IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 18, 2020, accepted June 26, 2020, date of publication June 30, 2020, date of current version July 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3006130

Targeted Speech Adversarial Example Generation With Generative Adversarial Network

DONGHUA WANG, (Student Member, IEEE), LI DONG, (Member, IEEE), RANGDING WANG, (Member, IEEE), DIQUN YAN, (Member, IEEE), AND JIE WANG, (Student Member, IEEE)

College of Information Science and Engineering, Ningbo University, Zhejiang 315211, China

Corresponding authors: Li Dong (dongli@nbu.edu.cn) and Rangding Wang (wangrangding@nbu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1736215, Grant 61672302, and Grant 61901237, in part by the Zhejiang Natural Science Foundation under Grant LY20F020010 and Grant LY17F020010, in part by the K.C. Wong Magna Fund, Ningbo University, and in part by the Natural Science Foundation of Ningbo under Grant 2019A610103.

ABSTRACT Although neural network-based speech recognition models have enjoyed significant success in many acoustic systems, they are susceptible to be attacked by the adversarial examples. In this work, we make first step towards using generative adversarial network (GAN) for constructing the targeted speech adversarial examples. Specifically, we integrate the target speech recognition network with GAN framework, which can then be formulated as a three-party game. The generator in GAN aims at generating perturbation that could make the target network misclassified to a specific target, while simultaneously fooling the discriminator treating the adversarial example as a beguine one. The discriminator is to distinguish the crafted examples from the geniue samples. The classification error of the target network is back-propagated via gradient flow to the generator for updating. The target network is responsible for back-propagating the classification error via gradients to the generator for updating, but the target network itself is freezed. With the carefully designed network architecture, loss function and training strategy, we successfully train a generator that could generate the adversarial perturbation for a given speech clip and a target label. Experiential results show that the generated adversarial examples could effectively fool the state-of-the-art speech classification networks, while attaining an acceptable auditory perception quality. In addition, our proposed method runs much faster than the prevalent optimization-based schemes. To facilitate reproducible research, codes, models and data are publicly available at https://github.com/winterwindwang/SpeechAdvGan.

INDEX TERMS Targeted adversarial example, generative adversarial network, speech recognition.

I. INTRODUCTION

Nowadays, the speech user interface is becoming one of the most prevalent human-machine-interaction ways. It has been widely adopted in numerous size-constrained smart equipments, wearable devices, and hand-free intelligent systems, where the user input via a physical or screen keyboard is often inconvenient. In general, these automatic speech recognition (ASR) systems are dependent on running a speech classification model in an always-on mode, receiving the voice and interpret it as commands. Due to the recent significant advances of deep neural networks, many state-of-the-art commercial products, e.g., Apple Siri, Google Now and Amazon Alexa, are shifting to the network-based speech classification model. Considering the widespread applications of ASR in real-world, the security concerns of these systems have drawn the attentions of researchers. Unfortunately, many works have demonstrated that neural networks are susceptible to the specially crafted speech adversarial examples, which are typically constructed by adding peculiar perturbation on the legitimate samples, causing the target acoustic system misbehave. One typical way is to generate a malicious perturbation based on the given legitimate sample, which thus could intentionally make the target acoustic system to produce the incorrect results. The existence of adversarial example is ubiquitous in many vision tasks such as image classification [1], [2], object detection [3], [4], face recognition and authentication [5], to name a few.

Compared with extensive research in the vision field, there are less investigations for the speech adversarial example

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad Ayoub Khan.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

generation. Vaidya *et al.* [6] is the first to report speech recognition system is vulnerable to adversarial examples. They constructed the adversarial examples by adding perturbation on the Mel-Frequency Cepstral Coefficient (MFCC)-transformed feature, and then converted the processed feature back to the waveform domain. Generally, their generated samples were noise-dominated, which cannot be interpreted by humans, but intelligible to the speech recognition system. Similarly, Carlini *et al.* [7] also suggested to perturb the MFCC feature vector of the given speech. Their results shown that the constructed adversarial examples could effectively attack the Hidden Markov Model-based ASR. Later on, Carlini and Wagner [8] extended the seminal fast gradient sign method (FGSM), which is well-known on image classification adversarial attack, to create the speech adversarial examples. By solving an iterative optimization problem, they could found a proper adversarial sample that indistinguishable to human but can fool the target ASR network. Alzantot *et al.* [9] proposed a genetic algorithm-based method to generate speech adversarial examples. Their attack modifies the least significant bits of a subset of the given audio clip, which is equivalent to injecting the background noise directly into the original speech. More recently, Du *et al.* [10] devised a method termed as SirenAttack, which employed the heuristic particle swarm optimization algorithm, to search the adversarial examples in a stochastic fashion. Experiments have shown the sought examples could trigger many CNN-based speech classification systems to produce incorrect results.

However, most of the aforementioned speech adversarial example generation methods have at least one of the following two drawbacks: First, the computational cost for generating the adversarial example for a given speech is quite high (e.g., the method [8] spends almost one hour to generate a single adversarial example, even with GPU acceleration), due to the high-complicated iterative optimization problem to be solved. This inefficiency issue barriers the practical usage of the adversarial examples. Second, the perception quality of the adversarial examples was not always satisfactory (some generated examples are even noise-dominated [6], [7]), which could clearly draw the attention of a common listener. In this work, we attempt to tackle the above issues using GAN. Specifically, we propose to integrate the target network with a GAN framework, which is formulated as a three-party game. The generator aims at generating perturbation that meets two goals simultaneously: 1) fooling the discriminator treating the adversarial example as the geniue one, and 2) making the target network misclassified to a specific target. The discriminator is to distinguish the crafted adversarial examples from the geniue samples. It could guide the generator to generate limited perturbation that resembles to the distribution of geniue ones. The target network participate the training as fixed role. Its classification error is feedback via gradient flow to the generator, guiding the generator to adjust the perturbation towards the misclassification directions. To implement the three goals, the GAN network architecture is carefully designed, with deliberately designed loss function. We also suggest an effective training strategy to train the proposed GAN networks stably. Experimental results validate the attacking capability of proposed method on some widely-used ASR networks, while retaining acceptable perception quality. The contributions of this work are summarized as follows

- By incorporating the target ASR into our specially devised GAN framework, we successfully trained a generator that could generate the adversarial perturbation. Moreover, the loss function and training strategy were carefully designed, which could stabilize the training of our proposed GAN framework.
- The proposed GAN-based attacking method could *efficiently* generate adversarial examples with pre-specified target label, comparing with the recent state-of-the-art speech adversarial example generation schemes.
- We conduct extensive experiments to demonstrate our generated adversarial examples could effectively fool the ASR systems with high confidence, while retaining reasonably good perception quality.

The rest of this work is organized as follows. We first briefly review related work in Section II, and then formulate the adversarial example generation problem in Section III. Our proposed targeted speech adversarial example generation using GAN is presented in Section IV, with a thorough discussion on the network architecture, loss function and training strategy. Experimental results are provided in Section V, and finally we conclude this work in Section VI.

