End-to-End Deep Learning for Audio Classification: From Waveforms to a Security Perspective

by

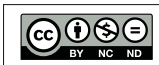
Sajjad ABDOLI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, DECEMBER 21, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC


Sajjad Abdoli, 2021



This Creative Commons license allows readers to download this work and share it with others as long as the author is credited. The content of this work cannot be modified in any way or used commercially.

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Alessandro Lameiras Koerich , Thesis supervisor
Department of Software and IT Engineering, École de technologie supérieure

Mr. Patrick Cardinal, co-Supervisor
Department of Software and IT Engineering, École de technologie supérieure

Mr. Jérémie Voix, President of the board of examiners
Department of Mechanical Engineering, École de technologie supérieure

Mr. Marco Pedersoli, Internal examiner
Department of Systems Engineering, École de technologie supérieure

Mr. François Grondin, External independent examiner
Department of Electrical Engineering and Computer Engineering, Université de Sherbrooke

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON DECEMBER 6, 2021

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

Dedicated to

My Mom, Dad and my beloved Sister,

for their unconditional love

Dedicated to the memory of

Milad Ghasemi Ariani,

a kind friend,

PhD student in marketing, University of Guelph,

the passenger of flight PS752

Apprentissage profond de bout en bout pour la classification audio : des formes d'onde aux perspectives de sécurité

Sajjad ABDOLI

RÉSUMÉ

Le traitement audio est un problème majeur de l'apprentissage automatique. Les modèles d'apprentissage profond ont récemment eu un impact significatif sur les problèmes de traitement du son tels que la classification audio, la reconnaissance vocale ou l'identification du locuteur. La plupart des méthodes basées sur les modèles profonds reposent sur l'utilisation de représentations 2D comme les spectrogrammes. Ces méthodes sont inspirées des modèles d'apprentissage en profondeur conçus pour la vision par ordinateur et traitent les représentations 2D comme des images. Récemment, des modèles de traitement audio de bout en bout ont été développés pour traiter des signaux à une dimension. Dans ce cas, le signal audio brut, représenté par un vecteur 1D, est utilisé comme entrée des modèles profonds pour tirer pleinement parti du potentiel de ces modèles et éliminer les modules de traitement du signal habituellement utilisés pour extraire les caractéristiques des signaux audio. Cette étude explore comment les modèles profonds peuvent être utilisés pour le traitement audio, notamment, les problèmes de classification audio. Les méthodologies générales de traitement audio basées sur une extraction de caractéristiques prédéfinies de signaux audio bruts sont passées en revue. Une nouvelle architecture basée sur un réseau neuronal convolutif pour la classification des bruits environnementaux est proposée. Ce modèle élimine les modules de traitement du signal habituellement nécessaires pour l'extraction de caractéristiques prédéfinies et surpasse la plupart des approches existantes. Les résultats expérimentaux démontrent la puissance et le potentiel d'un tel modèle pour le problème de classification.

De plus, des études récentes démontrent la vulnérabilité des modèles d'apprentissage en profondeur aux attaques adverses, qui menacent la fonctionnalité et la crédibilité de ces modèles. De telles menaces couvrent un large éventail d'objectifs tels que l'évasion, les données, l'empoisonnement du modèle ou l'extraction des caractéristiques. Cette étude explore ces attaques en se concentrant sur les forces et faiblesses des modèles de traitement audio de bout en bout. La perturbation adverse est l'une des attaques d'évasion et constitue une menace grave. Cette attaque injecte une perturbation quasi imperceptible dans l'échantillon d'entrée pour tromper le modèle, qui va mal le classer. Dans ce contexte, la perturbation adverse universelle est une perturbation adverse unique qui peut tromper le classificateur une fois ajoutée aux échantillons d'entrée. Dans cette étude, deux méthodes pour créer des perturbations audio adverses universelles sont proposées. Une méthode est inspirée d'un algorithme glouton itératif bien connu en vision par ordinateur, et l'autre est basée sur une nouvelle formulation de pénalité. Les résultats expérimentaux indiquent l'efficacité d'une telle perturbation pour tromper une famille de modèles ESC de reconnaissance vocale de bout en bout. À cet égard, des mécanismes défensifs et des stratégies d'atténuation appropriées doivent être envisagés pour concevoir et mettre en œuvre de tels modèles.

Mots-clés: l'apprentissage en profondeur, traitement audio, apprentissage automatique, perturbation adverse

End-to-End Deep Learning for Audio Classification: From Waveforms to a Security Perspective

Sajjad ABDOLI

ABSTRACT

Audio processing is one of the challenging problems in machine learning. Deep learning models have recently had a significant impact on sound processing problems such as audio classification, speech recognition, speaker identification, *etc.* Most of the methods based on deep models usually rely on feeding the models by 2D representations like spectrograms. Inspired by deep learning models designed for computer vision, these 2D representations are treated as images. Recently, end-to-end audio processing models have been developed for various tasks. In this case, as a 1D vector, the raw audio signal is used as the input to deep models to benefit from the full potential of such models and eliminate signal processing modules for generating the handcrafted features from audio signals. This study explores how deep models can be used for audio processing, notably, audio classification problems. The general methodologies for audio processing based on handcrafted representations and raw audio signals as inputs to the models are reviewed. A novel end-to-end architecture based on a convolutional neural network for Environmental Sound Classification (ESC) is proposed. The proposed model eliminates the necessary signal processing modules for generating handcrafted features and outperforms most state-of-the-art approaches that use handcrafted features as input. The experimental results demonstrate the power and the potential of such an end-to-end model for the classification problem.

Moreover, recent studies demonstrate the vulnerability of deep learning models to a range of adversarial attacks, which threaten the functionality and credibility of such models. Such threats cover a wide range of goals such as evasion, data, model poisoning, model extraction, *etc.* This study also explores such attacks focusing on end-to-end audio processing models by mentioning their strengths and weaknesses. Adversarial perturbation is one of the evasion attacks, which is a severe threat against such models. This attack injects a quasi-imperceptible perturbation to the input sample to deceive the model into misclassifying it. In this context, universal adversarial perturbation is a single adversarial perturbation that can fool the classifier for most input samples once added to the input samples. This study proposes two methodologies for crafting universal adversarial audio perturbations. One method is inspired by an iterative greedy algorithm, which is well-known in computer vision, and the other is based on a novel penalty formulation. The experimental results indicate the effectiveness of such perturbation for fooling a family of end-to-end ESC and speech recognition models. In this regard, suitable defensive mechanisms and mitigation strategies must be considered for designing and implementing such end-to-end models.

Keywords: deep learning, audio processing, machine learning, adversarial perturbation.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

AI	Air conditioner
ACC	Accuracy
ASR	Attack Success Rate
AUC	Area Under Curve
C&W	Carlini and Wanger attack
CA	Car horn
CER	Charachter Error Rate
CH	Children playing
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classication
DCT	Discrete Cosine Transform
DDN	Decoupled Direction and Norm
DFT	Discrete Fourier Transform
DO	Dog bark
DR	Drilling
EN	Engine idling
ESC	Environmental Sound Classification
FC	Fully Connected
FCN	Fully Convolutional Network

FGSM	Fast Gradient Sign Method
FMA	Free Music Archive
GU	Gun shot
HVC	Hidden Voice Commands
JA	Jackhammer
LEAF	Learnable Frontend for audio classification
MFCC	Mel Frequency Cepstral Coefficient
MI-FGM	Momentum based Iterative Fast Gradient Method
MitE	Man in the Elevator
MSD	Million Song Dataset
MAP	Mean Average Precision
MSE	Mean Squared Error
PCEN	Per-Channel Energy Normalization
PSD	Power Spectral Density
RMS	Root Mean Square
SER	Sentence Error Rate
SID	Speaker Identification
SIR	Siren
SMS	Short Message Service
SNR	Signal to Noise Ratio

SPL	Sound Pressure Level
SR	Automatic Speech Recognition
SSA	Singular Spectrum Analysis
ST	Street music
STFT	Short-Time Fourier Transform
SVM	Support Vector Machines
TFSR	TensorFlow Speech Recognition
UAP	Universal Adversarial Perturbation
WER	Word Error Rate

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

dB	Decibel
$\mathbf{W}^{net}$	Weights matrix of the neural network
b^{net}	Bias parameters of the neural network
X	Set of data samples of the dataset
Y	Legitimate labels of the the data samples, X, of the dataset
$\mathbf{x}$	Input sample to the neural network from the dataset, X
x_i	i_{th} element of the input $\mathbf{x}$ to the neural network
y	Possible outcomes of probability estimation of classifier function for given input $\mathbf{x}$
C	Classes of the dataset
t	Target class from C classes of the dataset
y_t	The output of the classifier for target class t
K^{fl}	Number of the filters of CNN
F^{fl}	Size of the filters of CNN
M	Number of samples in the dataset
N	Dimensionalty of the input $\mathbf{x}$ to the neural network
L	Number of layers of neural network
l	l-th layer of neural network
a_m^L	Output of the layer L of the neural network for sample m
J	Cost function

θ	Set of all of the parameters of the neural network
F^{net}	Transformation function by neural network layers on the input
z	Output of transformation function
fr	Frequency in Hz scale
N_{fb}	Number of MFCC filters
$fb_{n_{fb}}$	$n_{fb_{th}}$ output of MFCC filter-bank
ct_{ρ}	DCT function
ρ	DCT coefficient
$\mathbf{I}_m$	m_th frame of video
W^{vid}	Rows (heights) in video frame
H^{vid}	Columns (widths) in video frame
T^{vid}	Number of sampled frames in video
Ψ	Number of concepts which are transferred from visual to audio network
TC_{ψ}	Teacher vision network for the the concept ψ
ST_{ψ}	Student audio network for the the concept ψ
L_{KL}	KL divergence loss function
$\tilde{\mathbf{x}}$	Adversarial sample
δ	Adversarial perturbation
ℓ_p	p norm
ϵ^{fgsm}	Small constant to make FGSM perturbation undetectable

y_l	Legitimate labels of the input samples
η^{fgsm}	FGSM learning rate
$\nabla_{\mathbf{x}} J$	Gradient of cost function w.r.t to input $\mathbf{x}$
dist	Function to compute the distance between legitimate and adversarial samples
$\mathbf{w}^{cw}$	Auxiliary variable in (C&W) ℓ_2 attack variable change
G	Linear hinge function
f	Logit (pre-softmax) output of neural network
c	Positive stability constant in attack objective function
κ	Confidence level of sample misclassification in attack objective function
ϵ_k^{ddn}	Norm in DDN attack at iteration k
γ^{att}	Factor to modify norm at each iteration of DDN attack
K^{ddn}	Number of iterations of DNN attack algorithm
α^{ddn}	Step of size of each iteration of DDN attack
g^{ddn}	Direction of the adversarial perturbation at each step of DDN attack
P^{sig}	Power of signal computed by RMS function
$l_{\mathrm{dB}_{\mathbf{x}}}$	Relative loudness of perturbation with respect to audio sample
l_{dB}	Log of ℓ_∞ norm of the signal
τ_{att}	Magnitude threshold of the perturbation signal
S	Number of the frames of splitted audio signal
s	Audio frame

h	Activation function
F^m	Number of feature maps
o_{sc}	The CNN prediction for the s segment of the audio for class c
μ^{sig}	The distribution of audio samples
$\hat{k}$	classifier that predicts the class of the audio sample
$\mathbf{v}$	UAP vector
v_n	The n-th component of the UAP array, $\mathbf{v}$
ξ	Parameter to control the magnitude of the UAP
r	Minimum perturbation that sends the sum of the sample and UAP to the decision boundary at each step of iterative algorithm
$\mathcal{P}_{p,\xi}$	Projection operator to project the adversarial sample ℓ_p ball of radius ξ and centered at 0
$\mathbf{1}\{\cdot\}$	True-or-false indicator function
$\mathbf{w}$	Auxiliary parameter in change of paramters to satisfy the box constraint in UAP generation by penalty method
$\mathbf{x}'_i$	Audio sample i presented in tanh space
$\mathbf{v}'$	Audio UAP presented in tanh space
ϵ^{uap}	Small constant that ensures that samples does not assume infinity values
k	k-th iteration of UAP penalty attack algorithm
L^{uap}	Penalty objective function for penalty-based method of UAP generation
L^{uap^*}	Optimal value of objective function, L^{uap}, for penalty-based method of UAP generation

g^{uap}	Cumulative gradient of penalty objective function, L^{uap}, for UAP generation
Z	Statistic test value
$\widehat{p_l}$	ASR of attacking method with lower ASR
$\widehat{p_h}$	ASR of attacking method with higher ASR
$\widehat{X_l}$	Number of successfully attacked samples by by attacking method with lower ASR
$\widehat{X_h}$	Number of successfully attacked samples by by attacking method with higher ASR
H_0	Null hypothesis
H_a	Alternative hypothesis
α	Level of significance in Z test
z_α	Z test critical value related to level of significance, α

INTRODUCTION

The sounds around us provide helpful information about the environment we live in. For example, when the sound of a siren and non-stop sounds of honks of fire trucks or ambulances are heard in the street, there might be an incident that happened nearby. So, it is necessary to be vigilant and cautious. In this context, automatic alarm systems based on environmental sounds help to empower citizens, especially in dangerous incidents like explosions or shootings. Such systems may also be a valuable tool for security guards and police officers since they can detect the source of the sounds and produce practical information on the approximate location of the incident.

Moreover, recent studies show that living in a noisy environment like big cities has a significant negative impact on our mental health (Jensen, Rasmussen & Ekholm, 2019). In this case, a noise monitoring system that can detect the noise source in busy cities is quite helpful for possibly removing the noise source to improve the citizens' life quality. Moreover, having such systems near sensitive areas like schools or hospitals is quite helpful. For example, such a system may notify the authorities about construction in illegal periods by detecting the source of the noisy sounds like drilling and jackhammer. Then, the government may plan noise mitigation programs.

Audio is also ubiquitous content on the internet. The music sharing platforms facilitate listening to any type of music. For instance, Spotify, an audio streaming website, has more than 157 million subscribers and provides more than 30 million songs. The large amount of data on the internet magnifies the need for robust and efficient approaches to archive and classify audio files and retrieve information. In this context, audio processing systems play an essential role in this environment.

0.1 Problem Statement

In this thesis, the problem of audio classification, such as environmental sounds, is presented. Several different tasks may be performed on audio. For example, audio speech may be classified

according to the speaker's gender, or music may also be classified according to the type of instruments present in the music. Moreover, the source detection of different environmental sounds could also be considered an interesting problem. Some examples of target classes for classification problem in speech and music are shown in Tables 0.1 and 0.2. In general, the main goal of a system that processes audio automatically is to classify audio into the appropriate class in a supervised fashion. Classical audio classification systems are based on two fundamental steps: feature extraction and classification. Feature extraction attempts to represent the audio signal in a compact and relevant manner for the classification point of view. For instance, different audio features have been proposed in the last years, such as Mel frequency cepstral coefficients (MFCCs), pitch histograms, *etc.* In the classification step, the features extracted from the audio signal are firstly used to build a model that must predict a class, given an input. For such an aim, different types of machine learning algorithms such as support vector machines (SVMs), neural networks, k-nearest neighbors *etc.*, may be used.

Table 0.1 Music Classification Targets

Music	**Targets**
Genre	Pop, Rock, Jazz, ...
Mood detection	Sad, Sleepy, Happy, ...
Instrument Recognition	Piano, Cello, Viola , ...
Latin Music modes	Salsa, Tango, Bolero , ...
Iranian music modes	Shur, Segah, Homayun , ...

Table 0.2 Speech Classification Targets

Speech	**Targets**
Gender	Male, Female
Speaker identification (SID)	Name of the speaker
Emotion Detection	Sadness, anger, fear,...

Most recent studies on deep learning systems for audio processing, especially for environmental sound classification, do not fully utilize deep learning algorithms such as convolutional neural networks (CNNs) to automatically learn the low-level and high-level audio representation

directly from the audio signal. Instead, most of the approaches based on CNNs take some 2D representation, such as spectrograms, produced by a separated algorithm as input to a CNN. Then, such a CNN is only used for learning high-level representations from handcrafted feature representations.

Moreover, recent studies have shown that deep learning models are vulnerable to a range of adversarial attacks (Biggio & Roli, 2018). Such attacks cover a wide range of threats. For instance, an attacker may try to cripple such systems by injecting some engineered, crafted perturbation to the input data of the model (adversarial perturbations) to fool the targeted neural network. Poisoning the training samples, stealing the weights of the trained model, or violating privacy are other types of attacks (Biggio & Roli, 2018). Detection of such attacks and proposing an effective defensive methodology against them require comprehensive knowledge of the current attacks. Recent studies show that breaking the deep learning-based models is relatively easier than defending them (Biggio & Roli, 2018). Thus, it is an ongoing cat and mouse game between the attacking and defending algorithms to attack and protect the deep-learning-based models.

Adversarial perturbations are usually generated by defining an appropriate objective function to solve an optimization problem. This optimization problem has two objectives: minimizing the magnitude of the injected noise and maximizing the success rate of fooling the target model. This function may use a range of knowledge from the target model, including the model architecture (computational graph) and weights, input examples, and the model's predictions. This function is usually minimized by the use of some gradient-based optimization algorithms. Recent studies (Biggio & Roli, 2018; Carlini & Wagner, 2017) have shown that by having an end-to-end model where all of the model layers are differentiable, this objective function can be straightforwardly used to attack the model using gradient-based methods. This type of attack injects adversarial perturbations to the model's inputs to deceive the model into misclassifying them. This issue poses a significant threat against such end-to-end models. Accordingly, this issue must be

considered for designing any end-to-end model for audio classification. The vulnerability of such models against adversarial attacks has not been widely addressed by community Abdullah, Warren, Bindschaedler, Papernot & Traynor (2020). Since such audio classification systems, notably ESC systems, might be used in sensitive environments for surveillance applications, the possible threats against such models must also be recognized and addressed.

0.2 Objectives of Research

The main research question posed for this thesis is: Is it possible to utilize an end-to-end deep learning algorithm for effective representation learning for audio waveforms while addressing the vulnerability of such an end-to-end model against adversarial perturbations?

For processing other modalities than audio, like image, deep learning models are used for useful representation learning from the raw image pixel matrix. No representation or feature of the input image, like color histograms, is used as the inputs to the deep learning model such as a CNN. Using the back-propagation algorithm, the parameters of the CNN are adjusted to the dataset samples, making it a powerful tool for representation learning. Considering the advances in utilizing CNNs for audio signal processing, most audio/music processing researchers are interested in using handcrafted features or 2D representations of the audio signal as the inputs to CNN. The research community primarily uses spectrograms as inputs to CNN. We argue that CNN is powerful enough for representation learning directly from the raw audio signal. This research aims to design, implement, and evaluate an end-to-end system based on CNN for environmental audio classification.

This idea is justified by reviewing the recent advances in CNN-based audio classification systems. As an instance, the end-to-end system presented by Dieleman & Schrauwen (2014), the performance of an end-to-end CNN-based system for music tag detection is comparable with a CNN-based system that uses the spectrogram of the audio as input. Moreover, our results

on the ESC problem (Salamon & Bello, 2017) based on an end-to-end CNN system have shown that better results than state-of-the-art methods can be obtained with an end-to-end approach. Therefore, such a system has some benefits. Since feature extraction modules can be replaced with CNNs, this method eliminates the need for writing codes for signal processing modules and tuning the parameters. Moreover, since signal processing modules are omitted in hardware development, such an end-to-end system may reduce the cost of building hardware modules for audio classification. Additionally, there are no guarantees that the engineered features like MFCCs are optimal for all audio classification tasks since these features are designed initially from perceptual evidence (Ravanelli & Bengio, 2018). On the other hand, such a system may have some downsides according to the security perspectives which must be considered in utilizing such models.

Recent studies demonstrate that machine learning models are vulnerable to a range of adversarial attacks (Biggio & Roli, 2018). These adversarial attacks pose a potential challenge on end-to-end audio processing systems, which has not been widely addressed in the audio processing domain (Abdullah *et al.*, 2020; Carlini & Wagner, 2018). As a part of the main research question of this thesis, a summary of current attacks on audio processing and classification models with an emphasis on adversarial perturbations is presented. We also question whether it is possible to effectively attack an end-to-end audio classifier by crafting a Universal Adversarial Perturbation (UAP) (Moosavi-Dezfooli, Fawzi, Fawzi & Frossard, 2017) vector. This kind of attack is challenging since a single perturbation vector must be crafted to perturb most dataset samples effectively. However, this attack has been widely addressed in targeting image processing models (Moosavi-Dezfooli *et al.*, 2017). One of the aims of this thesis is to expand this type of attack to audio classification models, notably a family of ESC systems, and comprehensively evaluate the effectiveness of such an attack.

0.3 Contribution on Research

In this thesis, we contribute to advancing the field of audio processing with the following contributions:

- Design and implementation of a CNN-based ESC model. Such a model obtains comparable results with systems that use 2D representations as input for the classifier.
- Demonstrate the existence of UAPs, which can fool a family of end-to-end audio classification models. We demonstrate the effectiveness of two methods for crafting such perturbations; the first is based on an iterative algorithm which is well-known in computer vision (Moosavi-Dezfooli *et al.*, 2017). The second is a novel penalty formulation for finding such a perturbation vector. We also demonstrate that the penalty method is more effective when the number of audio samples is limited. A theoretical proof also validates that the proposed penalty method converges to a solution that corresponds to universal adversarial perturbations. We also address some new challenges for crafting UAPs, such as transferability of the attacks between different models and their effectiveness in physical environments.

0.4 Organization of Document

The rest of this thesis is organized as follows: In Chapter 1 a comprehensive literature review regarding the thesis research questions is presented. General methodologies on audio classification models using an end-to-end methodology and low-level features and 2D representations as input to deep learning models are reviewed. In Chapter 2 the proposed end-to-end model for ESC problem is presented.

The trustworthiness of deep models against adversarial attacks, especially for targeting audio processing models, is discussed in detail in Chapter 1. Based on this survey, in Chapter 3 we address the problem of crafting UAPs for targeting a family of audio classification models.

In Chapter 4 the experimental protocol and results of the evaluation of the proposed method for ESC and also the proposed method for generating UAPs are presented. Then, the benchmark datasets are presented, and the effectiveness of the proposed methods are evaluated and discussed in detail. The conclusion and recommendations of future work are presented in the last chapter.

CHAPTER 1

LITERATURE REVIEW

In the last ten years, deep learning systems attracted a lot of attention in the machine learning community by providing a powerful learning framework for a large number of tasks in image, audio, and natural language processing. In audio processing, CNNs have had significant impact on several audio and music processing tasks such as automatic music tagging (Dieleman & Schrauwen, 2014), large-scale video clip classification based on audio information (Hershey et al., 2017), music genre classification (Costa, Oliveira & Silla, 2017), SID (Ravanelli & Bengio, 2018), ESC (Piczak, 2015b; Salamon & Bello, 2017; Pons & Serra, 2018; Simonyan & Zisserman, 2014; Tokozume & Harada, 2017) , among others. ESC is also an interesting problem. (Sigtia, Stark, Krstulović & Plumbley, 2016a; Stowell, Giannoulis, Benetos, Lagrange & Plumbley, 2015) which has different applications ranging from crime detection (Radhakrishnan, Divakaran & Smaragdis, 2005) to environmental context aware processing (Chu, Narayanan & Kuo, 2009). Moreover, with the increasing interest in smart cities, IOT devices embedding automatic audio classification can be very useful for urban acoustic monitoring (Mydlarz, Salamon & Bello, 2017) like intelligent audio-based surveillance system in public transportation (Laffitte, Wang, Sodoyer & Girin, 2019).

Several handcrafted low-level features have been proposed over the years in audio processing. The most popular feature is the well-known MFCCs (Jurafsky & Martin, 2014). Other techniques such as spectrogram, pitch histograms, fundamental frequency (F0) curves, and zero-crossing rates are also widely used in this domain. These features can be used as inputs to any audio processing models for appropriate tasks like classification. Most of the approaches for ESC also rely on handcrafted features and representations such as spectro-temporal representations (Ludeña-Choez & Gallardo-Antolín, 2016; Costa, Oliveira, Koerich, Gouyon & Martins, 2012). Spectral representations have been used as features in several approaches based on matrix factorization (Mesaros, Heittola, Dikmen & Virtanen, 2015; Benetos, Lafay, Lagrange & Plumbley, 2016; Bisot, Serizel, Essid & Richard, 2016; Salamon & Bello, 2015; Geiger & Helwani, 2015). In this

regard, Mesaros *et al.* (2015) presented an approach for overlapping sound event detection based on learning non-negative dictionaries through joint use of spectrum and class activity annotation. Benetos *et al.* (2016) also presented an approach for overlapping acoustic event detection based on probabilistic latent component analysis where each exemplar in a sound event dictionary consists of a succession of spectral templates. The method proposed by Bisot *et al.* (2016) learns features from time-frequency 2D representations in an unsupervised manner. The representations are decomposed using matrix factorization methods to build a dictionary and the projection coefficients are used as features for classification. Salamon & Bello (2015) also proposed a dictionary learning method based on the Spherical K-Means (SKM) algorithm, which used log-Mel spectrograms as 2D representations as inputs. Geiger & Helwani (2015) used Gabor filter-bank features and Gaussian mixture models for event detection. Mulimani & Koolagudi (2019) used a singular value decomposition method for extracting acoustic event-specific features from spectrograms. These features are used as inputs to an SVM classifier. Xie & Zhu (2019) proposed a method for aggregation of acoustic and visual features for acoustic scene classification. Several acoustic features like spectral centroid, spectral entropy, as well as several visual features like local binary pattern, histogram of gradients, are proposed. A suitable feature selection algorithm like principle component analysis is also used. The selected feature set is used as input to an SVM classifier.

