Universal Adversarial Audio Perturbations

Sajjad Abdoli, Luiz G. Hafemann, Jérôme Rony, Ismail Ben Ayed, Patrick Cardinal and Alessandro L. Koerich

Abstract—We demonstrate the existence of universal adversarial perturbations, which can fool a family of audio classification architectures, for both targeted and untargeted attack scenarios. We propose two methods for finding such perturbations. The first method is based on an iterative, greedy approach that is well-known in computer vision: it aggregates small perturbations to the input so as to push it to the decision boundary. The second method, which is the main contribution of this work, is a novel penalty formulation, which finds targeted and untargeted universal adversarial perturbations. Differently from the greedy approach, the penalty method minimizes an appropriate objective function on a batch of samples. Therefore, it produces more successful attacks when the number of training samples is limited. Moreover, we provide a proof that the proposed penalty method theoretically converges to a solution that corresponds to universal adversarial perturbations. We also demonstrate that it is possible to provide successful attacks using the penalty method when only one sample from the target dataset is available for the attacker. Experimental results on attacking five 1D CNN architectures have shown attack success rates higher than 85.4% and 83.1% for targeted and untargeted attacks, respectively using the proposed penalty method.

Index Terms—adversarial perturbation, deep learning, audio processing, audio classification.

✦

1 INTRODUCTION

DEEP learning models have been achieving state-of-the-art performance in various problems, notably in image recognition [1], natural language processing [2], [3] and speech processing [4], [5]. However, recent studies have demonstrated that such deep models are vulnerable to adversarial attacks [6], [7], [8], [9], [10]. Adversarial examples are carefully perturbed input examples that can fool a machine learning model at test time [6], [11], posing security and reliability concerns for such models. The threat of such attacks has been mainly addressed for computer vision tasks [9], [12]. For instance, Moosavi-Dezfooli *et al.* [13] have shown the existence of *universal* adversarial perturbation, which, when added to an input image, causes the input to be misclassified with high probability. For these universal attacks, the generated vector is independent from the input examples.

End-to-end audio classification systems have been gaining more attention recently [14], [15], [16], [17]. In such systems, the input to the classifier is the audio waveform. Moreover, there have been some studies that embed traditional signal processing techniques into the layers of Convolutional Neural Network (CNN) [18], [19], [20]. For such audio classification systems, the effect of adversarial attacks is not widely addressed [21]. Creating attacks to threaten audio classification systems is challenging, due mainly to the signal variability in the time domain [21].

In this paper, we demonstrate the existence of universal adversarial perturbations, which can fool a family of audio classification architectures, for both targeted and untargeted attack scenarios. We propose two methods for finding such

perturbations. The first method is based on the greedy-approach principle proposed by Moosavi-Dezfooli *et al.* [13], which finds the minimum perturbation that sends examples to the decision boundary. The second method, which is the main contribution of this work, is a novel penalty formulation, which finds targeted and untargeted universal adversarial perturbations. Differently from the greedy approach, the penalty method minimizes an appropriate objective function on a batch of samples. Therefore, it produces more successful attacks than the previous method when the number of training samples is limited. We also show that using this method, it is possible to provide successful attacks when only one sample from the target dataset is available to the attacker. Moreover, we provide a proof that the proposed penalty method theoretically converges to a solution that corresponds to universal adversarial perturbations. Both methods are evaluated on a family of audio classifiers based on deep models for environmental sound classification. The experimental results have shown that both methods can attack the target models with a high success rate.

This paper is organized as follows: Section 2 presents an overview of adversarial attacks on deep learning models. Section 3 presents the proposed methods to craft universal audio adversarial perturbations. Section 4 presents the dataset and the target models used to evaluate the proposed methods, while the experimental results are presented in Section 4.1. The conclusion and perspective of future work is presented in the last section.

2 ADVERSARIAL MACHINE LEARNING

Research on adversarial attacks has attracted considerable attention recently, due to its impact on the reliability of shallow and deep learning models for computer vision tasks [8]. For a given example $\mathbf{x}$, an attack can find a small perturbation δ, often imperceptible to a human observer, so that an example $\tilde{\mathbf{x}} = \mathbf{x} + \delta$ is misclassified by a machine

• *Authors are with École de Technologie Supérieure (ÉTS), Université du Québec, Montreal, QC, Canada*
E-mail: sajjad.abdoli.1@ens.etsmtl.ca, luiz.gh@mailbox.org, jerome.rony.1 @etsmtl.net, {ismail.benayed, patrick.cardinal, alessandro.koerich} @etsmtl.ca

learning model [6], [22], [23]. The attacker's goal may cover a wide range of threats like *privacy* violation, *availability* violation and *integrity* violation [11]. Moreover, the attacker's goal may be *specific* (targeted), with inputs misclassified as a specific class or *generic* (untargeted), where the attacker simply wants to have an example classified differently from its true class [11]. Attacks generated by targeting a specific classifier are often *transferable* to other classifiers (i.e. also induce misclassification on them) [24], even if they have different architectures and do not use the same training set [6]. It has been also shown that such attacks can be applied in the physical world too. For instance, Kurakin *et al.* [25] showed that printed adversarial examples were still misclassified after being captured by a camera; Athalye *et al.* [26] presented a method to generate adversarial examples that are robust under translation and scale transformations, showing that such attacks induce misclassification even when the examples correspond to different viewpoints.

Research on adversarial attacks on audio classification systems is quite recent. Some studies focused mainly on providing inaudible and hidden targeted attacks on the systems. In such attacks, new audio examples are synthesized, instead of adding perturbations to actual inputs [27], [28]. Other works focus on untargeted attacks on speech and music classification systems [29], [30]. Du *et al.* [31] proposed a method based on particle swarm optimization for targeted and untargeted attacks. They evaluated their attacks on a range of applications such as speech command recognition, speaker recognition, sound event detection and music genre classification. Alzantot *et al.* [32] proposed a similar approach that uses a genetic algorithm for crafting targeted black-box attacks on a speech command classification system [33], which achieved 87% of success. Carlini *et al.* [21] proposed a targeted attack on the speech processing system DeepSpeech [14], which is a state-of-the-art speech-to-text transcription neural network. A penalty method is used in such attacks, which achieved 100% of success.

Most studies consider that the attacker can directly manipulate the input of the classifiers. Crafting audio attacks that can work on the physical world (i.e. played over-the-air) presents challenges such as being robust to background noise and reverberations. Yakura *et al.* [34] showed that over-the-air attacks are possible, at least for a dataset of short phrases. Qin *et al.* [35] also reported successful adversarial attacks on automatic speech recognition systems, which remain effective after applying realistic simulated environmental distortions.

Universal adversarial perturbations are effective type of attacks [13] because the additive noise is independent of the input examples, but once added to any example, it may fool a deep model to misclassify such an example. Moosavi-Dezfooli *et al.* [13] proposed a greedy algorithm to provide such perturbations for untargeted attacks to images. The perturbation is generated by aggregating atomic perturbation vectors, which send the data points to the decision boundary of the classifier. Recently, Moosavi-Dezfooli *et al.* [36] provided a formal relationship between the geometry of the decision boundary and robustness to universal perturbations. They have also shown the strong vulnerability of state-of-the-art deep models to universal perturbations. Metzen *et al.* [37] generalized this idea to pro-

vide attacks against semantic image segmentation models. Recently, Behjati *et al.* [38] generalized this idea to attack a text classifier in both targeted and non-targeted scenarios. In a different approach, Hayes *et al.* [39] proposed a generative model for providing such universal perturbations. Recently, Neekhara *et al.* [40] were also inspired by this idea to generate universal adversarial perturbations which cause mis-transcription of audio signals of automatic transcription systems. They reported success rates up to 88.24% on the hold-out set of the pre-trained Mozilla DeepSpeech model [14]. Nonetheless, their method is designed only for untargeted attacks. For audio classification systems, the impact of such perturbations can be very strong if these perturbations can be played over the air even without knowing what the test examples would look like.

3 UNIVERSAL ADVERSARIAL AUDIO PERTURBATIONS

In this section, we formalize the problem of crafting universal audio adversarial perturbations and propose two methods for finding such perturbations. The first method is based on the greedy-approach principle proposed by Moosavi-Dezfooli *et al.* [13] which finds the minimum perturbation that sends examples to the decision boundary of the classifier or inside the boundary of the target class for untargeted and targeted perturbations, respectively. The second method, which is the main contribution of this work, is a penalty formulation, which finds a universal perturbation vector that minimizes an objective function.