II. RELATE WORK

In this section, we first review the generative adversarial networks, and then survey the recent advances of the speech adversarial example generation task.

A. GENERATIVE ADVERSARIAL NETWORK

The generative adversarial network (GAN) was first proposed by Goodfellow *et al.* [11]. The essential idea of GAN is to establish a contest between two networks, i.e., the generator and discriminator networks. Through adequate competing, the generator acquires the capability to generate the instances that (approximately) follow the distribution of training data. GAN has harvested great success in many vision tasks such as realistic image generation [12], image-to-image translation [13], and text-to-image synthesis [14]. Recently, the methods using GAN to generate image adversarial example emerge. For example, Hu and Tan [15] proposed to use GANs to generate adversarial image examples, which could bypass detection systems. Xiao *et al.* [16] utilized GAN to generate realistic adversarial image examples, which could learn and approximate the distribution of the original data samples. Compared with extensive research of GAN in the image domain, there are far less works in the speech domain. Pascual *et al.* [17] devised a generative adversarial network for speech enhancement task, which could remove noise from

speech and obtain much more cleaner clips. Latif *et al.* [18] employed GAN as defense measures to the adversarial attack. They used GAN to eliminate suspicious noise exists in a given speech, which make the target network perform classification correctly as intended. Tang *et al.* [19] proposed GAN-based data augmentation method. Li *et al.* [20] utilized improved GAN to efficiently address signal processing task. Note that, GAN has also entered into the digital forensics filed. For instance, to resist the audio source identification (ASI) forensics, Li *et al.* [21] proposed to use GAN to falsify the source information of an audio clip by adding specific disturbance. The doctored audio clips can effectively deceive numerous ASI forensic methods.

B. SPEECH ADVERSARIAL EXAMPLE GENERATION

Recently, the adversarial examples generation in the speech domain has also drawn attention of researchers. Carlini and Wagner [8] employed the conventional fast gradient sign method (FGSM) method to generate adversarial perturbation. The results have shown that the generated adversarial examples can successfully attack a speech recognition network. Qin *et al.* [22] also proposed optimized-based algorithms to find the optimal perturbations under the small-magnitude perturbation constraint. Du *et al.* [10] then proposed a particle swarm optimization algorithm based method to construct speech adversarial examples. This method involves solving complicated optimization problem with multiple heuristic tricks. Note that, the computational cost of the above optimization-based methods are all huge. This is mainly because, for each sample, a complex optimization problem must be solved once to generate its corresponding adversarial example. Instead, our proposed framework implants the target network to a generative adversarial network, and uses the gradient information feedback by the target network to guide the training of the generator. Once completing the training, the generator can efficiently generate perturbation for a given speech.

III. PROBLEM FORMULATION

Currently, adversarial attacks can be roughly classified into untargeted attack and targeted attack. The untargeted adversarial examples generation aim to make the output of system deviate from the original classification, to arbitrary incorrect label, e.g., introducing spelling errors to attack the ASR systems. In contrast, the targeted attack attempts to make the recognition systems outputs a pre-specified target label. Compared with the untargeted attack, the target attack is practical and much more challenging. Therefore, in this work, we focus on investigating the targeted attack. Furthermore, for technical tractability, we assume a white-box attacking scenario, where one can access the necessary information about the target system $f(\cdot)$, e.g., the network architecture, weight parameters, hyperparameter settings.

Specifically, let $\mathcal{X}$ be the speech audio waveform space, and $\mathcal{Y}$ be its ground-truth text label space. A well-learned ASR is expected to receive a speech sample clip $x \in \mathcal{X}$, and outputs its transcription $y \in \mathcal{Y}$, i.e., $f(x) = y$. Given such ASR $f(\cdot)$, an input speech sample x, and a pre-specified transcription $y_t \neq y$, the goal of our targeted speech adversarial example generation scheme is two-folds. First, we shall to generate a speech adversarial example x^{adv} that could effectively fool the given ASR classification, i.e.,

$$f(x^{\text{adv}}) = y_t, \quad \text{where } y_t \neq y. \tag{1}$$

Second, the generated adversarial example x^{adv} should be similar to the original sample x, in terms of human auditory perception. That is, a common human listener cannot distinguish x^{adv} from x when listening to them.

More formally, to fulfill the aforementioned two goals, we aims at finding a small perturbation δ such that $x^{\text{adv}} = x + \delta$. Under this setting, the adversarial example generation algorithm can be converted into solving the following optimization problem

$$\delta^* = \arg\min_{\delta} l(f(x+\delta), y_t) \tag{2}$$

$$\text{s.t.: } y_t \neq y, \quad \text{and } \|\delta\| < \epsilon. \tag{3}$$

where $l(\cdot)$ is the loss function to evaluate the prediction/classification accuracy, which achieves the minima when $f(x^{\text{adv}}) = y_t$. Here $\|\cdot\|$ is the norm operator, e.g., l_1, l_2, l_∞, and ϵ denotes the maximum incurred perturbation magnitude.

However, most of the existing adversarial example generation schemes seek the perturbation by solving complicated optimization problems [9], [10]. Thus, for a given speech sample, it is costly for generating its adversarial example. Furthermore, their results have shown that the adversarial perturbation is often large-magnitude, making the audio clip sounds much noisy, especially in the silent frames. Instead, as will be detailed shortly in next section, we propose to employ a generative adversarial network to generate such perturbation, which could atomically determines where is suitable to add adversarial noise for the given speech clip. In addition, once completing the training procedure, our method could generate the adversarial examples quite efficiently.

IV. TARGETED SPEECH ADVERSARIAL EXAMPLE GENERATION USING GAN

In this section, we first present the network architecture of our proposed method. The loss function is then thoroughly discussed and finally the training strategy is provided.

A. FRAMEWORK OVERVIEW

Figure 1 illustrates the overall architecture of our proposed method. The key idea of our method is to employ the generative adversarial network (GAN) to produce a specific perturbation for the given speech sample. In general, our framework is mainly composed of three components: a generator $\mathcal{G}$, a discriminator $\mathcal{D}$, and the target ASR network f. This is a game among three players: generator $\mathcal{G}$, discriminator $\mathcal{D}$ and ASR network f, where the ASR part is fixed. The goal of the GAN components is to learn the latent knowledge to fool

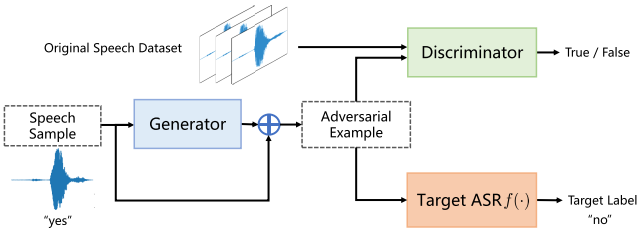


FIGURE 1. The overview of the proposed framework.

the target ASR network, under small perturbation constraint. Mathematically, the original speech x is fed into generator and its output $\mathcal{G}(x)$ as the adversarial perturbation, which can be expressed as

$$x^{\text{adv}} = x + \mathcal{G}(x). \tag{4}$$

Then, x^{adv} is fed into the discriminator $\mathcal{D}$ and the target ASR network f, respectively. The function of discriminator $\mathcal{D}$ is to distinguish the constructed adversarial examples from the geniue ones, driving the generator $\mathcal{G}$ to produce the perturbation that is unnoticeable to the discriminator. The target classifier f is involved to guide the generator to craft adversarial examples that would be misclassified to a pre-specified target label. The detailed network architecture of the generator $\mathcal{G}$ and discriminator $\mathcal{D}$ are presented as follows.

