SA: Sliding attack for synthetic speech detection with resistance to clipping and self-splicing

Jiacheng Deng*, Li Dong**, Yan Diquan*, Rangding Wang* and Jiaming Zeng*

*Department of Computer Science, Ningbo University, Ningbo 315000, China

ARTICLE INFO

Keywords:
adversarial examples
synthetic speech detection
clipping and self-splicing
transferability

ABSTRACT

Deep neural networks are vulnerable to adversarial examples that mislead models with imperceptible perturbations. In audio, although adversarial examples have achieved incredible attack success rates on white-box settings and black-box settings, most existing adversarial attacks are constrained by the input length. A more practical scenario is that the adversarial examples must be clipped or self-spliced and input into the black-box model. Therefore, it is necessary to explore how to improve transferability in different input length settings. In this paper, we take the synthetic speech detection task as an example and consider two representative SOTA models. We observe that the gradients of fragments with the same sample value are similar in different models via analyzing the gradients obtained by feeding samples into the model after cropping or self-splicing. Inspired by the above observation, we propose a new adversarial attack method termed sliding attack. Specifically, we make each sampling point aware of gradients at different locations, which can simulate the situation where adversarial examples are input to black-box models with varying input lengths. Therefore, instead of using the current gradient directly in each iteration of the gradient calculation, we go through the following three steps. First, we extract subsegments of different lengths using sliding windows. We then augment the subsegments with data from the adjacent domains. Finally, we feed the sub-segments into different models to obtain aggregate gradients to update adversarial examples. Empirical results demonstrate that our method could significantly improve the transferability of adversarial examples after clipping or self-splicing. Besides, our method could also enhance the transferability between models based on different features. Code is available at https://gitee.com/djc_QRICK/Adversarial-1D.

1. Introduction

Deep neural networks (DNNs) are vulnerable to adversarial examples [9, 30] that, by adding small perturbations, are indistinguishable from legitimate examples but lead to incorrect model predictions. Making adversarial examples has received increasing attention in recent years [2, 3, 18, 21, 25], as it can not only identify model vulnerabilities [25, 30], but also help improve model robustness. In addition, adversarial examples also show good transferability between models [23, 26], i.e., adversarial examples made for one model can still fool other models, enabling black-box attacks in real-world applications without any knowledge of the victim model. The existing adversarial attacks show significant effectiveness in the white-box setting, in which the attacker can access the structure and parameters of the target model [3, 18], but the transferability in the black-box model is poor. To address this issue, recent works focus on improving the transferability of adversarial examples by advanced gradient calculation (e.g., Momentum, Nesterov’s accelerated gradient, etc.) [5, 22], attacking multiple models [23], or adopting various input transformations (e.g., random resizing and padding, translation, scale, admix, etc.) [6, 22, 36]. However, the above work is mainly applied in the image field, and there is a lack of related work on improving the transferability of adversarial examples in audio. Existing works [16, 28, 29] in audio only consider cases where the input lengths of black and white-box models are consistent. In images, the resizing operation can effectively solve the problem of varying input size and retaining most of the image information. However, in audio classification, it is common to clip too long or self-splice too short audio and input it to the model of the corresponding length, which makes the processed audio very different from the original input audio and dramatically reduces the transferability in black-box models.

In this paper, we evaluate several typical adversarial attack methods and observe that they are less transferable in the black-box model after clipping or self-splicing. Since attacking ensemble models effectively improves the transferability of adversarial examples, we intuitively think of attacking different input length models. However, previous ensemble attacks only consider integrating models with the same input length, and we need to input samples into models with different input lengths after clipping or self-splicing. Clipping and self-splicing lead to widely different input samples, and aggregating the gradients of these samples has the potential to produce misleading gradients. Therefore, we first analyze the sample’s gradients after clipping and self-splicing. Although clipped samples or self-spliced samples are quite different from the original samples, their sampling points are all derived from the original samples. Then we analyze the gradient of the common area of the original sample, the clipped sample, and the self-spliced sample, and the results show that the gradients of the fragments with the same sampling value are similar in different models. Hence aggregating these similar gradients can stabilize the gradient update direction and improve the attack success rate on white-box models. Considering that the input length and cutting...
method of the black-box models are uncertain, the positions of the same sampling point on the samples before and after the preprocessing of the black-box model are different.

Based on the above observations, we propose a novel sliding attack (SA), which extracts sub-segments at different positions of the original sample through a sliding window and feeds them into different models to obtain gradients and aggregate these gradients to update adversarial examples. Furthermore, inspired by VI-FGSM [35], in each iteration, we not only compute the gradient of the current data point but also use the gradient of the data in the neighboring domain of the current data point. The rationale behind this design includes three aspects. First of all, we use sliding windows of different lengths to select sub-segments and input models of different input lengths, which simulates the change of the position of the sampling points after the black box preprocesses the input samples and allows each sampling point to perceive the gradient at different positions in models. Second, we observe that the gradients of segments with the same sampling value are similar across different models. Therefore, aggregating the gradients of these sub-segments selected by sliding windows can enhance the transferability of adversarial examples and improve the success rate of attacking white-box models. Finally, the data in the adjacent domain is similar to the original data point, so the gradient of the data point in the adjacent domain is similar to the gradient of the original data point. Aggregating these gradients can further effectively stabilize the gradient update direction and improve the threat of adversarial examples.

We implemented extensive experiments on two representative SOTA models (RawNets [31], LCNNs [19, 39]) based on the ASVspoof2021 dataset [34, 38]. Ablation experiments show that our proposed method significantly improves the current sota attack success rate on black-box Rawnets by 192%. In addition, SA can effectively improve the attack success rate of white-box models; for example, the attack success rate on the ensemble Rawnet is increased by 11.6%.

Our contributions are summarized as follows.

- We make the first steps to study the effect of clipping and self-splicing on adversarial examples. By conducting a comprehensive experiment, we observe that existing adversarial attacks are difficult to maintain threatening after clipping or self-splicing.
- We define gradient similarity, study the gradients of adversarial examples after clipping and self-splicing, and show that subsegments with the same sampling value are similar across models.
- We propose an adversarial attack method against audio, called a sliding attack (SA). The proposed method can craft adversarial examples that are still fool black-box models after being clipped or self-spliced.

