the actual setting where the number of normal samples far exceeds that of adversarial samples, the storage space consumption is deemed acceptable.

Table 4: Comparison of the detection efficiency among LSH-GPD, Voting, and WaveGuard Algorithms

Detection algorithm	Detection time(seconds)	CPU(%)	GPU(MB)	Memory(GB)	Storage(KB)
LSH-GPD	0.12	148.3%	-	2.69	67.2
Voting	24.03	172.3%	404MB	11.34	-
Waveguard	15.12	748.8%	404MB	11.40	-

5 Conclusion

Addressing the poor performance of adversarial sample detection algorithms in the early-stage of adversarial sample generation, we propose the LSH-GPD algorithm.The algorithm leverages the characteristic of similar speech produced during the sample generation process by optimization-based adversarial sample methods to detect adversarial samples. Furthermore, to facilitate rapid search for similar speech samples, we designs an adversarial locality-sensitive hashing algorithm in this paper, named Ad-LSH.

Experimental results demonstrate that, compared with traditional detection methods (such as the Voting and WaveGuard algorithm), the LSH-GPD algorithm achieves higher Accuracy, lower FNR and FPR. It can effectively detect adversarial samples in the early-stage stages of generation, and significantly lower than other detection algorithms for detection time, CPU usage, GPU usage, and memory consumption.

Additionally, since the LSH-GPD algorithm does not rely on the targeted model and its detection mechanism is based on the similar speech features generated during the adversarial sample creation process, it achieves good generalization capability and can be applied to various speaker recognition systems.

Funding

1. This work is supported by the National Natural Science Foundation of China (Grant Nos. 62472047, 62172055).

About The Author

Jia Zheng was born in Shangrao, Jiangxi. China, in 1991. He obtained a master's degree from Beijing University of Posts and Telecommunications in china.He currently working at the Network Business Unit of China Mobile Communications Group Corporation. He main

research direction is network security.

Hua Zhang received the B.S. degree in telecommunications engineering from the Xidian University in 1998, the M.S. degree in cryptology from Xidian University in 2005, and the Ph.D. degree in cryptology from Beijing University of Posts and Telecommunications in 2008. Now she is a professor at Beijing University of Posts and Telecommunications. Her research interests include cryptography, information security, and network security.

Ming Lv was born in Runan, Henan. China, in 1981. He obtained a master's degree from Beijing University of Posts and Telecommunications in china.He currently working at the Network Business Unit of China Mobile Communications Group Corporation. He main research direction is network security.

Wanjin Hou was born in Cangzhou, Hebei. He obtained a master's degree from Beijing University of Posts and Telecommunications. His main research direction is artificial intelligence security.

Huiting Hou was born in Ganzhou, Jiangxi, China, in 2000. She obtained a master's degree from Peking University in China. She is currently working at the Network Business Unit of China Mobile Communications Group Corporation. Her main research direction is network security.

Lili Zhang was born in Yichang, Hubei. China, in 1989. She obtained a master's degree from Beijing University of Posts and Telecommunications in china.She currently working at China Satelite Network Application Research Co., Ltd. Her main research direction is satellite communication.

References

[1] Zheng B, Jiang P, Wang Q, et al. Black-box adversarial attacks on commercial speech platforms with minimal information[C]. ACM SIGSAC Conference on Computer and Communications Security. 2021: 86-107.

[2] Tan H, Wang L, Zhang H, et al. Adversarial attack and defense strategies of speaker recognition systems: A survey[J]. Electronics. 2022: 11(14): 2183.

[3] Yuan X, Chen Y, Zhao Y, et al. CommanderSong: A systematic Approach for Practical Adversarial Voice Recognition[C]. USENIX security symposium. 2018: 49-64.

[4] Wang D, Wang R, Dong L, et al. Adversarial examples attack and countermeasure for speech recognition system: A survey[C]. International Conference on Security and Privacy. 2020: 443-468.

[5] Sun S, Yeh C F, Ostendorf M, et al. Training augmentation with adversarial examples for robust speech recognition[C]. International Speech Communication Association. 2018: 2404-2408.