B. GENERATOR

As aforementioned, the aim of generator $\mathcal{G}$ is to generate adversarial perturbation, making the adversarial example misclassfied but not harming the perception quality of speech severely. To realize this goal, we adopt the *encoder-decoder like* network structure to design our generator. In practice, such encoder-decoder framework has successfully applied to many applications, e.g., U-Net for image segmentation [23], SEGAN for speech enhancement [17].

The architecture of the generator is illustrated in Figure 2. It contains 8 convolutional layers as encoding component and 8 deconvolutional layers as the decoding component. Specifically, each convolutional layer contains multiple convolution filters,[1] with the kernel size 1×32 and stride 2. The parametric rectified linear units (PReLU) activation function [24] is used, which could benefit the training stability in our experiments. The deconvolutional component is composed of the quasi-symmetrical structure of the convolutional component. Each deconvolutional layer also contains multiple deconvolution filters, with the kernel size 1×32 and stride 2, followed by PReLU activation. Inspired by SEGAN [17], in order to make the gradients flow through the network without suffering vanishing, we add the skip connection that wires the convolutional layer with its deconvolutional layer counterpart. Skip connection could make the deconvolutional layer directly share the features that extracted by the convolutional layer, preventing to generate exaggerate perturbation that deviate from original signal. Finally, it is worth pointing out that, in the output of last layer, we use tangent activation function to replace PReLU. This goal of this action is to the enforce the generated perturbation into the same sampling values of the speech, i.e., $[-1, 1]$. To ensure the final adversarial sample being a valid speech sample, the generated adversarial example will be hard truncated into $[-1, 1]$ when overflow occurs.

[1]The number of filters is labeled right below the output of the convolutional layer. For instance, the first layer contains 16 filters, which transforms the input of 1×16384 into a tensor of size 16×8192.

C. DISCRIMINATOR

The discriminator $\mathcal{D}$ in our framework is required to distinguish the original speech samples from the constructed adversarial ones. It could motivate the generator to produces peculiar perturbation that mimics the noise distribution exists in geniue samples. The detailed architecture of the discriminator is presented in Figure 3. In more detail, the discriminator is formed by 12 convolutional blocks. For the first 11 convolutional blocks, each contains certain number of filters with kernel size 1×31, followed by batch-norm and Leaky-ReLU. For the last convolution layer, there is only filter with 1×31 and Leaky-ReLU, aiming to squeeze the feature maps into a one-dimensional vector. No batch-norm operation is used in this layer. The last two layers are fully connected layer and softmax layer, respectively, which are used to output the final prediction probability in the range [0, 1]. In other words, the output of $\mathcal{D}(x)$ indicates the confidence that the input speech sample is classified as a genuine one.

D. LOSS FUNCTION

To learn a generator that mislead the ASR to recognize adversarial example as a specified target, the loss is the critical element to drive the training process. Thus, we design the loss function elaborately. In general, there are two networks, i.e., generator $\mathcal{G}(\cdot)$ and discriminator $\mathcal{D}(\cdot)$, to be trained. Each network shall be trained with a well designed loss function. More specifically, for the generator, we define its loss function as

$$L_{\mathcal{G}} = L_{\text{adv}}^{f} + \alpha L_{\text{fool}} + \beta L_{\text{hinge}} + \gamma L_2, \tag{5}$$

where L_{adv}^{f} is the adversarial loss, representing the attacking ability of generator on the target classifier; L_{fool} is the loss to denote the fooling capability of generator on the discriminator. The last two terms L_{hinge} and L_2 are the hinge and l_2-norm loss of the adversarial example, respectively. These two regularization terms are adopted here to expect an acceptable auditory quality of the adversarial speech example. The parameters α, β and γ are the weights to balance the importance among the four loss terms. In the next, we give more detailed explanation on each term in (5).

1) TARGET CLASSIFIER LOSS L_{adv}^{f}

One primary goal of the generator is to fool the target ASR network. Then the target classifier loss L_{adv}^{f} shall measure the

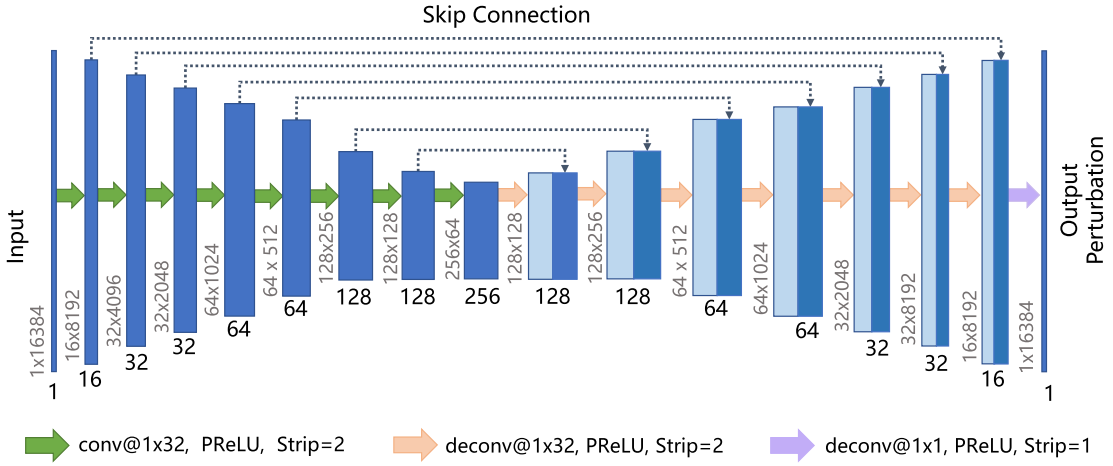


FIGURE 2. The architecture of the generator.