The rest of the paper is organized as follows: We present related work in Section 2 and formulate the problem mathematically in Section 3. We introduce the detailed design of SA in Section 4. We then offer experimental results and some discussions in Section 5. Finally, we summarize our work in Section 6.

2. Related work

2.1. Typical Adversarial Attacks

Here we introduce several typical white-box adversarial attacks. The Fast Gradient Sign Method (FGSM) assumes the linear behavior in high-dimensional spaces is sufficient to generate adversarial inputs [9]. Therefore, it constructs adversarial samples by applying a first-order approximation of the loss function. An adversarial example can be generated with a one-step update. As an iterative version of FGSM, the Projected Gradient Descent (PGD) [17] selects the original sample as a starting point and generates adversarial examples with multi-step. It is a powerful adversarial attack method and is therefore often used as a baseline attack for evaluating defense designs. Inspired by the momentum optimizer, Dong et al. [5] proposed to integrate the momentum memory into the iterative process and derived a new iterative algorithm, called momentum iterative FGSM (MI-FGSM). Generally, momentum-based adversarial examples are more transferable on black-box models. DeepFool is also an iterative attack method, which has been proposed to generate an adversarial example with the minimum perturbation on the decision boundary of a classifier. The Carlini & Wagner (CW) attack method [3] takes a Lagrangian form and adopts Adam [15] for optimization.

2.2. Synthetic speech detection

Synthetic speech detection [38] focuses on detecting spoof speech in non-auto speaker verification scenarios. It reflects the scenario in which an attacker has access to the voice data of a targeted victim, e.g., data posted to social media. The victim might be a celebrity, a social media influencer, or an ordinary citizen. The attackers’ incentive might be, e.g., to blackmail the victim or to denigrate his or her reputation in some way by spreading spoken misinformation. Synthetic speech techniques commonly use by attackers include text-to-speech (TTS) [7] and speech conversion (VC) [24, 32]. In order to detect this kind of attack, researchers have proposed a variety of countermeasures [12, 27]. In this paper, considering the practical hazards of synthetic speech, our research is based on the standard ASVspoof2021 dataset, a practical dataset for promoting the development of synthetic speech detection.

3. Problem Formulation

In the section, we first define the basic notations, clipping and self-splicing. Then, the problem to be solved will be described.

We denote initial input sample as $x$, its ground-truth label as $y$, the corresponding adversarial example as $x^{adv}$, the model with input length $l$, as $F$, where $F(x) = y$ and $F(x) \neq F(x^{adv})$. We use $\theta$ to denote parameters of network $F$ and...
use $L(x, y; \theta)$ denote the loss for $x$. To generate the adversarial example, the goal is to maximize the loss $L(x, y; \theta)$ for the input sample $x$.

Considering that, in audio, the speeches' length and requirements of the input model are usually inconsistent, the input speeches will be clipped or self-spliced. We formally define the operation of clipping and self-splicing as follows.

**Clipping:** When the length of the input speech exceeds the input length required by the model, the speech needs to be clipped. The clipping rule is unknown to attackers under the black-box setting. Therefore, we assume one practical case where the victim model employs random clipping for extremely long inputs. The operation of clipping is shown in Fig. 1(a) and abbreviated as $C(\cdot)$ and formulated as follows.

$$C(x, p, l) = x[p : p + l],$$  \hspace{1cm} (1)

where $l$ is clipped length, $p$ is start sampling index of clipping operation, $x[p : p + l]$ is the clipped segment from the $p$ to $p + l$ in $x$, and $p + l$ must be less than or equal to the length of $x$.

**Self-splicing:** When the input speech's length is shorter than the input length required, the speech needs to be self-spliced, clipping a segment from the input speech and splicing this segment after the input speech. Although the victim model can arbitrarily choose the region to be self-spliced, one of the most widely-adopted strategies is to select the head segment to splice the tail of the input. Therefore, the operation of self-splicing is shown in Fig. 1(b) and abbreviated as $S(\cdot)$ and formulated as follows.

$$S(x, l) = x \oplus x[0 : l],$$ \hspace{1cm} (2)

where $x \oplus x[0 : l]$ means splice $x[0 : l]$ to the tail of $x$.

The goal of prior works in the literature is to generate the adversarial example that satisfies:

$$\arg \max_{x^{adv}} L(x^{adv}, y; \theta), \ s.t. \|x^{adv} - x\|_\infty < \epsilon,$$ \hspace{1cm} (3)

where $\epsilon$ is a tiny distortion that keeps $x^{adv}$ sound similar to the original audio $x$. However, we consider a more practical scenario where the input length of the victim model is unknown. Therefore, after the adversarial example is generated by the attacker, it needs to be clipped or self-spliced before it can be input into the victim model. More specifically, when the input length of the victim model is required to be shorter than $x^{adv}$, we assume that the victim model is $T_{l_c}$, its input length is $l_c$, and its parameter is $\theta_{l_c}$. We require that the adversarial example $x^{adv}$ is still adversarial after self-clipping, which is defined as follows:

$$\arg \max_{x^{adv}} L(C(x^{adv}, p, l_c), y; \theta_{l_c}), \ s.t. \|x^{adv} - x\|_\infty < \epsilon.$$ \hspace{1cm} (4)

When the input length of the victim model is required to be longer than $x^{adv}$, we assume that the victim model is $F_{l_s}$, its input length is $l_s$, and its parameter is $\theta_{l_s}$. We require that the adversarial example $x^{adv}$ is still adversarial after clipping, which is defined as follows:

$$\arg \max_{x^{adv}} L(S(x^{adv}, l_s - l_n), y; \theta_{l_s}), \ s.t. \|x^{adv} - x\|_\infty < \epsilon.$$ \hspace{1cm} (5)

However, existing adversarial attacking methods cannot strike a good balance between the quality of adversarial examples and the adversarial after clipping or self-clipping operations. As shown in Table 2 and Table 3, adversarial examples generated by typical adversarial attacks are much less threatening to the victim model after clipping or self-clipping. More details can be found in Section 5.3. In the following, we propose an adversarial attacking method, which can effectively improve the threat of adversarial examples after clipping or self-clipping.