Let μ be the distribution of audio samples in $\mathbb{R}^d$ and $\hat{k}(\mathbf{x}) = \arg\max_y \mathbb{P}(y|\mathbf{x}, \theta)$ be a classifier that predicts the class of the audio sample $\mathbf{x}$, where y is the predicted label of $\mathbf{x}$ and θ denotes the parameters of the classifier. Our goal is to find a vector $\mathbf{v}$ that, once added to the audio samples can fool the classifier for most of the samples. This vector is called universal as it is a fixed perturbation that is independent of the audio samples and, therefore, it can be added to any sample in order to fool a classifier. The problem can be defined such that $\hat{k}(\mathbf{x} + \mathbf{v}) \neq \hat{k}(\mathbf{x})$ for a untargeted attack and, for a targeted attack, $\hat{k}(\mathbf{x} + \mathbf{v}) = y_t$, where y_t denotes the target class. In this context, the universal perturbation is a vector with a sufficiently small l_p norm, where $p \in [1, \infty)$, which satisfies two constraints [13]: $\|\mathbf{v}\|_p \leq \xi$ and $\mathbb{P}_{\mathbf{x} \sim \mu}(\hat{k}(\mathbf{x} + \mathbf{v}) \neq \hat{k}(\mathbf{x})) \geq 1 - \delta$, where ξ controls the magnitude of the perturbation and δ controls the desired fooling rate. For a targeted attack, the second constraint is defined as $\mathbb{P}_{\mathbf{x} \sim \mu}(\hat{k}(\mathbf{x} + \mathbf{v}) = y_t) \geq 1 - \delta$.

3.1 Iterative Greedy Algorithm

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be a set of m audio files sampled from the distribution μ. The greedy algorithm proposed by Moosavi-Dezfooli *et al.* [13] gradually crafts adversarial perturbations in an iterative manner. For untargeted attacks, at each iteration, the algorithm finds the minimal perturbation $\Delta \mathbf{v}_i$ that pushes an example $\mathbf{x}_i$ to the decision boundary, and adds the current perturbation to the universal perturbation. In this study, a targeted version of the algorithm is also proposed such that the universal perturbation added to the example must push it toward the decision boundary

of the target class. In more details, at each iteration of the algorithm, if the universal perturbation makes the model misclassify the example, the algorithm ignores it, otherwise, an extra $\Delta \mathbf{v}_i$ is found and aggregated to the universal perturbation by solving the minimization problem with the following constraints for untargeted and targeted attacks respectively:

$$\Delta \mathbf{v}_i \leftarrow \arg \min_{\mathbf{r}} \|\mathbf{r}\|_2$$
$$\text{s.t.} \quad \hat{k}\left(\mathbf{x}_i + \mathbf{v} + \mathbf{r}\right) \neq \hat{k}\left(\mathbf{x}_i\right), \qquad (1)$$
$$\text{or} \quad \hat{k}\left(\mathbf{x}_i + \mathbf{v} + \mathbf{r}\right) = y_t.$$

In order to find $\Delta \mathbf{v}_i$ for each sample of the dataset, any attack that provides perturbation that misclassifies the sample, such as Carlini and Wanger L_2 attack [8] or Decoupled Direction and Norm (DDN) attack [41], can be used. Moosavi-Dezfooli *et al.* [42] used Deepfool to find such a vector.

In order to satisfy the first constraint ($\|\mathbf{v}\|_p \leq \xi$), the universal perturbation is projected on the l_p ball of radius ξ and centered at 0. The projection function $\mathcal{P}_{p,\xi}$ is formulated as:

$$\mathcal{P}_{p,\xi}(\mathbf{v}) = \arg \min_{\mathbf{v}'} \|\mathbf{v} - \mathbf{v}'\|_2 \quad \text{s.t.} \quad \|\mathbf{v}'\|_p \leq \xi. \qquad (2)$$

The termination criteria for the algorithm is defined such that the Attack Success Rate (ASR) on the perturbed training set exceeds a threshold $1 - \delta$. In this protocol, the algorithm stops for untargeted perturbations when:

$$\text{ASR}\left(X, \mathbf{v}\right) := \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\{\hat{k}\left(\mathbf{x}_i + \mathbf{v}\right) \neq \hat{k}\left(\mathbf{x}_i\right)\} \geq 1 - \delta, \quad (3)$$

where $\mathbf{1}\{\cdot\}$ is the true-or-false indicator function. For a targeted attack, we replace inequality $\hat{k}\left(\mathbf{x}_i + \mathbf{v}\right) \neq \hat{k}(\mathbf{x}_i)$ by $\hat{k}\left(\mathbf{x}_i + \mathbf{v}\right) = y_t$ in Eq. (3). The problem with iterative Greedy formulation is that the constraint is only defined on universal perturbation ($\|\mathbf{v}\|_p \leq \xi$). Therefore, the summation of the universal perturbation with the data points results in an audio signal that is out of a specific range such as $[0, 1]$, for almost all audio samples, even by selecting a small value for ξ. In order to solve this problem, we may clip the value of each resulting data point to a valid range.

3.2 Penalty Method

The proposed penalty method minimizes an appropriate objective function on a batch of samples from a dataset for finding universal adversarial perturbations. In the case of noise perception in audio systems, the level of noise perception can be measured using a realistic metric such as the sound pressure level (SPL). Therefore, the SPL is used instead of the L_p norm. In this paper, such a measure is used in one of the objective functions of the optimization problem, where one of the goals is to minimize the SPL of the perturbation, which is measured in decibel (dB) [21]. The problem of crafting a perturbation in a targeted attack can be reformulated as the following constrained optimization problem:

$$\text{minimize} \quad \text{SPL}(\mathbf{v})$$
$$\text{s.t.} \quad y_t = \arg \max_{y} \mathbb{P}\left(y|\mathbf{x}_i + \mathbf{v}, \theta\right) \qquad (4)$$
$$\text{and} \quad 0 \leq \mathbf{x}_i + \mathbf{v} \leq 1 \quad \forall i$$

For untargeted attacks we use $y_l \neq \arg \max_y \mathbb{P}\left(y|\mathbf{x}_i + \mathbf{v}, \theta\right)$, where y_l is the legitimate class.

Different from the iterative Greedy formulation, in the proposed penalty method the second constraint is defined on the summation of the data points and universal perturbation $(\mathbf{x}_i + \mathbf{v})$ to keep the perturbed example in a valid range. As the constraint of the DDN attack, which is used in the iterative greedy algorithm, is that the data must be in range $[0, 1]$, for a fair comparison between methods, we impose the same constraint on the penalty method. This box constraint should be valid for all audio samples.

In Eq. (4), the power of an audio waveform (noise) can be computed as:

$$\text{SPL}(\mathbf{v}) = 20 \log_{10} P(\mathbf{v}), \qquad (5)$$

where $P(\mathbf{v})$ is the power of the perturbation signal of length N, which is given by:

$$P(\mathbf{v}) = \sqrt{\frac{1}{N} \sum_{n=1}^{N} v_n^2}, \qquad (6)$$

where v_n denotes the n-th component of the array $\mathbf{v}$. We will show later that a gradient-based method is used for solving the optimization problem introduced in Eq. (4). As the gradient-based algorithm does not enforce the box constraint, we need to introduce a new parameter $\mathbf{w}$, which is defined in Eq. (7) to ensure that the box constraint is satisfied. This variable change is inspired by the L_2 attack of Carlini and Wagner [8].

$$\mathbf{w}_i = \frac{1}{2}(\tanh(\mathbf{x}'_i + \mathbf{v}) + 1), \qquad (7)$$

where $\mathbf{x}'_i$ is the transformed audio example, $\mathbf{x}_i$, to $\tanh$ space[1]. Since $-1 \leq \tanh(\mathbf{x}'_i + \mathbf{v}) \leq 1$ then $0 \leq \mathbf{w}_i \leq 1$ and the solution will be valid according to the box constraint.

In order to solve the optimization problem defined in Eq. (4), we rewrite variable $\mathbf{v}$ as follows:

$$\mathbf{v} = \frac{1}{2} \ln \left(\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}\right) - \mathbf{x}'_i. \qquad (8)$$

The details of expressing $\mathbf{v}$ as a function of $\mathbf{w}_i$ are shown in Appendix A. Therefore, we propose a penalty method that optimizes the following objective function:

$$\min_{\mathbf{w}_i} \left\{ \begin{array}{l} L(\mathbf{w}_i, t) = \text{SPL}\left(\frac{1}{2} \ln \left(\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}\right) - \mathbf{x}'_i\right) + c.G(\mathbf{w}_i, t), \\ G(\mathbf{w}_i, t) = \max\{\max_{j \neq t} \{f(\mathbf{w}_i)_j\} - f(\mathbf{w}_i)_t, -\kappa\} \end{array} \right\} \qquad (9)$$

where t is the target class, $f(\mathbf{w}_i)_j$ is the output of the pre-softmax layer (logit) of a neural network for class j, c is a positive constant known as "*penalty coefficient*" and κ controls the confidence level of sample misclassification. This formulation enables the attacker to control the confidence

---

1. In order to use Eq. (7), the audio example $\mathbf{x}_i$ must be transformed to $\tanh$ space and then Eq. (7) can be used to transform the perturbed data to the valid range of $[0, 1]$. The audio samples can be transformed to $\tanh$ space by $\mathbf{x}'_i = \text{arctanh}\left((2\mathbf{x}_i - 1) * (1 - \epsilon)\right)$, where ϵ is a small constant that depends on the extreme values of the transformed signal that ensures that $\mathbf{x}'_i$ does not assume infinity values. For our dataset, $\epsilon = 1e-7$ is a suitable value.