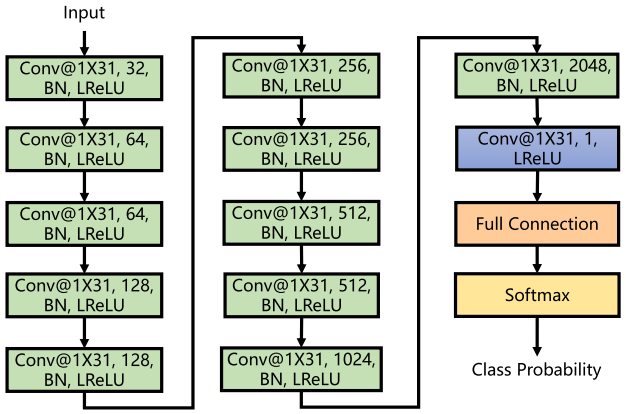


FIGURE 3. The architecture of the discriminator. Different color represents different layer types. The text in green blocks all follow the syntax: *K* convolution filters with kernel size 1 × 31 followed by batch-norm (BN) and Leaky-ReLU (LReLU) is abbreviated as Conv@1 × 31, *K*, BN, LReLU.

difference between the targeted label with the classifier prediction of the adversarial example. Mathematically, we have

$$L_{\text{adv}}^{f} = \mathbb{E}_x \left[l_{\text{ce}} \left(f(x^{\text{adv}}), y_t \right) \right], \tag{6}$$

where $\mathbb{E}_x[\cdot]$ denotes the expectation operator over the training samples; y_t is the pre-specified target labeled of an adversary, x^{adv} is the generated adversarial examples, and $l_{\text{ce}}(\cdot)$ denotes the cross-entropy loss function. As one can see, minimizing the loss L_{adv}^{f} encourages the ASR network classifier to classify the adversarial speech as the target label y_t.

2) ADVERSARIAL LOSS L_{fool}

To encourage the generator to seek a suitable perturbation that misleads the classification of discriminator, the generator should receive the feedback of its adversary (i.e., the discriminator) to improve its fooling ability consistently. To this end, the adversarial loss is designed as follows [11]

$$L_{\text{fool}} = \mathbb{E}_x \left[\log(1 - \mathcal{D}(x^{\text{adv}})) \right]. \tag{7}$$

It can be seen that minimizing L_{fool} is equivalent to maximizing the classification probability $\mathcal{D}(x^{\text{adv}})$ towards 1. This essentially guides the generated adversarial example to mimic the geniue speech data distribution.

3) REGULARIZATION TERMS L_{hinge} AND L_2

The key goal of this two terms is to ensure the auditory quality of constructed adversarial examples. First, the soft hinge loss on L_{hinge} norm is imposed on the generated perturbation, which can be expressed as follows

$$L_{\text{hinge}} = \mathbb{E}_x \left[\max(0, \|\mathcal{G}(x)\|_2 - c) \right], \tag{8}$$

where c denotes a user-specified bound. Intuitively, this loss function pushes the generator to generate more *sparse* perturbation throughout the all the sampling locations. i.e., the perturbation that modifies fewer speech sampling values is preferred. In addition, as shown in [13], such hinge loss could also stabilize the GAN training process.

Second, to make the adversarial example similar to the geniue one, we further incorporate following l_2-norm to bound the magnitude of the adversarial perturbation, i.e.,

$$L_2 = \|x^{\text{adv}} - x\|_2. \tag{9}$$

This loss function could control the total energy of the perturbation, avoiding the situation that perturbation explodes. That is, the perturbation with smaller magnitude is preferred when training the generator.

For the discriminator, its goal is to distinguish the generated adversarial example (fake) from the geniue ones (true). We adopt the following loss function

$$L_{\mathcal{D}} = \frac{1}{2} \left(\mathbb{E}_x \left[\log(1 - \mathcal{D}(x)) \right] + \mathbb{E}_x \left[\log(\mathcal{D}(x^{\text{adv}})) \right] \right), \tag{10}$$

where $\mathcal{D}(x^{\text{adv}}) \in [0, 1]$ denotes the probability that the crafted adversarial example is classified as geniue by the discriminator $\mathcal{D}$. Note that this loss function for discriminator is similar to the one employed in LSGAN [25], in which its effectiveness was validated for the GAN training procedure.

Algorithm 1 Training Strategy to Solve Problem (11)

Input: Target label y_t, training dataset, iteration index i, number of warm-up epochs N, number of maximum epochs M, min-batch size m, hyperparameters α, β, γ.

Output: The well trained generator $\mathcal{G}^*$ parameterized by $\theta_{\mathcal{G}}$.

Initialization: Initialize the network parameters of generator $\theta_{\mathcal{G}}$ and discriminator $\theta_{\mathcal{D}}$ using Xavier.
for ith iteration smaller than M **do**
  Randomly draw m samples from the training dataset.
  Generate m adversarial examples via $x^{\text{adv}} = x + \mathcal{G}(x)$.
  if $(i \leq N)$ **then**
    Update the generator parameters $\theta_{\mathcal{G}}$ via the back-propagated gradients $\nabla_{\theta_{\mathcal{G}}} \alpha L_{\text{fool}} + \beta L_{\text{hinge}} + \gamma L_2$.
  else
    Update the generator parameters $\theta_{\mathcal{G}}$ via the back-propagated gradients $\nabla_{\theta_{\mathcal{G}}} L_{\mathcal{G}}$. *// ASR joins the training, receives and backpropagates the gradients but without updating.*
  end if
  Update the discriminator parameters $\theta_{\mathcal{D}}$ via the back-propagated gradients $\nabla_{\theta_{\mathcal{D}}} L_{\mathcal{D}}$.
end for

Finally, by combining the loss function for generator ($L_{\mathcal{G}}$) and discriminator ($L_{\mathcal{D}}$), we can obtain the generator by solving the following min-max optimization problem

$$\mathcal{G}^* = \arg\min_{\mathcal{G}} \max_{\mathcal{D}} \left(L_{\mathcal{G}} + L_{\mathcal{D}} \right), \tag{11}$$

where $\mathcal{G}^*$ is the well-trained generator we sought. Upon training completion, one can efficiently generate the adversarial for the given input and specified target with $\mathcal{G}^*$. However, solving (11) is not a trivial task. We in the next discuss the training strategy that deliberately designed for the targeted adversarial example generation task.

E. TRAINING STRATEGY

The GAN network component in our proposed framework (i.e., the generator $\mathcal{G}$ and $\mathcal{D}$) is first trained for N epochs (N is empirically set as 3 in our experiments). Then, the target ASR system, i.e., the speech classifier network, joins for the remaining training. The underlying motivation of this training strategy is to ensure the discriminator could has reasonably good capability for distinguishing the adversarial samples from the geniue ones, which could warm-start the training process. The generator and discriminator are trained in an alternating way, i.e., when training the generator, the discriminator is fixed and vice versa. It is also worth pointing out that the network parameters of the target network are fixed during the entire training stage. That is, the ASR prediction errors can be backpropagated via gradients to the generator $\mathcal{G}$, but the weights of target network itself are not updated. Algorithm 1 presents the complete description of our training strategy.