### Table 1: Accuracy of models based on different input lengths

| Model   | Input length | Train(%) | Test(%) |
|---------|--------------|----------|---------|
| Rawnet_4 | 40000        | 99.4     | 98.0    |
| Rawnet_8 | 48000        | 99.2     | 97.8    |
| Rawnet_6 | 56000        | 99.6     | 99.4    |
| Rawnet_16| 64600        | 99.7     | 99.5    |
| Rawnet_22| 72600        | 99.7     | 99.3    |
| LCNN_4  | 40000        | 99.8     | 99.6    |
| LCNN_8  | 48000        | 99.8     | 99.8    |
| LCNN_6  | 56000        | 99.7     | 99.3    |
| LCNN_16 | 64600        | 99.5     | 99.8    |
| LCNN_22 | 72600        | 99.9     | 99.8    |

### 4. Proposed Method

In this section, we are inspired by prior work in the literature in Section 4.1 and hope to improve the threat of adversarial examples by attacking the ensemble model. However, previous ensembled attacks only consider integrating models with the same input length, and we need to input samples into models with different input lengths after clipping or self-clipping. Clipping and self-clipping lead to widely different input samples, and aggregating the gradients of these samples has the potential to produce misleading gradients. Therefore, we analyze the sample's gradients after clipping and self-clipping in Section 4.2 and, based on the above observations, propose the SA attack method and describe it in detail in Section 4.3.
4.1. Gradient-based Adversarial Attack Method

Empirical studies of most gradient-based attacks (such as DIFGSM [36], TIFGSM [6], NIFGSM [22] and VIFGSM [35]) show that ensembles of more gradients can generate more powerful adversarial examples. Taking a typical ensemble adversarial attack as an example, the attacker inputs \( x_{t}^{adv} \) into multiple models to obtain gradients and then aggregate the gradients to update the adversarial examples. The aggregation formula of the gradient is as follows:

\[
G_{t+1} = \frac{1}{I} \sum_{i=1}^{I} w^i \nabla_{x} L(x_{t}^{adv}, y, \theta^i) \tag{6}
\]

\[
x_{t+1}^{adv} = x_{t}^{adv} + \alpha \cdot \text{sign}(G_{t+1}) \tag{7}
\]

where \( I \) is the number of models, \( w^i \) is the weights of gradients, and \( \theta^i \) represents parameters from different models. Inspired by ensemble attacks, we hope to improve the transferability of adversarial examples after clipping or self-splicing by aggregating gradients from models of different input lengths.

Convolutional neural networks are supposed to have the translation-invariant property, according to [20], that an object in the input can be recognized in spite of its position. In practice, CNNs are not truly translation-invariant [8, 13]. We follow the assumption of [18] that the translation-invariant property is nearly held with very small translations. Previous work improves the transferability of adversarial examples by attacking ensemble models with the same input length. In this case, aggregating gradients from different models can improve the transferability of adversarial examples since the samples input to models is the same, and the gradients obtained by backpropagation are also similar. However, we consider a more practical case where we need to input samples into models after clipping or self-splicing. This results in widely different input samples and cannot preserve translation-invariant properties. This means that the gradients we obtain may be dissimilar and the aggregated gradient will deviate from the correct direction, as shown in Fig. 2.

Therefore, we define the gradient aggregation and gradient similarity formula in the following subsections and then analyze the gradient of the common area of the original sample, the clipped sample, and the self-spliced sample.

4.2. Gradient Similarity Analysis

We first define the gradient aggregation formula for models with different input lengths as follows:

\[
G_{t+1} = \sum_{i=1}^{I_1} w^i \nabla_{x} L(x_{t}^{adv}, y, \theta^i)+ \\
\sum_{i=1}^{I_2} w^i \nabla_{x} L(C(x_{t}^{adv}, p, I_c), y, \theta^i)+ \\
\sum_{i=1}^{I_3} w^i \nabla_{x} L(S(x_{t}^{adv}, l_n), y, \theta^i) \tag{8}
\]

where \( I_1, I_2, I_3 \) are the number of different models, \( w^1, w^2, w^3 \) are weights. Then, given two gradients \( g^a \) and \( g^b \) of length \( l' \), we define the gradient similarity calculation formula as follows:

\[
GS(g^a, g^b) = \frac{\sum_{i=0}^{l'} (\text{sign}(g_{t_i}^a) \odot \text{sign}(g_{t_i}^b))}{l'} \tag{9}
\]

where \( \odot \) is Exclusive NOR, \( \text{sign}(\cdot) \) denotes the sign function. Since gradient-based adversarial attacks usually take the gradient sign, we do not directly count the similarity of the actual value of the gradient (e.g., using the cosine distance) but first do the sign for gradients and then count the proportion of sampling points with consistent gradient. The gradient similarity of two randomly input samples is about 0.5 because the result of the sign only includes -1 and 1. Therefore, the value of \( GS \) is distributed between 0 and 1. When \( GS(g^a, g^b) > 0.5 \), it means that the \( g^a \) is similar to \( g^b \), and when \( GS \) is equal to 1, it means that the two gradients are completely consistent.

Next, to better observe the gradients of the clipped samples, the self-spliced samples, and the original samples, we select the gradients of their common areas for analysis. Since \( p \) in \( C(\cdot) \) is arbitrary, we need to fix \( p \) for a coherent analysis of gradient similarity. We set \( p = 0 \), where the head gradient aggregation of adversarial examples has the highest complexity. As shown in the Fig. 3, when \( p = 0 \), the head of the aggregated gradient is aggregated from the gradients of four regions, including \( x_{t}^{adv}[0 : l_5 - l_n], S(x_{t}^{adv}, l_5 - l_n)[0 : l_5 - l_n], S(x_{t}^{adv}, l_3 - l_n)[0 : l_3 - l_n], \) and \( C(x_{t}^{adv}, 0)[0 : l_3 - l_n] \), and the sample values of these four segments are the same. For ease of representation, we denote the gradients of A, B, C, and D and the gradient of the head of the aggregated gradient as \( g^A, g^B, g^C, g^D \) and \( g^* \). Suppose the aggregated gradient is similar to the gradients of the other four regions. In
Leveraging social media news

Figure 3: Gradient aggregation process when $p = 0$. The head of the aggregated gradient is aggregated from the gradients of four regions, namely A, B, C, and D, and these four regions have the same sampling value.