level of the attack. For untargeted attacks, we modify the objective function of Eq. (9) as:

$$
\min_{\mathbf{w}_i}
\begin{cases}
L(\mathbf{w}_i, y_l) = \text{SPL}\left(\frac{1}{2}\ln\left(\frac{\mathbf{w}_i}{1-\mathbf{w}_i}\right) - \mathbf{x}'_i\right) + c.G(\mathbf{w}_i, y_l), \\
G(\mathbf{w}_i, y_l) = \max\{f(\mathbf{w}_i)_{y_l} - \max_{j \neq y_l}\{f(\mathbf{w}_i)_j\}, -\kappa\}
\end{cases}
$$
(10)

where y_l is the legitimate label for the i-th sample of the batch. $G(\mathbf{w}_i, t)$ is the hinge loss penalty function, which for a targeted attack and $\kappa = 0$, must satisfy:

$$
\begin{aligned}
G(\mathbf{w}_i, t) = 0 \quad &\text{if} \quad y_t = \arg\max_y \mathbb{P}(y|\mathbf{w}_i, \theta), \\
G(\mathbf{w}_i, t) > 0 \quad &\text{if} \quad y_t \neq \arg\max_y \mathbb{P}(y|\mathbf{w}_i, \theta),
\end{aligned}
$$
(11)

The same properties of the penalty function are also valid for untargeted perturbations. This penalty function is convex and has subgradients therefore, a gradient-based optimization algorithm, such as the Adam algorithm [43] can be used to minimize Eqs. (9) and (10). Several other optimization algorithms like AdaGrad [44], standard gradient descent, gradient descent with Nesterov momentum [45] and RMSProp [46] have also been evaluated but Adam converges in fewer iterations and it produces relatively similar solutions.

Algorithm 1 presents the pseudo-code of the proposed penalty method. The algorithm minimizes the objective function introduced in Eq. (9) or Eq. (10) using a set of training samples and the Adam optimizer [43], [46].

Theorem 1: Let $\{\mathbf{v}^k\}$, $k = 1, ..., \infty$ be the sequence generated by the proposed penalty method in Algorithm 1 for k iterations. Let $\bar{\mathbf{v}}$ be the limit point of $\{\mathbf{v}^k\}$. Then any limit point of the sequence is a solution to the original optimization problem defined in Eq. (4)[2].

Proof: According to Eq. (8), $\mathbf{v}^k$ can be defined as:

$$
\mathbf{v}^k = \frac{1}{2}\ln\left(\frac{\mathbf{w}_i^k}{1-\mathbf{w}_i^k}\right) - \mathbf{x}'_i.
$$
(12)

Before proving Theorem 1, a useful Lemma is also presented and proved.

Lemma 1: Let $\mathbf{v}^*$ be the optimal value of the original constrained problem defined in Eq. (4). Then $\text{SPL}(\mathbf{v}^*) \geq L(\mathbf{w}_i^k, t) \geq \text{SPL}(\mathbf{v}^k) \, \forall k$.

Proof of Lemma 1:

$$
\begin{aligned}
\text{SPL}(\mathbf{v}^*) \;&= \text{SPL}(\mathbf{v}^*) + c.G(\mathbf{w}_i^*, t) \quad (\because G(\mathbf{w}_i^*, t) = 0) \\
&\geq \text{SPL}(\mathbf{v}^k) + c.G(\mathbf{w}_i^k, t) \quad (\because c > 0, G(\mathbf{w}_i^k, t) \geq 0, \\
&\qquad\qquad\qquad\qquad\qquad \mathbf{w}_i^k \text{ minimizes } L(\mathbf{w}_i^k, t)) \\
&\geq \text{SPL}(\mathbf{v}^k) \\
\therefore \text{SPL}(\mathbf{v}^*) &\geq L(\mathbf{w}_i^k, t) \geq \text{SPL}(\mathbf{v}^k) \, \forall k.
\end{aligned}
$$

Proof of Theorem 1. SPL is a monotonically increasing function and continuous. Also, G is a hinge function, which is continuous. L is the summation of two continuous functions. Therefore, it is also a continuous function. The limit point of $\{\mathbf{v}^k\}$ is defined as: $\bar{\mathbf{v}} = \lim_{k\to\infty}\mathbf{v}^k$ and since SPL is

2. Theorem 1 applies to the context of convex optimization. The neural network is defined as a functional constraint on the optimization problem defined in Eq. (4). Since neural networks are not convex, a feasible solution to the optimization problem may not be unique and is not guaranteed to be a global optimum.

a continuous function, $\text{SPL}(\bar{\mathbf{v}}) = \lim_{k\to\infty}\text{SPL}(\mathbf{v}^k)$. We can conclude that:

$$
\begin{aligned}
L^* &= \lim_{k\to\infty} L(\mathbf{w}_i^k, t) \leq \text{SPL}(\mathbf{v}^*) \quad (\because \text{Lemma 1}) \\
L^* &= \lim_{k\to\infty} \text{SPL}(\mathbf{v}^k) + \lim_{k\to\infty} c.G(\mathbf{w}_i^k, t) \leq \text{SPL}(\mathbf{v}^*) \\
L^* &= \text{SPL}(\bar{\mathbf{v}}) + \lim_{k\to\infty} c.G(\mathbf{w}_i^k, t) \leq \text{SPL}(\mathbf{v}^*).
\end{aligned}
$$

If $\mathbf{v}^k$ is a feasible point for the constrained optimization problem defined in Eq. (4), then, from the definition of function $G(.)$, one can conclude that $\lim_{k\to\infty} c.G(\mathbf{w}_i^k, t) = 0$. Then:

$$
L^* = \text{SPL}(\bar{\mathbf{v}}) \leq \text{SPL}(\mathbf{v}^*)
$$

$\boxed{\therefore \bar{\mathbf{v}} \text{ is a solution of the problem defined in Eq. (4)}}$

Algorithm 1: Penalty method for universal adversarial audio perturbations.

Input: Data points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ with corresponding legitimate labels Y, desired fooling rate on perturbed samples δ, and target class t (for targeted attacks)

Output: Universal perturbation signal $\mathbf{v}$

1 **initialize** $\mathbf{v} \leftarrow 0$,
2 **while** $\text{ASR}(X, \mathbf{v}) \leq 1 - \delta$ **do**
3   Sample a mini-batch of size S from (X, Y)
4   $\mathbf{g} \leftarrow 0$
5   **for** $i \leftarrow 1 \, to \, S$ **do**
6   Transform the audio signal to $\tanh$ space:
   $\mathbf{x}'_i = \text{arctanh}((2\mathbf{x}_i - 1) * (1 - \epsilon))$,
7   Compute the transformation of the perturbed signal for each sample i from mini-batch:
   $\mathbf{w}_i = \frac{1}{2}(\tanh(\mathbf{x}'_i + \mathbf{v}) + 1)$,
   Compute the gradient of the objective function, i.e., Eq. (9) or Eq. (10), w.r.t. $\mathbf{w}_i$:
8   **if** *targeted attack:* **then**
9     $\mathbf{g} \leftarrow \mathbf{g} + \frac{\partial L(\mathbf{w}_i, t)}{\partial \mathbf{w}_i}$
   else
10     $\mathbf{g} \leftarrow \mathbf{g} + \frac{\partial L(\mathbf{w}_i, y_i)}{\partial \mathbf{w}_i}$
11   Compute update $\Delta\mathbf{v}$ using $\mathbf{g}$ according to Adam update rule [43]
12   apply update: $\mathbf{v} \leftarrow \mathbf{v} + \Delta\mathbf{v}$
return $\mathbf{v}$

---

4 EXPERIMENTAL PROTOCOL AND RESULTS

For this study, we have used the full audio recordings of UrbanSound8k dataset [47] downsampled to 16 kHz for training and evaluating the models as well as for generating adversarial perturbations. This dataset consists of 7.3 hours of audio recordings split into 8,732 audio clips of up to slightly more than three seconds. Therefore, each audio sample is represented by a 50,999-dimensional array. The dataset was split into training (80%), validation (10%) and test (10%) set. For generating perturbations, 1,000 samples of the training set were randomly selected, and for penalty-based method a mini-batch size of 100 samples is used. The perturbations were evaluated on the whole test set (874 samples).
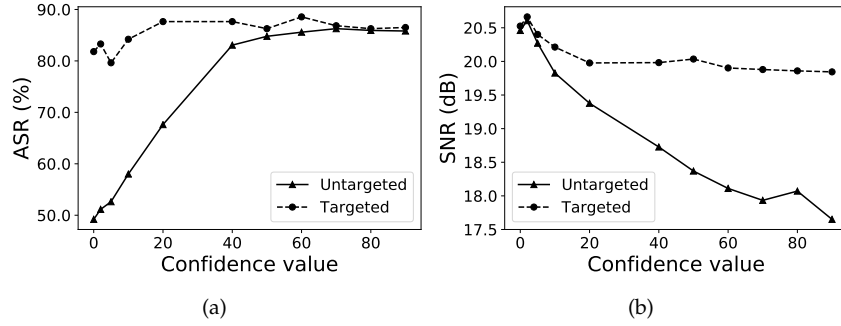
Fig. 1. Effect of different confidence values on (a) mean ASR and (b) mean SNR for targeted and untargeted attacks. ENVnet-V2 [48] is used as the target model.