V. EXPERIMENTAL RESULTS

In this section, we first introduce the dataset and experiential settings that are used throughout the experiments. Then, an adversarial attack on the state-of-the-art network-based ASR is implemented, to verify the effectiveness of our proposed method. Following that, the perceptual quality of the generated adversarial example is evaluated. Afterwards, ablation study on the loss terms L_2 and hinge loss is conducted. Finally, robustness of our proposed method is analyzed.

A. EXPERIENTIAL SETTINGS

1) DATASET

Our experiments are conducted on two datasets: the Google single-word speech command dataset `SpeechCmd` [26] and the musical genre collection `GTZAN` [27]. The speech commands dataset `SpeechCmd` is composed of 65000 speech files, where each file is one-second speech clip that falls into the following ten classes: ''yes'', ''no'', ''up'', ''down'', ''left'', ''right'', ''on'', ''off'', ''stop'', and ''go''. The `GTZAN` genre collection contains 1000 speech files, where each file is about 30-second music recording excerpts belonging to one of the following ten genres: ''blues'', ''classical'', ''country'', ''disco'', ''hiphop'', ''jazz'', ''metal'', ''pop'', ''reggae'', and ''rock''. To unfiy the length of the input speech clip, in our experiments, each file of `GTZAN` is spitted into multiple one-second clip. Finally, the total speech files in the `GTZAN` is 29000.[2] For each dataset, we split it into a training set, a validation set and a test set in the ratio 80% : 10% : 10%. As practiced in previous work [17], before feeding into the generator, each speech shall be first normalized into $[-1, 1]$ with

$$\text{Normlize}(x) = \frac{2}{65535} \cdot (x - 32767) + 1, \tag{12}$$

and the final adversarial speech examples can be reconstructed using the inverse-normalize function,

$$\text{Inversion}(x) = (x - 1) \cdot \frac{65535}{2} + 32767. \tag{13}$$

2) IMPLEMENTATION DETAILS

a: PARAMETER SETTINGS

The weight hyperparameters α, β and γ in Eqn. (5) are set to 1, 1 and 100, respectively. Note that, our experiments shown that the setting of γ is crucial to preserve the quality of generated adversarial examples. In general, a large value γ leads to higher perception quality of the generated speech samples. The parameter c in Eqn. (8) is empirically set to 0. The initial random seed is set to 1024. The learning rate is set to 2×10^{-4}. The number of training epoch is set to 50. The min-batch size is 64. Throughout the entire training process, the Adam optimizer [28] is adopted.

[2]Notice that, in the original `GTZAN` dataset, the duration of some files are not exactly the 30 seconds. To account this fact, we in practice take the first 29 seconds of each file and then divide it into multiple non-overlap clip of one second duration. This yields $29 \times 10000 = 29000$ files.

b: STOPPING CRITERION

The training process of our method would be halted when meets one of the following two conditions: First, the number of the training epoch reaches the specified maximum value. Second, the attacking success rate does not change within 5 epochs and no training instability such as gradient vanish or explosion exists.

3) PERFORMANCE METRIC

The performance of our proposed method is evaluated by the attack success rate of the generated adversarial examples, which can be expressed as

$$\text{success\_rate} = \frac{\#\{\text{misclassified samples}\}}{\#\{\text{test samples}\}}, \tag{14}$$

where $\#\{\cdot\}$ returns the cardinality of the input set.

To evaluate the objective quality of generated adversarial example, the Signal-to-Noise Ratio (SNR) metric is used,

$$\text{SNR}(x^{\text{adv}}) = 10 \cdot \log_{10} \frac{P_x}{P_\delta}, \tag{15}$$

where x^{adv}, x are the adversarial example and the geniue speech, respectively; δ is the generated perturbation noise. P_x and P_δ denote the energies of the intrinsic speech signal and the perturbation noise, respectively. In general, SNR compares the level of the intrinsic signal to level of perturbation noise. Clearly, a larger SNR value indicates a cleaner speech. In addition, the Perceptual Evaluation of Speech Quality (PESQ) is employed to assess the human perception quality of the generated adversarial example. This metric ranges from -0.5 (bad) to 4.5 (excellent).

B. ATTACKING TO THE NETWORK-BASED ASR

The attacking is assumed implemented under white-box settings, where an adversary can exploit the gradient of the target network for attacking. More specifically, considering the goal of this work is to implement a targeted attack, we train our proposed network model for each target on each target network.[3] For a given target label, one can feed the training data except the ones belonging to the target label into the generator to obtain perturbation. Then, the constructed adversarial examples are fed into the target network to compute the loss between the output and the designated target label. The network parameters of the generator is then updated with the gradient information that back-propagated from the loss. Finally, we can obtain a well-trained model to realize the targeted attack on a ASR network. That is, our trained model can be used to generate adversarial examples for a given speech clip, making it misclassfied as the specific target label with quite high confidence.

The attacking experiments are conducted on two speech recognition models WideResNet [29] and SampleCNN [30]. These two representative networks are selected as target due to their prevalent usage in practice. WideResNet is a ResNet-based speech classification neural network, whose goal is to classify the given speech into different classes. It is reported that the classification accuracy of WideResNet reaches 96.03% on the `SpeechCmd` dataset [29]. SampleCNN is a convolutional-based music genre classification neural network, aiming to classify the music audio clips into various genres. It can achieve 92.90% classification accuracy on the `GTZAN` dataset [30].

Figure 4 shows the attack success rate results for the WideResNet on `SpeechCmd` dataset, and Figure 5 shows the attack success rate results for SampleCNN on `GTZAN` dataset. The results are presented in the form of confusion matrix. Note that, the diagonal values of the confusion matrix are all deliberately set as zeros. This is because diagonal entry indicates the source equals target, which is a correct classification rather than an intended attack. As shown

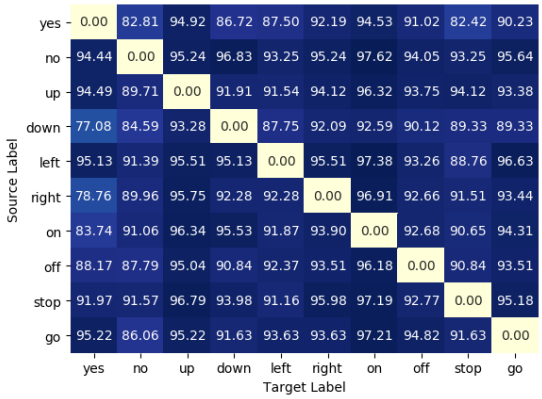


FIGURE 4. The confusion matrix of the attack success_rate for the WideResNet speech classification network [29] on the `SpeechCmd` dataset. Our targeted attack enforces a given speech that originally belongs to the source label misclassified as the target label. The diagonal entries are all set as zeros because they indicate the source equals target, which is a correct classification rather than an intended attack.

FIGURE 5. The confusion matrix of the attack success_rate for the SampleCNN speech classification network [30] on the `GTZAN` dataset. Our targeted attack enforces a given speech that originally belongs to the source label misclassified as the target label. The diagonal entries are all set as zeros because they indicate the source equals target, which is a correct classification rather than an intended attack.