Figure 4: Average gradient similarity with $PGD_{ens}$.

that case, we assume that the gradients of the sub-segments with the same sampling value are similar in different models, and aggregating these gradients can improve the attack success rate of the white-box model. We select $PGD_{ens}$ to attack representative model Rawnets on standard ASVspoof 2019 and calculate the average similarity between $g^*$ and $g^A$, $g^B$, $g^C$ and $g^D$. We select three required input length models in this experiment, including 40000, 56000, and 72600. These models' accuracy and required input length are shown in Table 1. Moreover, we set all $w_j$ and all $I_j$ equal to 1. For simplicity, we abbreviate $GS(g^*_{i-1}, s^{A}_{i}),... ,GS(g^*_{i-1}, s^{D}_{i})$ as $GS_A,...,GS_D$, where $i$ represents $i$-th iteration. The result is shown in Fig. 4. We can find that the gradient finally stabilizes above 0.6 as the number of iterations increases, proving that sub-segments with the same sampling value have similar gradients on models with different input lengths.

Based on the above observations, aggregating these similar gradients can stabilize the gradient update direction and improve the attack success rate on white-box models. We proposed a novel sliding attack named SA and introduced the detail in the following subsection 4.3.

4.3. Sliding Attack

This subsection provides a comprehensive introduction to the SA implementation process. The purpose of SA is to generate an adversarial example $x^{adv}$ of length $l_n$, and we hope to enhance the transferability of $x^{adv}$ after being clipped or self-spliced. Attacking the ensemble model effectively improves the attack success rate of adversarial examples on the white-box model and the transferability on the black-box model. Therefore, the proposed method attacks the ensemble model, consisting of three models with different input lengths, namely $F$, $F_{l_c}$, and $F_{l_s}$. The $F$, $F_{l_c}$, and $F_{l_s}$ input lengths are $l_n$, $l_c$, and $l_s$, respectively. In the experiment, we choose the first quartile, median and third quartile as $l_c$, $l_n$, and $l_s$ according to the length distribution of the audio in the dataset.

Reviewing our scenario, we hope adversarial examples retain transferability after clipping or self-splicing. However, the input length of the black-box models and the clipping method are uncertain. We cannot predict where a particular sampling point on the adversarial example will appear in the processed sample and design the corresponding algorithm to improve transferability. Although we can improve the transferability of adversarial examples by integrating a large number of models with different input lengths, training the model takes a lot of time and computational resources. Therefore, we hope to simulate more black-box situations with a limited model. Specifically, as shown in Fig. 5, SA uses sliding windows of different lengths to sample sub-segments of different lengths on the original sample. For ease of understanding, we arbitrarily select a sample point on the $x^{adv}_{i}$ and mark it with a blue vertical line and trace it. We can observe that after the sliding window extraction, the marked sampling points are located at different positions of the sub-segments, which simulates the different situations of black-box models and make the sampling point preceptive more gradients with a different location. According to our observation in Section 4.2, the gradients of segments with the same sampling values are similar across different models. Therefore, we aggregate the gradients of these sub-segments to increase the attack success rate on the white-box model and improve the transferability on the black-box model. The process of extracting sub-segments is shown in lines 8-14 of Algorithm 1. We use sliding windows of different lengths ($l_c$, $l_s$) to extract $S_n$ sub-segments on $x^{adv}_{i}$ respectively. We can directly extract
Leveraging social media news

Figure 5: The framework of SA, where we use blue vertical lines to mark and track sampling points. SA includes the following steps. First, sub-segments are extracted with different sliding windows. Then, we collect adjacent data points through the function $V(\cdot; r)$ to avoid getting stuck in the local optima of the gradient. Finally, sub-segments are fed into the model of the corresponding length, and the gradients are aggregated to update the adversarial examples.

Figure 6: Distribution of the lengths of audios in the ASVspoof2021 dataset.

5.1. Experimental Settings

Datasets: We use the standard ASVspoof2021 dataset introduced for the speech deepfake challenge. The training dataset contains 2580 bonafide speeches and 22800 spoofed speeches. The test dataset contains 2548 bonafide speech and 22206 spoofed speech. This dataset is suitable for our study because the input length of the victim model is inaccessible, but an approximate input range can be predicted based on the dataset. We selected five data lengths, including 40000, 48000, 56000, 64000, and 72000, as input length requirements of the model according to the audio data length distribution of ASVspoof2021 in Fig. 6. In particular, pay attention to the following two points:

- In all experiments in this paper, we generate adversar
Leveraging social media news

Algorithm 1: SA

Input: Classifiers $F$, $F_1$, and $F_2$ with required input length $l_n$, $l_c$, and $l_a$; model parameters $\theta_1$, $\theta_2$, and $\theta_3$.
Input: A raw input sample $x$ with ground-truth label $y$.
Input: The magnitude of perturbation $\epsilon$; number of iteration $T$; step length $\alpha$; decay factor $\mu$.
Input: Number of sub-segments $S_n$; noise function $V(\cdot; r)$.
Output: An adversarial audio $x^{adv}$ with length of $l_n$.