Deep-leaning CNN-based approaches significantly improved over traditional handcrafted feature-based methods, notably for ESC. (Piczak, 2015b; Salamon & Bello, 2017; Pons & Serra, 2018; Simonyan & Zisserman, 2014; Tokozume & Harada, 2017). However, most of these approaches first convert the audio signal into a 2D representation and use 2D CNN architectures that were originally designed for object recognition, such as AlexNet (Krizhevsky, Sutskever & Hinton, 2012) and VGG (Simonyan & Zisserman, 2014). One of the main advantages of using 2D representations is that spectrograms can summarize high dimensional waveforms into a compact representation. Furthermore, due to the high dimensionality of an audio signal, 1D representations would not be complete representations so that perceptually related sounds would generally be near neighbours in the vector space (Stowell & Plumbley, 2014). So, a

powerful representation learning method is needed to provide suitable representations from 1D representations. Piczak (Piczak, 2015b) presented a CNN with two layers followed by three dense layers. The network operates on two input channels: log-Mel spectra and their deltas. However, Considering the proposed model has millions of parameters, one of the challenges in using 2D CNNs for such a task is that the modeling capacity of such networks depends on the availability of a large amount of annotated training data to learn the parameters.

However, deep learning approaches introduce the possibility of learning the best representation from the raw input values. This approach is widely used in image and video processing where only raw images or video frames are used as inputs to the models (Simonyan & Zisserman, 2014). In addition, one-dimensional CNNs that learn acoustic models directly from audio waveforms are becoming a popular method in audio processing due to the ability of these networks to take advantage of the signal's fine time structure (Hoshen, Weiss & Wilson, 2015; Ravanelli & Bengio, 2018; Zeghidour, Usunier, Synnaeve, Collobert & Dupoux, 2018; Sainath, Weiss, Senior, Wilson & Vinyals, 2015; Dai, Dai, Qu, Li & Das, 2017).

This chapter presents a review of general methodologies in using deep architectures for audio processing based on handcrafted audio representations and raw audio signals as inputs to deep learning models. Some interesting studies on these two categories are also discussed and explained in detail.

Recent studies have also demonstrated that deep models are vulnerable to adversarial attacks (Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow & Fergus, 2013; Goodfellow, Shlens & Szegedy, 2015; Carlini & Wagner, 2017; Akhtar & Mian, 2018; Grosse, Trost, Mosbach, Backes & Klakow, 2019; Koerich, Esmaeilpour, Abdoli, Britto Jr. & Koerich, 2020). Adversarial examples are carefully perturbed input examples that can fool a machine learning model at test time (Biggio & Roli, 2018; Szegedy *et al.*, 2013), posing security and reliability concerns for such models. The threat of such attacks has been mainly addressed for computer vision tasks (Akhtar & Mian, 2018; Orekondy, Schiele & Fritz, 2019). For instance, Moosavi-Dezfooli *et al.* (2017) 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 of the input examples. For audio processing systems, notably end-to-end models, the effect of adversarial attacks is not widely addressed (Carlini & Wagner, 2018; Abdullah *et al.*, 2020). Creating attacks to threaten audio classification systems is challenging, due mainly to the signal variability in the time domain (Carlini & Wagner, 2018). This chapter presents an overview of such attacks on end-to-end audio classification models and challenges for generating such attacks.

1.1 Handcrafted audio features as inputs to deep learning models

Since the major research studies on CNNs are for image classification, many methods for audio classification are inspired by this general methodology. In this way, some 2D representations such as constant Q transform (Han, Kim, Lee, Han, Kim & Lee, 2017), Mel-frequency spectrograms (Salamon & Bello, 2017) or MFCC (Sigtia, Benetos & Dixon, 2016b) are extracted from the input signal and be used as the input to CNNs. In this case, the signal's representations may be considered as an image and be fed into the CNNs which is usually followed by several FC layers. As is mentioned, These FC layers are regular neural networks. The output of the FC layers is usually fed into the softmax classifier for classifying the audio signals to the appropriate classes. Due to the dimensionality reduction during extracting the low-level features, the network might be trained faster. The general methodology in the classification of audio signals based on the extracted low-level features as the input is depicted in Figure 1.1. In this section, several approaches for training neural networks in this fashion are reviewed and discussed. Two important handcrafted representations from audio signals are discussed.

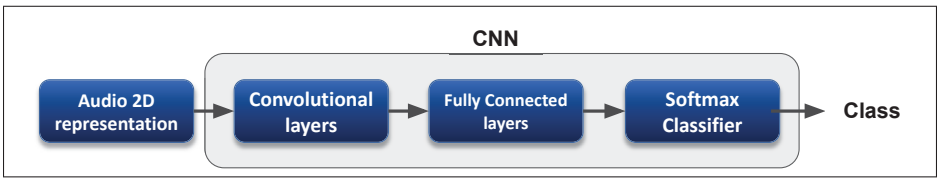


Figure 1.1 General audio feature based CNN for classification

1.1.1 Handcrafted features types

Several handcrafted features and representations can be extracted from the raw audio signal. This section discusses two important low-level audio features, such as spectrogram, a 2D time/frequency representation, and MFCC, which is one of the important low-level audio features. The next chapter reviews current methodologies for using such features and representations for audio processing.

1.1.1.1 Spectrograms

One of the popular representations of the audio signals widely used is the spectrogram. Spectrograms can be considered 2D representations that characterize frequencies present in the signal over time. The color intensity represents the strength and the power of each frequency component. A low-power component will have a light color, while a high-power component will be darker. The spectrogram can also be defined as an intensity plot of the magnitude of the Short-Time Fourier Transform (STFT). Several variations of spectrograms exist, and most of them can be used as input for CNNs. Generally, using this two-dimensional representation, the CNNs, which are mainly designed for image processing, can be leveraged for audio processing tasks. A typical spectrogram is shown in Figure 1.2 . The spectrogram is extracted from the sound file of a dog barking. The duration of the signal is around 3 seconds, and the frequency range of 0 to 5 kHz is covered.

1.1.1.2 Mel Frequency Cepstral Coefficients (MFCCs)

The most commonly used feature set in speech and audio processing is MFCC (Jurafsky & Martin, 2014). The MFCC can be considered as the sequence of feature vectors. Each vector represents the information in a small time window of the input signal. Combining these feature vectors along the time axis generates a 2D representation from the input signal. This section briefly describes and discusses the extraction of MFCC features from audio signals.

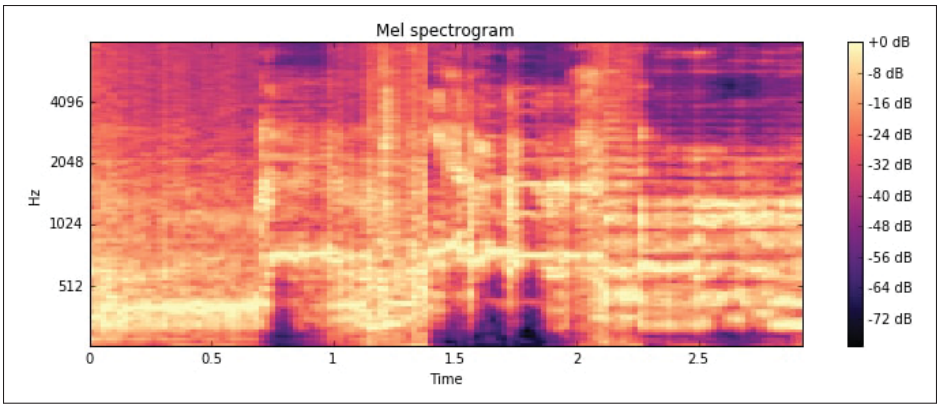


Figure 1.2 Typical spectrogram of the audio file of a dog's bark

The steps of MFCC extraction is shown in Figure 1.3. The first step is pre-emphasis. In this stage, a temporal filter is used to boost the energy level in high frequencies since, in speech signals, the amount of energy in lower frequencies is higher than in high-frequencies. This drop in energy across different frequencies is considered as spectral tilt. Boosting the high-frequency energy makes information from the higher formants more available and improves the accuracy level of detection. This process is done by applying a high pass filter.

The next step in the process is windowing. The signal is divided into small parts before being processed. The length of the windows has to be long enough to contain relevant information but also short enough to be considered stationary. The typical window size is 25 ms. The window is then moved to 10 ms, creating an overlap of 15 ms. This overlap ensures that transitions between sounds type are not missed. One way of windowing is to use a rectangular window. In this case, the signal could be simply cut into several frames. However, this type of window introduces discontinuities at the boundaries of the frames. In order to avoid this problem, a Hamming window function is applied, which shrinks the values of the signal toward zero at each end of the frame. This function attenuates the signal at the window's beginning and ending, avoiding an abrupt drop in the signal.

The next stage is the computation of the Discrete Fourier Transform (DFT). This step transforms the signal from the time domain to the frequency domain. The input of the DFT is the windowed

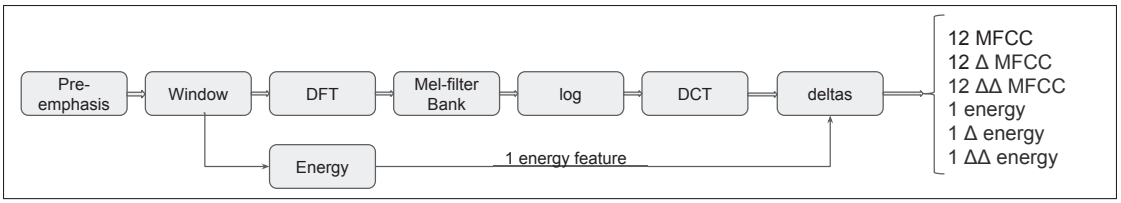


Figure 1.3 MFCC extraction steps Adapted from Jurafsky & Martin (2014)

signal, and the output is a complex number consisting of the phase and the magnitude of each frequency component of the signal. The magnitude is the energy of every frequency component in the signal. So, for example, if a signal is composed of sinusoidal, one at 5 kHz and another at 10 kHz, the representation in the frequency domain will show energy at both 5 kHz and 10 kHz.

The next step consists of applying Mel filters. Human hearing is not equally sensitive in every frequency band, and it is less sensitive at higher frequencies (roughly above 1 kHz). The Mel filters are designed to take this property into account, and it is proved that modeling this property improves the speech recognition performance (Jurafsky & Martin, 2014). In this stage, the frequencies output by the DFT step are transformed from the Hertz scale to the Mel scale. For example, the frequency in Hertz scale, fr, could be converted to the Mel scale as:

$$Mel(fr) = 1,127 \cdot ln(1 + \frac{fr}{700}) \tag{1.1}$$

This transformation is implemented by using a filter-bank consisting of triangular filters that are spaced along with the frequency range following the Mel-scale. This filter bank is applied to the signal frame's spectrum generated by the DFT stage. Each magnitude coefficient belonging to a filter is scaled according to its corresponding filter gain. The results are accumulated to create a vector of numbers. The vector size is equivalent to the number of filters in the filter bank. Figure 1.4 shows a set of these filters, which covers the frequency range from 0 Hz to 4 kHz. Each filter in the filter-bank is a triangular filter having a response of one at the center frequency and decreasing linearly towards zero (Fayek, 2016). The logarithm of filter-bank values is then computed to imitate human senses, which also have logarithmic behavior.

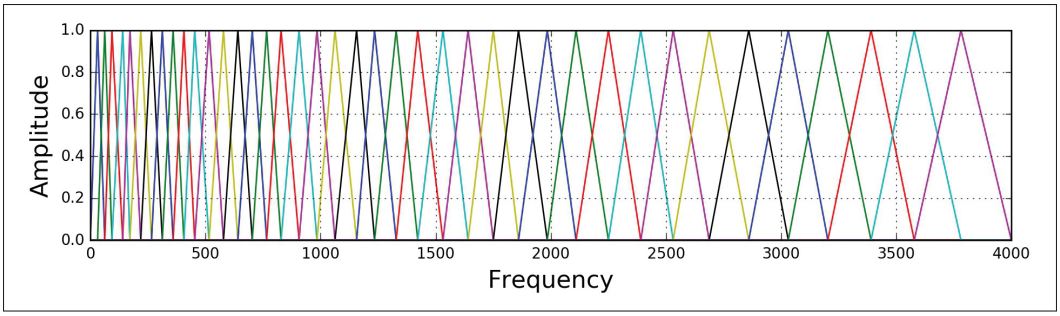


Figure 1.4 MFCC filter-bank Adapted from Fayek (2016)

The next step is to compute the cepstrum. Generally speaking, the speech waveform is generated for the voiced phonemes when a glottal source waveform of a specific fundamental frequency is passed through the vocal tract, which has some filtering characteristics. For speech processing, the characteristics of the source (such as fundamental frequency, the details of glottal pulse *etc.*) are not important, and the most useful information for phone detection is the filter, such as the exact position of the vocal tract. Knowing the shape of the vocal tract is crucial in phone detection. The cepstrum is one way to separate the source and the filter and have only the vocal tract filter. The cepstrum is computed using Discrete Cosine Transform (DCT). Consider $fb_{n_{fb}}$, $n_{fb} = 1 \ldots N_{fb}$ as the output of each filter of the filter-bank (filter channels). The MFCCs are then calculated from the log filter-bank amplitude by applying a DCT as described by the following formula:

$$ct_\rho = \sqrt{\frac{2}{N_{fb}}} \sum_{n_{fb}=1}^{N_{fb}} fb_{n_{fb}} \cos(\frac{\pi\rho}{N_{fb}}(n_{fb} - 0.5)), \rho = 1..12, \tag{1.2}$$

where ρ is each coefficient of the DCT function. For Speech Recognition (SR) only 12 coefficients are usually kept where these 12 coefficients represent information solely about the vocal tract.

The energy from the frame is added as the thirteenth feature. There is a correlation between energy and the phone identity, so it is useful for phone detection (Jurafsky & Martin, 2014).

The energy is computed by taking the sum over time of the power of the samples in the frame. Another fact is that the speech signal is not constant from frame to frame. So these changes between the frames have useful information for phone detection. Features related to these changes are added by computing the delta (velocity) feature and double delta (acceleration) feature of the 13 features (12 cepstral features and energy feature). The delta features represent the changes between the cepstral/energy features, and the double deltas also represent the changes between the delta features. The deltas are simply computed by computing the difference between the frames (Jurafsky & Martin, 2014).

In summary, after adding the energy, delta, and double delta vectors to the 12 cepstral features, the MFCC features consist of 39 features for each audio signal frame. MFCCs are widely used in speech processing, and they are also used as input for the deep neural network.

1.1.2 CNNs for high level representation learning from low level features

This section reviews several approaches based on CNNs for audio processing. Generally speaking, representations like spectrograms and MFCC features are used as a two-dimensional input to such models.

Sigtia *et al.* (2016b) presented a neural network model for polyphonic piano music transcription. They proposed an architecture similar to those used in SR. The acoustic model is a neural network used to estimate pitches' probabilities in an audio frame. They incorporate a language model, implemented with a recurrent neural network, for modeling the correlations between pitch combinations over time. The proposed model can be used for the transcription of polyphonic music. They reported 58.87% accuracy on the MAPS dataset, which consists of audio and appropriate corresponding annotations for individual sounds, chords, and complete pieces of piano music (Emiya, Badeau & David, 2009).

Li, Chan & Chun (2010) also used MFCC features as input to CNNs for music genre classification. In this study, MFCC features may be considered an image to be used as an input to CNNs. The CNNs are responsible for learning high-level representations of the MFCC matrix, which is

then used as input for a second classifier. The proposed architecture is shown in Figure 1.5. They reported 84% of accuracy on the GITZAN dataset (Tzanetakis & Cook, 2002) for music genre classification. The task of the network is to classify the audio samples into ten genres like (Classical, Jazz, Rock *etc.*).

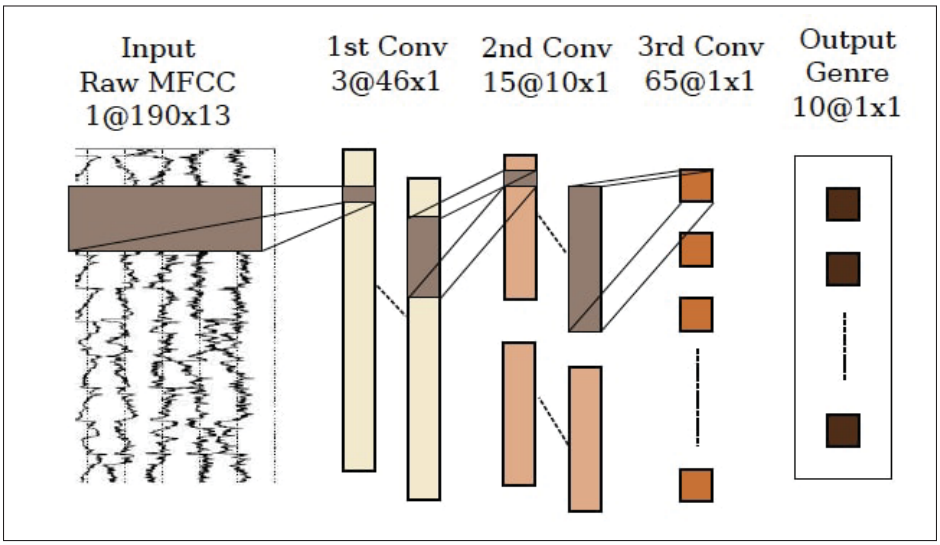


Figure 1.5 MFCC features as the input to the CNN
Taken from Fayek (2016)

Han *et al.* (2017) also used a CNN for identifying the predominant instrument in polyphonic music. In this study, they used spectrograms (using 128 channels) as the input of the CNN for representation learning. Different activation functions and different window sizes have been investigated. The best-reported score is 0.62 in terms of F-score. IRMAS (Bosch, Janer, Fuhrmann & Herrera, 2012) data set is used to train and test the network. The dataset contains polyphonic music clips, and the task is to detect the predominant instrument in each audio clip like cello, clarinet, flute *etc.* The proposed model uses four general blocks of two 3×3 convolution layers, one max-pooling layer with dropout layers in the middle of each block. The number of kernels for each block is 32, 64, and 128 filters. The output of these four blocks is flattened and then used as the input for two FC layers with 1024 and 11 (target classes) neurons with a dropout layer at the middle.

In another interesting study, Salamon, Jacoby & Bello (2014) used CNNs to classify organisms based on their vocalizations automatically. They built an automated c1assification system for clustering the migrating birds' sounds that the birds use for navigation guiding (flight calls). In this research, they explored a shallow learning approach with a CNN and used a data augmentation approach to increase the size of the training set. Mel-frequency spectrograms are also used as inputs for the system. They show that the two models perform comparably on a dataset of 5,428 flight calls which consist of 43 different species. Based on the collected dataset, they reported an accuracy of about 96%.

Salamon & Bello (2017) proposed a method for ESC (SB-CNN) based on the spectrogram representations of audio signals. They reported a mean accuracy of about 79%. The dataset used for these experiments consists of audio recordings of 8,732 environmental sounds (UrbanSound8K dataset). Each audio recording is divided into approximately 3-second frames, and the spectrogram representations of frames are extracted. The spectrogram patches are then used as the inputs to a CNN, and a pooling strategy is then applied. Finally, the prediction is made at the sample level, and the class with the highest mean output activation over all frames is selected. They also applied several data augmentation techniques such as adding background noise (from 4 different field recordings), dynamic range compression, pitch shifting, and time stretching. The used architecture is shown in Table 1.1.

Table 1.1 CNN based ESC model Adapted from Salamon & Bello (2017)

Layer	**Conv1**	**Pool1**	**Conv2**	**Pool2**	**Conv3**	**FC1**	**FC2**
# of Filters	24	-	48	-	48	64	10
Filter size	(5,5)	(4,2)	(5,5)	(4,2)	(5,5)	-	-
Activation	-	Relu	-	Relu	Relu	Relu	Softmax

The application of DNNs for large-scale audio processing tasks has also been studied in the literature. Hershey *et al.* (2017) used a dataset made of YouTube video clips. The dataset consists of 70 million videos totaling 5.24 million hours, each tagged from a set of 30,871 labels. This dataset is called YouTube-100M. The primary task is to predict the video labels

using the audio information of the video clips. Labels like "music", "speech", "bird", "siren" *etc.* are assigned to video clips. The authors investigated how popular DNN architectures can be used for video clip classification based on their audio track. The CNNs such as FC (3 layers with 1k units per each layer), AlexNet (Krizhevsky, Sutskever & Hinton, 2017), VGG (Simonyan & Zisserman, 2014), Inception V3 (Szegedy, Vanhoucke, Ioffe, Shlens & Wojna, 2016), ResNet-50 (He, Zhang, Ren & Sun, 2016) are used. The video clip's audio is transformed into spectrograms and then used as 2D representations to the inputs for training the deep models. The results of the evaluation of different networks, the number of steps in training, the duration of training of the networks, and also the obtained Area Under Curve (AUC) level of each network are summarized in Table 1.2. The authors used a different number of training videos as training sets. The best-reported result is obtained by using a data set that contains 70M videos. This study magnifies the importance of the number of samples for training CNNs. Figure 1.6 also shows the predicted labels by the best classifier. Sixteen classifier outputs with the greatest peaks are also shown. The bottom left box in each example shows the spectrogram of the audio signal corresponding to the video clip.

Table 1.2 Results of training with different amounts of data for large scale audio classification Adapted from Hershey *et al.* (2017)

Architecture	**Steps**	**Time (h)**	**AUC**
FC	5M	35	0.851
AlexNet	5M	82	0.894
VGG	5M	184	0.911
InceptionV3	5M	137	**0.918**
ResNet-50	5M	119	0.916
ResNet-50	5M	356	**0.926**

In a similar approach, Pons & Serra (2018) also used randomly weighted 2D CNNs (non-trained) for extracting features from audio spectrograms and raw audio samples for sound classification. Several experiments have been conducted to find the best architectures for this method. In the case of ESC, the best results have been obtained by using a VGG 2D CNN (Simonyan & Zisserman,

Table 1.3 ResNet50 for large scale audio classification Adapted from Hershey *et al.* (2017)

Trainig Videos	Accuracy
70M	0.923
7M	0.922
700K	0.921
70K	0.909
23K	0.686

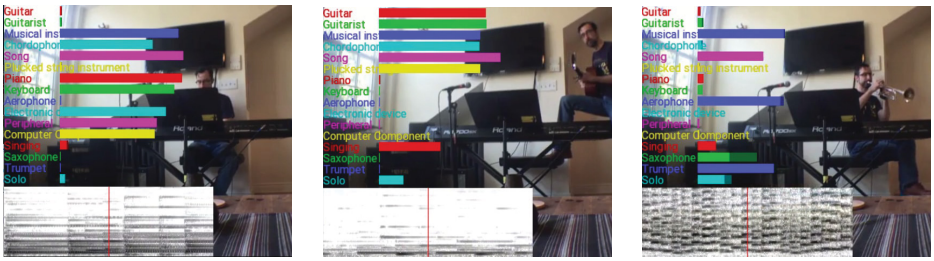


Figure 1.6 Audio based video classification. Three examples of excerpts from videos and the corresponding classes Taken from Hershey *et al.* (2017)

2014) as a feature extractor and SVMs as classifiers. They reported mean accuracy of 70% for this problem on the UrbanSound8k dataset (Salamon *et al.*, 2014).

Learned representations from 2D representations of audio signals using the models originally designed for image processing can also be used in self-supervised systems inspired by computer vision. Saeed, Grangier & Zeghidour (2021) proposed a general methodology for self-supervised representation learning from handcrafted representations from Mel filter-bank. EfficientNet-B0 (Tan & Le, 2019), which is a lightweight model that was originally proposed for computer vision, is used as an encoder to map the representations from the Mel filter-bank to a latent representation. After using a shallow network as a projection head, the bilinear similarity is used as a similarity measure between pairs of the representations of input samples. After that, the

contrastive loss is used to maximize the similarity between the representations of the inputs from the same audio clip, positive pairs, and minimize the similarity between the representations not from the same audio clip, negative pairs. After self-supervised training, the encoder can be used as a rich feature extractor where the weights of the model are fixed. It can also be fine-tuned and used for representation learning for other downstream tasks like birdsong detection, music instruments detection, language identification *etc.*. The embedding of the encoder is pre-trained on AudioSet (Gemmeke et al., 2017). The downstream evaluation is performed on various tasks, and the authors reported average accuracy of 85.1% on the downstream tasks.