[3]Note that one can also train a unified network to realize targeted attack on all targets under our proposed network framework.

TABLE 1. Comparison the proposed method with Alzantot *et al.* [9] and SirenAttack [10].

Attacking Method	Attacking WideResNet on SpeechCmd			Attacking SampleCNN on GTZAN		
	success_rate	SNR(dB)	Time(s)	success_rate	SNR(dB)	Time(s)
Alzantot *et al.* [9]	84.96%	15.72	231.46	82.76%	14.15	215.36
SirenAttack [10]	89.25%	17.57	368.29	89.10%	15.39	452.21
Proposed	**92.33%**	**20.27**	**0.009**	**90.58%**	**26.92**	**0.01**

in Figure 4, the attack success_rate of our method against the WideResNet network generally larger than 90% (the average attack success_rate is 92.33%). In other words, the average accuracy of the target classification network on generated adversarial examples is about 7.67%, which means that our generated adversarial examples could almost completely paralyze the WideResNet ASR system. Take the targeted attack "no" → "yes" as an example. Our generated adversarial example which sounds like "no" could be misclassified as the opposite "yes" by WideResNet ASR system of high chance 94.44%. This could causes high security risks for some critical ASR systems, e.g., the ASR equipped in the automobile. With more careful examination, one can notice that, on one hand, the attack success_rate's of the target "up", "right", "on" and "off" are all exceed 90% (the average success_rate's for "up", "right", "on" and "off" are 95.34%, 94.02%, 96.21% and 92.79%, respectively). This implies that these particular words are more prone to be as the targets when attacking the WideResNet ASR system. On the other hand, when the source words are "no", "left", and "stop", the average attack success_rate's are 95.06%, 94.30% and 94.07%, all exceeding 90%. This phenomenon suggests that these words would be much easier to be attacked, which may guide an adversary to exploit such *vulnerable* words when implementing an attack. The similar observations can also be made Figure 5, which represents the attack success_rate for SampleCNN on GTZAN dataset, where the maximum attack success_rate reaches 99.66%. In a short summary, the attack success_rate results verify that our method can effectively attack the target network-based ASR systems.

To better illustrate the results for attack success rates, we further present a boxplot to describe the error margin (note that, attack error rate = 1 – success_rate) on two ASR networks, i.e., WideResNet and SampleCNN. Please refer to Figure 6. As one can see, the average error rates for both networks are below 10%, with a few outliers exceeding 15%. More specifically, the margins of the attack error rate on WideResNet and SampleCNN are [2.84%, 22.92%] and [0.34%, 35.52%], respectively. However, the number of outliers on WideResNet and SampleCNN are 6 and 5, respectively, merely accounting for 6.67% (6/90) and 5.56% (5/90) of the entire testing cases. Furthermore, the average error rate for WideResNet is 7.67%, with standard deviation 3.9%; and on SampleCNN is 9.42%, with standard deviation 6.45%. This suggests that, despite there exist a few outliers, our proposed method can effectively paralyze the target ASR models for most cases.

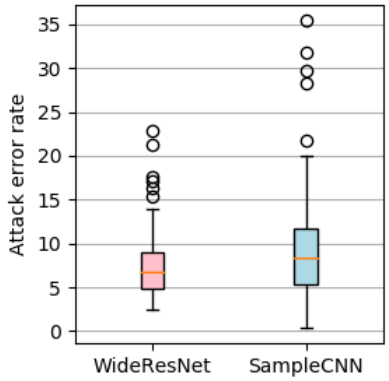
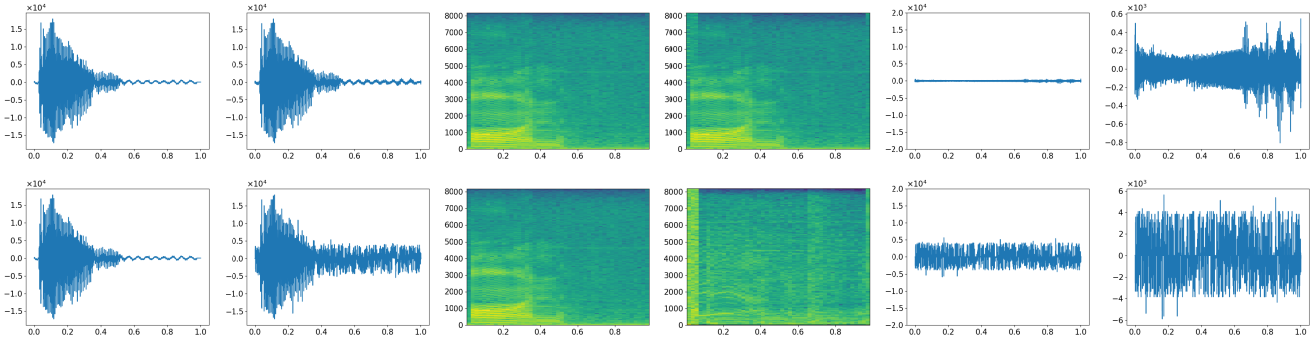


FIGURE 6. Attack error rates(%) of proposed method on WideResNet and SampleCNN.

Furthermore, we compare the proposed method with two recent state-of-the-art works, Alzantot *et al.* [9] and SirenAttack [10], under two same attacking settings: 1) attacking WideRestNet model on SpeechCmd dataset, and 2) attacking SampleCNN model on GTZAN dataset. The comparison results are listed in Table 1, from which one can observe that our method generally achieves superior performance. For instance, when attacking WideResNet on SpeechCmd, the average attack success rate for [9] is 84.96% with the average SNR 15.72dB, and the average time for generating an adversarial audio clip is 231.46 seconds; The average attack success rate for [10] is 89.25% with the average SNR 17.57dB, and the average time for generating an adversarial audio clip is 368.29 seconds. For comparison, our proposed method yields attack success rate 92.33% with the average SNR 20.27dB, and only takes 0.009 seconds for generating an adversarial example in average.

C. QUALITY EVALUATION OF THE GENERATED ADVERSARIAL SPEECH EXAMPLE

First, we compute the average SNR and PESQ values between the original sample with the generated adversarial example for each target. The results are listed in Table 2. Specifically, the average SNR for WideResNet on SpeechCmd is 20.27dB, and the average SNR for SampleCNN on GTZAN is 26.92dB. For reference, the average SNR values for WideResNet on SpeechCmd of the recent state-of-the-art attacking method [9] and [10] is 15.72 and 17.57dB respectively, which is inferior to ours. Not surprisingly, adding



(a) Original (b) Adversarial (c) Original (d) Adversarial (e) Residual (f) Enlarged residual

FIGURE 7. Comparison between the original speech sample with its generated adversarial example counterpart. The two rows show the results for attack "on" → "go" on WideResNet network using our method (top row) and Alzantot *et al.* (bottom row) [9], respectively. From left to right: (a) and (b) are the waveforms of the original and the adversarial sample, respectively; (c) and (d) are the spectrograms of the original and adversarial sample, respectively; (e) is the residual map between the original and adversarial sample, and (f) is 10× enlarged residual map.