Signal to Noise Ratio (SNR) is used as a metric to measure the level of noise with respect to the original signal. This metric, which is also measured in dB, is used for measuring the level of the perturbation of the signal after adding the universal perturbation. This measure is also used in previous works for evaluating the quality of the generated adversarial audio attacks [30], [31], and it is defined as:

$$\text{SNR}(\mathbf{x}, \mathbf{v}) = 20 \log_{10} \frac{P(\mathbf{x})}{P(\mathbf{v})}, \tag{13}$$

where $P(.)$ is the power of the signal defined in Eq. (6). A high SNR indicates that a low level of noise is added to the audio sample by the universal adversarial perturbation.

We have chosen a family of diverse end-to-end architectures as our target models. This selection is based on choosing architectures which might learn representations directly from the audio signal. We briefly describe the architecture of each model as follows. A detailed description of the architectures can be found in Appendix C.

1D CNN Rand [49]: This model consists of five one-dimensional convolutional layers (CL). The output of CLs is used as input to two fully connected (FC) layers followed by an output layer with softmax activation function. The weights of all of the layers are initialized randomly. This model was proposed by Abdoli *et al.* [49] for environmental sound classification.

1D CNN Gamma [49]: This model is similar to 1D CNN Rand except that it employs a Gammatone filter-bank in its first layer. Furthermore, this layer is kept frozen during the training process. Gammatone filters are used to decompose the input signal to appropriate frequency bands.

ENVnet-V2 [48]: The architecture proposed by Tokozume *et al.* [48] for sound recognition was slightly modified to make it compatible with the input size of the downsampled audio samples of UrbanSound8k dataset. This architecture uses the raw audio signal as input, and it extracts short-time frequency features by using two one-dimensional CLs followed by a pooling layer (PL). It then swaps axes and convolves in time and in frequency domain the features using five two-dimensional CLs. Two FC layers and an output layer with softmax activation function complete the network.

SincNet [18]: The end-to-end architecture proposed by Ravanelli and Bengio [18] for sound processing extracts meaningful features from the audio signal at its first layer.

In this model, several sinc functions are used as band-pass filters and only low and high cutoff frequencies are learned from audio. After that, two one-dimensional CLs are applied. Two FC layers followed by an output layer with softmax activation are used for classification.

SincNet+VGG19 [18]: This model uses sinc filters to extract features from the raw audio signal as in SincNet [18]. After an one-dimensional maxpooling layer, the output is stacked along time axis to form a 2D representation. This time-frequency representation is used as the input to a VGG19 network [50] followed by a FC layer and an output layer with softmax activation for classification. This time-frequency representation resembles a spectrogram representation of the audio signal.

For the iterative method, several parameters must be chosen. In order to find the minimal perturbation $\Delta \mathbf{v}_i$, we used the DDN L_2 attack [41]. This attack is designed to efficiently find small perturbations that fool the model. The difference with DeepFool [42] is that it can be used for both untargeted and targeted attacks, extending the iterative method to the targeted scenario. DDN was used with a budget of 50 steps and an initial norm of 0.2. Results are reported for $p = \infty$ and we set ξ to 0.2 and 0.12 for untargeted and targeted attack scenarios, respectively. These values were chosen to craft perturbations in which the norm is much lower than the norm of the audio samples in the dataset.

For evaluating the penalty method, we set the penalty coefficient c to 0.2 and 0.15 for untargeted and targeted attack scenarios, respectively. The confidence value κ is also set to 40 and 10 for crafting untargeted and targeted perturbations, respectively. Fig. 1 shows the effect of different confidence values on both mean ASR and mean SNR for targeted and untargeted attacks on the ENVnet-V2 model [48] for the test set. For this experiment, 1,000 and 500 training samples for untargeted and targeted attack scenarios are used, respectively. For targeted attacks, the target class is *"Gun shot"*. Fig. 1 shows that the ASR increases as confidence value increases. However, the SNR also decreases in the same way. For both iterative and penalty methods, we set the desired fooling rate on perturbed training samples to $\delta = 0.1$. Both algorithms terminate execution whether they achieve the desired fooling ratio, or they reach 100 iterations. Both algorithms have been trained and tested using a TITAN Xp GPU.

4.1 Results

Table 1 shows the results of the iterative and penalty methods against the five models considered in this study. We evaluate both targeted and untargeted attacks in terms of mean ASR on the training and test sets, as well as mean SNR of the perturbed samples of the test set. For both untargeted and targeted attacks, the penalty method produces the highest ASR for all target models. Both methods produce relatively similar mean SNR on the test set. The crafted universal perturbations produced from the test set by the penalty method generalize better than those produced by the iterative method for both attacking scenarios. We also observe a relatively low difference in ASR between the training and test sets. The detailed results of the targeted attack scenario for each model is reported in Appendix B. For a better assessment of the perturbations produced by both proposed methods, several randomly chosen examples of perturbed audio samples are presented in Appendix D.

We now consider the influence of the number of training data points on the quality of the universal perturbations. Fig. 2 shows the ASR and mean SNR achieved on the test set with different number of data points, considering two target models. The untargeted attack is evaluated on SincNet and the targeted attack is evaluated on 1D CNN Gamma model. For targeted attacks, the target class is *"Gun shot"*. For both targeted and untargeted scenarios, penalty method produces better ASR when the perturbations are crafted with a lower number of data points. For the untargeted scenario, iterative method produces perturbations with a slightly better mean SNR than those produced by the penalty method. However, this difference is perceptually negligible. For the targeted attack scenario, when the data points are limited, e.g. lower than 100 samples, the iterative method also produces perturbations with a slightly better mean SNR. However when the number of data points increases, the penalty method produces better perturbations in terms of SNR than those produced by the iterative method. The Greedy algorithm used in the iterative method is designed to generate perturbations with the least possible power level. At each iteration, if the perturbation vector misclassifies the example, it will be ignored for the next iterations. Therefore, it is impossible for the attacker to obtain higher ASRs at the expense of having a universal perturbation with slightly higher sound power level, especially when the number of audio samples is limited. However in the penalty method, the algorithm is able to generate more successful universal perturbations at the expense of having a universal perturbation with a negligible higher power level as long as the gradient-based algorithm can minimize the objective functions defined in Eqs. 9 and 10. Moreover, for all iterations of the penalty method, the algorithm exploits all available data to minimize the objective function for generating the universal perturbation.

Another advantage of the proposed penalty method is that it is also able to generate perturbations when a single audio example is available to the attacker. For such an aim, a single audio sample of each class is randomly selected from the training set, and the objective functions defined in Eqs. (9) and (10) are minimized iteratively using Algorithm 1 for targeted and untargeted attacks, respectively. For this experiment, we set the penalty coefficient $c = 0.2$ and the confidence value $\kappa = 90$ for both untargeted and targeted attack scenarios. The algorithm is executed for 86 iterations and the perturbation produced is used to perturb all the audio samples of the test set, which are then used to fool the 1D CNN Gamma model.

Table 2 shows the results of the untargeted attack in terms of ASR and SNR on the perturbed samples of the test set. Fig. 3 shows the results of the targeted attack in terms of SNR and ASR, considering all classes. In this case, the perturbation is also generated from a single randomly selected audio sample of each class in order to attack the model for a specific target class. Mean ASR of 0.708 and 0.826 are achieved for untargeted and targeted attack scenarios, respectively. Moreover, mean SNR of 16.85 dB and 16.25 dB are also achieved for untargeted and targeted attack scenarios, respectively. As it is shown in Table 2 and Fig. 3, it is possible to produce the most successful attacks in terms of mean ASR using samples from *"Children playing"* class and *"Engine idling"* class for untargeted and targeted attack scenarios, respectively. Moreover, Fig. 3 also shows that the 1D CNN Gamma model is more vulnerable to targeted attacks when the target class is *"Gun shot"*. Similar results were also obtained for all other target models.

5 CONCLUSION

In this study, we proposed an iterative method and a penalty method for generating targeted and untargeted universal adversarial audio perturbations. Both methods were used for attacking a diverse family of end-to-end audio classifiers based on deep models and the experimental results have shown that both methods can easily fool all models with a high success rate. It is also proved that the proposed penalty method converges to an optimal solution.

Although the perturbations produced by both methods can target the models with high success rate, in listening tests humans can detect the additive noise but the adversarial audios can be recognized as their original labels. Developing effective universal adversarial audio examples using the principle of auditory masking [35] to reduce the level of noise introduced by the adversarial perturbation is our current goal.