It is also possible to use multiple types of audio features to feed several CNNs and then concatenate mid-level features and use them as input for a final CNN which will provide the final classification. An example of this approach is explained by Dieleman, Brakel & Schrauwen (2011). They used both timbral and chroma features as inputs to two independent CNNs. The Million Song Dataset (MSD) (Bertin-Mahieux, Ellis, Whitman & Lamere, 2011) is used to train the networks. The task is to perform artist recognition, genre recognition, and key detection. Chroma features describe the pitch content of the music. Each of the 12 components (the notes of one octave) corresponds to a pitch class (from note C to B). The timbre features are the coefficients of 12 basis functions that capture certain timbral characteristics like brightness, flatness, and attack. The architecture of such a network is depicted in Figure 1.7. The numbers on the sides indicate the size of the feature maps at each layer. The network starts with extracting information from the beats and continues to extract more high-level features from the bars. Considering learning rate as a hyperparameter, multiple learning rates of 0.005, 0.0005, $5 * 10^{-5}$, $5 * 10^{-6}$ and $5 * 10^{-7}$ for training the network for different tasks are used. For the genre recognition task, the best accuracy of 29.5% by the use of a learning rate of $5 * 10^{-6}$ is reported. The neural network achieved an accuracy of 35.74%, using a learning rate of 0.05, in the task of artist identification. For the task of key detection, the network obtained an accuracy of 86.53% using a learning rate of $5 * 10^{-5}$. Table 1.4 compares the performance of this architecture with the naive Bayes algorithm as the baseline. This shows that such architecture improves the baseline by 19.32 %, 28.94%, and 12.79% for genre classification, artist recognition, and key

detection, respectively. This shows that the network produces the best improvement with respect to baseline for artist recognition compared to other tasks.

Table 1.4 Results of using multiple of audio features as input to a CNN in accuracy level Adapted from Dieleman *et al.* (2011)

Task	**Baseline**	**Proposed architecture**
Genre Recognition	10.02%	29.52%
Artist recognition	6.80%	35.74%
Key detection	73.74%	86.53%

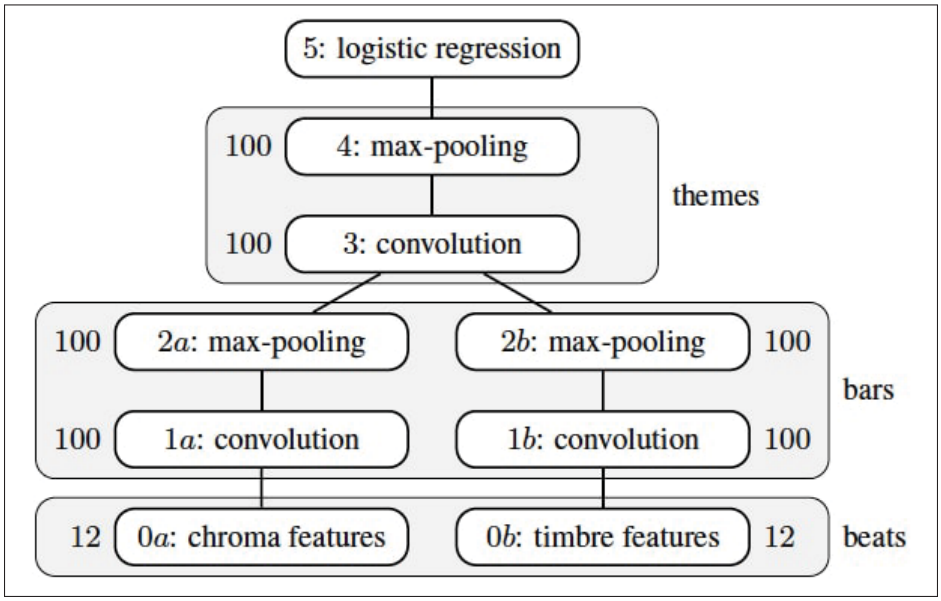


Figure 1.7 Using multiple type of low level features as the inputs for training the CNNs Taken from Dieleman *et al.* (2011)

Choi, Fazekas & Sandler (2016) proposed a content-based automatic music tagging algorithm using Fully Convolutional Networks (FCNs). Automatic tagging in music information retrieval is a multi-label classification task. In other words, an audio clip may be tagged with multiple tags, and an audio file may be the member of two classes simultaneously. It is different from other audio classification problems such as genre or artist classification, which are often formalized as a single-label classification problem. Different types of FCNs were evaluated. Table 1.5

shows the FCN-4 configuration which is a FCN with 4 convolutional layers. Mel-spectrograms are used as input to the network. The dimensionality of the input is $96 * 1,366$. Max pooling layers are also used after convolutional layers. The output is also an FC layer with 50 neurons for the appropriate classification. The block diagram of this network is also shown in Figure 1.8. Several other modifications of FCNs with 5, 6, and 7 layers (FCN-5, FCN-6, FCN-7) were also used and tested. Two datasets were used to evaluate the proposed architectures, the MagnaTagATune dataset (Law & Von Ahn, 2009) and MSD (Bertin-Mahieux *et al.*, 2011). Table 1.6 shows the accuracy level of different configurations by the use of the MagnaTagATune dataset as the input. The best result is produced by the FCN-4 and using the Mel-spectrogram as the input to the FCN. Table 1.7 also represents the accuracy level of different configurations by the use of the MSD dataset as the input. The FCN-6 produces the best result. In this case, the Mel-spectrogram also is used as the input to the FCN.

Table 1.5 FCN-4 deep neural network configuration
Adapted from Choi *et al.* (2016)

Layer	**Conv1**	**Pool1**	**Conv2**	**Pool2**	**Conv3**	**Pool3**	**Conv4**	**Pool4**	**FC**
# of filters	128	-	384	-	768	-	2048	-	-
Filter size	(3,3)	(2,4)	(3,3)	(4,5)	(3,3)	(3,8)	(3,3)	(4,8)	-
Output		(48,341,128)		(24,85,384)		(12,21,768)		(1,1,2048)	50

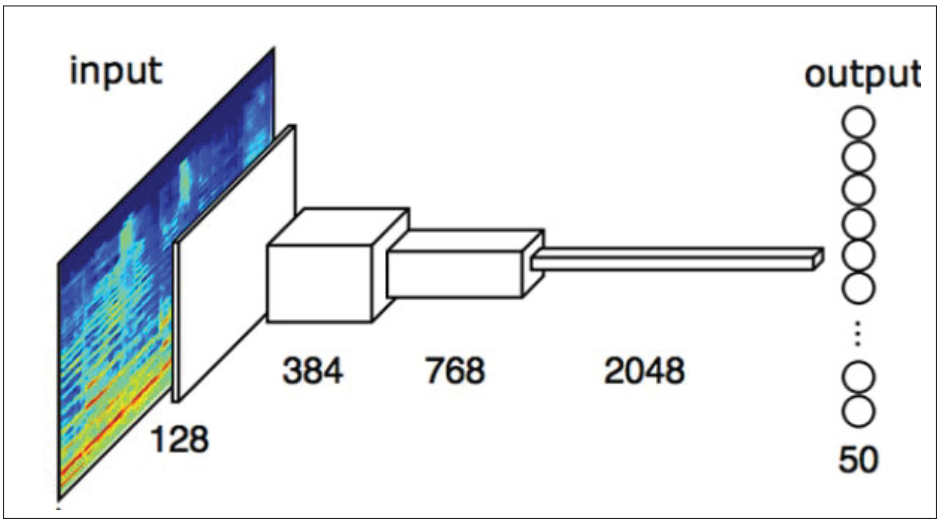


Figure 1.8 Block diagram of the proposed 4-layer
architecture for music tagging Taken from Choi *et al.* (2016)

Table 1.6 Results of the network based on FCNs on MagnaTagATune dataset Adapted from Choi *et al.* (2016)

Method	**AUC**
FCN-3 Mel-spectrogram	0.852
FCN-4 Mel-spectrogram	**0.894**
FCN-5 Mel-spectrogram	0.890
FCN-4 STFT	0.846
FCN-3 MFCC	0.862

Table 1.7 Results of the network based on FCNs on MSD dataset Adapted from Choi *et al.* (2016)

Method	**AUC**
FCN-3 Mel-spectrogram	0.852
FCN-4 Mel-spectrogram	**0.894**
FCN-5 Mel-spectrogram	0.890
FCN-4 STFT	0.846
FCN-3 MFCC	0.862

Figure 1.9 also shows the learning curves of the AUC scores on the validation set on the MSD dataset. As indicated by Choi *et al.* (2016) at the beginning of the training, the simpler and more shallow networks show better performance since there is a fewer number of parameters to learn. FCN-4 and FCN-5 also show similar performance between around 20–40 epochs.

In this section, some of the studies for using CNNs for high-level representation learning from handcrafted features of the audio signals were presented. The following section focuses on general methodologies for using CNNs in an end-to-end fashion for audio processing. Finally, the capability of such architectures in producing comparable results with other models that use representations from the audio signal is discussed.

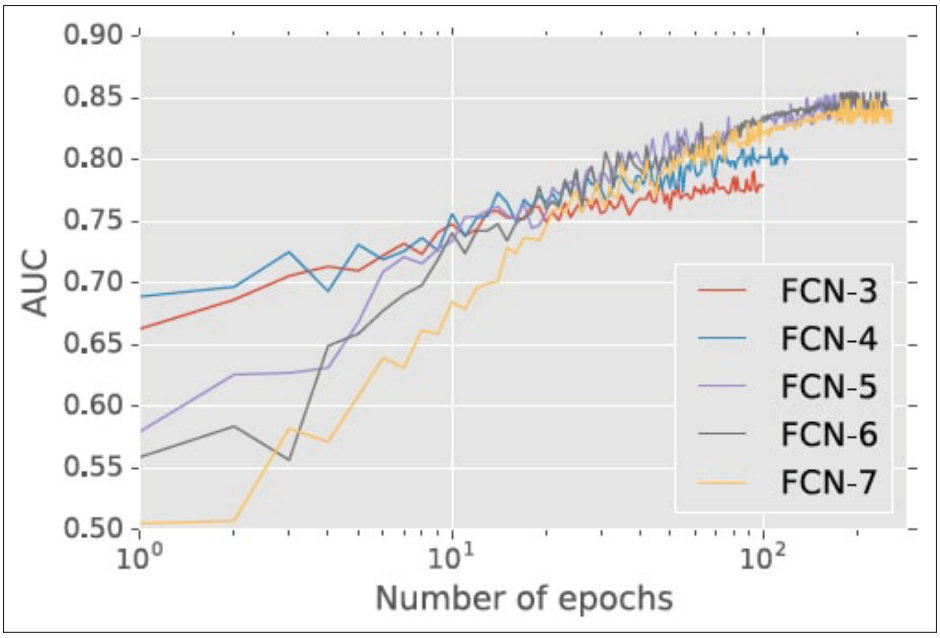


Figure 1.9 The learning curves of the AUC scored measured on validation set (MSD data set) Taken from Choi *et al.* (2016)

1.2 End-to-end approaches for audio processing

This section reviews different end-to-end audio processing approaches proposed in the literature. End-to-end systems are based on deep neural networks, and most of the proposed architectures follow this general scheme. Without any transformation, the raw audio signal is used as the input to the neural networks. The general architecture of such systems is depicted in Figure 1.10. The raw audio is first filtered using convolutional layers. These layers can be modified for a specific task. The representation learning part is followed by FC layers, which aim to transform the representations provided by convolutional layers into a linearly separable problem, which can be classified using the simple softmax layer. In this section, several end-to-end architectures are reviewed and discussed.

Zhu, Enge & Hannun (2016) proposed an end-to-end learning approach for speech recognition based on multiscale convolutions that learns the representation directly from audio waveforms. Three 1D convolutional layers with different kernel sizes are used for suitable representation learning. A pooling layer concatenates the representations to ensure a consistent sampling

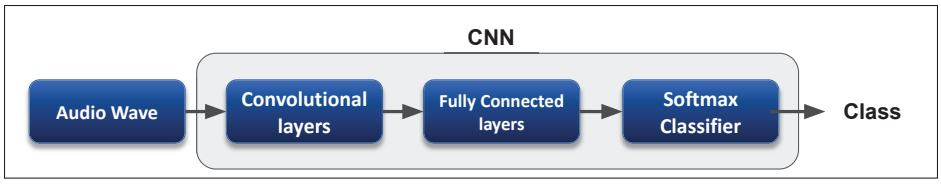


Figure 1.10 General end-to-end CNN for classification

frequency for the rest of the network. They reported 23.28% of word error rate on a dataset drawn from a collection of sources including reading, conversational, accented, and noisy speech (Amodei et al., 2016).

Learning parameters of filter-banks by the use CNNs is an interesting approach. In this regard, Ravanelli and Bengio (Ravanelli & Bengio, 2018) proposed the SincNet, an end-to-end approach for SID and verification. The first layer of such a model is based on parametric Sinc functions, which are band-pass filters. Only the filters' low and high cutoff frequencies are learned from the data. This model learns meaningful filters for the first layer and decreases the number of parameters of the model. This model achieves a Sentence Error Rate (SER) of 0.85% on TIMIT dataset (Garofolo, Lamel, Fisher, Fiscus & Pallett, 1993).

By the use of SincNet as filter-bank, Lavechin, Bousbib, Bredin, Dupoux & Cristia (2020) proposed a method for voice type classification for child-centered daylong recordings. In this research, a neural network is proposed to classify the audio segments recorded by child-wearing recording device into five classes such as the child wearing the device (key-child), all the vocalizations produced by other children in the environment, adult female speech, adult male speech and speech produced by any speaker. The end-to-end model is based on a SincNet filter-bank two recurrent networks followed by three FC layers. The model's output is an FC layer with five units and a Sigmoid activation function for classification. A dataset consisting of multiple child-centered corpora data is collected. The dataset is called Babytrain, and it has 20h of audio recordings. The F-measure of 57.3 is reported for a multi-class classification setup.

Hoshen *et al.* (2015) also proposed an end-to-end multichannel 1D CNN for speech recognition. They also found that the timing difference between channels indicates the location of the input

in space. They reported 27.1% of single-channel word error rate on a large vocabulary voice search dataset.

In this regard, Zeghidour *et al.* (2018) proposed an end-to-end 1D CNN architecture for speech recognition by learning a filter-bank which is considered as a replacement of Mel-filter-banks. Zeghidour, Teboul, Quitry & Tagliasacchi (2021) extended this work and proposed a Learnable Frontend for audio classification (LEAF). In this work, the authors demonstrated that a single learnable frontend could be trained to outperform Mel filter-banks by introducing an entirely learnable architecture to replace such filter-banks on various audio signals, including speech, music, audio events, and animal sounds. The model leverages normalized 1D-convolution with learnable Gabor filters. The filters are performed directly on raw input audio signals. Similar to SincNet (Ravanelli & Bengio, 2018) the parameters of the filter-banks such as center frequency and bandwidth are learned during the training process. After convolving the input signal with the Gabor filters, lowpass filters are used to downsample the output of the filter-bank to a lower sampling rate. The lowpass filters are specified for each filter of the filter-bank and are formed to have a Gaussian impulse response with learnable bandwidth. This lowpass filtering is used instead of a simple average or max-pooling since it has a systematic improvement over those pooling functions (Zeghidour *et al.*, 2018). After that, Per-Channel Energy Normalization (PCEN) with learnable parameters is used for compression. The method is evaluated on various tasks such as single task classification, multi-task classification, and multi-label classification on the Audioset (Gemmeke *et al.*, 2017). For single task classification, the model is evaluated on eight distinctive tasks like acoustic scenes detection, birdsong detection, emotion detection *etc.* For single task classification, The average accuracy of 76.9% is reported. For multi-task classification, the single backbone frontend model is trained along with task-specific heads for eight specific tasks as in single task classification. For multi-task classification, the average accuracy of 79.3% is reported. As the authors demonstrated, the LEAF frontend outperformed the Mel filter-banks over several tasks. The authors also used LEAF as the frontend to feed a CNN14 (Kong, Cao, Iqbal, Wang, Wang & Plumbley, 2020) model for multi-label classification. The authors reported 0.97 AUC on AudioSet for multi-label classification.

Similar end-to-end pipelines can also be used for healthcare purposes. Millet & Zeghidour (2019) proposed an end-to-end pipeline that jointly learns the filter-bank, the compression, the normalization, and the classifier for detecting dysarthria from raw audio speech signals. Trainable Gabor wavelets are used as filter-bank. PCEN is also used for compression. A recurrent model along with FC layers is used for dysarthria level detection. The average recall of 76.4% is reported on TORGO dataset (Rudzicz et al., 2008).

Image processing networks can also teach audio classification models in a teacher/student model. SoundNet is an end-to-end audio processing CNN proposed by Aytar, Vondrick & Torralba (2016) based on this methodology. The proposed network aims to take advantage of the natural synchronization between images and sounds to learn acoustic representation directed by unlabeled videos. They proposed a student-teacher training procedure that transfers discriminative visual knowledge (Teacher) into the sound modality using unlabeled video as an interface. In this case, the visual information of the input data is used as a guide for training the CNN, which is used for feature learning of audio signals.

Let the $\mathbf{x}_m \in \mathbb{R}^N$ be the waveform signal with N dimensions and $\mathbf{I}_m \in \mathbb{R}^{3\times T^{vid}\times W^{vid}\times H^{vid}}$ be its corresponding video for $1 \leq m \leq M$ where W^{vid}, H^{vid}, and T^{vid} are width, height and number of sampled frames in the video, respectively. During the training process, the goal is to use the probabilities from a teacher vision network $TC_\psi\left(\mathbf{I}_m\right)$ for training the student audio network $ST_\psi\left(\mathbf{x}_m\right)$. During the learning process, the following function needs to be optimized for the set of parameters of the student network, θ:

$$\min_{\theta} = \sum_{\psi=1}^{\Psi}\sum_{m=1}^{M} L_{KL}\left(TC_\psi\left(\mathbf{I}_m\right)||ST_\psi\left(\mathbf{x}_m;\theta\right)\right), \tag{1.3}$$

where $L_{KL}\left(.||.\right)$ is the KL divergence loss function and M is the number of samples in the training set. Ψ is also the concepts transferred from visual networks to audio networks. Both scene and object visual networks are used for concept transferring ($\Psi = 2$). This training protocol transfers semantic meaning from visual networks to the sound processing network. Weights

from a pre-trained neural network are also used for extracting visual information from video clips. Two widely used networks have been tested for this task: ImageNet CNN (Krizhevsky *et al.*, 2017) and Places CNN (Zhou, Lapedriza, Xiao, Torralba & Oliva, 2014). The best results are obtained when both networks extract visual information. The extracted features from the Soundnet are used as the inputs to an SVM classifier. For ESC, the accuracy of 88% on the DCASE database has been achieved (Stowell *et al.*, 2015). This result outperforms handcrafted features by 10% on this dataset. They also reported a 74% and 92% of accuracy on ESC-50 and ESC-10 (Piczak, 2015b) datasets, respectively. The SoundNet architecture is shown in Figure 1.11.

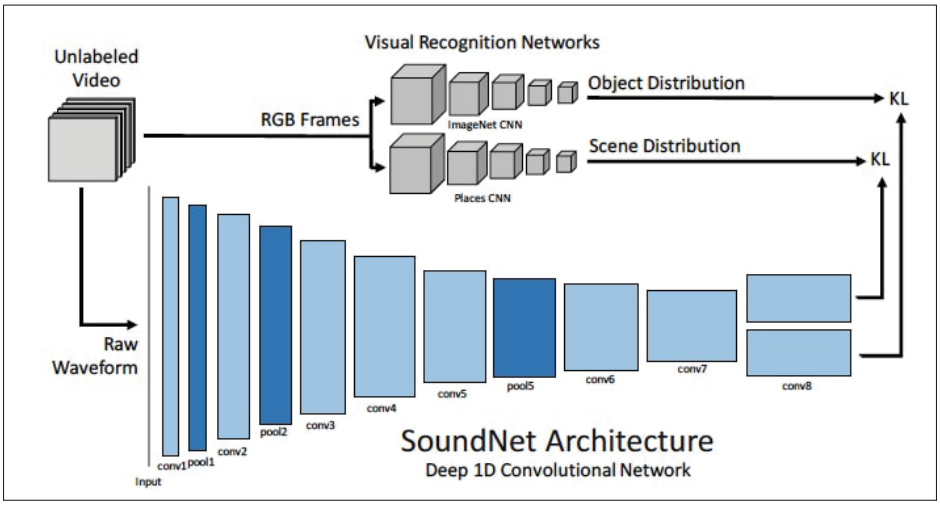


Figure 1.11 SoundNet Architecture, a deep convolutional architecture for natural sound recognition Taken from Aytar *et al.* (2016)

Dieleman & Schrauwen (2014) proposed a network with both spectrograms and raw audio files as inputs to a neural network trained for music classification. Spectrograms and raw audio signals are used for representation learning using one-dimensional convolutional filters. The architecture of such a network is depicted in Figure 1.12. Spectrograms with 128 components are used for this approach. Two different sub-networks are selected to extract high-level information from raw audio signals. One has only one strided convolution, and the other has a feature pooling on top of the strided convolution layer. The Magnatagatune dataset (Law & Von Ahn, 2009)

was used to train the network. The task is to classify the music tracks according to appropriate tags such as acoustic, loud, classic, string, drums, electronic, *etc.* The data set consists of 25,863, 29-second audio clips. The reported accuracies of 88.15%, 84.87% and 83.87% for the spectrogram fed and raw audio fed without feature pooling and raw audio fed with feature pooling networks, respectively. Some samples of the one-dimensional kernels at the lowest layer of the model that processes raw audio signals are also shown in Figure 1.13.

Dai, Dai, Qu, Li & Das (2016) presented several architectures for ESC (M3 Net, M5 Net, M18 Net, and M34-res net). The architectures are based on end-to-end CNNs, and the models consist of up to 34 layers. The idea of M3 net and M5 net is based on using large receptive fields. The models are fully convolutional, and no FC layers are used. Instead, a single global average pooling layer is used. The average of each feature map is taken, and the resulting vector is fed directly into the softmax layer. The model uses a large receptive field in the first convolutional layer to simulate band-pass filters but very small receptive fields for the other layers to control the model's capacity. To unify the shape of the input signal to the CNN, the audio files were resampled to 16 kHz. The input vector to the models has 32,000 elements. Batch normalization (Ioffe & Szegedy, 2015) is also applied to the output of each convolutional layer to prevent the model from overfitting. M5 net is the extended version of the M3 net. It has two more convolutional and pooling layers than the M3 net. The proposed M34-res architecture is based on residual learning, which effectively trains deep convolutional neural networks. Consider $\mathbf{x}$ as the input vector to the network. In regular network, the layer fits the mapping (transformation) of $F(\mathbf{x})$. In residual mapping, the original mapping is also recast to $F(\mathbf{x}) + \mathbf{x}$. The residual learning is achieved by specifying a skip connection in a residual block, as it is shown in Figure 1.14. M34-res net is based on this methodology, and it consists of three residual blocks. ReLu function is also used as the activation function. After each block, maxpooling is also applied.

Table 1.8 shows several different architectures proposed by Dai *et al.* (2016). The appropriate values of the hyperparameters of the networks are also presented in detail. The proposed networks are trained and tested on the UrbanSound8K (Salamon *et al.*, 2014) dataset. The

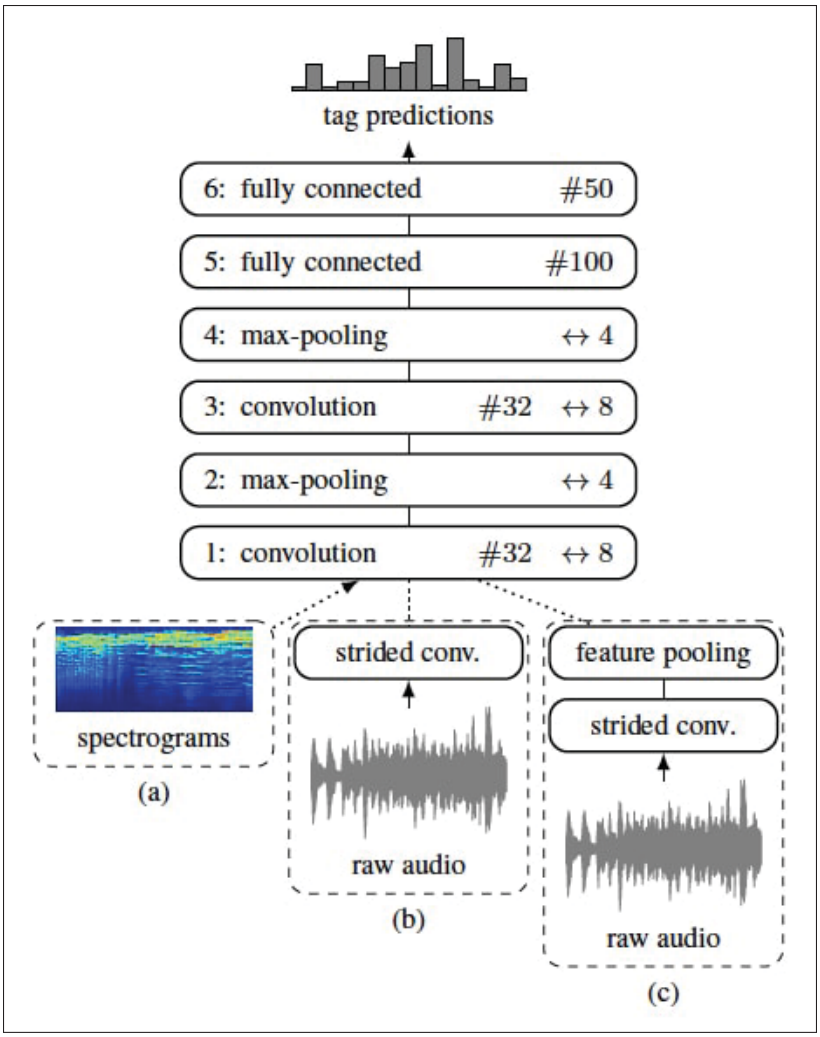


Figure 1.12 Spectrogram and raw audio fed CNN. The filter sized and number of filters are indicated by ↔ and #. Three approaches are considered: (a) spectrograms as input, (b) raw audio as input with additional strided convolutional layer, and (c) raw audio with feature pooling Taken from Dieleman & Schrauwen (2014)

proposed architectures produce comparable results to the state-of-the-art algorithms. The use of M18 architecture shows an accuracy of 71.68% on the test set.