TABLE 2. The SNR(dB)/PESQ of the generated adversarial examples.

ASR Network	Metric	SpeechCmd Dataset										
		yes	no	up	down	left	right	on	off	stop	go	Average
WideResNet	SNR(dB)	16.73	20.80	21.55	19.69	19.59	23.41	19.4	21.89	17.66	22.02	20.27
	PESQ	2.808	3.179	3.151	3.040	3.208	3.303	2.858	3.151	3.116	3.060	3.087
ASR Network	**Metric**	**GTZAN Dataset**										
		blues	classical	country	disco	hiphop	jazz	metal	pop	reggage	rock	Average
SampleCNN	SNR(dB)	26.91	24.67	27.21	27.36	26.44	27.32	26.74	29.79	24.12	28.66	26.92
	PESQ	3.840	3.731	3.984	4.062	3.985	3.689	3.792	4.230	3.896	4.262	3.947

perturbation to the original speech sample would cause certain degradation of the quality of adversarial speech sample. However, by examining the perception quality index PESQ, we can see that the average PESQ for WideResNet on `SpeechCmd` is 3.087, and the average PESQ for SampleCNN on `GTZAN` is 3.947, indicating an acceptable auditory perception experience for a common human listener. For more detailed listening comparison, the readers are suggested to check the exemplar audio clips provided in our project website *https://winterwindwang.github.io/SpeechAdvGan/*.

Moreover, as concrete examples, we visually compare the original speech and its counterpart adversarial example in Figure 7. As shown in Figure 7-(a)(b)(c) and (d), the waveforms and spectrograms of the original and adversarial example are almost identical. To better illustrate the difference, in Figure 7-(e), we compute and plot the residual (i.e., the perturbation) between original speech waveform and the one of adversarial example. It can be seen that the perturbation magnitude is quite small when comparing with the original audio sample. Furthermore, compared with Alzantot *et al.* [9] (bottom row), the magnitude of our produced perturbation is much smaller. With more detailed inspection on the 10× of the residual map Figure 7-(f), one can notice that, similar to [9], the adversarial noise will be added throughout entire speech clip. However, unlike the work [9] uniformly enforcing noise on the all sampling points, our method demonstrates certain preference at some sampling locations. One possible explanation for the behavior of our generated perturbation can be as follows. In our loss function (5) for generator, there are two regularization terms, i.e., L_{hinge} and L_2. The L_2 loss prefers the perturbation distribute uniformly across the entire speech with limited energy, while L_{hinge} loss favors finding some critical sampling points that could make the ASR systems misbehave. By trade-off those two losses, the sought adversarial noise would exhibit the above-mentioned phenomenon. We also would like to point that, this finding is consistent with the results that reported in previous work [22], in which the authors also suggested to add perturbation non-uniformly, in order to obtain an imperceptible adversarial example.

D. ABLATION STUDY

We have conducted an ablation study on the loss terms L_2 and hinge loss. As a concrete example, we implement the attack "yes" -> "right" under three different loss settings, i.e., L_2 loss, hinge loss and *L2&Hinge* loss. Figure 8 illustrates the perturbation of the generated adversarial examples. As one can observe, the perturbation yielded merely with L_2 loss is somewhat uniformly, almost all the sampling points have perturbation noise. In contrast, the perturbation yielded merely with hinge loss is relatively sparse, the perturbations of many sampling points are close to zero, while some sampling points have large-magnitude noise, and such points can be referred to critical points. By jointly employing L_2 and hinge loss, the perturbation trends to trade-off among the two objectives,

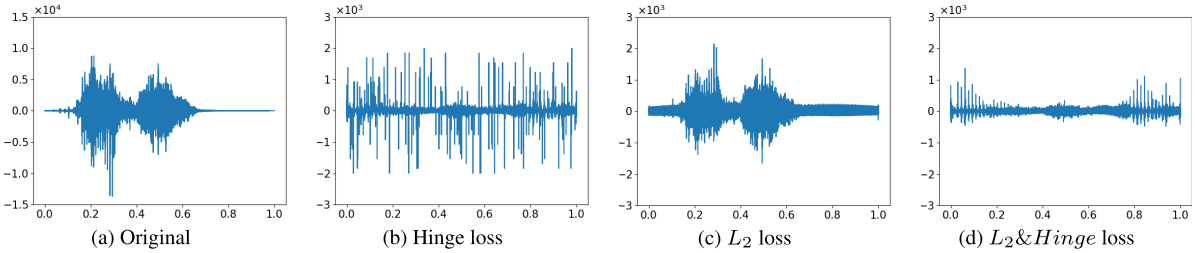


FIGURE 8. Comparison of the generated adversarial perturbation for the attack "yes" -> "right" with three different loss settings: Hinge loss, L_2 loss and L_2&*Hinge* loss. (a) The waveform of the original audio clip "yes"; (b) Generated perturbation merely using hinge loss; (c) Generated perturbation merely using L_2 loss; (d) Generated perturbation merely using both hinge loss and L_2 loss.

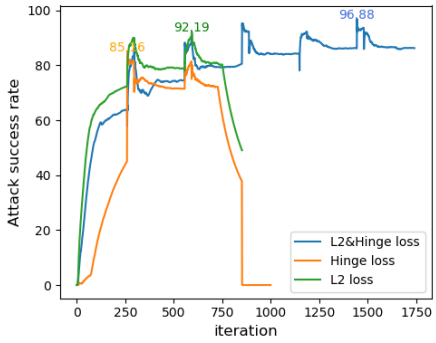


FIGURE 9. Comparison of generator loss and attack success rate for attack "yes" -> "right" with different loss. As shown, with "L_2&*Hinge* loss" can achieve higher attack success rate; while another two loss will lead generator to explode.

i.e., finding critical sampling points while generating limited perturbation across the entire speech.

In addition, we also would like to note that another important motivation of jointly using both L_2 and hinge loss is to stabilize the GAN training process. To see this, we recorded the training attack success rates for different loss settings for several hundred iterations. As can be seen from Figure 9, when merely using L_2 loss or hinge loss, the training could explode or vanish at early stage (less than 1000 iterations). The maximum success attack rates attained with L_2 loss and hinge loss are 92.19% and 85.16%, but the performance would significantly drop after 700 iterations. In contrast, when jointly using L_2 loss and hinge loss, the success attack rate could grow up for a large number of iterations (in our experiments, at least 2000 iterations), and the maximum success attack rate achieved is 96.88%, outperforming usage of L_2 loss or hinge loss alone.