1. $g_0 = 0$; $x_0^{adv} = x$;
2. for $t = 1 \rightarrow T$ do
3. $X = V(x_0^{adv}; r)$
4. $\hat{g}_{t+1} = \sum_{j=0}^{T} \nabla_{x_j} L(X_j; y; \theta)$
5. Set start positions and steps for sliding windows
6. $s1 = 0$, $s2 = 0$
7. $step1 = (l_n - s_l)/S_n$, step2 = $(l_s/l_n) * (l_n - l_s)/S_n$
8. for $i = 1 \rightarrow S_n$ do
9. $X1 = V(x_i^{adv}[s1 : s1 + l_s]; r)$
10. $\hat{g}_{i+1} = \hat{g}_{i+1} + \sum_{j=0}^{T} \nabla_{x_j} L(X1_j; y; \theta)_{[l_s/l_n]}$
11. $X2 = V(x_i^{adv} \oplus \cdots \oplus x_i^{adv}[s2 : s2 + l_c]; r)$
12. $\hat{g}_{i+1} = \hat{g}_{i+1} + \sum_{j=0}^{T} \nabla_{x_j} L(X2_j; y; \theta)_{[l_c]}$
13. $s = s1 + step1$, $s2 = s2 + step1$
14. end
15. Update $g_{i+1}$ by based momentum
16. $\hat{g}_{i+1} = \mu \cdot g_{i} + \frac{\hat{g}_{i+1}}{||g_{i+1}||}$
17. Update $x_{i+1}^{adv}$ by applying the sign of gradient
18. $x_{i+1}^{adv} = \text{clip}(\{x_i^{adv} + \alpha \cdot \text{sign}(g_{i+1})\})$
19. $x^{adv} = x_T^{adv}$;
20. return $x^{adv}$;

Note that when the length of the adversarial example exceeds the model input requirements, it needs to be clipped by function $C(\cdot)$. However, since the attacker cannot access the starting position of the clipping taken by the victim model, $p$ in $C(\cdot)$ is random. In all experiments of the paper, we cut the adversarial example’s head, middle, and tail and then input these samples into the victim model. For instance, given an adversarial example $x^{adv}$ of length $l_2$ and a model $F_2$, with input length $l_c$, where $l_2 > l_c$. We select $x^{adv}[0 : l_2]$, $x^{adv}[l_2-l_c : l_2-l_c+1 : l_c]$ and $x^{adv}[l_2-l_c : l_2-l_c+1 : l_2]$, as input to the victim model and calculate the average TSR as the final result.

Victim Models: Considering that the attacker does not have access to the model structure and there are various feature extraction methods in audio, we refer to the baseline of ASVspoof2021 and select two representative models, including waveform+Rawnet and LFCC+LCNN. These two models represent the widely used one-dimensional and two-dimensional features in audio classification models, respectively. Rawnet is direct modeling of raw waveforms using deep neural networks, which has achieved state-of-the-art in speaker classification and synthetic speech detection. Linear frequency cepstral coefficients (LFCC) [14, 37] are a popular feature used for synthetic speech detection. The LFCC process has two steps: 1) Using linear filters and logarithmic compression power spectrum; 2) Perform DCT to produce the cepstral coefficients. In the experiments, we retained 60 LFCC coefficients and concatenated LFCC, Delta-LFCC, and Delta-Delta-LFCC as input features. Where delta and delta-delta [4, 10] can record some dynamic information, such as the change of LFCC over time. LCNN is a typical CNN model, which performs better in the classification of two-dimensional features such as LFCC, spectrogram [1], CQCC [33], and MFCC [40]. More specifically, accuracies, input lengths, and model names are shown in Table 1.

5.2. Metrics

The experimental evaluation criteria include the quantity of adversarial examples, the attack success rate (ASR), and transfer success rate (TSR).

SNR: To quantify the distortion introduced by an adversarial audio, we use the signal-to-noise ratio (SNR) [11] measure. SNR is a logarithmic scale that measures the relative loudness of an audio sample. This measure is a relative measure, hence we use the original waveform $x^a$ as the reference point for the distortion $x^d$.

$$\text{SNR}_{dB}(x^a, x^d) = 10 \log_{10} \frac{\sum_{n=1}^{N} x_n^a}{\sqrt{\sum_{n=1}^{N} x_n^d}}$$

ASR: Attack success rate is an important indicator to evaluate the performance of adversarial attack algorithms. Considering that there are a small number of misclassified samples, the attack success rate of model $a$ is calculated as follows:

$$\text{ASR} = \frac{\text{num}(\hat{X}_a^{ADV} - X_{adv}^{ADV})}{\text{num}(X^{ADV} - X_{adv}^{ADV})} \times 100\%.$$
Leveraging social media news

Figure 7: Comparing perturbations from various adversarial attacks in Rawnet, blue waveforms represent adversarial examples (except clean synthetic speech), and orange waveforms represent residuals. The full average transfer success rate (ASRxAvg.TSR) are 2.94% (CW), 4.68% (DeepFool), 0.81% (FGSM), 7.87% (PGD-20), 13.2% (MI-FGSM-20), 21.25% (SA, $S_n = 0, r = 0$) and 47.2% (SA, $S_n = 8, r = 5$) respectively. The adversarial perturbation generated by our method shows better transferability on black-box models.

Figure 8: Compare the spectrogram of various adversarial examples on Rawnet.

as follows:

$$X^* = \hat{X}_A^{c_{All}} - X_{err}^{err}$$

$$\text{TSR} = \frac{\text{num}(\hat{X}_A^{c_{All}} - X_{err}^{err})}{\text{num}(X^* - X_{err}^{err})} \times 100\%.$$  \quad (13)

5.3. Baseline Adversarial Attack

To test the effect of clipping and self-splicing on the transferability of adversarial examples, we conduct extensive experiments on representative Rawnet and LCNN models with typical adversarial attack methods. We first perform five adversarial attacks on a single neural network: CW, DeepFool, FGSM, PGD, and MIFGSM. We craft adversarial examples on normally trained networks (e.g., Rawnet56 or LCNN56) and test them on different input length neural networks we consider. The results are shown in Table 2 and Table 3. More specifically, we generate adversarial examples based on the white-box model in the first column of the table and then input them into five black-box models after clipping or self-splicing. The second and third columns show the attack method and its attack success rate on the white-box model, respectively.

From Table 2 and Table 3, a first glance shows that gradient-based adversarial attacks outperform other baseline attacks (e.g., CW, DeepFool) by a large margin on all black-box models and maintains high success rates on black-box model with same input length. We can observe that after clipping or self-splicing, adversarial examples almost lose their transferability on Rawnet models, and the average TSR drops by 41% on LCNN models. Meanwhile, we find that the adversarial samples generated by gradient-based adversarial attacks are the most threatening, such as PGD and MIFGSM.