Hertel, Phan & Mertins (2016) also presented an end-to-end architecture for audio event detection. The network is based on four convolutional layers followed by max-pooling layers. The network

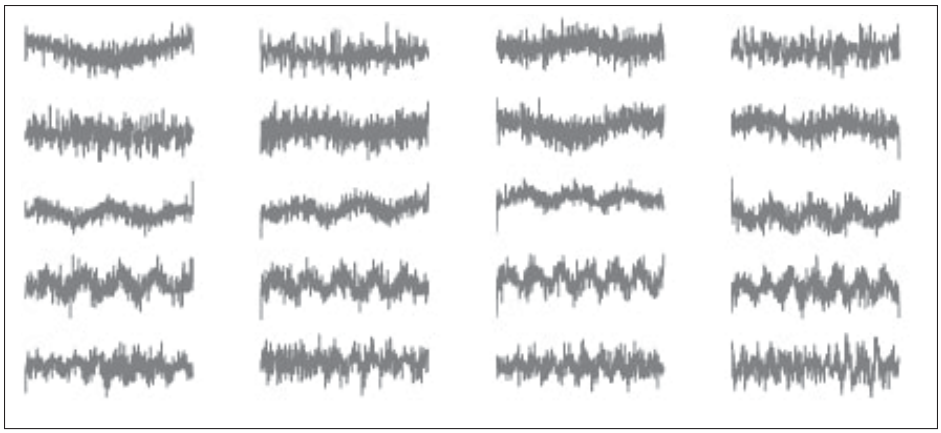


Figure 1.13 One dimensional Kernels after training the CNN that processes raw audio signals Taken from Dieleman & Schrauwen (2014)

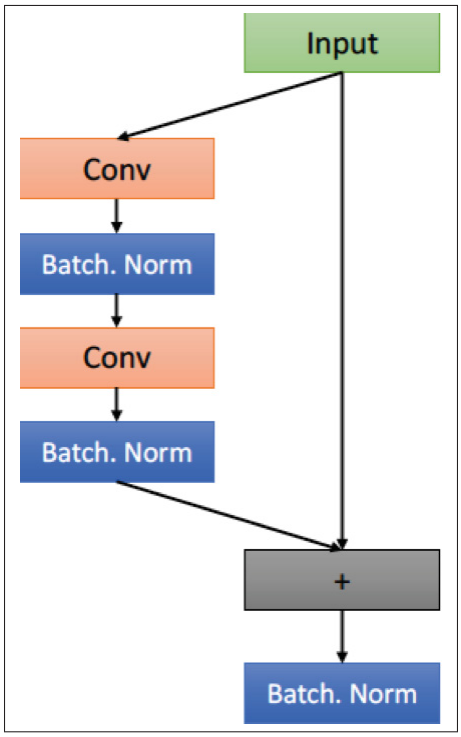


Figure 1.14 Residual block used in M34-res net

is originally trained on ESC-10 dataset (Piczak, 2015a) to classify the audio events such as *"Dog bark"*, *"Sea waves"*, *"Helicopter"* and so on. The accuracy of 83.7% is reported.

Table 1.8 Arcitecture of M3 Net, M5 Net, M18 Net M34-res net. (80/4,256) denotes a convolution layer with kernel size of 80 and 256 filters with stride of 4. Stride is omitted for stride 1 (e.g., (3, 256) has stride 1). For M34-res net the double layers in (..)*bl indicates the residual blocks and bl denotes the number of blocks Adapted from Dai *et al.* (2016)

<table>
<tr><th colspan="4">Architecture</th></tr>
<tr><th>M3</th><th>M5</th><th>M18</th><th>M34-res</th></tr>
<tr><td colspan="4">Input: 1D signal (32,000*1, Time domain waveform)</td></tr>
<tr><td>(80/4,256)</td><td>(80/4,128)</td><td>(80/4,64)</td><td>(80/4,48)</td></tr>
<tr><td colspan="4">Max-pool: 4*1</td></tr>
<tr><td>(3,256)</td><td>(3,128)</td><td>(3,64)*4</td><td>$\binom{3,48}{3,48} * 3$</td></tr>
<tr><td colspan="4">Max-pool: 4*1</td></tr>
<tr><td rowspan="5"></td><td>(3,256)</td><td>(3,128)*4</td><td>$\binom{3,96}{3,96} * 4$</td></tr>
<tr><td colspan="3">Max-pool: 4*1</td></tr>
<tr><td>(3,512)</td><td>(3,256)*4</td><td>$\binom{3,192}{3,192} * 6$</td></tr>
<tr><td colspan="3">Max-pool: 4*1</td></tr>
<tr><td></td><td>(3,512)*4</td><td>$\binom{3,384}{3,384} * 3$</td></tr>
<tr><td colspan="4">Global average pooling</td></tr>
<tr><td colspan="4">Softmax (No of units: 10)</td></tr>
</table>

Several studies have shown that if the raw audio is used as the input to a CNN, the network extracts some information like log-Mel filter-bank magnitudes. In this regard, Hoshen *et al.* (2015) used multichannel raw audio signals as the input to a CNN. They reported that the first convolutional layer of the proposed model naturally learns a filter-bank that is selective in frequency domain *i.e.* a bank of filters with an auditory-like frequency scale. They use a network with one convolution layer, a max-pooling layer, and a non-linearity layer. The output of the non-linearity is also used as the input to four FC layers. The task of the network is SR. The network is trained on the Voice search dataset, which has 400 hours of the training set and 36 hours of test set (Schalkwyk et al., 2010). For noisy inputs, single-channel input, and using raw audio as input, the authors reported 41.5% of Word Error Rate (WER). For a multichannel input, the error rate of 38.1% is reported.

Sainath *et al.* (2015) also observed a similar behavior by the filters of the first layer of CNN. They trained a similar CNN for SR on Google's voice search dataset. They test the network with a 20-hour test set. The output of the convolution layer is used as the input of a recurrent neural network and, finally, a deep neural network. They reported that if the network is trained on 40,000 hours of data, the WER of 15.5% can be achieved with raw audio files as input.

Tokozume & Harada (2017) also demonstrated a similar behavior from the first convolution layer of a CNN designed for ESC. The proposed CNN has two parts. The first part operates a one-dimensional convolution operation. After two convolution layers and a pooling layer, the pooling layer's output is reshaped and converted from $40 \times 1 \times 150$ to $1 \times 40 \times 150$ shape in channel× frequency×time format. After reshaping, two 2D pooling layers are operated. The output of pooling layers is used as the inputs to FC layers, followed by a Softmax layer. The model uses a new Between-Class (BC) learning method for training neural networks. The network, for which the input is a mixture of two audio samples, is trained to predict the mixing ratio of the samples. Their experiments show that BC learning has improved performance for various architectures used for sound identification tasks. They also proposed an end-to-end 1D CNN (EnvNet-v2) that performs well on various environmental sound datasets when trained with the BC learning approach, compared to conventional learning techniques. The best accuracy rates of 71% and 78% are reported on ESC-10 (Piczak, 2015a) and UrbanSound8k (Salamon *et al.*, 2014) datasets, respectively.

Transformer architecture recently produced state-of-the-art results in various machine learning tasks like language understanding (Devlin, Chang, Lee & Toutanova, 2018), image understanding (Dosovitskiy et al., 2020), music generation (Huang et al., 2018), video/language representation learning (Sun, Myers, Vondrick, Murphy & Schmid, 2019) and human action localization (Girdhar, Carreira, Doersch & Zisserman, 2019). Inspired by these advances, Verma & Berger (2021) proposed a transformer-based architecture for audio classification. The core idea of the model is to replace the usual CNN-based architecture with a purely transformer-based architecture. The architecture is based on using three modules of transformers. An average pooling layer follows each module. The input of the architecture is the signal of one second,

which is divided into 25ms patches. Each patch is fed into a feed-forward Fully Connected (FC) encoder. After that, a sinusoidal function is used to provide the positional encoding to feed the representations into the transformers. Finally, an FC dense layer is used to predict an appropriate label of the input signal. Such a model is shown in Figure 1.15. The FSD50K dataset (Fonseca, Favory, Pons, Font & Serra, 2020) is used to train the model. The dataset consists of 51,197 audio signals of 200 classes. The best Mean Average Precision (MAP) of 0.537 is reported on this dataset. The proposed architecture outperforms traditional convolutional architectures. The proposed architecture has 2.3 million parameters, so the large parameter space could be considered a downside of the methodology.

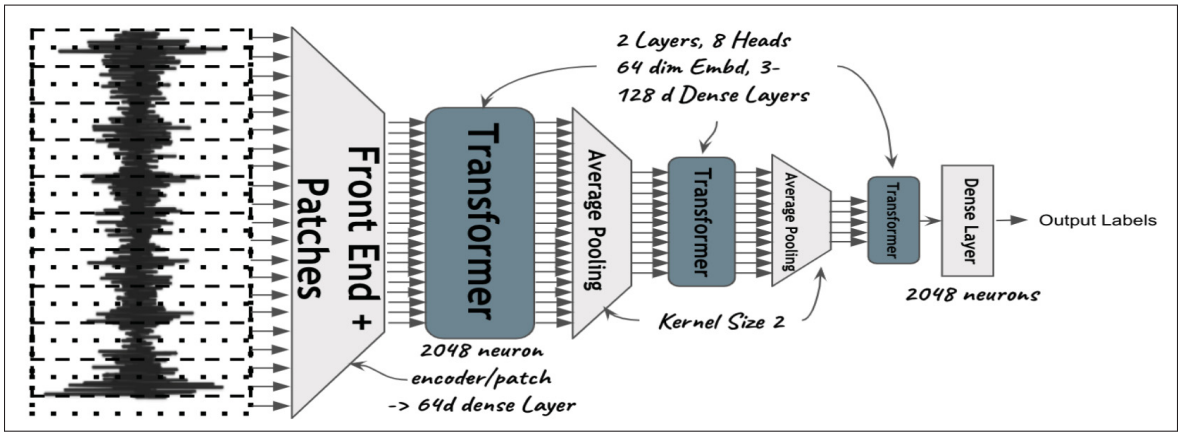


Figure 1.15 Transformer architecture for audio classification. The model takes audio input of one second. Three blocks of transformers with averegae pooling is used. Thanks to the transformers, no convolutional layers are used in this methodology Taken from Verma & Berger (2021)

The previous two sections of this chapter provided an overview of general methodologies on using CNNs for audio processing. Two major methodologies, such as end-to-end models and models for representation learning from audio representations, were discussed. The next section reviews the audio processing models from a security perspective. The main focus is attacking end-to-end models, which is aligned with the research question of this thesis.

1.3 Deep learning from security perspective

The main question of this thesis is assessing the possibility of designing end-to-end models for audio processing tasks. The ingredients of such a model are the raw audio signals as inputs, appropriate annotations of audio signals like classes *etc.*. An end-to-end model and a suitable training recipe. A deep end-to-end model in general consists of several layers of filters. All layers are based on differentiable functions to use a back-propagation algorithm to fit the filter parameters based on an appropriate loss function and optimization algorithm. In this case, all of the model layers are differentiable from the input signal to the final loss function. Models based on differentiable layers, model parameters, and training/test data availability are useful ingredients for a successful attack against such systems.

1.3.1 Adversarial machine learning

Recent studies have shown that deep learning models are quite vulnerable to some sort of adversarial attacks. There is an arms race in this field where the attacker constantly analyses the target machine learning model in order to generate effective adversarial attacks. On the other side of this game, the designer must be aware of such attacks against the models and develop suitable countermeasures. Biggio & Roli (2018) analyze the two strategies which the system designers might take. One of them is a *reactive* approach where the attacker and designer adapt their behavior in response to the opponent's activities. In this case, if the adversary generates new attacks, the system designer adapts the system's behavior by providing solutions as a reaction to the attack. The other approach which is more effective is a *proactive* strategy where the system designer acts as if she is in the attacker's shoes. In this case, the designer constantly evaluates the system's trustworthiness to predict the attacker's behavior and develop suitable countermeasures and defensive mechanisms. Figure 1.16 depicts the schematic overview of two such approaches.

The attacker's goals can be categorized in the following aspects (Biggio & Roli, 2018):

- **Security violation.** The attacker may choose to provide an *integrity* violation attack where the normal system operation is not compromised; *availability* violation where the usual

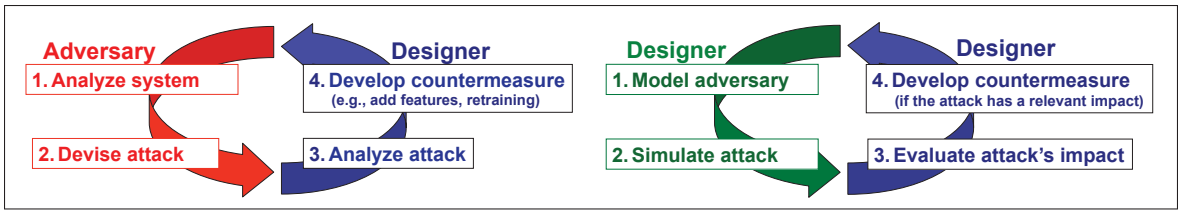


Figure 1.16 Strategies against adversarial attacks (left) *reactive* (right) *proactive* strategies Taken from Biggio & Roli (2018)

functionality of the system is violated; or *privacy* violation where the adversary tries to obtain some sensitive private information from the model, the users or data by reverse-engineering.

- **Attack specificity.** The intention of the attacker might be generating adversarial attacks that are effective for a specific group of data samples or all of the samples. As an instance targeting the information from specific people of the society like underrepresented groups, women or black community which is used as inputs to the machine learning model (like photos, voice audio samples, biometric data, *etc.*). In this case, the attacks are just effective for those specific people, *e.g.*, criminalizing and demonization of the members of the black community.
- **Error specificity.** The goal of the adversary might be *specific* (targeted) where the attacker aims to fool the model to provide specific output. For example, in the classification, task after attacking the model, the model should predict all of the samples as a specific class. The goal could also be *generic* (untargeted) where the goal is casing the model just to misclassify the input samples different from their legitimate class.

Generally speaking, the main components to generate such attacks are the trained model and its weights. If the adversary has access to such information, the process of adversarial attack generation will be pretty straightforward. First, an appropriate objective function should be defined, and then the function should be minimized by an optimization algorithm. The algorithm's output is an adversarial perturbation, and the input to such algorithms is the data and, in some cases, the labels of the input sample. A spectrum of information might be available for the attacker.

The attacker may have different levels of information about the target model such as, *training data*, *feature set*, *learning algorithm* as well as the *objective function* that is used for training the model and probably the *parameters* of the trained model. Biggio & Roli (2018) suggest the following scenarios in terms of the attacker's knowledge of the model that is under invasion:

- **Perfect-knowledge white-box attacks.** In this scenario, the attacker has access to all of the information about the targeted system, such as training data, feature set, learning algorithm, objective function, and model parameters. In this case, attacking the model. is quite straightforward, and it provides the worst-case invasion scenario on the model.
- **Limited-knowledge gray box attack.** In this case, a different setting in terms of the available knowledge for the attacker may be proposed. Roughly speaking, the attacker may have access to the feature set and learning algorithm (for example, the neural network architecture) but, the attacker may not have access to the training data or the model parameters. In this situation, the attacker may collect some *surrogate* data from the same source of the original training data (ideally from the same data distribution). The attacker may have some estimation of the original model's parameters by training the surrogate classifier. Another situation could be considered when the attacker has no information on the learning algorithm as well. In this case, the adversary may generate some successful attacks against the surrogate models, and hopefully, the attacks are effective in targeting the target model as well. In this case, we may say the attacks generated on the surrogate model are *transferable* to the target model as well. The effective attacks in this setting are so-called *model agnostic*. The transferability of the attacks is an interesting problem in this field.
- **Zero-knowledge black-box attacks.** In this form of attack, the model may be invaded without any substantial knowledge from the feature space, the learning algorithm, and the training data, and the model. In this case, the attacking algorithm provides some queries to the model and gets some feed-backs on the labels or the confidence scores (Papernot et al., 2017). The attacker may only have some ideas about the task of the model. The model's output is just used to guide some changes on the input data to the model or some modifications on the feature space *e.g.* some modifications on image pixels. This type of attack, to some extent, is similar to the gray box attack where the adversary may use surrogate

model(s), as well as surrogate data, to perform an *active learning* process. In this process, the transferability of attacks generated by the surrogate model can be tested on the actual target model. One may argue that one of the objectives of this form of attack is to limit the number of queries that the attacker may provide to the target model since querying the target model might be expensive or the access to the model is limited to the owner of the system. For example, some commercial models are accessible by using some APIs that may not be free. Moreover, since the attacker has no, or quite limited, information on the data samples of the target model, the process of attack generation could be time-consuming. Attacking the model within a reasonable time span is crucial in this context.

Table 1.9 shows a summary of different categories of adversarial attacks on machine learning systems based on the mentioned threat model in terms of attacker's goals and attacker's knowledge. In the next section, adversarial perturbations, one of the major threats to machine learning models, are discussed.

Table 1.9 Adversarial attacks taxonomy against machine learning models in terms of attacker's goals and attacker's knowledge
Adapted from Biggio & Roli (2018)

	Attacker's Goal		
Attacker's Capability	**Integrity**	**Availability**	**Privacy/Confidentiality**
Test data	Evasion (adversarial perturbations)	-	Model extraction or stealing
Training data	Poisioning (backdoor attacks, trojans)	Poisioning (to maximize classification error)	-

1.3.2 Adversarial perturbations

Adversarial perturbation is one of the major threats designed to cripple machine learning models. In this form of attack, a quasi-imperceptible perturbation is added to the input samples of the trained machine learning model in order to deceive the trained model into predicting the wrong class category output (Biggio & Roli, 2018). For a given example $\mathbf{x}$, a small perturbation is defined as δ, often imperceptible to a human observer, so that an example $\tilde{\mathbf{x}} = \mathbf{x}+\delta$ is misclassified

by a machine learning model (Szegedy *et al.*, 2013), (Peck, Roels, Goossens & Saeys, 2017), (Shafahi, Huang, Studer, Feizi & Goldstein, 2019). As it is mentioned by Szegedy *et al.* (2013), it is generally possible to find a similar input $\tilde{\mathbf{x}}$, which can fool the classifier to come up with the wrong decision. This section reviews some of the methods for generating such perturbations.

In the definition of adversarial attacks, a suitable distance should be defined to measure the quality of the examples and quantify the similarity between the legitimate sample and the perturbed sample. As it is mentioned by Carlini & Wagner (2017), three different measures in terms of ℓ_p norm are widely used. The ℓ_p distance between legitimate and perturbed sample is indicated as $\|\tilde{\mathbf{x}} - \mathbf{x}\|_p$ where the p-norm between two vectors of size N is defined as:

$$\|\tilde{\mathbf{x}} - \mathbf{x}\|_p = \left(\sum_{n=1}^{N} |\tilde{\mathbf{x}_n} - \mathbf{x}_n| \right)^{\frac{1}{p}} \tag{1.4}$$

In summary:

- ℓ_0 distance computes the number of the elements which are changed in the legitimate sample ($\mathbf{x}_n \neq \mathbf{x}_n$).
- ℓ_2 distance measures the Euclidean distance between two samples. One caveat is that the measure can be small when there are many small changes to many elements of the vector. Nevertheless, this measure is the backbone of generating many adversarial examples in the literature. Moreover, as will be discussed later in this thesis, we will use this measure also to generate adversarial perturbations against CNN-based models for audio processing.
- ℓ_∞ measures the maximum changes, which are applied to the samples to any of the components as:

  $$\|\tilde{\mathbf{x}} - \mathbf{x}\|_p = \max \left(|\tilde{x_1} - x_1| \dots |\tilde{x_n} - x_n| \right). \tag{1.5}$$

  This measure could be considered the maximum allowed budget to change the legitimate input sample to make it adversarial.

As it is emphasized by Carlini & Wagner (2017) no distance measure can measure the human perceptional difference between legitimate samples and adversarial ones. Instead, these measures are used to have a first rough quantitative assessment for a reasonable comparison between the attacks. Second, since these measures are differentiable, they may be used as one of the components of an optimization-based objective function. Such objective function in this formulation can be minimized by using gradient-based optimization algorithms like gradient descent.

Goodfellow, Shlens & Szegedy (2014) proposed a Fast Gradient Sign Method (FGSM) for generating adversarial examples which are optimized for the ℓ_2 distance metric. In this type of attack, the adversarial perturbation is found by executing the following operation for a single step:

$$\delta = \epsilon^{fgsm} \operatorname{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y_l)), \tag{1.6}$$

where ϵ^{fgsm} is a small constant to make the adversarial perturbation undetectable, J is the cost function which is used for training the neural network, $\mathbf{x}$ is the input, and y_l is the legitimate labels. This type of attack is designed to be fast, and it is not quite optimal (Carlini & Wagner, 2017). As shown in Figure 1.17, such perturbation can fool an image classification model, GoogLenet (Szegedy et al., 2015), to produce the wrong prediction with high confidence.

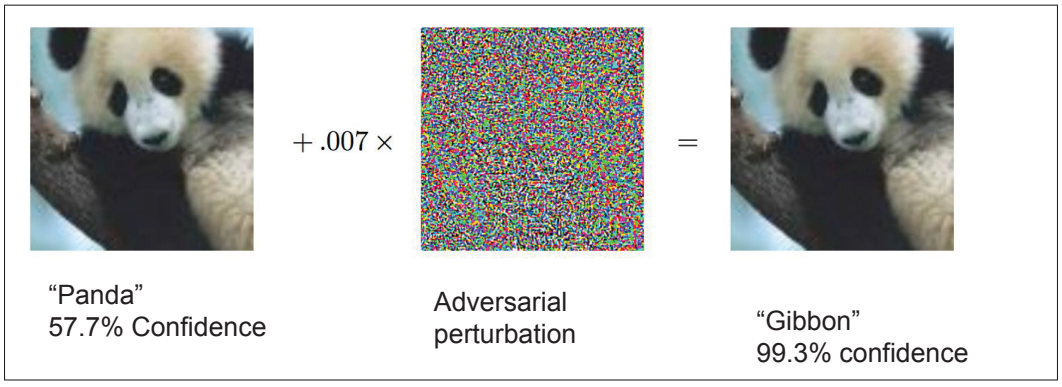


Figure 1.17 Adversarial example capable of fooling GoogLenet (Szegedy *et al.*, 2015). By introducing a small amount of perturbation, generated by FGSM, to the input image from ImageNet dataset (Krizhevsky *et al.*, 2017) , the model produces wrong prediction with high confidence score Adapted from Goodfellow *et al.* (2014)

Kurakin, Goodfellow & Bengio (2017) refined this attack where instead of taking one single step toward the gradient sign, multiple smaller steps are taken and the final result is clipped by ϵ^{fgsm}. This procedure starts with setting $\tilde{\mathbf{x}}_0 = 0$ and taking the following steps:

$$\tilde{\mathbf{x}}_i = \tilde{\mathbf{x}}_{i-1} + \text{clip}_{\epsilon^{fgsm}}(\eta^{fgsm}\,\text{sign}(\nabla_{\mathbf{x}} J(\theta, \tilde{\mathbf{x}}_{i-1}, y_l))), \tag{1.7}$$

Where η^{fgsm} is a small constant similar to the learning rate in order to take quite small steps toward the adversarial example.

Moosavi-Dezfooli, Fawzi & Frossard (2016) presented the DeepFool algorithm for finding untargeted adversarial perturbation optimized for ℓ_2 distance metric. This attack uses the assumption that neural networks are completely linear. The algorithm iteratively finds the adversarial perturbation that the perturbed sample would surpass the decision boundary under this simplified linear approximation.