E. RUNNING TIME

One merit of our proposed method is its inference efficiency. This is because our method is network-based. Once completing the network training, the testing (inference) is quite fast. In contrast, the work Alzantot *et al.* [9] employed genetic algorithm and SirenAttack [10] used particle swarm optimization, which are all computationally expensive evolutionary algorithms. Specifically, our method spends approximately 0.009 seconds (i.e., 9 milliseconds) to generate one adversarial example (Nvidia Titan X GPU) on attack WideResNet. For comparison, Alzantot *et al.* [9] needs 231.46 seconds (Nvidia Titan X GPU), and SirenAttack [10] reported 368.29 seconds shall be spent (Nvidia GeForce GTX 1080Ti GPU), to generate a single adversarial example.

VI. CONCLUSION

In this work, we make the first attempt to employ generative adversarial networks to generate speech adversarial examples, which could successfully paralyze network-based speech classification networks. Our proposed method implants the target network into a generative adversarial network framework. The generator utilizes the gradient information provided from the target network to generate perturbations that can deceive the target network. Such perturbations are directly added to the original speech waveform to form an adversarial example. Meanwhile, the discriminator constantly stimulates the generator to generate the limited perturbation, so that the adversarial examples it constitutes can simultaneously fool the discriminator and attain acceptable perception quality. Upon completing the training, the well-trained generator could efficiently generate adversarial samples for a specified target. Extensive experiments demonstrate the effectiveness and efficiency of our proposed method. As a future work, we would like to extend the proposed method to handle the adversarial generation problem under the setting that the speech is played over-the-air.

REFERENCES

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: http://arxiv.org/abs/1312.6199

[2] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[3] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," 2016, *arXiv:1605.05396*. [Online]. Available: http://arxiv.org/abs/1605.05396

[4] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1369–1378.

[5] Y. Dong, H. Su, B. Wu, Z. Li, W. Liu, T. Zhang, and J. Zhu, "Efficient decision-based black-box adversarial attacks on face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7714–7722.

[6] T. Vaidya, Y. Zhang, M. Sherr, and C. Shields, "Cocaine noodles: Exploiting the gap between human and machine speech recognition," in *Proc. 9th USENIX Workshop Offensive Technol. (WOOT)*, 2015, pp. 1–4.

[7] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. and Zhou, "Hidden voice commands," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 513–530.

[8] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on Speech-to-Text," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2018, pp. 1–7.

[9] M. Alzantot, B. Balaji, and M. Srivastava, "Did you hear that? Adversarial examples against automatic speech recognition," 2018, *arXiv:1801.00554*. [Online]. Available: http://arxiv.org/abs/1801.00554

[10] T. Du, S. Ji, J. Li, Q. Gu, T. Wang, and R. Beyah, "SirenAttack: Generating adversarial audio for End-to-End acoustic systems," 2019, *arXiv:1901.07846*. [Online]. Available: http://arxiv.org/abs/1901.07846

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[12] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*. [Online]. Available: http://arxiv.org/abs/1511.06434

[13] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1125–1134.

[14] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," 2016, *arXiv:1605.05396*. [Online]. Available: http://arxiv.org/abs/1605.05396

[15] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, *arXiv:1702.05983*. [Online]. Available: http://arxiv.org/abs/1702.05983

[16] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," 2018, *arXiv:1801.02610*. [Online]. Available: http://arxiv.org/abs/1801.02610

[17] S. Pascual, A. Bonafonte, and J. Serrá, "SEGAN: Speech enhancement generative adversarial network," 2017, *arXiv:1703.09452*. [Online]. Available: http://arxiv.org/abs/1703.09452

[18] S. Latif, R. Rana, and J. Qadir, "Adversarial machine learning and speech emotion recognition: Utilizing generative adversarial networks for robustness," 2018, *arXiv:1811.11402*. [Online]. Available: http://arxiv.org/abs/1811.11402

[19] B. Tang, Y. Tu, Z. Zhang, and Y. Lin, "Digital signal modulation classification with data augmentation using generative adversarial nets in cognitive radio networks," *IEEE Access*, vol. 6, pp. 15713–15722, 2018.

[20] M. Li, O. Li, G. Liu, and C. Zhang, "Generative adversarial networks-based semi-supervised automatic modulation recognition for cognitive radio networks," *Sensors*, vol. 18, no. 11, p. 3913, Nov. 2018.

[21] X. Li, D. Yan, L. Dong, and R. Wang, "Anti-forensics of audio source identification using generative adversarial network," *IEEE Access*, vol. 7, pp. 184332–184339, 2019.

[22] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," 2019, *arXiv:1903.10346*. [Online]. Available: http://arxiv.org/abs/1903.10346

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2015, pp. 234–241.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[25] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2794–2802.

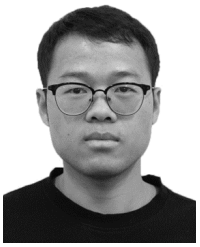
[26] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018, *arXiv:1804.03209*. [Online]. Available: http://arxiv.org/abs/1804.03209

[27] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293–302, Jul. 2002.

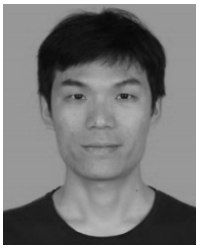
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[29] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*. [Online]. Available: http://arxiv.org/abs/1605.07146

[30] J. Lee, J. Park, K. Luke Kim, and J. Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," 2017, *arXiv:1703.01789*. [Online]. Available: http://arxiv.org/abs/1703.01789



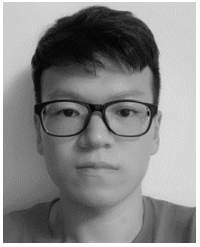
DONGHUA WANG (Student Member, IEEE) received the B.S. degree from Jiangxi Normal University, in 2018. He is currently pursuing the master's degree with the Faculty of Electrical Engineering and Computer Science, Ningbo University. His research interests include adversarial machine learning and information security.



LI DONG (Member, IEEE) received the B.Eng. degree from Chongqing University, in 2012, and the M.S. and Ph.D. degrees from the University of Macau, in 2014 and 2018, respectively. He is currently an Assistant Professor with the Department of Computer Science, Faculty of Electrical Engineering and Computer Science, Ningbo University. His research interests include statistical image modeling and processing, multimedia security and forensic, and machine learning.

RANGDING WANG (Member, IEEE) received the Ph.D. degree from Tongji University, in 2004. He is currently a Full Professor with Ningbo University, China. His research interests mainly include multimedia security, digital watermarking for digital rights management, data hiding, and steganography.

DIQUN YAN (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Ningbo University, in 2002, 2008, 2012, respectively. He was a Visiting Scholar with the New Jersey Institute of Technology, Newark, NJ, USA, from 2014 to 2015. He is currently an Associate Professor with the Faculty of Electrical Engineering and Computer Science, Ningbo University, China. He is also the Head of the Computer Science Department. His current research interests include speech processing and multimedia forensics.



JIE WANG (Student Member, IEEE) received the B.S. degree from the College of Science and Technology, Ningbo University, in 2018, where he is currently pursuing the master's degree with the Faculty of Electrical Engineering and Computer Science. His research interests include data hiding, steganography and information security.