[6] Pal M, Jati A, Peri R, et al. Adversarial defense for deep speaker recognition using hybrid adversarial training[C]// IEEE International Conference on Acoustics, Speech and Signal Processing. 2021: 6164-6168.

[7] Yang C, Ahmed Z, Gu Y, et al. Mitigating closed-model adversarial examples with Bayesian neural modeling for enhanced end-to-end speech recognition[C]. IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2022: 6302-6306.

[8] Hussain S, Neekhara P, Dubnov S, et al. WaveGuard: Understanding and mitigating audio adversarial examples[C]// USENIX Security Symposium. 2021: 2273-2290.

[9] Rajaratnam K, Shah K, Kalita J. Isolated and ensemble audio preprocessing methods for detecting adversarial examples against automatic speech recognition[J]// arXiv preprint arXiv:1809.04397, 2018.

[10] Tamura K, Ito H. Protection Method based on Multiple Sub-Detectors against Audio Adversarial Examples[C]// International Workshop on Computational Intelligence and Applications. 2021: 1-7.

[11] Wu H, Zhang Y, Wu Z, et al. Voting for the right answer: Adversarial defense for speaker verification[J]// International Speech Communication Association. 2021: 4294-4298.

[12] Samizade S, Tan Z H, Shen C, et al. Adversarial example detection by classification for deep speech recognition[C]// IEEE International Conference on Acoustics, Speech and Signal Processing. 2020: 3102-3106.

[13] Carlini N, Wagner D. Audio adversarial examples: Targeted attacks on speech-to-text[C]// IEEE security and privacy workshops. 2018: 1-7.

[14] Alzantot M, Balaji B, Srivastava M. Did you hear that? Adversarial examples against automatic speech recognition[J]// arXiv preprint arXiv:1801.00554, 2018.

[15] Goodfellow I J, Shlens J, Szegedy C, et al. EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES[J]// stat, 2015, 1050: 20.

[16] Pal M, Jati A, Peri R, et al. Adversarial defense for deep speaker recognition using hybrid adversarial training[C]// IEEE International Conference on Acoustics, Speech and Signal Processing. 2021: 6164-6168.

[17] Carlini N, Wagner D. Towards evaluating the robustness of neural networks[C]// IEEE symposium on security and privacy. 2017: 39-57.

[18] Zeng Q, Su J, Fu C, et al. A multiversion programming inspired approach to detecting audio adversarial examples[C]// IEEE/IFIP international conference on dependable systems and networks. 2019: 39-51.

[19] Carlini N, Mishra P, Vaidya T, et al. Hidden voice commands[C]// USENIX security symposium. 2016: 513-530.

[20] Taori R, Kamsetty A, Chu B, et al. Targeted adversarial examples for black box audio systems[C]// IEEE security and privacy workshops. 2019: 15-20.

[21] Garofolo J, Lamel L, Fisher W, et al.. Getting started with the DARPA TIMIT CD-ROM: An acoustic phonetic continuous speech database[J]. Natl Inst Stand Technol, 1988: 107: 16.

[22] Ardila R, Branson M, Davis K, et al. Common voice: A massively-multilingual speech corpus[J]. arXiv preprint arXiv:1912.06670, 2019.

[23] Nagrani A, Chung J S, Xie W, et al. Voxceleb: Large-scale speaker verification in the wild[J]. Computer Speech & Language, 2020, 60: 101027.

[24] Desplanques B, Thienpondt J, Demuynck K. Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification[J]. arXiv preprint arXiv:2005.07143, 2020.

[25] Koluguri N R, Park T, Ginsburg B. TitaNet: Neural Model for speaker representation with 1D Depth-wise separable convolutions and global context[C]. IEEE International Conference on Acoustics, Speech and Signal Processing. 2022: 8102-8106.

[26] Chen G, Chen S, Fan L, et al. Who is real bob? adversarial attacks on speaker recognition systems[C]. IEEE Symposium on Security and Privacy. 2021: 694-711.

[27] Carlini N, Wagner D. Towards evaluating the robustness of neural networks[C]. IEEE symposium on security and privacy. 2017: 39-57.

[28] Alzantot M, Balaji B, Srivastava M. Did you hear that? Adversarial examples against automatic speech recognition[J]. arXiv preprint arXiv:1801.00554, 2018.