The process of crafting an adversarial example for a targeted attack scenario can be formally defined as a constrained optimization problem such as:

$$\begin{aligned} \text{minimize} \quad & \text{dist}(\mathbf{x}, \mathbf{x} + \delta) \\ \text{s.t.} \quad & y_t = \arg\max_y \mathbb{P}\,(y|\mathbf{x} + \delta, \theta) \\ \text{and} \quad & 0 \leq \mathbf{x} + \delta \leq 1 \end{aligned} \tag{1.8}$$

where θ is the set of the neural network parameters and dist(.) can be considered as any function for measuring the distance between the legitimate sample and the adversarial example. The mentioned distance measures like ℓ_0, ℓ_2 or ℓ_∞ can be used. One of the objectives of this optimization problem is to minimize the distance between the legitimate and perturbed audio samples. The other objective is to generate adversarial samples and fool the network to predict the desired class in favor of the adversary. For untargeted attacks we use $y_l \neq \arg\max_y \mathbb{P}\,(y|\mathbf{x} + \delta, \theta)$, where y_l is the legitimate class. Such optimization problem is hard to

be solved as $y_t = \arg\max_y \mathbb{P}(y|\mathbf{x}+\delta, \theta)$ is a highly non-linear constraint. Another constraint of the optimization problem is to have an input sample, which is the summation of a legitimate audio sample and the adversarial perturbation, bounded in the range of $[0, 1]$. The appropriate algorithm must not generate samples out of this range.

Carlini & Wagner (2017) solved this problem by defining an appropriate unconstrained function to minimize. This formulation is based on a penalty optimization problem. In order to solve the problem of the non-linear function of the optimization problem defined in Equation 1.8, a linear function (hinge function) is defined. In order to ensure that the modification on the input image generates a valid image in the range of $[0, 1]$, a method to change variables is defined as:

$$\delta = \frac{1}{2}(\tanh(\mathbf{w}^{cw}) + 1) - \mathbf{x}. \tag{1.9}$$

Instead of optimizing over the variable δ, the optimization process is applied over $\mathbf{w}^{cw}$. Since $-1 \leq \tanh(\mathbf{w}^{cw}) \leq 1$, the perturbed sample is within the valid range, $0 \leq \mathbf{x}+\delta \leq 1$. Using ℓ_2 as the distance function, the Carlini and Wanger (C&W) ℓ_2 attack is defined as:

$$\text{minimize} \quad \|\frac{1}{2}(\tanh(\mathbf{w}^{cw}) + 1) - \mathbf{x}\|_2 + c.G(\frac{1}{2}(\tanh(\mathbf{w}^{cw}) + 1), t), \tag{1.10}$$

where G is the linear hinge function defined as:

$$G(\tilde{\mathbf{x}}, t) = \max\{\max_{j \neq t} \left\{f(\tilde{\mathbf{x}})_j\right\} - f(\tilde{\mathbf{x}})_t, -\kappa\} \tag{1.11}$$

where $f(\tilde{\mathbf{x}})_j$ is the output of the pre-softmax layer (logit) of the neural network for class j for perturbed sample, $\tilde{\mathbf{x}}$. Parameter c is a positive suitably chosen small constant known as *"penalty coefficient"*, t is the target class in targeted attacking scenario and, κ controls the confidence level of sample mis-classification. This parameter enables the attacker to control the confidence level of the attack so the instance will be misclassifed with high confidence. For the distance

measure, other measures like ℓ_∞ and ℓ_0 could also be used. Carlini & Wagner (2017) set $\kappa = 0$ in their experiments. This type of attack produces successful attacks on MNIST (LeCun, Bottou, Bengio & Haffner, 1998) and CIFAR-10 (Krizhevsky, Hinton et al., 2009) datasets with 100% success rate. The target models are CNNs with four convolutional layers followed by two FC layers and a softmax layer for classifying input images into 10 classes (Papernot, McDaniel, Wu, Jha & Swami, 2016) . Figure 1.18 shows several original and perturbed samples from MNIST (LeCun *et al.*, 1998) and CIFAR-10 (Krizhevsky *et al.*, 2009) for ℓ_2, ℓ_∞ and ℓ_0 attacks. Carlini & Wagner (2017) used Adam optimization algorithm (Kingma & Ba, 2014) to minimize such objective function.

This formulation is quite sensitive to the c parameter. If a small amount is chosen for this parameter, the algorithm may never find the optimal solution. On the other side, if a large amount is selected for this parameter, the algorithm may find successful perturbation vectors with a high budget. The algorithm possibly produces adversarial examples with a high distance value to the original samples (algorithm produces noisy adversarial samples). Figure 1.19 shows the sensitivity of this formulation on constant c.

Considering this issue, finding an optimal value for c is challenging. Carlini & Wagner (2017) proposed an *ad hoc* binary search over different values of parameter c in order to find the optimal value. This issue can be more problematic if there is a restriction on the number of queries sent to the target model or a restriction on the number of necessary steps to find the perturbation vector. Moreover, for each sample of the dataset, different values of the c parameter must be found (Rony et al., 2019).

In order to solve this issue, Rony *et al.* (2019) proposed an algorithm that a penalty on the ℓ_2 norm is not imposed during the training procedure. Instead, the perturbation vector is projected on a limited sphere around the original sample. After that, the algorithm uses a binary decision to modify the ℓ_2 norm. During this procedure, if the perturbed sample is not adversarial at step k, the norm is increased for the next step. Otherwise, it is decreased.

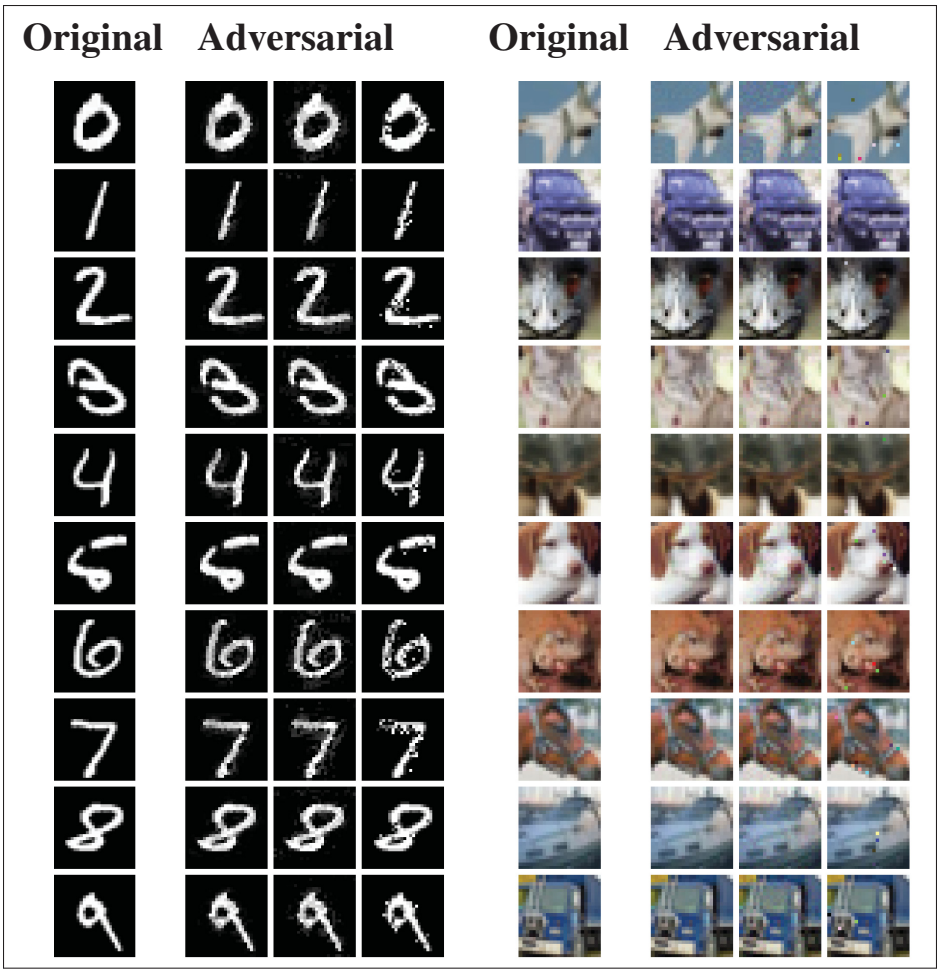


Figure 1.18 Adversarial examples from digit classification and object recognition datasets. The left most column depicts the original legitimate samples from the datasts. The next three columns shows corresponding perturbed samples based on ℓ_2, ℓ_∞ and ℓ_0 distance measures, respectively. The three misclassified instances share the same misclassified label Taken from Carlini & Wagner (2017)

Algorithm 1.1 and Figure 1.20 summarise the full procedure in finding an effective perturbation vector by the use of Decoupled Direction and Norm (DDN) attack method (Rony *et al.*, 2019). The algorithm starts with the legitimate input sample $\mathbf{x}$ and iteratively refines the perturbation vector. At each iteration, if the perturbed sample, $\tilde{\mathbf{x}}_k = \mathbf{x} + \delta_k$, is not adversarial the algorithm proposed a larger norm, $\epsilon_k^{ddn} \leftarrow (1 + \gamma^{att})\epsilon_{k-1}^{ddn}$. Otherwise, the algorithm considers a smaller norm. At each iteration, the algorithm takes a step according to the gradient of the loss function

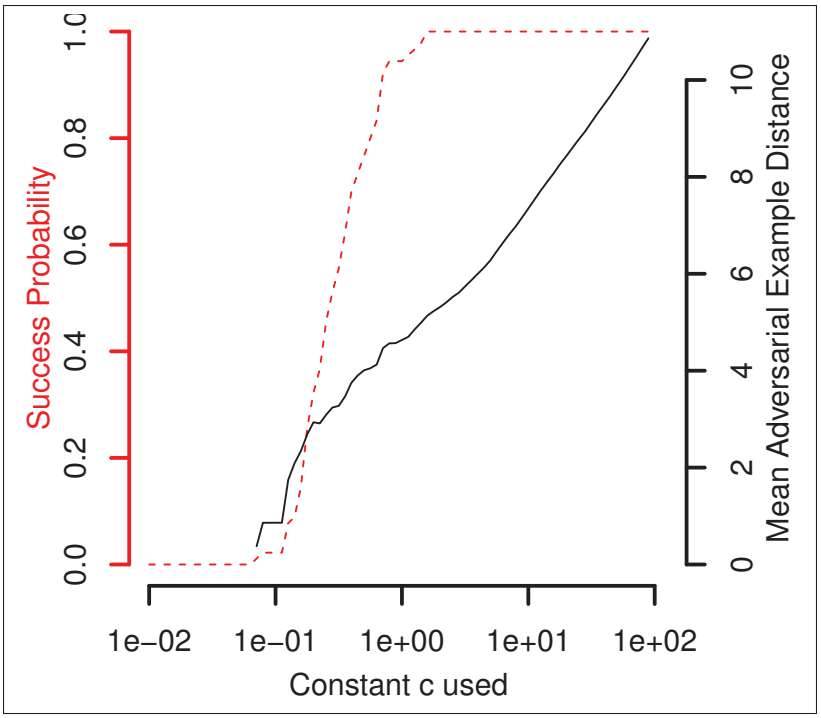


Figure 1.19 Sensitivity of C&W ℓ_2 attack to constant c. plot shows the effect of different values of c on the adversarial example distance and attack success probability. For $c < 1$ the attack rarely succeeds and for $c > 1$ the attack succeeds but with more amount of perturbation Taken from Carlini & Wagner (2017)

J that is used for training the model (red arrow of the Figure 1.20). Note that at each step of the algorithm, the result is also being projected back onto an ϵ_k^{ddn}-sphere around the original sample. At the end of the procedure, the algorithm makes sure that the perturbed sample is within the valid range of $[0, 1]$. This procedure makes sure that the perturbation vector puts the perturbed sample just outside of the decision boundary of the sample.

Rony *et al.* (2019) evaluated DDN attack for targeting several image classification datasets such as MNIST (LeCun *et al.*, 1998), CIFAR-10 (Krizhevsky *et al.*, 2009) and ImageNet (Krizhevsky *et al.*, 2017). One thousand samples from the datasets are selected for generating the attacks. The same model and parameters as defined by Carlini & Wagner (2017) are used for MNIST,

Algorithm 1.1 Decoupled Direction and Norm attack (DDN) Adapted from Rony *et al.* (2019)

Input: $\mathbf{x}$: original input sample to be attacked, y_l: true label, K^{ddn}: number of iterations, α^{ddn}: step size, γ^{ddn}: factor to modify norm at each iteration.

Output: $\tilde{\mathbf{x}}$: The adversarial input sample

Initialize $\delta_0 \leftarrow 0$, $\tilde{\mathbf{x}}_0 \leftarrow \mathbf{x}$, $\epsilon^{ddn} \leftarrow 1$

If targeted attack: $m^{ddn} \leftarrow -1$ else $m^{ddn} \leftarrow +1$

for $k \leftarrow 1$ *to* K^{ddn} **do**

  $g^{ddn} \leftarrow m \cdot \nabla_{\tilde{\mathbf{x}}_{k-1}} J(\tilde{\mathbf{x}}_{k-1}, y_l, \Theta)$

  $g^{ddn} \leftarrow \alpha^{ddn} \cdot \frac{g^{ddn}}{\|g^{ddn}\|_2}$. ▷ Step of size α^{ddn} in the direction of g^{ddn}

  $\delta_k \leftarrow \delta_{k-1} + g^{ddn}$

  if $\tilde{\mathbf{x}}_{k-1}$ *is adversarial* **then**

    $\epsilon_k^{ddn} \leftarrow (1 - \gamma^{att})\epsilon_{k-1}^{ddn}$ ▷ Decrease norm

  else

    $\epsilon_k^{ddn} \leftarrow (1 + \gamma^{att})\epsilon_{k-1}^{ddn}$ ▷ Increase norm.

  end if

  $\tilde{\mathbf{x}}_k \leftarrow \mathbf{x} + \epsilon_k^{ddn} \cdot \frac{\delta_k}{\|\delta_k\|_2}$ ▷ Project δ_k onto an ϵ_k^{ddn}-sphere around $\mathbf{x}$.

  $\tilde{\mathbf{x}}_k \leftarrow \text{clip}(\tilde{\mathbf{x}}_k, 0, 1)$ ▷ Ensure the perturbed sample is within the valid range.

end for

Return $\tilde{\mathbf{x}}_k$ that has the lowest ℓ_2 norm, $\|\tilde{\mathbf{x}}_k - \tilde{\mathbf{x}}\|_2$, and it is also adversarial.

CIFAR-10 datasets. For the ImageNet dataset, the Inception V3 (Szegedy *et al.*, 2016) model is used as the target model. For untargeted attacking scenario, Rony *et al.* (2019) compared the DDN attack to the state-of-the-art attacks like C&W ℓ_2 and DeepFool (Moosavi-Dezfooli *et al.*, 2016). The DDN attack generates successful attacks with a 100% success rate for all of the datasets with a mean ℓ_2 norm of 1.424, 0.148, and 0.361 for MNIST, CIFAR-10, and ImageNet datasets, respectively. As it is reported by Rony *et al.* (2019), DDN produces adversarial attacks comparable to other attacks in terms of ℓ_2 norm for MNIST and CIFAR-10 datasets. Nonetheless, on the ImageNet dataset, this attack produces better results in terms of ℓ_2 norm than the other attacks. DDN attack produces better results in terms of the number of times the attacking algorithm needs to compute the gradients compared to other attacking algorithms. For targeted

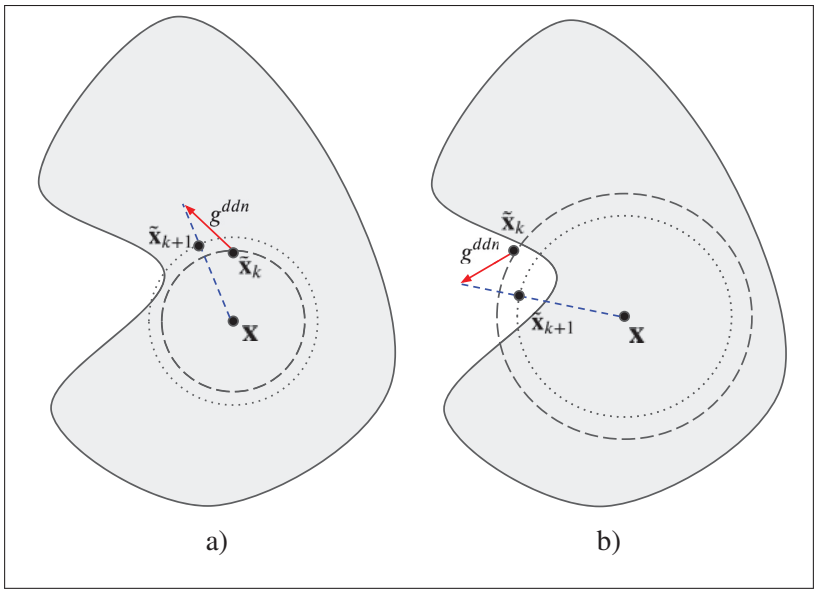


Figure 1.20 DDN untargeted attack illustration. The area that the sample $\mathbf{x}$ is classified as a true prediction is shown as the shaded area. In a) the perturbed sample $\tilde{\mathbf{x}}$ is not adversarial so, the norm, ϵ_{k+1}^{ddn}, is increased for the next iteration, otherwise it is decreased as in b). The algorithm takes a step g^{ddn} starting from the current sample $\tilde{\mathbf{x}}$. The magnitude of the step is bounded by an ϵ_k^{ddn}-sphere centered at $\mathbf{x}$ Adapted from Rony *et al.* (2019)

attacking scenarios, The DDN method also produces comparable results to the C&W ℓ_2 attack. Moreover, on the ImageNet dataset, the DDN attack obtains superior performance than C&W ℓ_2 attack. Rony *et al.* (2019) reported the Attack Success Rate (ASR) of 100% for all of the datasets. They reported a mean ℓ_2 norm of 1.951, 0.286, and 0.844 for MNIST, CIFAR-10, and ImageNet datasets, respectively. Roughly speaking, the DDN attack provides a useful trade-off between the amount of perturbation, ASR, and the number gradient computations. It is quite useful when our access to the target model is limited or querying the model is considerably expensive. Figure 1.21 also shows an example of perturbing a sample from ImageNet dataset (Krizhevsky *et al.*, 2017) for targeted attacking scenario. The target model is Inception V3 (Szegedy *et al.*, 2016) model. After adding the perturbation vector, the model misclassifies the sample as *"microwave"*.

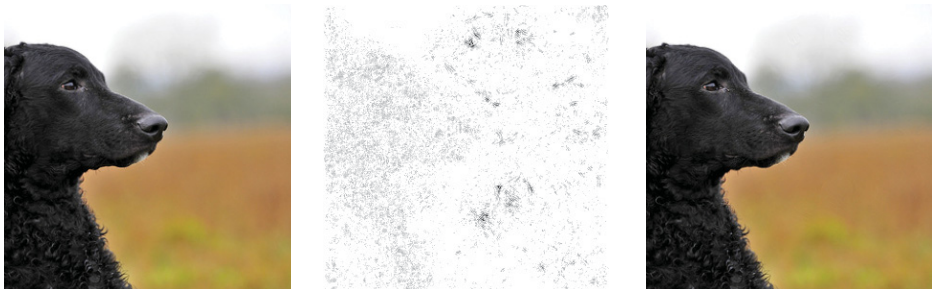


Figure 1.21 Example of adversarial image attacked by DDN algorithm. a) The legitimate sample $\mathbf{x}$, b) The perturbation vector δ, and c) the perturbed sample, $\tilde{\mathbf{x}} = \mathbf{x} + \delta$, misclassified as *"microwave"* Taken from Rony *et al.* (2019)

Current studies extensively evaluated the trustworthiness of image processing models against adversarial attacks (Biggio & Roli, 2018). This section provided a brief overview of popular attacking methods on such systems. However, this issue has not been widely addressed for audio processing models. In the following section, recent studies on the trustworthiness of audio processing models, especially end-to-end models against adversarial attacks, are reviewed and discussed.

Moosavi-Dezfooli *et al.* (2017) demonstrated the existence of a universal quasi-imperceptible adversarial perturbation that can fool a family of image classification models. In this form of attack, by adding a small UAP vector to input samples, the targeted model misclassifies the sample with high probability. The perturbation vector is considered universal since it is *image agnostic*, so it can be effective for almost all of the samples of the dataset. The algorithm proposed by (Moosavi-Dezfooli *et al.*, 2017) is designed for generating untargeted attacks. Figure 1.22 shows how such perturbation could be effective for targeting a typical image recognition model. In Chapter 3 this type of perturbation is discussed in detail, and the existence of this type of perturbation for targeting a family of audio classification models is shown. Two methodologies for crafting such perturbations for targeted and untargeted attacking scenarios are also discussed.

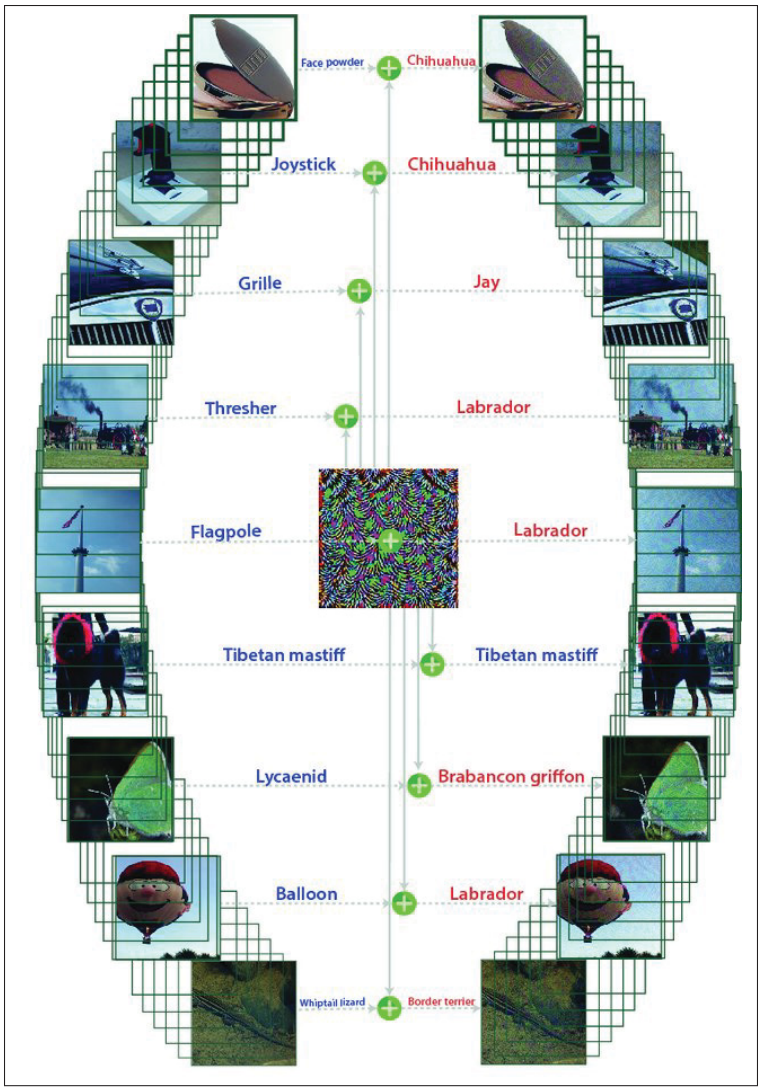


Figure 1.22 Universal adversarial perturbation when added to a natural image, the model misclassifies the sample with high probability. Left images are the legitimate samples. Central image is also an UAP vector. Right images are the perturbed samples. The estimated labels are also shown Taken from Moosavi-Dezfooli *et al.* (2017)

1.3.3 Adversarial machine learning for audio processing models

As we discussed in this chapter, deep neural networks have significantly improved the quality of audio processing and voice processing models. However, as we also mentioned, such deep

models are extremely vulnerable to adversarial attacks. The research community has mainly focused on demonstrating the threats such attacks may impose on image classification models. A few studies have recently proved the threats of such attacks on audio processing models based on deep neural networks. Recently, Abdullah *et al.* (2020) discussed the current studies on adversarial machine learning for audio processing models. They also taxonomized the audio processing threat model. This section reviews some recent studies on evasion attacks on audio processing models.

As we mentioned earlier in this chapter, the adversarial perturbation generation can be generally formulated by a constrained optimization problem as defined in Equation 1.8 and this problem can be solved by the use of an unconstrained objective function. This function can also be optimized by the use of an optimization algorithm (like C&W ℓ_2 attack). The same procedure for specific motivation can be applied for attacking audio processing models. For example, considering a speech transcription system, the attacker's motivation could be to perturb the input legitimate audio sample to fool the system into mistranscribing the audio sample. The input noise can be perceived as a quasi-imperceptible white noise to the human ear (Abdullah et al., 2019b). This procedure is depicted in Figure 1.23. The attacking algorithm generates the adversarial perturbation in an iterative procedure (Abdullah *et al.*, 2020). Note that the algorithm may be terminated as soon as an effective perturbation vector is found, or the algorithm may be executed for a specific predefined number of iterations. In this process, multiple numbers of effective perturbation vectors may be discovered. The algorithm may eventually select the perturbation vector that produces minimum distance to the original sample when it is added to the legitimate sample.