Different from image-based universal adversarial perturbations [51], the universal audio perturbations crafted in this study do not perform well in the physical world (played over-the-air). Combining the methods proposed in this paper with recent studies on generating robust adversarial attacks on several transformations on the audio signal [34], [35] may be a promising way to craft robust physical attacks against audio systems. Moreover, conducting an extensive study of the transferability of the universal perturbations over proposed models is also an important direction for future work. The transferability of perturbations may be examined for targeted and untargeted scenarios [52].

Finally, proposing a defensive mechanism against such universal perturbations is also an important issue. Using

TABLE 1
Mean ASR and mean SNR on the training and test sets for untargeted and targeted perturbations.

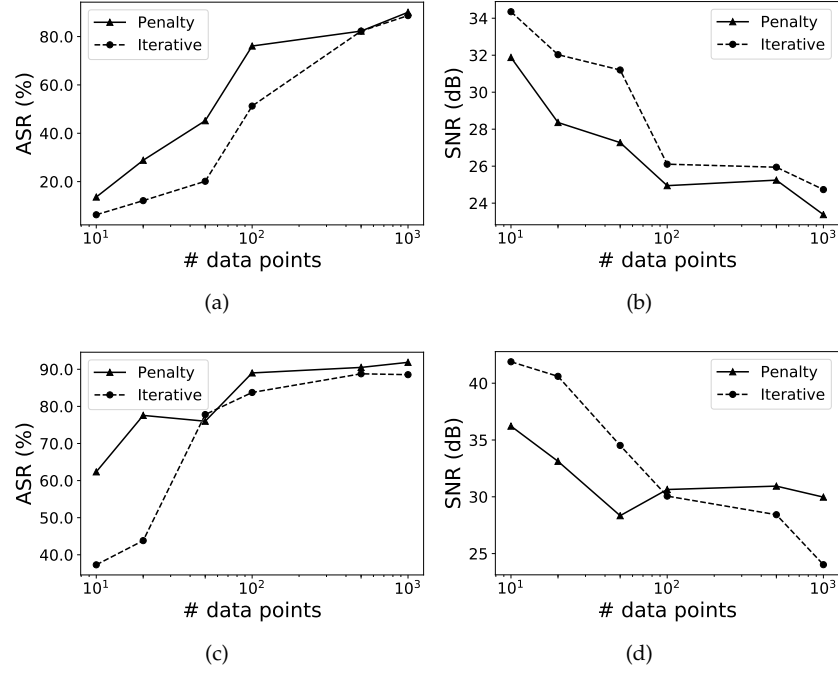| Method	Model	Targeted Attack			Untargeted Attack		
		Training Set	Test Set		Training Set	Test Set	
		ASR	ASR	SNR (dB)	ASR	ASR	SNR (dB)
Iterative	1D CNN Rand	0.926	0.672	25.321	0.911	0.412	25.244
	1D CNN Gamma	0.945	0.795	23.587	0.904	0.737	22.922
	ENVnet-V2	0.916	0.767	22.734	0.910	0.669	24.960
	SincNet	1.000	0.899	28.668	0.915	0.886	24.025
	SincNet+VGG19	0.985	0.872	26.203	0.924	0.838	26.362
Penalty	1D CNN Rand	0.917	**0.854**	23.468	0.900	**0.876**	20.350
	1D CNN Gamma	0.913	**0.888**	22.835	0.901	**0.858**	20.551
	ENVnet-V2	0.922	**0.877**	21.832	0.900	**0.831**	18.727
	SincNet	0.962	**0.971**	30.411	0.900	**0.919**	29.972
	SincNet+VGG19	0.916	**0.898**	26.736	0.902	**0.865**	23.555

Fig. 2. Targeted and untargeted ASR and mean SNR on the test set versus the number of training data points. Targeted attack: (a) ASR on 1D CNN Gamma model; (b) mean SNR on inputs of 1D CNN Gamma model. Untargeted attack: (c) ASR on SincNet; (d) mean SNR on inputs of SincNet.

methods like adversarial training [53] against such attacks is also considered as one of our future research directions.

APPENDIX A
SOLVING $\mathbf{w}_i$ FOR $\mathbf{v}$

$$\mathbf{w}_i = \frac{1}{2}\big(\tanh(\mathbf{x}_i' + \mathbf{v}) + 1\big)$$

Substituting $\mathbf{x}_i' + \mathbf{v}$ by $\mathbf{b}_i$:

$$\mathbf{w}_i = \frac{1}{2}\big(\tanh(\mathbf{b}_i) + 1\big)$$

By using the definition of $\tanh$ function:

$$\tanh(\mathbf{b}_i) = \frac{-e^{-\mathbf{b}_i} + e^{\mathbf{b}_i}}{e^{-\mathbf{b}_i} + e^{\mathbf{b}_i}},$$

we have:

$$2\mathbf{w}_i - 1 = \frac{-e^{-\mathbf{b}_i} + e^{\mathbf{b}_i}}{e^{-\mathbf{b}_i} + e^{\mathbf{b}_i}}.$$

Substituting $e^{\mathbf{b}_i}$ by $\mathbf{u}_i$:

$$2\mathbf{w}_i - 1 = \frac{-\mathbf{u}_i^{-1} + \mathbf{u}_i}{\mathbf{u}_i^{-1} + \mathbf{u}_i}$$

$$\Rightarrow 2\mathbf{w}_i - 1 = \frac{-1 + \mathbf{u}_i^2}{1 + \mathbf{u}_i^2}$$

$$\Rightarrow 2\mathbf{w}_i\left(1 + \mathbf{u}_i^2\right) - \left(1 + \mathbf{u}_i^2\right) = -1 + \mathbf{u}_i^2$$

$$\Rightarrow 2\mathbf{w}_i + 2\mathbf{w}_i\mathbf{u}_i^2 - \mathbf{u}_i^2 = \mathbf{u}_i^2$$

$$\Rightarrow 2\mathbf{w}_i = 2\mathbf{u}_i^2 - 2\mathbf{w}_i\mathbf{u}_i^2$$

$$\Rightarrow \mathbf{w}_i = \mathbf{u}_i^2\left(1 - \mathbf{w}_i\right)$$

$$\Rightarrow \frac{\mathbf{w}_i}{1 - \mathbf{w}_i} = \mathbf{u}_i^2$$

$\mathbf{u}_i$ has two solutions:

$$\Rightarrow \mathbf{u}_i = \pm\sqrt{\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}}$$

TABLE 2
The results of the untargeted attack in terms of ASR and SNR on the test set. The perturbation is generated by a single randomly selected sample of each class. Proposed penalty method is used to generate attacks against 1D CNN Gamma model [49].

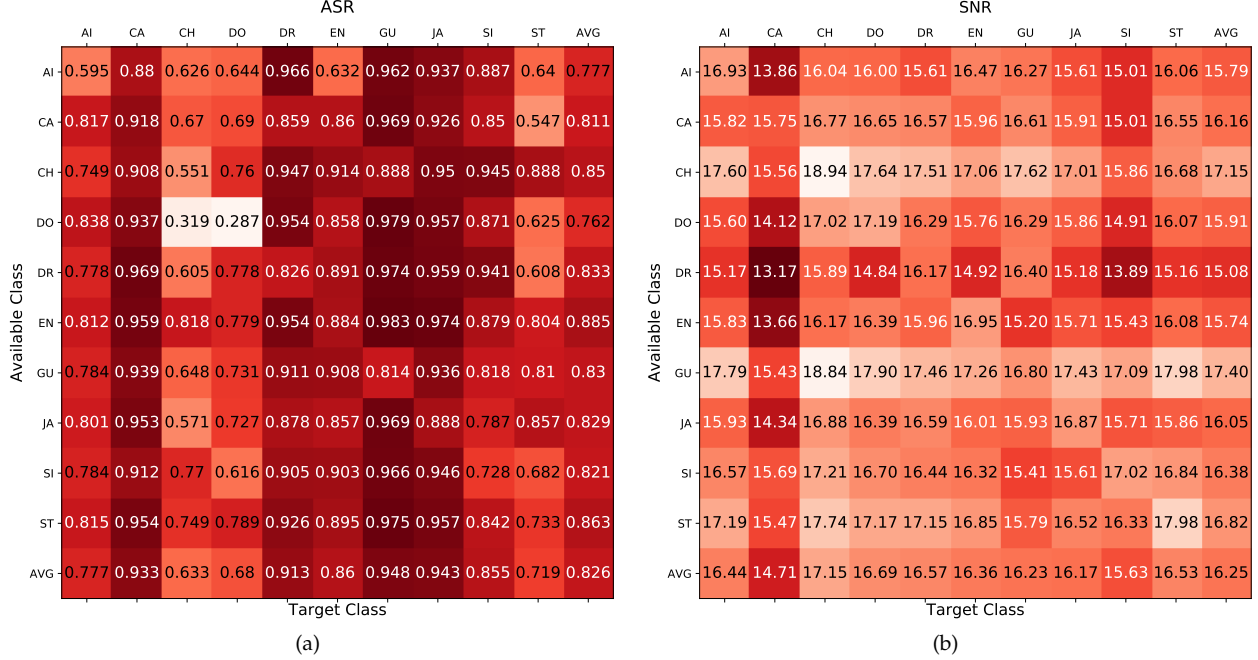| | Available Class | | | | | | | | | | |
	AI	CA	CH	DO	DR	EN	GU	JA	SI	ST	Mean
ASR test set	0.614	0.784	0.867	0.836	0.727	0.635	0.590	0.577	0.856	0.593	0.708
SNR test set	16.08	16.55	14.98	17.64	16.19	16.07	18.47	19.98	15.31	17.20	16.85

Fig. 3. The results of the targeted attack in terms of ASR (a) and SNR (b) on the test set. Proposed penalty method is used for crafting the perturbation. The perturbation is generated by a single randomly selected sample of each class in order to attack the 1D CNN Gamma model [49] for a specific target class.