Hyper-parameters: For CW, we set $c = 10$, learning rate $lr = 0.001$ and Optimization steps=100. For FGSM, we set step size $\alpha = 0.015$. For PGD-20, we set the step size $\alpha = 0.01$ and the number of iteration $T = 20$. For PGD-20,
Leveraging social media news

| White-box Model | Attack Method | ASR (%) | SNR (dB) |
|-----------------|--------------|---------|----------|
| Rawnet<sub>56</sub> | CW | 99.5 | 38.2 |
| | Deepfool | 100 | 27.8 |
| | PGD-20 | 92.9 | 28.0 |
| | PGD-40 | 96.9 | 25.3 |
| | MIFGSM-10 | 97.2 | 21.8 |
| | MIFGSM-20 | 97.7 | 19.6 |

| Black-box Models | Rawnet<sub>4</sub> | Rawnet<sub>48</sub> | Rawnet<sub>56</sub> | Rawnet<sub>646</sub> | Rawnet<sub>T=256</sub> |
|------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
|                  | 0.0 | 3.2 | 10.3 | 1.2 | 0.0 |
|                  | 0.0 | 13.2 | 6.1 | 4.1 | 0.0 |
|                  | 0.0 | 17.2 | 46.8 | 0.0 | 0.0 |
|                  | 0.1 | 4.8 | 35.8 | 1.5 | 0.2 |
|                  | 0.2 | 7.1 | 29.5 | 3.0 | 0.1 |
|                  | 0.2 | 16.5 | 34.6 | 3.4 | 0.2 |
|                  | 0.2 | 25.6 | 37.8 | 3.6 | 0.5 |

Table 2
The TSR(%) on Rawnets with different input lengths, where we attack a single network Rawnet<sub>56</sub> using typical adversarial attacks. * indicates the model with the different epoch in training.

| White-box Model | Attack Method | ASR (%) | SNR (dB) |
|-----------------|--------------|---------|----------|
| LCNN<sub>56</sub> | CW | 99.8 | 37.27 |
| | Deepfool | 100 | 54.4 |
| | PGD-20 | 100 | 32.57 |
| | PGD-40 | 100 | 32.58 |
| | MIFGSM-10 | 100 | 26.2 |
| | MIFGSM-20 | 100 | 25.3 |

| Black-box Models | LCNN<sub>56</sub> | LCNN<sub>56</sub><sup>*</sup> | LCNN<sub>48</sub> | LCNN<sub>646</sub> | LCNN<sub>T=256</sub> |
|------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
|                  | 25.2 | 11.2 | 59.8 | 0.11 | 0.65 |
|                  | 0.4 | 1.1 | 15.4 | 6.7 | 8.5 |
|                  | 47.8 | 75.3 | 99.9 | 51.3 | 69.6 |
|                  | 49.8 | 76.5 | 99.9 | 50.8 | 71.1 |
|                  | 40.4 | 66.2 | 99.9 | 52.8 | 63.2 |
|                  | 41.2 | 69.1 | 99.9 | 54.4 | 64.1 |

Table 3
The TSR(%) on LCNNs with different input lengths, where we attack a single network LCNN<sub>56</sub> using typical adversarial attacks. We did not calculate FGSM’s TSR because the ASR of FGSM is too low. * indicates the model with the different epoch in training.

we set the number of iteration $T = 40$. For MIFGSM-10, we set the decay factor $\mu = 1.0$ and $T = 10$.

5.4. Transferability of proposed method among models with the same architecture

In this subsection, we conduct a series of ablation experiments to study the impact of different parameters on the effectiveness of SA. We only consider attacking an ensemble of networks here, since it is much stronger than attacking a single network and can provide a more accurate evaluation of the network’s robustness. We choose models with lengths of 40,000, 56,000 and 72,600 as ensemble white-box models and craft adversarial examples with the length of 56000. Then we input the adversarial examples into five black-box models of different lengths after clipping or self-splicing, such as columns 6 to 10 in the second row in Tables 4 and 5, where * denotes the model with the different epoch in training. In the experiments, we only perform ablation experiments on the number of sub-segments $S_n$ and the number of noises $r$ in $V(\cdot; r)$. Other hyper-parameters are set as follows: the number of iterations $T = 50$, decay factor $\mu = 0.1$, the magnitude of perturbation $\epsilon = 0.05$ and step size $\alpha = 0.001$.

The results of the ablation experiments are shown in Tables 4 and 5. Note that we have marked the row with parameter $S_n = 0$ and $r = 0$ in grey. This is because SA is equivalent to MIFGSM when $S_n = 0$ and $r = 0$. The third column shows the attack success rate of the ensemble model in the white-box, and the fourth column shows the quality of the generated adversarial examples, the higher the better for both metrics. In Table 4, we attacked Rawnets based on raw waveforms using SA. We can observe that as $S_n$ and $r$ increase, the TSR of adversarial examples also increases. Compared with MIFGSM($S_n = 0$, $r = 0$), SA ($S_n = 8$, $r = 5$) improves TSR by 40.9% on Rawnet<sub>4</sub>, 25.4% on Rawnet<sub>48</sub>, 10.5% on Rawnet<sub>56</sub>, 32.8% on Rawnet<sub>646</sub>, 22.0% on Rawnet<sub>T=256</sub>. Our proposed method improves from 27.6% to 53.1%, 1.92 times that of MIFGSM. Moreover, our proposed method improves the ASR from 77.3% to 88.9%, an increase of 11.6%. In Table 5, we attacked LCNNs based on LFCC using SA. Similar trends can also be observed for SA on LCNNs. The TSR on black-box always increases as the $S_n$ and $r$. Compared with the first row ($S_n = 0$, $r = 0$), SA ($S_n = 8$, $r = 5$) improves TSR by 20.5% on LCNN<sub>56</sub>, 13.8% on LCNN<sub>48</sub>, 16.4% on LCNN<sub>646</sub>, 2.8% on LCNN<sub>T=256</sub>. Our proposed method improves from 27.6% to 53.1%, 1.92 times that of MIFGSM. Moreover, our proposed method improves the ASR from 77.3% to 88.9%, an increase of 11.6%.