1.3.3.1 Challenges on attacking audio processing models

There are several challenges for generalizing the attacks designed for targeting image processing models to audio processing models. Abdullah *et al.* (2020) mentioned some issues, which should be taken into account in this regard:

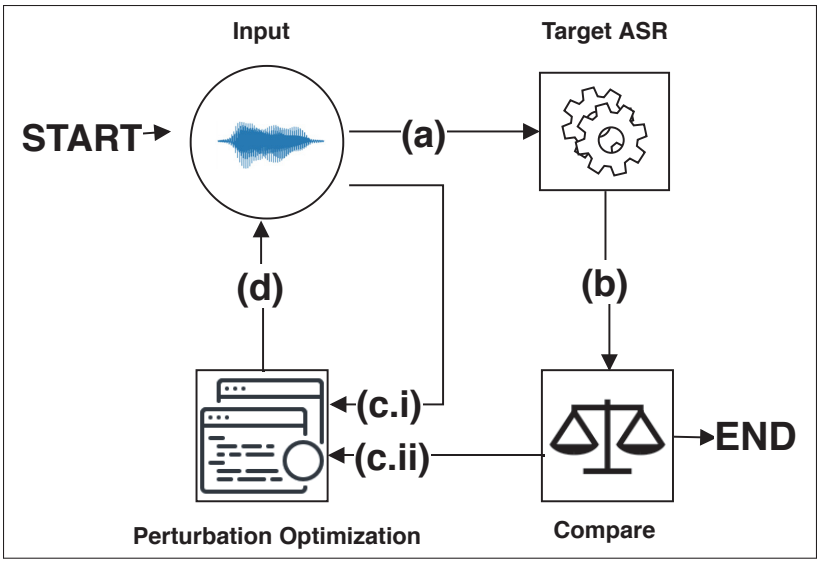


Figure 1.23 Iterative adversarial perturbation generation. The target model uses the input (a) and makes a prediction (b). If the produced prediction does not match the adversary expectation, the perturbed input (c.i) and the model's prediction are passed to the algorithm and it produces a new perturbation vector (c.ii) Taken from Abdullah *et al.* (2020)

- **Pre-processing modules:** As we discussed in detail in this chapter, most of the audio processing models work based on using handcrafted features of the audio signal as the input to the deep learning models (Refer to Figure 1.1). In contrast, image processing models based on deep architectures directly use the raw image as input. Therefore, most of the proposed attacks are optimization-based methods. However, suppose we want effective perturbations for targeting raw audio signals. In that case, we need to compute the gradient of the defined objective function with respect to the raw audio input. Considering MFCC generation, the process uses some functions that are not differentiable, so it is impossible to attack MFCC-fed models directly. This issue is considered *gradient obfuscation* (Athalye, Carlini & Wagner, 2018a), where some non-differentiable functions are used at some levels of audio processing, which prevent the adversary from executing the gradient-based algorithms. One way to circumvent such a problem is to simulate the non-differentiable function with similar differentiable counterparts. This challenge also enables a new type of

attacking method where the adversary uses some signal processing techniques to generate some noise-like attacks, which are recognized like legitimate samples by the target model (Abdullah *et al.*, 2019b). In this procedure, the adversary needs only to tweak some of the parameters of the signal processing modules, and the gradients of the target model are not required.

- **Sequential models:** In the case of SR pipelines, the structure of the models is different from image processing models. In this case, the temporal dependency between the components of the input audio signal is important. The speech processing models usually handle this dependency by the use of *recurrent* models where the connection between the neurons of the same layer are used. The temporal dimension is a challenge for generating successful attacks against such models. The adversary needs to consider the time steps as well the model needs to be unrolled. The process of unrolling causes some difficulties for generating optimization-based successful attacks like vanishing gradients and gradient explosion.

1.3.3.2 Taxonomy of adversarial attacks on audio processing models

For the evasion attacks on audio processing models, several goals for the attacker can be enumerated. The attacker's goals for different tasks in terms of *error specificity* can be categorized as *untargeted* and *targeted* scenarios as in general adversarial machine learning.

1. **Untargeted:** The goal is to fool the audio processing model to generate an output different from the original output.
   a. **SI:** The goal is to cause the model to misidentify the speaker.
   b. **SR:** The goal is to fool the model to transcribe the audio sample other than the original one for the perturbed audio input.
   c. **Audio classification:** In this mode, the attacker's goal is to deceive the model into misclassifying the perturbed audio sample other than the legitimate sample. For example, for ESC for the legitimate input audio clip from *"Dog bark"* class, if the model predicts the perturbed sample as any other class, the attack is successful.
2. **Targeted:** The goal is to fool the audio processing model to generate a *specific* output.

a. **SI:** The goal is to cause the model to classify any perturbed input sample as a specific speaker.
b. **SR:** The goal is to fool the model to only generate a specific transcription for any perturbed input sample.
c. **Audio classification:** In this mode, the attacker's goal is to deceive the model into classifying the perturbed input sample as a specific predefined class. For example, for ESC for all of the perturbed samples, if the model predicts the *"Gun shot"* class, which is the intention of the adversary, the attack is successful.

1.3.3.3 Types of audio adversarial attack generation

Two main methodologies for crafting adversarial perturbations may be considered: *Optimization-based attacks* and *signal processing-based attacks* (Abdullah *et al.*, 2020).

1. **Optimization-based attacks:**
   a. **Direct:** The attacker uses the model architectures and their weights to generate adversarial perturbations. As mentioned in this chapter, a suitable objective function is defined, and such a function is minimized based on an optimization algorithm, like gradient descent.
   b. **Indirect:** When the gradients of all or some parts of the model are not available to the attacker, some estimation methods can be used to have an understanding of the under-laying gradients. This can be done by using some surrogate functions to mimic the behavior of the original function (Athalye *et al.*, 2018a).
2. **Signal processing-based attacks:** These types of attacks use signal processing techniques to perturb the signal to generate adversarial samples. For example, perturbation techniques like *time domain inversion*, *random phase generation*, *high frequency addition* or *time scaling* can be used to manipulate the audio signal. The parameters of the perturbation techniques can be modified by querying the model multiple times. These perturbations are usually unintelligible to the human ear. These types of attacks are also less model-dependent,

and Abdullah *et al.* (2019b) generated successful attacks in the physical environment by the use of this methodology.

1.3.3.4 Adversarial capabilities

Several capabilities might be available for the attacker to target audio processing models. The attacker may also face several limitations for attack generation. Depending on the scenario, the adversary may produce several types of audio perturbations (Abdullah *et al.*, 2020):

- **Inaudible:** Human auditory system is able to detect the frequencies in the range of 20 Hz to 20 kHz. In contrast, microphones can detect to capture the frequencies beyond 20 kHz (Abdullah *et al.*, 2020). By using this difference, the attacker may generate some adversarial perturbations beyond the intelligible frequency by the human ear. However, the perturbed audio sample is processed by the audio processing model (Zhang et al., 2017).
- **Noise:** The attacker generates the perturbation that is similar to white noise to the human ear while it is considered as a legitimate sample and it is processed by the audio processing model.
- **Clean:** In this category, the audio perturbation is quite tiny so it causes a minimum amount of distortion to the audio sample. The human ear can not detect the added noise, and the human listener considers the perturbed audio sample a clean and intact sample. However, the model processes additive perturbation, and it can deceive it according to the attacker's intention.

1.3.3.5 Attack medium

The generated adversarial sample may be transferred to the target model through different mediums based on the attacking scenario. Each medium has its unique specifications, and there are specific challenges for transferring the adversarial audio sample through them. Abdullah *et al.* (2020) enumerated four main medium types for adversarial audio transferring including *Over-Line*, *Over-Air*, *Over-Telephony-Network* and *Over-Others*. Figure 1.24 shows a schematic view of different scenarios for transferring adversarial examples to the target model.

- **Over-Line:** The adversarial sample is transferred to the model directly through a waveform array or an audio file in `.wav` format. In this category, no compression of the audio signal is proposed. This is considered the easiest way to attack the audio processing model.
- **Over-Air:** In this type, the adversarial audio sample is played using a speaker, and a microphone records the sample. After that, the audio sample is transferred to the target model. A scenario for this type of attack is to target Amazon Alexa or Apple Siri, where the attacker plays the adversarial sample over the speaker. The challenge is that the adversarial sample needs to be robust against environmental background noise, audio reverberations, and also not efficient acoustic equipment (Qin, Carlini, Cottrell, Goodfellow & Raffel, 2019; Abdullah *et al.*, 2020). Moreover, the acoustic environment to test such attacks is also important. For example, the attack may be tested in an acoustic sound-proof room or examined in a real-world scenario with various background noise levels.
- **Over-Telephony-Network:** This type of attack involves transferring the adversarial sample through the telephony network in order to attack the audio processing model on the other side of the network. The network is a lossy environment because of problems like packet loss (Abdullah *et al.*, 2020). Crafting effective adversarial samples robust to the signal distortions in this harsh environment is challenging for for the attacker.
- **Over-Others:** This scenario includes mediums that do not fall into the other scenarios. As an instance, the audio file can be converted to different formats for example, `.wav` file can be converted to `.mp3` format. The `.mp3` format uses a compression technique to reduce the size of the file. The adversarial example must be robust against the file conversions and potential distortions associated with them.

Other issues like the maximum distance that the adversarial attack is effective, the necessary acoustic equipment, and the required acoustic environment are also crucial for evaluating the effectiveness and reproducibility of the adversarial perturbations. For example, adversarial attacks transferred through Over-Air and Over-Telephony-Network mediums might be effective for specific types of equipment such as microphones or speakers, which may not be transferable to other kinds of equipment (Abdullah *et al.*, 2020). Moreover, the proposed methods may be

effective for a specific room environment. For example, the environmental background noise may harm the performance of an adversarial attack. These issues are important for reporting any adversarial machine learning method effective in physical environments.

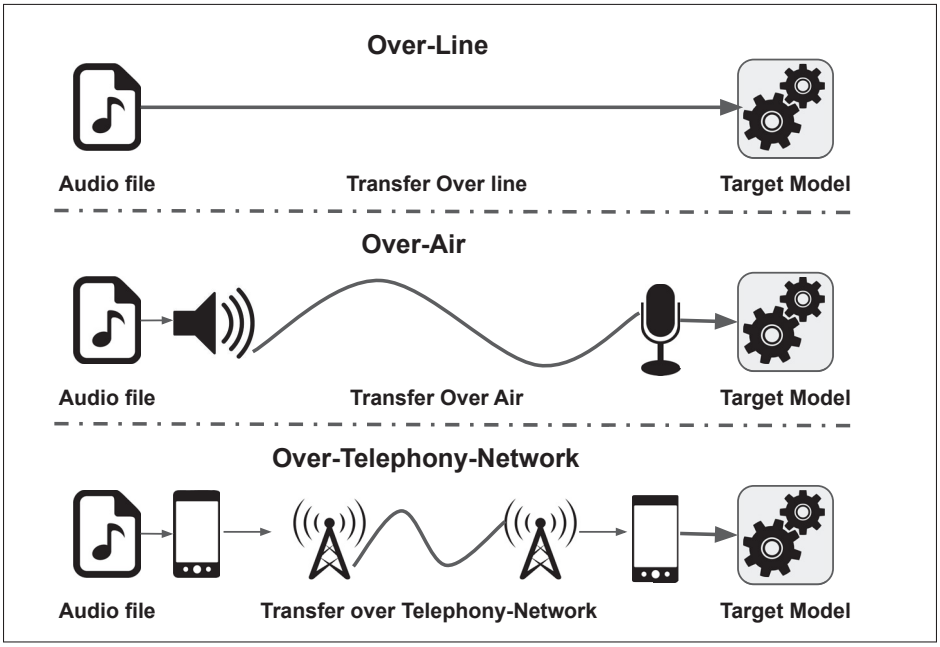


Figure 1.24 Audio adversarial sample transferring scenarios. Top) Over-Line attack where the audio file is passed to the model directly. Middle) Over-Air attack where the adversarial sample is played over the air through a speaker. Bottom) Over-Telephony-Network attack where the adversarial sample is transmitted over a Telephony-Network Adapted from Abdullah *et al.* (2020)

1.3.3.6 Audio adversarial attack quality assessment

Two measures are generally used to evaluate the quality of the perturbations on audio processing models. Signal to Noise Ratio (SNR) (Yakura & Sakuma, 2019; Kereliuk, Sturm & Larsen, 2015; Du, Ji, Li, Gu, Wang & Beyah, 2020) and the relative loudness of perturbation with respect to audio sample (Yang, Li, Chen & Song, 2019; Carlini & Wagner, 2018). Even though both measures are not quite accurate in assessing the audio intelligibility of the perturbed signal

by human ears. They may provide a rough estimation of the quality of the additive perturbation, which is introduced to the legitimate signal (Abdullah *et al.*, 2020).

SNR is measured in decibel (dB) and is used for measuring the level of the perturbation of the signal after adding the perturbation. It is defined as:

$$\mathrm{SNR}(\mathbf{x}, \delta) = 20 \log_{10} \frac{P^{sig}(\mathbf{x})}{P^{sig}(\delta)}, \tag{1.12}$$

where $P^{sig}(.)$ is the power of the signal measured by computing the the Root Mean Square (RMS) of the signal as:

$$P^{sig}(\mathbf{x}) = \sqrt{\frac{1}{N} \sum_{n=1}^{N} x_n^2}, \tag{1.13}$$

where x_n denotes the n-th component of the signal array $\mathbf{x}$. A high SNR indicates that a low noise level is added to the audio sample by the adversarial perturbation.

The relative loudness of perturbation with respect to the audio sample $l_{\mathrm{dB}_\mathbf{x}}(\delta)$ is also another perturbation quality assessment measured in *dB*, which is computed as:

$$l_{\mathrm{dB}_\mathbf{x}}(\delta) = l_{\mathrm{dB}}(\delta) - l_{\mathrm{dB}}(\mathbf{x}) \tag{1.14}$$

where $l_{\mathrm{dB}}(\mathbf{x}) = \max_n(20 \log_{10}(x_n))$ and x_n denotes the n-th component of the array $\mathbf{x}$. This measure is similar to ℓ_∞ norm in image domain. Lower $l_{\mathrm{dB}_\mathbf{x}}(\delta)$ values indicate quieter perturbations

1.3.3.7 White box attacks

Most of the attacking methodologies for targeting audio processing models are white-box attacks. As we mentioned earlier in this chapter, the adversary has all of the perfect knowledge from the target model in this attack, including the model architecture and the optimized weights, to provide a successful attack against the audio processing model. Carlini & Wagner (2018) demonstrated the existence of adversarial audio perturbations to fool SR systems. After adding

the perturbation to the input audio sample, the recognition system should mistranscribe the audio sample as it is shown schematically in Figure 1.25. The attacks are crafted in the targeted attacking scenario.

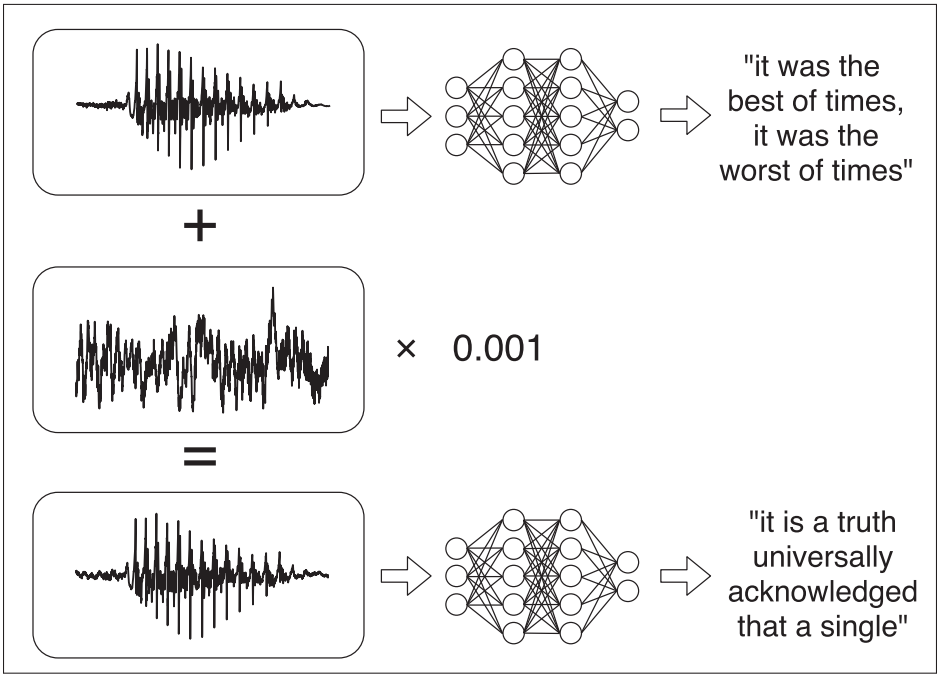


Figure 1.25 Audio adversarial attacks against speech-to-text system. After adding a small perturbation to the audio signal, the model should mis-transcribe the sample Taken from Carlini & Wagner (2018)

The methodology of crafting the perturbations is similar to C&W ell_2 attack for targeting image processing systems with few considerations to adapt the attacks to be used for targeting audio processing models. Let t be the target sequence of this attack for the input $\mathbf{x}$ can be defined as:

$$\begin{aligned} \text{minimize} \quad & \|\delta\|_2 + c \cdot J(\mathbf{x}+\delta, t) \\ \text{s.t.} \quad & l_{\text{dB}_\mathbf{x}}(\delta) \leq \tau_{att}, \end{aligned} \tag{1.15}$$

where J can be any suitable loss function for sequence-to-sequence matching that is used by the target model and the relative loudness of perturbation with respect to audio sample $l_{\text{dB}_\mathbf{x}}(\delta)$ computed by Equation 1.14.

The $l_{\mathrm{dB_x}}(\delta) \leq \tau_{att}$ constraint is to make sure that the magnitude of the perturbation does not exceed a specific threshold, τ_{att}. This constraint can be held by simply clipping the perturbation during the optimization procedure using a gradient descent algorithm. Lower values indicate quieter perturbations. This algorithm generates relatively quiet perturbation vectors. The algorithm can generate successful attacks against Mozilla's implementation (Mozilla, 2017) of DeepSpeech (Hannun et al., 2014) with a 100% success rate. The mean distortion level of -38 dB is reported. Figure 1.26 shows the legitimate audio sample and the perturbation vector overlaid. This sample shows that it is possible to generate successful attacks by adding a small perturbation.

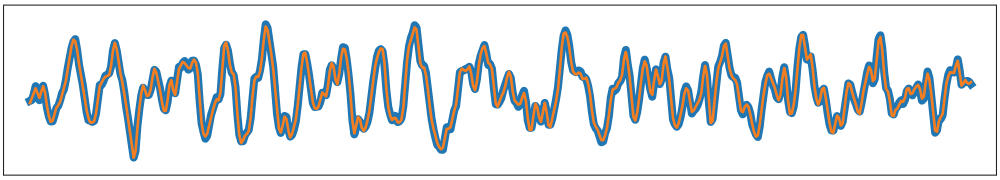


Figure 1.26 Original legitimate audio sample waveform (blue, thick line) with adversarial perturbation (orange, thin line) overlaid. A limited amount of perturbation can generate successful adversarial samples Taken from Carlini & Wagner (2018)

Generating adversarial attacks using *psychoacoustic hiding* is an interesting approach to put the perturbation below the thresholds of human perception (Schönherr, Kohls, Zeiler, Holz & Kolossa, 2018; Qin *et al.*, 2019). Qin *et al.* (2019) used a similar formulation to Carlini & Wagner (2018) method for generating targeted adversarial perturbations against SR system. The authors propose two main contributions. The first one is to reduce the noise level, which is distinguishable by the human listener. The second improvement is developing a framework to generate the adversarial attacks robust to room reverberations to generate effective attacks in physical world scenarios. The first contribution is made by leveraging the auditory masking principle. To achieve such a goal, the attack is added to specific parts of the audio where the audio signal masks the perturbation to hide it from the human listener hopefully. The attack uses the *frequency masking* to generate the attacks. The algorithm is encouraged to craft the perturbation under the audio signal's masking threshold, which is computed in the frequency domain. The perturbation's

normalized Power Spectral Density (PSD) is computed from the short-time Fourier transform of the perturbation vector. When adding the adversarial perturbation, if the PSD of the perturbation is under the frequency masking threshold of the original legitimate audio signal, the perturbation will be masked out by the legitimate audio signal and therefore be indistinguishable to humans (Refer to Qin *et al.* (2019) for more details). A high loss function is used to compute the difference between the perturbation's PSD and the masking threshold of the signal. The second contribution is making the adversarial sample robust to potential distortions by harsh physical environments. The major distortion to the audio signal is room reverberations.

Similar to the expectation over transformation (Athalye, Engstrom, Ilyas & Kwok, 2018b) methodology, several transformations are introduced to the perturbed audio signal by convolving it with various simulated room impulse responses (Allen & Berkley, 1979; Scheibler, Bezzam & Dokmanić, 2018). The transformed signal is used as the input to the loss function used to train the model. The Linear combination of this loss function and the hinge loss function used to compute the difference between the PSD of the perturbation and the masking threshold of the signal forms the final loss function to generate the adversarial perturbation. For evaluating the attacks, Librispeech (Panayotov, Chen, Povey & Khudanpur, 2015) is used to train the Lingvo model (Shen, et al., 2019). From the test set, 1,000 samples are randomly chosen and perturbed to evaluate the effectiveness of the attack. For targeted attacks, a 100% success rate is reported. Using some simulated test rooms, the targeted ASR of 49.65% is also reported for generating imperceptible and reverberation-robust attacks. [1]

Yakura & Sakuma (2019) also used expectation over transformation methodology to generate robust adversarial attacks against SR model effective in physical environments by simulating the room environment using room impulse response signals. The main contribution of the research is an effort to expand Carlini & Wagner (2018) attack to be effective in the physical environment. The DeepSpeech (Mozilla, 2017) is used as the target model. The method is tested to generate targeted adversarial attacks for three target phrases *"hello world"*, *"open the door"*, and *"ok*

[1] We encourage the interested reader to listen to some of the audio examples generated by the attack proposed by Qin *et al.* (2019) here:http://cseweb.ucsd.edu/~yaq007/imperceptible-robust-adv.html

google". These phrases are used since they are used as a trigger word of Google Home. For the input samples, two different music audio clips of four seconds cut from *"Cello Suite No. 1"* by Bach and *"To The Sky"* by Owl City were selected. The authors reported successful attacks by 100% success rate against the model. The attack can also generate perturbations with SNRs between 2.6 dB and 13 dB. One of the limitations of this attack is that it is only effective for short two or three-word phrases but not on the full sentence phrases. The other limitation is the high perturbation magnitude of the attacks.

Cisse, Adi, Neverova & Keshet (2017) proposed an algorithm for generating attacks against various target models designed for different tasks such as SR and image processing models. The method defines a target function: the multiplication of a novel loss, Houdini loss, and task loss. The Houdini loss computes the distance between the difference between the scores assigned by the network to the desired target and the model's prediction. The task loss is also the loss that is originally designed for the specific task. An analytical derivation of this target function is defined to be used in the optimization algorithm. This attack targets a deep learning model based on two convolutional layers and seven recurrent layers followed by one FC layer. The model is trained on Libispeech dataset (Panayotov *et al.*, 2015). For generating the attacks, 2,620 samples from the dataset are used. Average WERs between 46.5 to 96.1 are reported for the untargeted attacking scenario. In addition, average Character Error Rates (CERs) between 4.5 to 12 are also reported. Unfortunately, the method failed to generate satisfactory targeted attacks against the model.

Kreuk, Adi, Cisse & Keshet (2018) also used the FGSM attack for targeting a SID model. The effectiveness of the attack is evaluated on a model based on recurrent layers and a component for computing the cosine similarity between the embeddings. A binary classifier also produces the membership degree between the unknown speaker, the test sample, and the other enrollment utterances from several known speakers. The attacks are evaluated for untargeted attacking scenarios based on utterances from YOHO (Campbell, 1995) and NTIMIT (Jankowski, Kalyanswamy, Basson & Spitz, 1990) datasets. First, the adversarial perturbation is added to the acoustic features, MFCC, and spectrograms, of the test sample, and then the reconstructed

waveform is used to attack the model. On the YOHO dataset, after adding the perturbation, the model's accuracy dropped by 48.00% and 61.75% for Mel-spectrum, and MFCC features as inputs, respectively. For targeting the samples from the NTIMIT dataset, after adding the perturbation, the model's accuracy also dropped by 59.86% and 69.94% for Mel-spectrum and MFCC features as inputs, respectively.

As mentioned in this chapter generating UAP is one of the interesting problems in adversarial machine learning. Recently, Neekhara et al. (2019) demonstrated the existence of perturbations that fool SR systems to mistranscribe the audio signals. As in attacking image processing models, an additive single perturbation vector must be crafted to make most of the input audio signals adversarial for targeting audio recognition models. In order to generate such a vector, a training set and a hold-out set is composed. The proposed method is based on the greedy approach proposed by Moosavi-Dezfooli (Moosavi-Dezfooli *et al.*, 2017) which finds the minimum perturbation that sends examples of the training set to the decision boundary. The aggregation of these individual perturbation vectors generates the UAP vector. The UAP generated for the training set must also be transferable to and effective on the test set as well. For finding the perturbation vectors for each sample, a formulation similar to the method proposed by Carlini & Wagner (2018) is used. The pre-trained Mozilla DeepSpeech (Mozilla, 2017) model is used as the target. Neekhara *et al.* (2019) trained the algorithm on 5,000 samples from The Common Voice dataset (Ardila et al., 2020) for UAP generation. They reported success rates up to 88.24% on the hold-out set. Their method is designed only for untargeted attacks. In Chapter 3, we will discuss this problem in detail, and two methods (iterative and penalty) are proposed for crafting such perturbations for both targeted and untargeted scenarios. We will show the effectiveness of such perturbations to fool a family of audio recognition models.