Considering $e^{\mathbf{b}_i} = \mathbf{u}_i$, for the first solution we have:

$$e^{\mathbf{b}_i} = \sqrt{\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}}$$

$$\Rightarrow e^{\mathbf{b}_i} = \left(\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}\right)^{\frac{1}{2}}$$

$$\Rightarrow \ln\left(e^{\mathbf{b}_i}\right) = \ln\left(\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}\right)^{\frac{1}{2}}$$

$$\Rightarrow \mathbf{b}_i = \frac{1}{2}\ln\left(\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}\right).$$

By substituting back $\mathbf{b}_i$ by $\mathbf{x}'_i + \mathbf{v}$, we have:

$$\Rightarrow \mathbf{x}'_i + \mathbf{v} = \frac{1}{2}\ln\left(\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}\right)$$

$$\Rightarrow \mathbf{v} = \frac{1}{2}\ln\left(\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}\right) - \mathbf{x}'_i.$$

For the second solution we have:

$$\mathbf{b}_i = \frac{1}{2}\ln\left(-\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}\right)$$

However, since $0 \leq \mathbf{w}_i \leq 1$ and the natural logarithm of a negative number is undefined, the second solution for $\mathbf{u}_i$ is invalid.

APPENDIX B
DETAILED TARGETED ATTACK RESULTS

Tables 3 to 7 show the detailed ASR on the target models in the targeted attack scenario for training and test sets. ASRs are reported for each specific target class of UrbanSound8k [47]. Mean SNRs of the examples after adding universal perturbation are also reported. The target classes are: Air conditioner (AI), Car horn (CA), Children playing (CH), Dog bark (DO), Drilling (DR), Engine (EN) idling, Gun shot (GU), Jackhammer (JA), Siren (SI), Street music (ST).

APPENDIX C
TARGET MODELS

In this study five types of models are targeted. For training all models, categorical crossentropy is used as loss function and Adadelta [54] is used for optimizing the parameters

TABLE 3
ASR and mean SNR for targeting each label of UrbanSound8k [47] dataset. The target model is 1D CNN Rand.

Method		AI	CA	CH	DO	DR	EN	GU	JA	SI	ST
						Target Classes					
Iterative	ASR training set	0.962	0.900	0.924	0.920	0.941	0.937	0.919	0.907	0.936	0.909
	ASR test set	0.636	0.741	0.602	0.666	0.716	0.656	0.808	0.662	0.640	0.593
	SNR (dB) test set	25.396	24.234	25.751	25.256	26.494	25.715	24.689	25.399	25.623	24.651
Penalty	ASR training set	0.907	0.917	0.916	0.919	0.906	0.932	0.929	0.921	0.906	0.917
	ASR test set	**0.846**	**0.872**	**0.807**	**0.832**	**0.860**	**0.890**	**0.905**	**0.871**	**0.822**	**0.834**
	SNR (dB) test set	24.170	22.492	23.239	22.532	24.371	23.647	24.688	23.445	23.172	22.920

TABLE 4
ASR and mean SNR for targeting each label of UrbanSound8k [47] dataset. The target model is 1D CNN Gamma.

Method		AI	CA	CH	DO	DR	EN	GU	JA	SI	ST
						Target Classes					
Iterative	ASR training set	0.905	0.938	0.952	0.941	0.968	0.936	0.959	0.974	0.941	0.934
	ASR test set	0.745	0.874	0.744	0.810	0.815	0.746	0.887	0.795	0.772	0.761
	SNR (dB) test set	21.766	23.091	23.384	24.829	25.339	22.620	24.734	24.530	22.519	23.058
Penalty	ASR training set	0.916	0.928	0.923	0.912	0.909	0.902	0.926	0.910	0.903	0.900
	ASR test set	**0.871**	**0.920**	**0.895**	**0.891**	**0.879**	**0.886**	**0.900**	**0.899**	**0.883**	**0.855**
	SNR (dB) test set	21.351	21.730	22.392	22.648	24.257	22.141	23.380	24.663	21.401	22.902

TABLE 5
ASR and mean SNR for targeting each label of UrbanSound8k [47] dataset. The target model is ENVnet-V2.

Method		AI	CA	CH	DO	DR	EN	GU	JA	SI	ST
						Target Classes					
Iterative	ASR training set	0.939	0.928	0.921	0.925	0.882	0.923	0.902	0.929	0.901	0.906
	ASR test set	0.797	0.835	0.697	0.767	0.764	0.763	0.804	0.744	0.747	0.754
	SNR (dB) test set	22.852	22.932	22.371	23.183	22.702	23.387	20.868	23.355	23.049	22.637
Penalty	ASR training set	0.926	0.935	0.916	0.928	0.923	0.904	0.929	0.904	0.936	0.919
	ASR test set	**0.873**	**0.902**	**0.860**	**0.873**	**0.867**	**0.888**	**0.895**	**0.879**	**0.866**	**0.863**
	SNR (dB) test set	22.143	21.208	22.241	21.601	22.251	21.917	20.798	22.367	21.971	21.818

TABLE 6
ASR and mean SNR for targeting each label of UrbanSound8k [47] dataset. The target model is SincNet.

Method		AI	CA	CH	DO	DR	EN	GU	JA	SI	ST
						Target Classes					
Iterative	ASR training set	1.000	0.998	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	ASR test set	0.754	0.955	0.931	0.854	0.916	0.919	**0.959**	0.878	0.905	0.920
	SNR (dB) test set	28.852	25.245	29.940	30.968	29.004	28.910	26.639	26.966	30.356	29.800
Penalty	ASR training set	0.935	0.974	0.964	0.948	0.985	0.994	0.924	0.970	0.966	0.961
	ASR test set	**0.941**	**0.990**	**0.974**	**0.957**	**0.987**	**0.994**	0.943	**0.975**	**0.978**	**0.975**
	SNR (dB) test set	33.329	25.554	32.919	32.652	28.420	28.579	28.437	28.946	32.663	32.616

of the models. In this section, we present the complete description of the models.

C.1 1D CNN Rand

Table 8 shows the configuration of 1D CNN Rand [49]. This model consists of 5 one-dimensional convolutional layers. The number of kernels of each convolutional layer is 16, 32, 64, 128 and 256. The size of the feature maps of each convolutional layer is 64, 32, 16, 8 and 4. The first, second and fifth convolutional layers are followed by a one-dimensional max-pooling layer of size of eight, eight and four, respectively. The output of the second pooling layer is used as input to two Fully Connected (FC) layers on which

a drop-out with probability of 0.5 is applied for both layers [55]. Relu is used as the activation function for all of the layers. The number of neurons of the FC layers are 128 and 64. In order to reduce the over-fitting, batch normalization is applied after the activation function of each convolution layer [56]. The output of the last fully connected layer is used as the input to a softmax layer with ten neurons for classification.

C.2 1D CNN Gamma

This model is similar to 1D CNN Rand except that a gamma-tone filter-bank is used for initialization of the filters of the first layer of this model [49]. Table 9 shows the configuration

TABLE 7
ASR and mean SNR for targeting each label of UrbanSound8k [47] dataset. The target model is SincNet+VGG.

Method		Target Classes									
		AI	CA	CH	DO	DR	EN	GU	JA	SI	ST
Iterative	ASR training set	0.990	0.992	0.993	0.994	0.995	0.972	0.925	0.996	0.999	0.994
	ASR test set	0.876	**0.918**	0.855	0.887	0.863	0.831	**0.899**	0.852	0.864	0.879
	SNR (dB) test set	25.571	26.434	27.734	25.674	28.890	24.276	22.930	26.621	27.292	26.607
Penalty	ASR training set	0.902	0.908	0.903	0.922	0.921	0.903	0.920	0.922	0.924	0.935
	ASR test set	**0.899**	0.889	**0.898**	**0.897**	**0.897**	**0.881**	0.882	**0.907**	**0.919**	**0.914**
	SNR (dB) test set	26.417	27.191	28.784	27.085	28.448	24.626	22.411	27.330	27.276	27.787

of this model. The gammatone filters are kept frozen and they are not trained during the backpropagation process. Sixty-four filters are used to decompose the input signal into appropriate frequency bands. This filter-bank covers the frequency range between 100 Hz to 8 kHz. After this layer, batch normalization is also applied [56].