Finally, comparing Tables 4 and 5, experiments show that LCNNs based on 2D features are more susceptible to adversarial examples. For instance, the attack success rate on the LCNN (ensemble white-box box) is always 100%, and the average TSR on other LCNNs (black-box) reaches 91.5%. However, the highest attack success rate on the Rawnet (ensemble white-box model) is only 88.9%, and the average TSR on Rawnets (black-box) is only 53.1%. In addition, the lowest SNR of adversarial examples generated by LCNN is 22.58, which is 3.67 higher than that generated by Rawnet, which indicates that Rawnet is more robust than LCNN.
Leveraging social media news

Table 4
The TSR(%) on Rawnets with different input lengths, where we attack an ensemble model using SA. * indicates the model with the different epoch in training.

| White-box Model | Hyper-parameters | ASR (%) | SNR (dB) | Black-box Models |
|-----------------|------------------|---------|----------|-----------------|
| Rawnet<sub>726</sub> | S<sub>n</sub> = 0, r = 0 | 77.3 | 21.7 | Rawnet<sub>4</sub> |
| + Rawnet<sub>56</sub> | 0 | 3 | 84.0 | 20.4 | Rawnet<sub>48</sub> |
| + Rawnet<sub>4</sub> | 5 | 0 | 80.2 | 20.92 | Rawnet<sub>56</sub> |
| 8 | 3 | 87.8 | 20.4 | Rawnet<sub>646</sub> |
| + Rawnet<sub>4</sub> | 8 | 3 | 88.9 | 19.1 | Rawnet<sub>726</sub> |

Table 5
The TSR(%) on LCNNs with different input lengths, where we attack an ensemble model using SA. * indicates the model with the different epoch in training.

| White-box Model | Hyper-parameters | ASR (%) | SNR (dB) | Black-box Models |
|-----------------|------------------|---------|----------|-----------------|
| LCNN<sub>726</sub> | S<sub>n</sub> = 0, r = 0 | 100 | 28.7 | LCNN<sub>4</sub> |
| + LCNN<sub>56</sub> | 0 | 3 | 100 | 22.8 | LCNN<sub>48</sub> |
| + LCNN<sub>4</sub> | 5 | 0 | 100 | 26.82 | LCNN<sub>56</sub> |
| 8 | 3 | 100 | 22.8 | LCNN<sub>646</sub> |
| + LCNN<sub>4</sub> | 8 | 5 | 100 | 22.58 | LCNN<sub>726</sub> |

5.5. Transferability of proposed method across models with different architecture

In subsection 5.4, we evaluate the transferability of adversarial examples generated by SA between the same model and input lengths. In this section, we conduct the more practical case where the victim’s model structure is different from the white-box model. We choose Rawnet as the white-box model and generate adversarial examples of length 56000 on the ensemble model Rawnet<sub>4</sub>+Rawnet<sub>56</sub>+Rawnet<sub>726</sub>. These adversarial examples are input into LCNNs of different lengths after clipped or self-spliced, and the corresponding TSRs are calculated.

The hyperparameter settings for this experiment follow the setting of subsection 5.4. The results are shown in Table 6, where we observe that the transferability of adversarial examples between different models is greatly reduced compared to Table 5. This is because although we attack the ensemble model, the structure of the ensemble model is the same, which is equivalent to attacking a single model. As in Table 4 and Table 5, the row marked gray in Table 6 represents MIFGSM (S<sub>n</sub> = 0, r = 0). Compared with the first row (MIFGSM), our proposed method improves TSR by 13.1% on LCNN<sub>4</sub>, 6.0% on LCNN<sub>48</sub>, 23.5% on LCNN<sub>56</sub>, 2.9% on LCNN<sub>646</sub>, and 30.0% on LCNN<sub>726</sub>. The results show that SA can effectively improve the transferability of adversarial examples between models with different structures after clipping or self-splicing.

6. CONCLUSION

In this paper, we consider the more practical case where the input length of the victim model is inconsistent with the input length of the white-box model used by the attacker. Therefore, the victim classifier usually needs to clipping or self-splicing the input samples to ensure that the input length is consistent. We extensively evaluate typical adversarial attacks and observe that they produce adversarial examples
that are clipped or self-spliced to be much less threatening to the victim model. We observe that segments with the same sample value have similar gradients in different models. We, therefore, propose an adversarial attack SA that attacks models of different lengths. SA is an ensemble adversarial attack, where the ensemble model consists of three models with different input lengths. We obtain the number of sub-segments of different lengths through a sliding window and noise function and feed them into models of different lengths, which allows us to capture more gradients and avoid local optima. Through extensive experiments, we show that the adversarial examples generated by SA are 25.5% and 10.7% higher in average transfer success rate for two representative models (Rawnets and LCNNs) after clipping or self-splicing. In addition, SA outperforms existing ensemble adversarial attack methods in attack success rate against white-box models (ensemble Rawnets), increasing from 77.3% to 88.9%. Finally, we evaluate the transferability of adversarial examples generated by SA between different model architectures. We use SA to generate 56,000 adversarial examples based on Rawnets, then clip or self-splice the adversarial examples and predict them with LCNNs of different input lengths. The results show that our proposed method effectively improves transferability by 15%. We hope our proposed attack strategy can serve as a benchmark for improving the transferability of adversarial examples after clipping or self-splicing.

CRediT authorship contribution statement

Jiacheng Deng: Conceptualization of this study, Methodology, Experiments. Li Dong: Supervision, Writing - original draft, Writing - review & editing, Validation. Yan Diqun: Writing - Original draft preparation. Ranging Wang: Supervision, Writing - original draft, Writing - review & editing, Validation. Jiaming Zeng: Assist experiments.