1.3.3.8 Black box attacks

Recently some solutions proposed to generate adversarial attacks in a black-box (model-agnostic) fashion. However, the dependence of the white box attacks on the models is a considerable downside of the existing attacks against audio processing models. The adversary queries the

model for generating the perturbation through a try and error procedure for crafting this type of attack.

Using evolutionary algorithms like genetic algorithms for generating adversarial attacks based on gradient-free methodology is an interesting approach (Taori, Kamsetty, Chu & Vemuri, 2019), (Alzantot, Balaji & Srivastava, 2018). Taori *et al.* (2019) developed a targeted attack on SR systems using genetic algorithms. The algorithm finds the perturbation vector through a try and error procedure. The algorithm is entirely independent of the model's gradients, so the attacker does not need to have access to the model and its parameters. This algorithm creates a population of candidates from the legitimate audio sample. At each step, the parents are selected based on the distance between the model's prediction for candidates and the target phrase. First, candidates with a minimum distance to the target phrase are selected. The distance measure is an appropriate distance measure for calculating the sequence to sequence similarity. In this algorithm, Connectionist Temporal Classification (CTC) loss is used. After that, the algorithm uses crossover from the best-selected candidates to create the next population. At this step, mutation with momentum is done by injecting random noise from normal distribution to the candidates. This procedure continues until the algorithm reaches the predefined maximum number of iterations, or one of the samples is decoded as the target, and the best sample is returned. The algorithm produces a 35% success rate for targeting a 2-word target phrase from the first 100 samples of CommonVoice test set (Ardila *et al.*, 2020). The victim model is Baidu's DeepSpeech model (Hannun *et al.*, 2014). The average distance between the legitimate samples and the perturbed ones is 2.3 using Levenshtein distance after 3,000 iterations. Figure 1.27 shows the schematic procedure of the algorithm.

The adversarial example can also be completely imperceptible to the human ear. Zhang *et al.* (2017) proposed an attack that modulates the voice commands on ultrasonic carriers (frequencies higher 20 kHz) by the use of appropriate signal processing techniques. The microphone easily detects the inaudible adversarial example and is processed as a legitimate audio sample. The

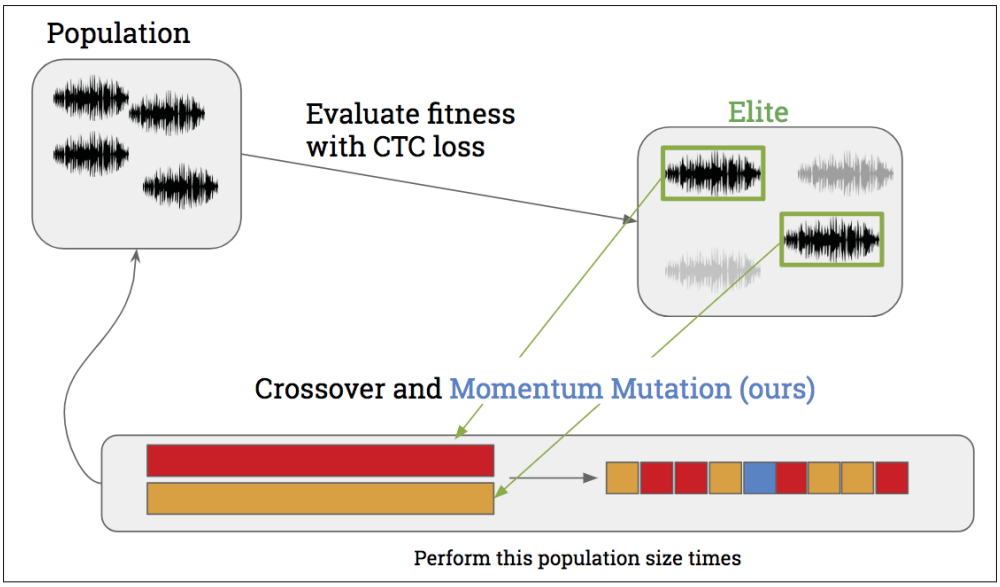


Figure 1.27 Genetic algorithm for adversarial attack generation. The parents are selected from the population based on the minimum distance between the candidates and the target. The crossover operation along with mutation is used to create the next generation Taken from Taori *et al.* (2019)

adversarial example can successfully attack commercial sound processing systems like Apple Siri to execute illegitimate commands [2].

Chen et al. (2020) proposed a targeted black-box attack, called *Devil's Whisper*, against commercial APIs, such as Google API and commercial devices like Amazon Echo. Alternate models based on the generation approach are proposed to achieve this goal. This methodology uses a local large base model and a substitute model as an ensemble for adversarial perturbation generation. The substitute model is a locally trained model to mimic the black box target model behavior. The first several audio files are selected and labeled by the black box target model. Then, the audio files and the generated labels are used to train the substitute model. The authors used 4.6-hour training data (about 1,500 queries with each audio about 25 seconds) for querying the target black-box models. The audio files can also be augmented by some techniques like adding white noise or twisting the audio by slowing or fasting speech rate. Moreover, a

[2] https://www.youtube.com/watch?v=21HjF4A3WE4

supplemental set of open-source audio files can be used to augment the audio training set. Since the substitute model is small and it can not be comparable to large commercial models that are targeted, a large based model like Kaldi Aspire Chain Model (Povey et al., 2011) is used as an ensemble along with a substitute model for adversarial sample generation. Momentum-based Iterative Fast Gradient Method (MI-FGM) (Dong et al., 2018) which is similar to the improved FGSM method (Kurakin *et al.*, 2017) is used for adversarial perturbation generation. Music is also used as the adversarial attack carrier to embed malicious commands into regular songs. This method attacks both the large base model and the substitute model simultaneously. Successful adversarial samples are added to the adversarial example collection. Google assistance, Google Home, Microsoft Cortana, Amazon Echo, and IBM Wav-To-Air devices are used as the target models. Ten phrases are used as targets, and a success rate of more than 90% is reported for targeting such models. SNR between 7.86 dB to 12.10 dB is also reported. The attacks are effective in the physical environment, and the effective distance between 5cm to 200cm is reported.

Using signal processing techniques is also an interesting methodology for crafting black-box attacks (Vaidya, Zhang, Sherr & Shields, 2015), (Abdullah *et al.*, 2020), (Abdullah *et al.*, 2019b). As a proof of concept, Vaidya *et al.* (2015) proposed *"Cocaine Noodles"* attack and posed the following question: *"whether there is audio that is interpreted as the human speech by machines but is perceived by humans as noise other than speech."* Using signal processing techniques, an interesting scenario for attacking SR models in the physical environment is proposed. In the *Man in the Elevator* (MitE) scenario, the victim user enters the elevator with a device that supports continuous speech recognition. During the ride, a user's unrecognizable sound is played. However, the mobile device of the user processes the sound and interprets it as a legitimate command. In such a scenario, Vaidya *et al.* (2015) enumerate several goals that a successful attacker may follow:

- **Initiate a drive-by-download:** The attacker may provide a command to open a web page to download a virus or malware.

- **Earn illegal money:** The attacker may craft an audio-based command to trigger a pay-based Short Message Service (SMS).
- **Enumerate devices in a physical area:** The adversary may also use a loudspeaker to cause the active phones in a physical area to send SMS messages to a specific number. The adversary eventually may have information about the number of active phones in the area.
- **Trigger a premium rate service:** The attack may generate a command to activate a premium rate service by a premium rate number. The attacker may use this service to earn illegal money.
- **Perform a denial-of-service attack:** In this scenario, the attacker may activate the victim's phone flight mode preventing them from receiving the call and other communications.

Vaidya *et al.* (2015) proposed an audio mangling approach based on appropriate signal processing techniques to generate such an attack. The outcome audio has acoustic features that the model processes as a legitimate audio signal. However, the audio file is difficult to be recognized by humans. Figure 1.28 shows the schematic view of the steps for crafting such attacks. The audio mangler generates the morphed version of the legitimate audio sample. The attack is then played to the SR model, which interprets the sample as a legitimate command. The authors used MFCC features to generate such attacks. The attack is generated by modifying the input signal by adjusting MFCC parameters, performing the feature extraction, and then reconstructing an audio signal using reverse MFCC to generate a new adversarial audio signal. Several parameters such as *window size* or *hop size* needs to be set to extract the MFCC features from the signal. This method uses a distance measure to assess the difference between the transcription of the legitimate and the perturbed audio signal to determine the range of the MFCC parameters.

Four types of attacks are created by Vaidya *et al.* (2015) to assess the effectiveness of the perturbations.

- Activate the voice command like *"OK Google"*.
- Calling a number.
- Sending a text message.
- Opening a website.

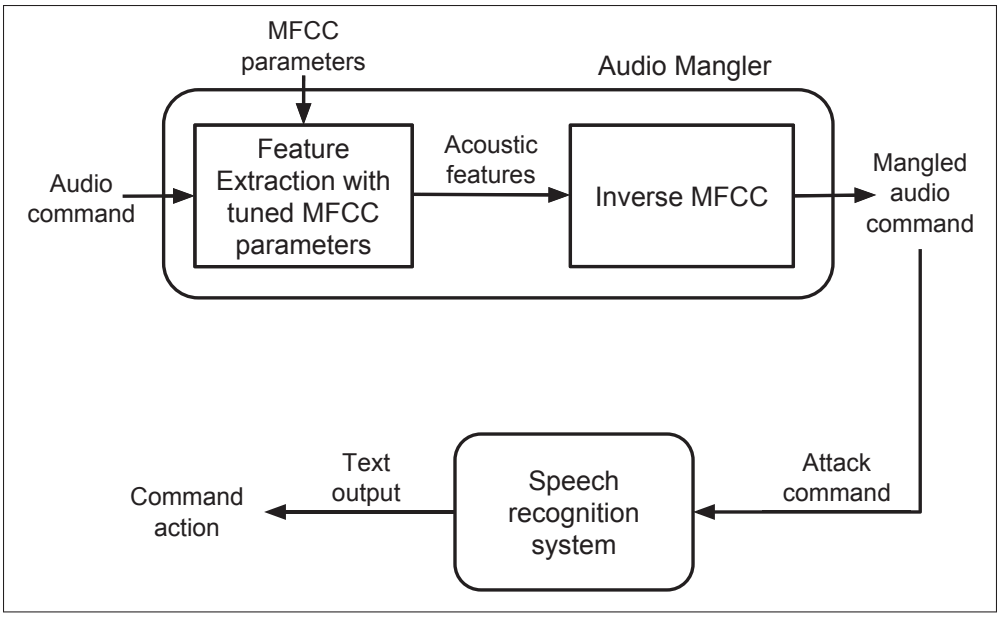


Figure 1.28 Audio mangling for adversarial attack generation. The method adjusts some MFCC parameters to extract features from the signal. The inverse MFCC module generates the adversarial audio signal which can fool the SR model Taken from Vaidya *et al.* (2015)

Google's speech recognition API (Payton, 2015) is used as the target model. According to mentioned scenarios, a set of audio commands consisting of 788 samples are used. The samples are played to a phone through a pair of speakers placed about 30cm to the Android phone to target the Google speech recognition model. As the authors mentioned, all of the selected attack samples successfully activated the corresponding functionalities on the smartphone.

Carlini et al. (2016) extended the research of Vaidya *et al.* (2015) by formalizing the method for creating Hidden Voice Commands (HVC) that are effective in a real-world scenario. Carlini *et al.* (2016) consider the adversarial audio samples that are processed by the SR model as the legitimate sample but are difficult for the human listeners to understand as *obfuscated commands*. The effect of background noise on such attacks is evaluated under a more practical scenario against Google's speech recognition service (Sak, Senior, Rao, Beaufays & Schalkwyk, 2015). The workflow of crafting the obfuscated commands is shown in Figure 1.29. The procedure is similar to the method proposed by Vaidya *et al.* (2015). The feedback process ensures that

first, the audio signal is adversarial, and second, the audio signal is not recognizable by a human listener. Here the attacker also uses an audio mangler to manipulate the MFCC features, and then the produced audio signal by the inverse method can fool the model. Suppose humans too easily understand the generated adversarial audio by the mangler. In that case, the attacker rejects the candidate and creates new candidates by re-adjusting the MFCC parameters to produce a new lower fidelity audio. Here the attacker uses the recognition model as the *oracle* and has no information on the model and its parameters. The whole process is conducted in a try-and-error procedure.

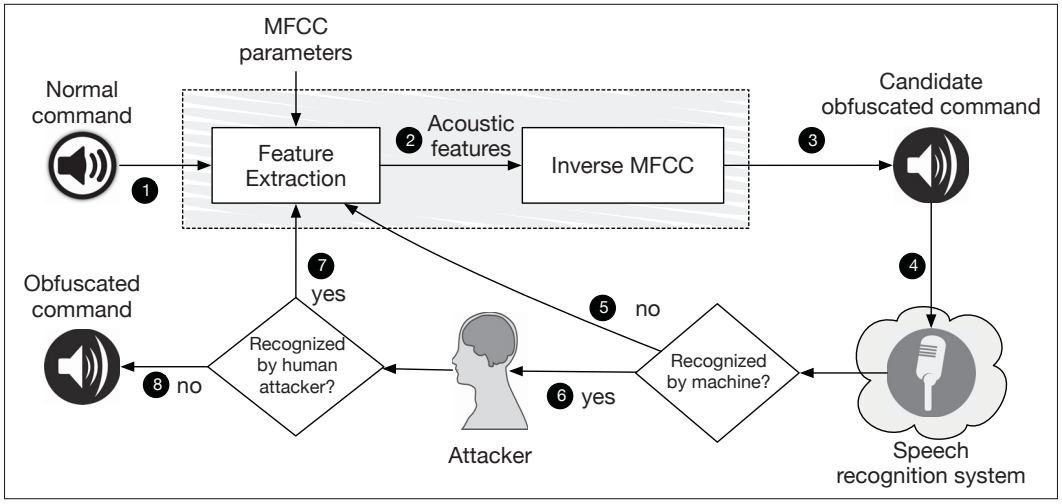


Figure 1.29 Obfuscated adversarial sample generation workflow. The adversarial sample is generated in a try-and-error feed-back loop. Several parameters of MFCC feature extraction module are adjusted to create a successful attack after inverting the extracted MFCC features. The sample must not be recognisable by humans. If the sample is recognisable by the human listener, new adjustment of the MFCC parameters must be performed. Numbers indicate the steps in this procedure Taken from Carlini *et al.* (2016)

Several voice commands including *"OK google"*, *"call 911"* and *"turn on airplane mode"* are used as targets. The voice commands are played by speakers 3 meters away from the target mobile phones. Several background noise signals with various intensity levels consist of audio signals from casinos, classrooms, shopping malls, and an event during which applause occurred are also projected during the experiment. The authors indicate that the attacks are effective when

the distance between the source and the target does not exceed 3.5 meters. In order to evaluate the effectiveness of the attacks, an experiment on Amazon Mechanical Turk was conducted. During the experiment, the listeners had to listen to some audio clips and separate the legitimate samples from illegitimate ones. Table 1.10 shows the percentage of the commands that are correctly interpreted by the SR model and the human listener, as well as the number of samples that are recognized as the target label along with the total number of queries. Results show that the attacks are effective for *"OK Google"* and *"Turn on airplane mode"* classes. However, for the *"Call 911"* command, the human listeners could also distinguish the obfuscated command. The authors suggested repeating several rounds of crowd-sourcing to find an adversarial sample that is not understandable to solve this problem.

Table 1.10 Black-box attacks based on audio obfuscation methodology. The "machine" columns indicate the percentage of the commands that are recognised as each specific target by the SR model. The percentage of the commands that are recognised by "humans" is also reported Taken from Carlini *et al.* (2016)

	OK Google		**Turn on airplane mode**		**Call 911**	
	Machine	*Human*	*Machine*	*Human*	*Machine*	*Human*
Normal	90% (36/40)	89% (356/400)	75% (30/40)	69% (315/456)	90% (36/40)	87% (283/324)
Obfuscated	95% (38/40)	22% (86/376)	45% (18/40)	24% (109/444)	40% (16/40)	94% (246/260)

Abdullah et al. (2019a) proposed an untargeted attack, called *"Kenansville"* attack against SR and SID systems by the use of appropriate signal processing techniques to perturb the audio sample. The main contribution of this work is that the generated attacks are designed in a full black-box scenario where the attacker has no information about the target model, and in contradiction to the methods based on evolutionary methods like genetic algorithm, this method needs to send quite fewer queries to the target model (fewer than 15 queries). Moreover, the adversarial attacks are transferable to unknown models. In addition to that, the attacks are tested over the cellular network, and they are effective in tricking SR and SID systems. The method involves several steps. First, the audio sample is decomposed into several components like DFT or Singular Spectrum Analysis (SSA). Next, a subset of the components whose intensity falls below a threshold are eliminated. After that, a new audio signal is created using the remaining components using the appropriate inverse transformation. Next, the audio signal is passed to the

model for processing. The thresholding constraints are updated according to the comparison between the model's outputs for legitimate and perturbed audio samples. To find the appropriate optimal threshold, more components are removed incrementally during this feedback loop process. Figure 1.30 shows this process.

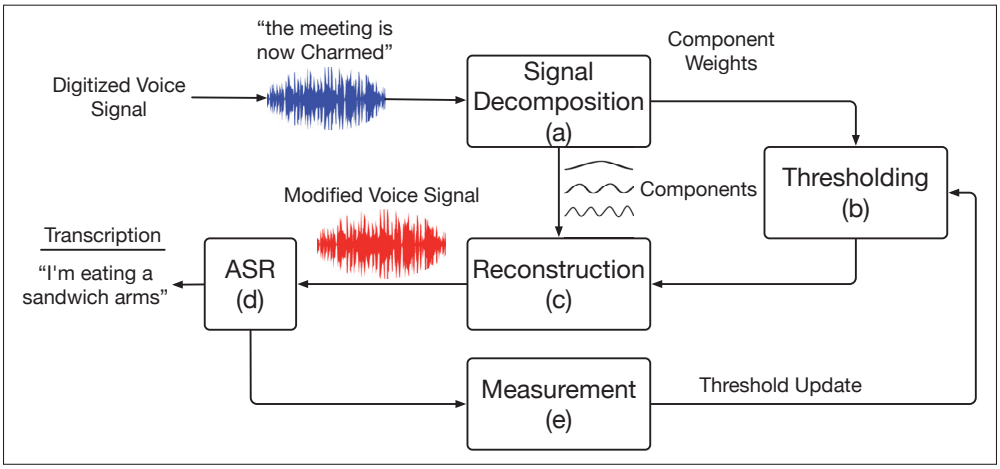


Figure 1.30 Kenansville untargeted attack against speech processing and SID systems. First, the audio signal is decomposed into several components (a). Then a set of components are eliminated based on a thresholding method (b). A new audio sample is reconstructed by the use of other components (c). The sample is processed by the target model. The difference between the model's predictions for legitimate and perturbed samples is used for updating the thresholding constraints (c), (d) Taken from Abdullah *et al.* (2019a)

As the authors claimed, the attacks generated by this method are *model-agnostic* and no information from the target model is needed to generate successful attacks. Considering this specification, the problem of the transferability of the attacks between different models is bypassed in this scenario. From TIMIT dataset (Jankowski *et al.*, 1990), 240 samples with 7600 samples are used for attacking. Another dataset consisting of 1,000 samples of common English language words is also used to attack the target model in word-level robustness (EducationFirst, 2019). Several models including Google transcription API (Google, 2019), Facebook Wit (Facebook, 2019), DeepSpeech (Mozilla, 2017), CMU Sphinx (Lamere et al., 2003) and Microsoft Azure (Microsoft, 2019) are used as the targets. For evaluating the effectiveness of the

attacks over the Telephony-Network, the adversarial samples are transferred through AT&T's LTE network and the Internet via Twilio (Twilio, 2019) to an iPhone 6.

For the *word level perturbations*, the method can successfully attack the models with an 85% success rate where the MSE between the legitimate and perturbed samples is around 0.017. The method can also successfully attack the models for *phoneme level perturbations*. The authors observed that for specific phonemes like *"zh"* or *"nx"* the attack is much more harder than *"eng"* or *"ow"* (refer to (Abdullah *et al.*, 2019a) for more details). The method can also generate successful attacks against Microsoft Azure (Microsoft, 2019) voice identification system. When DFT is used for signal decomposition, the method is able to attack *"t"* or *"d"* than *"zh"* or *"ch"*. Moreover, when SSA is used for signal decomposition, attacking *"oy"* or *"ng"* is more feasible *"zh"* or *"ch"*. Overall, SSA-based attacks are more successful than DFT-based attacks. Google transcription API, Facebook Wit, DeepSpeech, and CMU Sphinx models are used for the transferability experiment of the SSA-based attack. Transferability ratios between 3% to 87% are reported. The authors concluded that the attack has the highest transfer rate when a *harder* model is used to generate audio attack samples. By the hard model, we mean that a high average threshold is needed to attack the model. The attack can also generate successful attacks that are robust when played on the cellular network. For DFT-based attacks, success rates between ~60% to more than 80% for attacking DeepSpeech, Google transcription API, Sphinx, and wit models are reported. Moreover, ASRs between ~45% to more than 80% are also reported for SSA-based attacks.

This section discussed the vulnerability of deep learning models against adversarial attacks. Then, the recent approaches for attacking audio processing models and the problems that such attacks may cause were presented. In the next section, critical analysis and a summary of general audio processing methodologies based on end-to-end architectures and the models that work based on audio features and representations are presented. The following section also criticizes the methodologies for attacking audio processing models. Finally, the strength and weaknesses of the general methodologies are discussed.

1.4 Critical analysis

In this chapter, a summary of the current advances in CNN applied to audio processing methods was presented. Considering current advances of CNNs in audio processing, one can argue that CNNs are mostly used for representation learning from audio features, like MFCCs, or 2D representations, like spectrograms. Table 1.11 represents a summary of current studies using CNNs as representation learning for different tasks. Spectrograms are the most used representations as input to CNNs. Using CNNs for extracting low-level and high-level features from audio files is still an underrepresented issue in audio signal processing. This issue should be considered when designing a CNN architecture for an audio processing task. Audio and image processing share the same pipeline of processes: representation learning from the input sample followed by the classification part. In the case of image processing, the input of the CNN is a pixel matrix representing the image. To our best knowledge, no low-level feature or handcrafted representation of the input image like color histograms is used to feed a CNN. The CNN is used as a representation learning module. Using the effective error back-propagation method, the weights of the kernels of the CNN are adjusted to the dataset, making them a powerful tool for representation learning. There is room for improvement for audio signal processing, especially in the case of end-to-end systems.

For some fields of audio processing, using end-to-end models is challenging. For instance, one of the problems in music processing is the limited access to audio files of the datasets. Most of the popular data sets like MSD (Bertin-Mahieux *et al.*, 2011) provide only feature vectors (because of copyright laws). This might be one of the reasons that the music processing community is not interested in end-to-end systems. Another issue is the small number of recordings available in several audio processing datasets. This could be why CNNs are not so widely used in this field; using handcrafted features, like MFCCs, or 2D representations, like spectrograms, helps reduce the complexity of the neural network and make it easier to train when only a small amount of data is available. Nonetheless, Benzi, Defferrard, Vandergheynst & Bresson (2016) have introduced Free Music Archive (FMA) dataset for which raw audio signals are available. This dataset contains 343 days of audio.

Considering the hardware constraints and limitations on signal processing tools for extracting handcrafted features from the audio signal, it is quite convenient to focus on end-to-end systems based on CNNs for representation learning. Therefore, a question that should be answered is: Can handcrafted features, like MFCCs, or 2D representations, like spectrograms, be efficiently replaced by CNNs for the classification task, like ESC, or not? Therefore, one of the contributions of this thesis is to design an ESC model using an end-to-end architecture. Moreover, as discussed in this thesis, the security concern for developing and deploying such an end-to-end model is also an important issue that should be considered while designing and deploying such models.

Security issues on machine learning models are an emerging concern in this field. To assess current advances in adversarial machine learning, in this chapter, a summary of the threats against machine learning models with a focus on audio processing models was also presented. Adversarial attacks on audio processing models have not been widely addressed. Adversarial attacks on image processing techniques inspired the research community to propose successful methods for targeting audio processing models trained on raw audio signals (Carlini & Wagner, 2018; Kreuk *et al.*, 2018). This indicates that the audio processing models trained on raw audio signals are also vulnerable to such threats. Such threats must be addressed and assessed while designing and deploying such models. Table 1.12 shows a summary of recent attacks on audio processing models. The attacker may peruse different specific goals for targeting audio processing models. The major goals are fooling the SID systems and audio classifiers and deceiving the model to mistranscribe the speech signal.