C.3 ENVnet-V2

Table 10 shows the architecture of ENVnet-V2 [48]. This model extracts short-time frequency features from audio waveforms by using two one-dimensional convolutional layers with 32 and 64 filters, respectively followed by a one-dimensional max-pooling layer. The model then swaps axes and convolves in time and frequency domain features using two two-dimensional convolutional layers each with 32 filters. After the convolutional layers, a two-dimensional max-pooling layer is used. After that, two other two-dimensional convolutional layers followed by a max-pooling layer are used and finally another two-dimensional convolutional layer with 128 filters is used. After using two FC layers with 4,096 neurons, a softmax layer is applied for classification. Drop-out with probability of 0.5 is applied to FC layers [55]. Relu is used as the activation function for all of the layers.

C.4 SincNet

Table 11 shows the architecture of SincNet [18]. In this model, 80 sinc functions are used as band-pass filters for decomposing the audio signal into appropriate frequency bands. After that, two one-dimensional convolutional layers with 80 and 60 filters are applied. Layer normalization [57] is used after each convolutional layer. After each convolutional layer, max-pooling is used. Two FC layers followed by a softmax layer is used for classification. Drop-out with probability of 0.5 is applied to FC layers [55]. Batch normalization [56] is used after FC layers. In this model, all hidden layers use leaky-ReLU [58] non-linearity.

C.5 SincNet+VGG19

Table 12 shows the specification of this architecture. This model uses 227 Sinc filters to extract features from the raw audio signal as it is introduced in SincNet [18]. After applying one-dimensional max-pooling layer of size 218 with stride of one, and layer normalization [57], the output is stacked along time axis to form a 2D representation. This time-frequency representation is used as input to a VGG19 [50] network followed by a FC layer and softmax layer for classification. The parameters of the VGG19 are the same

TABLE 8
1D CNN Rand architecture.

Layer	Ksize	Stride	# of filters	Data shape
InputLayer	-	-	-	(50,999, 1)
Conv1D	64	2	16	(25,468, 16)
MaxPooling1D	8	8	16	(3,183, 16)
Conv1D	32	2	32	(1,576, 32)
MaxPooling1D	8	8	32	(197, 32)
Conv1D	16	2	64	(91, 64)
Conv1D	8	2	128	(42, 128)
Conv1D	4	2	256	(20, 256)
MaxPooling1D	4	4	128	(5, 256)
FC	-	-	128	(128)
FC	-	-	64	(64)
FC	-	-	10	(10)

TABLE 9
1D CNN Gamma architecture

Layer	Ksize	Stride	# of filters	Data shape
InputLayer	-	-	-	(50,999, 1)
Conv1D	512	1	64	(50,488, 64)
MaxPooling1D	8	8	64	(6,311, 64)
Conv1D	32	2	32	(3,140, 32)
MaxPooling1D	8	8	32	(392, 32)
Conv1D	16	2	64	(189, 64)
Conv1D	8	2	128	(91, 128)
Conv1D	4	2	256	(44, 256)
MaxPooling1D	4	4	128	(11, 256)
FC	-	-	128	(128)
FC	-	-	64	(64)
FC	-	-	10	(10)

as described in [50] and they are not changed in this study. The output of the VGG19 is used as input of a softmax layer with ten neurons for classification.

REFERENCES

[1] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans Neural Netw and Learn Systems*, 2019.

TABLE 10
ENVnet-V2 architecture

Layer	Ksize	Stride	# of filters	Data shape
InputLayer	-	-	-	(50,999, 1)
Conv1D	64	2	32	(25,468, 32)
Conv1D	16	2	64	(12,727, 64)
MaxPooling1D	64	64	64	(198, 64)
swapaxes	-	-	-	(64, 198, 1)
Conv2D	(8,8)	(1,1)	32	(57, 191, 32)
Conv2D	(8,8)	(1,1)	32	(50, 184, 32)
MaxPooling2D	(5,3)	(5,3)	32	(10, 61, 32)
Conv2D	(1,4)	(1,1)	64	(10, 58, 64)
Conv2D	(1,4)	(1,1)	64	(10, 55, 64)
MaxPooling2D	(1,2)	(1,2)	64	(10, 27, 64)
Conv2D	(1,2)	(1,1)	128	(10, 26, 128)
FC	-	-	4,096	(4,096)
FC	-	-	4,096	(4,096)
FC	-	-	10	(10)

TABLE 11
SincNet architecture

Layer	Ksize	Stride	# of filters	Data shape
InputLayer	-	-	-	(50,999, 1)
SincConv1D	251	1	80	(50,749, 80)
MaxPooling1D	3	1	80	(16,916, 80)
Conv1D	5	1	60	(16,912, 60)
MaxPooling1D	3	1	60	(5,637, 60)
Conv1D	5	1	60	(5,633, 60)
FC	-	-	128	(128)
FC	-	-	64	(64)
FC	-	-	10	(10)

TABLE 12
SincNet+VGG19 architecture

Layer	Ksize	Stride	# of filters	Data shape
InputLayer	-	-	-	(50,999, 1)
SincConv1D	251	1	227	(50,749, 1)
MaxPooling1D	218	1	227	(232, 1)
Reshape	-	-	-	(232, 227, 1)
VGG19 [50]	-	-	-	(4096)
FC	-	-	10	(10)

[2] M. Yuan, B. Van Durme, and J. L. Ying, "Multilingual anchoring: Interactive topic modeling and alignment across languages," in *Adv in Neural Inf Proc Systems*, 2018, pp. 8653–8663.

[3] Z. Yang, Z. Hu, C. Dyer, E. P. Xing, and T. Berg-Kirkpatrick, "Unsupervised text style transfer using language models as discriminators," in *Adv in Neural Inf Proc Systems*, 2018, pp. 7287–7298.

[4] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. L. Moreno, Y. Wu *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Adv in Neural Inf Proc Systems*, 2018, pp. 4480–4490.

[5] Y.-A. Chung, W.-H. Weng, S. Tong, and J. Glass, "Unsupervised cross-modal alignment of speech and text embedding spaces," in *Adv in Neural Inf Proc Systems*, 2018, pp. 7354–7364.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint 1312.6199*, 2013.

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *Intl Conf on Learning Representations*, 2015.

[8] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symp on Security and Privacy*, 2017, pp. 39–57.

[9] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[10] K. Grosse, T. A. Trost, M. Mosbach, M. Backes, and D. Klakow, "Adversarial initialization–when your network performs the way i want," *arXiv preprint arXiv:1902.03020*, 2019.

[11] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Patt Recog*, vol. 84, pp. 317–331, 2018.

[12] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4954–4963.

[13] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *IEEE Conf Comp Vis Patt Recog*, July 2017.

[14] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint 1412.5567*, 2014.

[15] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *IEEE Intl Conf on Acoustics, Speech and Signal Processing*, 2015, pp. 4624–4628.

[16] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," 2016.

[17] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *16th Annual Conf of the Intl Speech Communication Association*, 2015.

[18] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *IEEE Spoken Language Techn Works*, 2018, pp. 1021–1028.

[19] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, "Learning filterbanks from raw speech for phone recognition," in *IEEE Intl Conf on Acoustics, Speech and Signal Processing*, 2018, pp. 5509–5513.

[20] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux, "End-to-end speech recognition from the raw waveform," *arXiv preprint 1806.07098*, 2018.

[21] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *IEEE Security and Privacy Workshops*, 2018, pp. 1–7.

[22] J. Peck, J. Roels, B. Goossens, and Y. Saeys, "Lower bounds on the robustness to adversarial perturbations," in *Adv in Neural Inf Proc Systems*, 2017, pp. 804–813.

[23] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein, "Are adversarial examples inevitable?" in *Intl Conf on Learning Repres*, 2019.

[24] A. Fawzi, H. Fawzi, and O. Fawzi, "Adversarial vulnerability for any classifier," in *Adv in Neural Inf Proc Systems*, 2018, pp. 1178–1187.

[25] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Intl Conf on Learning Representations*, 2017.

[26] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing Robust Adversarial Examples," in *Intl Conf on Machine Learning*, 2018.

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

[28] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *2017 ACM SIGSAC Conf on Computer and Communications Security*, 2017, pp. 103–117.

[29] Y. Gong and C. Poellabauer, "Crafting adversarial examples for speech paralinguistics applications," *arXiv preprint 1711.03280*, 2017.

[30] C. Kereliuk, B. L. Sturm, and J. Larsen, "Deep learning and music adversaries," *IEEE Trans Multimedia*, vol. 17, no. 11, pp. 2059–2071, 2015.

[31] T. Du, S. Ji, J. Li, Q. Gu, T. Wang, and R. Beyah, "Sirenattack: Generating adversarial audio for end-to-end acoustic systems," *arXiv preprint 1901.07846*, 2019.

[32] M. Alzantot, B. Balaji, and M. Srivastava, "Did you hear that? adversarial examples against automatic speech recognition," *arXiv preprint 1801.00554*, 2018.