References

[1] Altes, Richard, A., 1980. Detection, estimation, and classification with spectrograms. Journal of the Acoustical Society of America 67, 1232–1246.
[2] Athalye, A., Carlini, N., Wagner, D., 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, in: International conference on machine learning, PMLR. pp. 274–283.
[3] Carlini, N., Wagner, D., 2017. Towards evaluating the robustness of neural networks, in: 2017 ieee symposium on security and privacy (sp), Ieee. pp. 39–57.
[4] Das, P.P., Allayear, S.M., Amin, R., Rahman, Z., 2016. Bangladeshi dialect recognition using mel frequency cepstral coefficient, delta, delta-delta and gaussian mixture model, in: 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI).
[5] Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J., 2018. Boosting adversarial attacks with momentum, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 9185–9193.
[6] Dong, Y., Pang, T., Su, H., Zhu, J., 2019. Evading defenses to transferable adversarial examples by translation-invariant attacks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4312–4321.
[7] Fan, Y., Qian, Y., Xie, F.L., Soong, F.K., 2014. Tts synthesis with bidirectional lstm based recurrent neural networks, in: Fifteenth annual conference of the international speech communication association.
[8] Goodfellow, I., Lee, H., Le, Q., Saxe, A., Ng, A., 2009. Measuring invariances in deep networks. Advances in neural information processing systems 22.
[9] Goodfellow, I.J., Shlens, J., Szegedy, C., 2015. Explaining and harnessing adversarial examples, in: International Conference on Learning Representations.
[10] Hossan, M.A., Memon, S., Gregory, M.A., 2010. A novel approach for mfcc feature extraction, in: International Conference on Signal Processing & Communication Systems.
[11] Houst, D.I., Richards, R.E., 2011. The signal-to-noise ratio of the nuclear magnetic resonance experiment. Journal of Magnetic Resonance 213, 329–343.
[12] Hua, G., Teoh, A.B.J., Zhang, H., 2021. Towards end-to-end synthetic speech detection. IEEE Signal Processing Letters 28, 1265–1269.
[13] Kauderer-Abrams, E., 2017. Quantifying translation-invariance in convolutional neural networks. arXiv preprint arXiv:1801.01450 .
[14] Kim, D.H., Seong, W.K., Hong, K.K., 2015. Performance comparison of deep feature based speaker verification systems. Phonetics and Speech Sciences 7, 9–16.
[15] Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: International Conference on Learning Representations (ICLR).
[16] Koerich, K.M., Esmailpour, M., Abdoli, S., Britto, A.d.S., Koerich, A.L., 2020. Cross-representation transferability of adversarial attacks: From spectrograms to audio waveforms, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–7.
[17] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems 25.
[18] Kurakin, A., Goodfellow, I.J., Bengio, S., 2018. Adversarial examples in the physical world, in: Artificial intelligence safety and security. Chapman and Hall/CRC, pp. 99–112.
[19] Lavrentyeva, G., Tseren, A., Volkova, M., Gorlanov, A., Kozlov, A., Novoselov, S., 2019. Stc antiSpoofing systems for the asvspoof2019 challenge, in: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, pp. 1033–1037.
[20] LeCun, Y., Bengio, Y., et al., 1995. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks 3361, 1995.
[21] Li, Y., Li, L., Wang, L., Zhang, T., Gong, B., 2019. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks, in: International Conference on Machine Learning, PMLR. pp. 3866–3876.
[22] Lin, J., Song, C., He, K., Wang, L., Hopcroft, J.E., 2019. Nesterov accelerated gradient and scale invariance for adversarial attacks, in: International Conference on Learning Representations.
[23] Liu, Y., Chen, X., Liu, C., Song, D., 2016. Delving into transferable adversarial examples and black-box attacks, in: International Conference on Learning Representations.
[24] Mohammadi, S.H., Kain, A., 2017. An overview of voice conversion systems. Speech Communication 88, 65–82.
[25] Naseer, M., Khan, S., Hayat, M., Khan, F.S., Porikli, F., 2020. A self-supervised approach for adversarial robustness, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 262–271.
[26] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A., 2017. Practical black-box attacks against machine learning, in: Proceedings of the 2017 ACM on Asia conference on computer and communications security, pp. 506–519.
[27] Sahidullah, M., Kinnunen, T., Hanliç, C., 2015. A comparison of features for synthetic speech detection, in: Interspeech 2015, International Speech Communication Association. pp. 2087–2091.
[28] Subramanian, V., Benetos, E., Sandler, M., 2019. Robustness of ad-
Leveraging social media news

[29] Subramanian, V., Pankajakshan, A., Benetos, E., Xu, N., McDonald, S., Sandler, M., 2020. A study on the transferability of adversarial attacks in sound event classification, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 301–305.

[30] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R., 2014. Intriguing properties of neural networks, in: 2nd International Conference on Learning Representations, ICLR 2014.

[31] Tak, H., Patino, J., Todisco, M., Nautsch, A., Evans, N., Larcher, A., 2021. End-to-end anti-spoofing with rawnet2, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 6369–6373.

[32] Toda, T., Chen, L.H., Saito, D., Villavicencio, F., Wester, M., Wu, Z., Yamagishi, J., 2016. The voice conversion challenge 2016, in: Interspeech 2016, International Speech Communication Association. pp. 1632–1636.

[33] Todisco, M., Delgado, H., Evans, N., 2017. Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification. Computer Speech & Language 45, 516–535.

[34] Todisco, M., Wang, X., Vestman, V., Sahidullah, M., Delgado, H., Nautsch, A., Yamagishi, J., Evans, N., Kinnunen, T., Lee, K.A., 2019. Asvspoof 2019: Future horizons in spoofed and fake audio detection, in: INTERSPEECH 2019-20th Annual Conference of the International Speech Communication Association.

[35] Wang, X., He, K., 2021. Enhancing the transferability of adversarial attacks through variance tuning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1924–1933.

[36] Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., Yuille, A.L., 2019. Improving transferability of adversarial examples with input diversity, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2730–2739.

[37] Xing, F., Hansen, 2015. Speaker identification with whispered speech based on modified ilcc parameters and feature mapping, in: IEEE.

[38] Yamagishi, J., Wang, X., Todisco, M., Sahidullah, M., Patino, J., Nautsch, A., Liu, X., Lee, K.A., Kinnunen, T., Evans, N., et al., 2021. Asvspoof 2021: accelerating progress in spoofed and deepfake speech detection, in: ASVspoof 2021 Workshop-Automatic Speaker Verification and Spoofing Countermeasures Challenge.

[39] Yang, Y., Wang, H., Dinkel, H., Chen, Z., Wang, S., Qian, Y., Yu, K., 2019. The sjtu robust anti-spoofing system for the asvspoof 2019 challenge, in: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, pp. 1038–1042.

[40] Zheng, T.F., Zhang, G., Song, Z., 2001. Comparison of different implementations of mfcc. Journal of Computer Science and Technology 16, 582–589.