[33] T. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," 2015.

[34] H. Yakura and J. Sakuma, "Robust audio adversarial example for a physical attack," *arXiv preprint 1810.11793*, 2018.

[35] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," *arXiv preprint 1903.10346*, 2019.

[36] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto, "Robustness of classifiers to universal perturbations: A geometric perspective," in *Intl Conf on Learn Repres*, 2018.

[37] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *IEEE Intl Conf on Computer Vision*, 2017, pp. 2774–2783.

[38] M. Behjati, S.-M. Moosavi-Dezfooli, M. S. Baghshah, and P. Frossard, "Universal adversarial attacks on text classifiers," in *IEEE Intl Conf on Acoustics, Speech and Signal Processing*, 2019, pp. 7345–7349.

[39] J. Hayes and G. Danezis, "Learning universal adversarial perturbations with generative models," in *IEEE Security and Privacy Workshops*, 2018, pp. 43–49.

[40] P. Neekhara, S. Hussain, P. Pandey, S. Dubnov, J. McAuley, and F. Koushanfar, "Universal adversarial perturbations for speech recognition systems," *Interspeech 2019*, Sep 2019.

[41] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," *arXiv preprint 1811.09600*, 2018.

[42] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *IEEE Conf Comp Vis Patt Recog*, 2016, pp. 2574–2582.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint 1412.6980*, 2014.

[44] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J Mach Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[45] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Intl Conf Mach Learning*, 2013, pp. 1139–1147.

[46] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[47] J. Salamon, C. Jacoby, and J. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM Intl Conf on Multimedia*, New York, NY, USA, 2014, pp. 1041–1044.

[48] Y. Tokozume, Y. Ushiku, and T. Harada, "Learning from between-class examples for deep sound recognition," *arXiv preprint 1711.10282*, 2017.

[49] S. Abdoli, P. Cardinal, and A. L. Koerich, "End-to-end environmental sound classification using a 1D convolutional neural network," *Expert Systems with Applic*, vol. 136, pp. 252–263, 2019.

[50] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint 1409.1556*, 2014.

[51] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *IEEE Conf Comp Vis Patt Recog*, 2017, pp. 1765–1773.

[52] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Intl Conf on Learn Repres*, 2017.

[53] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Intl Conf on Learn Repres*, 2018.

[54] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *J Mach Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[56] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint 1502.03167*, 2015.

[57] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint 1607.06450*, 2016.

[58] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc ICML*, vol. 30, no. 1, 2013, p. 3.

Sajjad Abdoli Received his Master's degree in computer engineering from QIAU, Qazvin, Iran in 2017. He is currently a Ph.D candidate at École de Technologie Supérieure (ÉTS), Université du Québec, Montreal, QC, Canada. His research interests include audio and speech processing, music information retrieval and developing adversarial attacks on machine learning systems.

Luiz Gustavo Hafemann received his B.S. and M.Sc. degrees in Computer Science from the Federal University of Paraná, Brazil, in the years of 2008 and 2014, respectively. He received his Ph.D. degree in Systems Engineering in 2019 from the École de Technologie Supérieure, Canada. He is currently a researcher at Sportlogiq, applying computer vision models for sports analytics. His interests include meta-learning, adversarial machine learning and group activity recognition.

Jérôme Rony received his M.A.Sc. degree in Systems Engineering in 2019 from the École de Technologie Supérieure (ÉTS), Université du Québec, Montreal, QC, Canada. He is currently a Ph.D candidate at ÉTS whose research interests include computer vision, adversarial examples and robust machine learning.

Ismail Ben Ayed received the PhD degree (with the highest honor) in computer vision from the Institut National de la Recherche Scientifique (INRS-EMT), Montreal, QC, in 2007. He is currently Associate Professor at the Ecole de Technologie Superieure (ETS), University of Quebec, where he holds a research chair on Artificial Intelligence in Medical Imaging. Before joining the ETS, he worked for 8 years as a research scientist at GE Healthcare, London, ON, conducting research in medical image analysis. His research interests are in computer vision, optimization, machine learning and their potential applications in medical image analysis.

Patrick Cardinal received the B. Eng. degree in electrical engineering in 2000 from École de Technologie Supérieure (ÉTS), M.Sc. from McGill University in 2003 and PhD from ÉTS in 2013. From 2000 to 2013, he has been involved in several projects related to speech processing, especially in the development of a closed-captioning system for live television shows based on automatic speech recognition. After his postdoc at MIT, he joined ÉTS as a professor. His research interests cover several aspects of speech processing for real life and medical applications.

Alessandro Lameiras Koerich is an Associate Professor in the Dept. of Software and IT Engineering of the École de Technologie Supérieure (ÉTS). He received the B.Eng. degree in electrical engineering from the Federal University of Santa Catarina, Brazil, in 1995, the M.Sc. in electrical engineering from the University of Campinas, Brazil, in 1997, and the Ph.D. in engineering from the ÉTS, in 2002. His current research interests include computer vision, machine learning and music information retrieval.

APPENDIX D
AUDIO EXAMPLES

Several randomly chosen examples of perturbed audio samples of Urbansound8k dataset [47] are also presented. The audio samples are perturbed based on two presented methods in this study. Targeted and untargeted perturbations are considered. Table 13 shows a list of the samples. Methodology of crafting the samples, target models, SNR of the perturbed samples, detected class of the sample by each model as well as the true class of the samples are also presented.

Sample	Detected Class	True Class	Target Model	Method	Targeted/Untargeted	SNR
DR_0_org.wav	Drilling	Drilling	SINCNet	N/A	N/A	N/A
DR_0_pert_itr.wav	Gun shot	Drilling	SINCNet	Iterative	Targeted	27.025
DR_0_pert_pen.wav	Gun shot	Drilling	SINCNet	penalty	Targeted	28.040
SI_0_org.wav	Siren	Siren	SINCNet	N/A	N/A	N/A
SI_0_pert_itr.wav	Airconditioner	Siren	SINCNet	Iterative	Targeted	29.244
SI_0_pert_pen.wav	Airconditioner	Siren	SINCNet	penalty	Targeted	33.766
CH_0_org.wav	Children playing	Children playing	SINCNet	N/A	N/A	N/A
CH_0_pert_itr.wav	Dog bark	Children playing	SINCNet	Iterative	Targeted	31.784
CH_0_pert_pen.wav	Dog bark	Children playing	SINCNet	penalty	Targeted	33.207
ST_0_org.wav	Street music	Street music	SINCNet	N/A	N/A	N/A
ST_0_pert_itr.wav	Airconditioner	Street music	SINCNet	Iterative	Targeted	28.994
ST_0_pert_pen.wav	Airconditioner	Street music	SINCNet	penalty	Targeted	33.539
DO_0_org.wav	Dog bark	Dog bark	SINCNet	N/A	N/A	N/A
DO_0_pert_itr.wav	Drilling	Dog bark	SINCNet	Iterative	Targeted	28.861
DO_0_pert_pen.wav	Drilling	Dog bark	SINCNet	penalty	Targeted	28.040
EN_0_org.wav	Engine idling	Engine idling	SINCNet+VGG19	N/A	N/A	N/A
EN_0_pert_itr.wav	Drilling	Engine idling	SINCNet+VGG19	Iterative	untargeted	26.378
EN_0_pert_pen.wav	Children playing	Engine idling	SINCNet+VGG19	penalty	untargeted	23.444
CA_0_org.wav	Car horn	Car horn	SINCNet+VGG19	N/A	N/A	N/A
CA_0_pert_itr.wav	Drilling	Car horn	SINCNet+VGG19	Iterative	untargeted	26.175
CA_0_pert_pen	Street music	Car horn	SINCNet+VGG19	penalty	untargeted	23.341
AI_0_org.wav	Airconditioner	Airconditioner	SINCNet+VGG19	N/A	N/A	N/A
AI_0_pert_itr.wav	Jackhammer	Airconditioner	SINCNet+VGG19	Iterative	untargeted	25.759
AI_0_pert_pen.wav	Children playing	Airconditioner	SINCNet+VGG19	penalty	untargeted	23.073
SI_1_org.wav	Siren	Siren	SINCNet+VGG19	N/A	N/A	N/A
SI_1_pert_itr.wav	Jackhammer	Siren	SINCNet+VGG19	Iterative	untargeted	26.334
SI_1_pert_pen.wav	Children playing	Siren	SINCNet+VGG19	penalty	untargeted	23.423
DR_1_org.wav	Drilling	Drilling	SINCNet+VGG19	N/A	N/A	N/A
DR_1_pert_itr.wav	Jackhammer	Drilling	SINCNet+VGG19	Iterative	untargeted	27.040
DR_1_pert_pen.wav	Children playing	Drilling	SINCNet+VGG19	penalty	untargeted	24.555

TABLE 13
List of examples of perturbed audio samples, Methodology of crafting the samples, SNR of the perturbed samples, target models, and also detected class of the sample by each model and the true class of the samples. The audio files belong to UrbanSound8k dateset [47]. N/A: Not Applicable