Specific challenges for attacking audio processing models, which need the research community's attention, were also reported in this chapter. One of the challenges is the attack's effectiveness on different mediums of audio signal transferring. As we discussed, the effectiveness of the adversarial attack when it is transferred over the line, air, or telephony network should be evaluated by the researchers in this field. There is no guarantee that a successful attack on the line can be transferred to the physical environment or over the telephony network. As it is shown in Table 1.12 only a small subset of attacks are effective to be played over the air or transferred through a telephony network *e.g.* the methods presented by Carlini *et al.* (2016) or Abdullah

et al. (2019b). One could conclude that adversarial attacks on end-to-end audio processing models are a lesser threat when the air or telephony network are the only mediums that the attacker may use.

Most of the recent attacks for targeting audio processing models are *"white box"* attacks. So, the amount of information that the attacker may have from the victim model directly affects the intensity of the threats on such models. In this form of the attack, the adversary can generate successful clean audio perturbations on the signal such as (Carlini & Wagner, 2018). In addition, the adversary may use the model's architecture and its weights to perform the attack using a direct optimization method. This issue must be considered in deploying and publishing end-to-end audio processing models and their parameters.

To generate effective adversarial attacks in real-world scenarios, the adversary must have the perfect knowledge from the target model, or the adversarial attack generated for one model must be transferable to other models (black-box scenario). For targeting audio processing models, the transferability of the successful attacks is not either evaluated or feasible (Abdullah *et al.*, 2020). Therefore, the research community must assess and report the transferability of the attacks among various audio processing models since it directly impacts the security of such models in deployment time.

As we mentioned, UAP is one of the effective attacks against image processing models (Moosavi-Dezfooli *et al.*, 2017). Recently Neekhara *et al.* (2019) demonstrated the existence of such perturbations on an SR model. However, the community has not evaluated such an attack targeting audio classification systems, like ESC systems. Moreover, the attack is just proposed for an untargeted attacking scenario. The transferability of UAPs between different models has not also been addressed. One of the contributions of this thesis is to expand this type of attack for targeting a family of audio and speech classification models and evaluating the transferability of the attacks between different models. The position of the proposed approach with respect to other studies is also shown in Table 1.12.

This chapter presented a literature review on general methodologies for audio classification based on end-to-end models and the models that use representations or handcrafted features from the audio signal. The benefits and limitations of each method were presented. This chapter presents a survey on the vulnerabilities of deep learning models to adversarial attacks. A review of the problems they may cause to audio processing models was also presented. The proposed methodology for an end-to-end ESC model will be given in the next chapter.

Table 1.11 Summary of current advances in CNN based audio processing

Study	Input	Application	Dataset	Measure	Best Result
Sigtia *et al.* (2016a)	Constant Q transform	Music Transcription	MAPS	ACC	73.57%
Li *et al.* (2010)	MFCC	Genre Rec.	GTZAN	ACC	84%
Han *et al.* (2017)	Mel-spectrogram	Instrument Rec.	IRMAS	F-Mes.	0.602
Salamon *et al.* (2014)	Mel-spectrogram	Organism Vocalization Rec.	Flight Calls	ACC	96%
Salamon & Bello (2017)	Mel-spectrogram	ESC	UrbanSound8k	ACC	79%
Hershey *et al.* (2017)	Spectrogram	Audio-based Label Pred.	Youtube Video Clips	ACC	93%
Dieleman *et al.* (2011)	Timberal and Chroma features	Artist and Genre Rec., Key Det.	MSD	ACC	Artist Rec.: 35.74%, Genre Rec.: 29.5%, Key Det.: 86.53%
Choi *et al.* (2016)	Mel-spectrogram	Music Tagging	MagnaTagATune, MSD	ACC	MagnaTagATune: 89.4%, MSD: 85.1%
Aytar *et al.* (2016)	Audio Signal	Audio Scene Class.	DCASE, ESC-50, ESC-10	ACC	DCASE: 88%, ESC-50: 74%, ESC-10: 92%
Dieleman & Schrauwen (2014)	Audio Signal, Spectrogram	Music Tagging	MagnaTagATune	ACC	Audio Signal: 84%, Spectrogram: 88%
Hoshen *et al.* (2015)	Audio Signal	SR	Voice Search Dataset	ACC	Single Channel Input: 41.5%, Multi Channel Input: 38.1%
Sainath *et al.* (2015)	Audio Signal	SR	Google's Voice search	ACC	15.5%
Tokozume, Ushiku & Harada (2018)	Audio Signal	ESC	UrbanSound8K, ESC-50	ACC	UrbanSound8k: 78%, ESC-50: 71%
Dai *et al.* (2016)	Audio Signal	ESC	UrbanSound8K	ACC	71.68%
Hertel *et al.* (2016)	Audio Signal	ESC	ESC-10	ACC	83.7%
Pons & Serra (2018)	Spectrograms	ESC	UrbanSound8K	ACC	70%
Zhu *et al.* (2016)	Audio Signal	SR	DeepSpeech2	WER	23.28%
Ravanelli & Bengio (2018)	Audio Signal	SID	TIMIT	SER	0.85%
Lavechin *et al.* (2020)	Audio Signal	Audio Class.	Babytrain	F-measure	57.3
Zeghidour *et al.* (2021)	Audio Signal	Audio Class.	Multiple Datasets	ACC, AUC	Single-task Class. ACC: 76.9%, Multi-task Class. ACC: 79.3%, AudioSet AUC: 0.97
Saeed *et al.* (2021)	2D Mel filter-bank representation	Audio Class./Detection	Multiple Datasets	ACC	85.1%
Millet & Zeghidour (2019)	Audio Signal	Dysarthria detection.	TORGO	recall	76.4%
Verma & Berger (2021)	Audio Signal	Audio Class.	FSD50K	MAP	0.537

Table 1.12 Summary of current attacks against audio processing models. "✓" indicates that the attack has the specification and "✗" indicates that either the attack lacks the specification or it is not reported. The attack may be effectively transferred to the target model over-line ,"L", over-air, "A" or over telephony-network, "T". "C": Clean, "IA": Inaudible, "N": Noisy; "D": Direct, "SP": Signal Processing, "ID": Indirect; "T": Targeted, "U": Untargeted; "B": Black, "W": White

Attack	**Audio Type**	**Attack Type**	**Goal**	**Knwl.**	**Med.**	**Dist.**	**Equip.**	**Env.**	**Transferable**	**App.**
Taori *et al.* (2019)	C	D	T	B	L	✗	✗	✗	✗	SR
Carlini & Wagner (2018)	C	D	T	W	L	✗	✗	✗	✗	SR
Cisse *et al.* (2017)	C	D	T	W	L	✗	✗	✗	✗	SR
Kreuk *et al.* (2018)	C	D	T	W	L	✗	✗	✗	✗	SID
Qin *et al.* (2019)	C	D	T	W	L	✗	✗	✗	✗	SR
Schönherr *et al.* (2018)	C	D	T	W	L	✗	✗	✗	✗	SR
Yakura & Sakuma (2019)	C	D	T	W	L,A	0.5 meters	✓	✗	✗	SR
Chen *et al.* (2020) (Devil's Whisper)	C	D	T	B	L,A	5-200 cm	✓	✓	✓	SR
Neekhara *et al.* (2019)	C	D	U	W	L	✗	✗	✗	✗	SR
Alzantot *et al.* (2018)	C	ID	T	B	L	✗	✗	✗	✗	SR
Abdullah *et al.* (2019a) (Kenansville)	C	SP	U	B	L,T	N/A	✓	✓	✓	SR/SID
Zhang *et al.* (2017) (Dolphin attack)	IA	SP	T	B	A	150 cm	✓	✓	✓	SR
Abdullah *et al.* (2019b)	N	SP	T	B	L,A	1 ft	✓	✓	✓	SR/SID
Vaidya *et al.* (2015) (Cocaine Noodles)	N	SP	T	B	L,A	30 cm	✓	✗	✗	SR
Carlini *et al.* (2016) (HVC)	N	SP	T	B	L,A	0.5 m	✓	✓	✗	SR
Audio UAP (Ours)	C	D	T, U	W	L	✗	✗	✗	✗	SR, Classification

CHAPTER 2

END-TO-END ESC USING A 1D CONVOLUTIONAL NEURAL NETWORK

In this chapter, we propose an end-to-end 1D CNN for ESC that learns the representation directly from the audio waveforms instead of from 2D representations (Piczak, 2015b; Salamon & Bello, 2017, 2015). The proposed end-to-end approach provides a compact architecture that reduces the computation cost and the amount of data required for training. With the aim of extracting relevant information directly from audio waveforms, several convolutional layers are used to learn low-level and high-level representations. The highest level of representation is then used for classifying the input signal by means of three FC layers. Experimental results on UrbanSound8k dataset, which contains 8,732 environmental sounds from 10 classes, have shown that the proposed approach outperforms other approaches based on 2D representations such as spectrograms (Piczak, 2015b; Salamon & Bello, 2017; Pons & Serra, 2018; Salamon & Bello, 2015) by between 11.24% (SB-CNN) and 27.14% (VGG) in terms of mean accuracy. Furthermore, the proposed approach does not require data augmentation or any signal pre-processing for extracting features.

In Section 2.1 of this chapter, the ideas behind the proposed end-to-end 1D CNN architecture and the proposed approach to deal with variable audio lengths is presented. We also present the variations in the architecture that may arise from different input dimensions as well the process of aggregating the predictions on audio frames.

2.1 Proposed end-to-end architecture

The aim of the proposed end-to-end architecture is to handle audio signals of variable lengths, learning directly from the audio signal, a discriminative representation that achieves a good classification performance on different environmental sounds.

2.1.1 Variable audio length

One of the challenges of using 1D CNNs in audio processing is that the length of the input sample must be fixed but the sound captured from the environment may have various duration. Therefore, it is necessary to adapt a CNN to be used with audio signals of different lengths. Moreover, a CNN must be used for continuous prediction of input audio signals of environmental sounds.

One way to circumvent this constraint imposed by the CNN input layer is to split the audio signal into several frames of fixed length using a sliding window of appropriate width. Therefore, in our approach we use a window of variable width to conditionate the audio signal to the input layer of the proposed 1D CNN. The window width depends mainly on the signal sampling rate. Furthermore, successive audio frames may also have a certain percentage of overlapping, which aim is to maximize the use of information. This naturally increases the number of samples as some parts of the audio signal are reused and that can be viewed as some sort of data augmentation. The process of framing the audio signal into appropriate frames is illustrated in Figure 2.1.

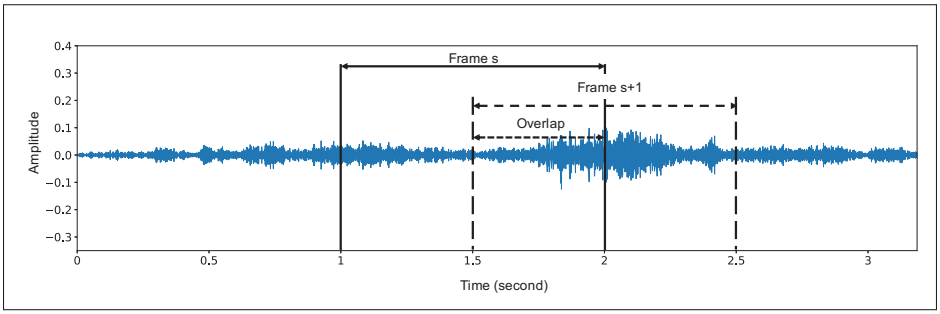


Figure 2.1 Framing the input audio signal into several frames $(s, s + 1)$ with appropriate overlapping percentage (50%)

Moreover, the sampling rate of the audio signals has a direct impact on the dimensionality of the input sample and eventually on the computational cost of model. For environmental sounds, a sampling rate of 16 kHz may be considered a good trade-off between the quality of the input sample and the computational cost of the model.

2.1.2 1D CNN topology

A 1D CNN is analogous to a regular neural network but but it has generally raw data as input instead of handcrafted features. Such an input data is processed through several trainable convolutional layers for learning an appropriate representation of the input. According to the *"local connectivity"* theorem (Li, 2021), the neurons in a layer are connected only to a small region of the previous layer. This small region of connectivity is called a receptive field. The input to out 1D CNN is an array representing the audio waveform, which is denoted as $\mathbf{x}$. The network is designed to learn a set of parameters θ to map the input to the prediction y according to a hierarchical representation learning given by Equation 2.1:

$$y = F^{net}(\mathbf{x} \mid \theta) = F_L^{net}(...F_2^{net}(F_1^{net}(\mathbf{x} \mid \theta_1) \mid \theta_2) \mid \theta_L) \tag{2.1}$$

where L is the number of hidden layers in the network. For the convolutional layers, the operation of the l-th layer can be expressed as:

$$F_l^{net}(\mathbf{x}_l \mid \theta_l) = h(\mathbf{W}^{net} \otimes \mathbf{x}_l + b^{net}), \quad \theta_l = [\mathbf{W}^{net}, b^{net}] \tag{2.2}$$

where $\otimes$ denotes the convolution operation, $\mathbf{x}_l$ is a two-dimensional input matrix of F^m feature maps with N dimension, $\mathbf{W}^{net}$ is a set of K^{fl} one dimensional kernels (receptive field) used for extracting a new set of features from the input array, b^{net} is the bias vector, and $h(\cdot)$ is the activation function. The shapes of $\mathbf{x}_l$, $\mathbf{W}^{net}$ and F_l^{net} are (F^m, N), (K^{fl}, F^{fl}) and $(F^m, N - F^{fl} + 1)$, respectively. Where, F^{fl} is the size of the filters of CNN. Several pooling layers are also applied between the convolutional layers for increasing the area covered by the next receptive fields. The output of the final convolutional layer is then flattened and used as input of several stacked FC layers, which can be described as:

$$F_l^{net}(\mathbf{x}_l \mid \theta_l) = h(\mathbf{W}^{net}\mathbf{x}_l + b^{net}), \quad \theta_l = [\mathbf{W}^{net}, b^{net}] \tag{2.3}$$

In the case of multiclass classification, the number of neurons of the output layer is the number of classes. Using softmax as the activation function for the output layer, each output neuron indicates the membership degree of the input samples for each class. During the training process, the parameters of the network are adjusted according to the back-propagated classification error and the parameters of the network are optimized to minimize an appropriate loss function (Goodfellow, Bengio & Courville, 2016).

The proposed topology aims a compact 1D CNN architecture with a reduced number of parameters. The number of parameters of a CNN is directly related to the computational effort to train such a network as well as to the need of a large amount of data for training. Therefore, the proposed architecture shown in Figure 2.2 is made of four convolutional layers, possibly interlaced with max pooling layers, followed by two FC layers and an output layer. The baseline model shown in Figure 2.2 has as input an array of 16,000 dimensions, which represents 1-second of audio sampled at 16 kHz. However, this is not a constraint since we can adapt the model for different audio lengths and sampling rates in two ways: (i) change the model architecture to adapt it to the characteristics of the audio inputs; (ii) padding or segmenting the audio piece to adapt it to the input dimensions of the network.

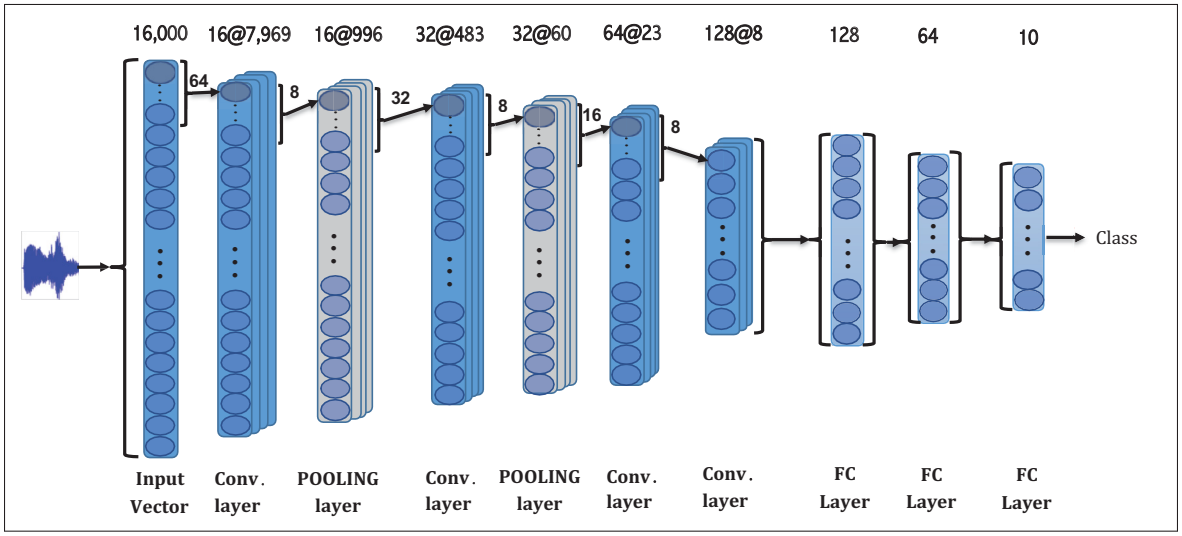


Figure 2.2 The architecture of the proposed end-to-end 1D CNN for ESC. The dimension, number of filters and filter size are given for the input size of 16,000. For other input sizes, the values are presented in Table 2.1

Several other configurations can also be derived from subtle modifications of the base model (shown in Figure 2.2) to adapt it to shorter or longer audio inputs, as shown in Table 2.1. This implies modifying the number of convolutional layers as well as the number and the dimension of filters and the stride. However, for long contiguous audio recordings, instead of increasing the input dimension of the network, which also implies increasing the number of parameters, and consequently its complexity, it is preferable to split the audio waveform into shorter frames by changing the window width as explained in Section 2.1.1. In this way, we keep the network compact and it can process audio waveforms of any length. In spite of that, in Section 4.1.2 we evaluate different audio lengths as input, keeping a fixed sampling rate of 16 kHz.

The proposed 1D CNN has large receptive fields in the first convolutional layers since it is assumed that the first layers should have a more global view of the audio signal. Moreover, the environmental sound signal is non-stationary *i.e.* the frequency or spectral contents of the signal changes with respect to time. Therefore, shorter filters do not provide a general view on the spectral contents of the signal. The output of the last pooling layer for all feature maps is flattened and used as input to a FC layer. In order to reduce the over-fitting, batch normalization is applied after the activation function of each convolution layer (Ioffe & Szegedy, 2015). The last FC layer has ten neurons. Mean squared logarithmic error, defined in Equation 2.4 is used as loss function (J) to assess the prediction of the model, a_m^L obtaind form the last layer of the model L with respect to target values, t_m:

$$J = \frac{1}{M} \sum_{i}^{M} log(\frac{a_m^L + 1}{t_m + 1})^2 \tag{2.4}$$

For all input sizes shown in Table 2.1, after the last pooling layer, there are two FC layers with 128 and 64 neurons respectively on which a drop-out is applied with a probability of 0.25 for both layers (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014). The ReLU activation function is used for all layers, except for the output layer where a softmax activation function is used. Since the amount of data for training is limited, it is not feasible to use deeper architectures without significant over-fitting. By the use of the architecture shown in Figure 2.2, it is possible to omit a signal processing module because the network is powerful enough to extract relevant

Table 2.1 The configuration of the CNN for ESC. The configuration of the convolutional layers (CL) and pooling layers (PL) for the end-to-end CNN considering different input sizes (audio lengths)

Input Size	**Specification**	**Layer**							
		CL1	PL1	CL2	PL2	CL3	CL4	CL5	PL3
50,999	Dim	25,468	3,183	1,576	197	91	42	20	5
	# Filters	16	16	32	32	64	128	256	256
	Filter Size	64	8	32	8	16	8	4	4
	Stride	2	8	2	8	2	2	2	4
32,000	Dim	15,969	1,996	983	122	54	24	11	2
	# Filters	16	16	32	32	64	128	256	256
	Filter Size	64	8	32	8	16	8	4	4
	Stride	2	8	2	8	2	2	2	4
16,000	Dim	7,969	996	483	60	23	8	NA	NA
	# Filters	16	16	32	32	64	128	NA	NA
	Filter Size	64	8	32	8	16	8	NA	NA
	Stride	2	8	2	8	2	2	NA	NA
16,000G	Dim	15,489	19,36	953	119	52	23	NA	NA
	# Filters	64	64	32	32	64	128	NA	NA
	Filter Size	512	8	32	8	16	8	NA	NA
	Stride	1	8	2	8	2	2	NA	NA
8,000	Dim	3,969	496	233	29	7	NA	NA	NA
	# Filters	16	16	32	32	64	NA	NA	NA
	Filter Size	64	8	32	8	16	NA	NA	NA
	Stride	2	8	2	8	2	NA	NA	NA
1,600	Dim	785	392	189	94	44	NA	NA	NA
	# Filters	16	16	32	32	64	NA	NA	NA
	Filter Size	32	2	16	2	8	NA	NA	NA
	Stride	2	2	2	2	2	NA	NA	NA

NA: Not Applicable. G: With Gammatone filter-bank for the first layer of CNN.

low-level and high-level information from the audio waveform. The convolutional layers of the proposed architecture are inspired in Aytar *et al.* (2016) who proposed a CNN architecture (SoundNet) for learning sound representations from unlabeled videos.

2.1.3 Gammatone filter-banks

Another interesting characteristic of such a 1D CNN is that its first layer can be initialized as a Gammatone filter-bank. A Gammatone filter is a linear filter described by an impulse response of a gamma distribution and a sinusoidal tone. This initialization can be viewed as a trade-off between handcrafted features and representation learning. In this configuration, the kernels of the first layer are initialized by 64 band-pass Gammatone filters with central frequency ranging from 100 Hz to 8 kHz. Such a filter-bank decomposes the input signal into 64 frequency bands.

Gammatone filters have been used in models of the human auditory system and are physiologically motivated to simulate the structure of peripheral auditory processing stage. For this reason, Gammatone filters have also been used to initialize the first layer of 1D CNNs for SR (Hoshen *et al.*, 2015; Zeghidour *et al.*, 2018; Sainath *et al.*, 2015). Figure 2.3 illustrates the frequency response of the Gammatone filter-bank, generated by the Gammatone-like spectrograms toolbox developed by Ellis (2009).

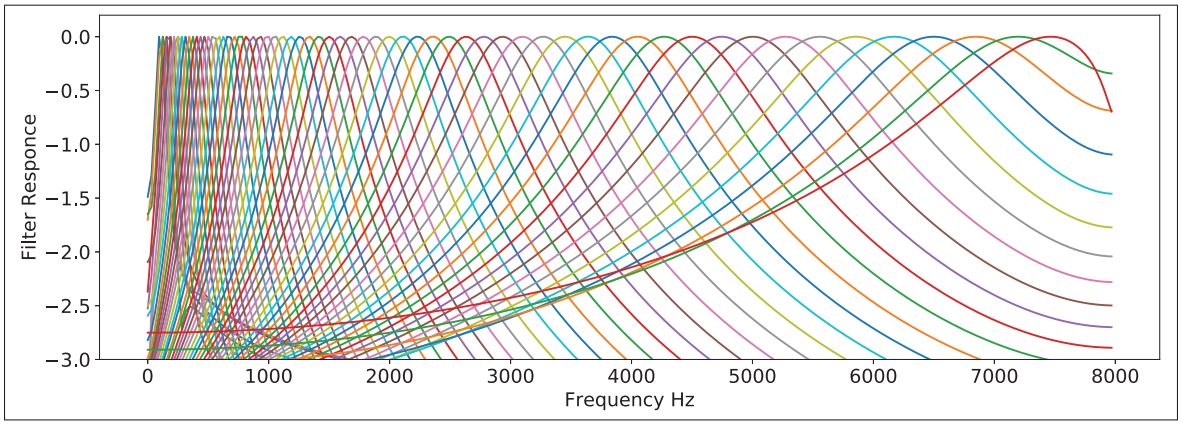


Figure 2.3 Frequency response of 64 filters of Gammatone filter-bank

2.1.4 Aggregation of audio frames

In the case where the input audio waveform $\mathbf{x}$ is split into S frames denoted as $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_S$, during the classification we need to aggregate the CNN predictions to come up to a decision on $\mathbf{x}$, as illustrated in Figure 2.4. For such an aim, different fusion rules can be used to reach a final

decision, such as the majority vote or the *sum* rule, which are denoted in Equations 2.5 and 2.6 respectively.

$$y_c = \sum_{s=1}^{S} o_{sc} \tag{2.5}$$

where o_{sc} is the CNN prediction for the $s = 1, \ldots, S$ segment of the audio waveform $\mathbf{x}$ and $c = 1, \ldots, C$ is the predicted class. S is the number of frames and C is the number of classes.

$$y_c = \frac{1}{S} \sum_{s=1}^{S} o_{sc} \tag{2.6}$$

When there are C classes, we generate C values and them for an audio input, we choose the class with the maximum y_c value:

$$\text{Choose} \quad C_c \quad \text{if} \quad y_c = \max_{c=1}^{C} y_c \tag{2.7}$$

In this chapter, the proposed model for ESC using a 1D convolutional neural network was

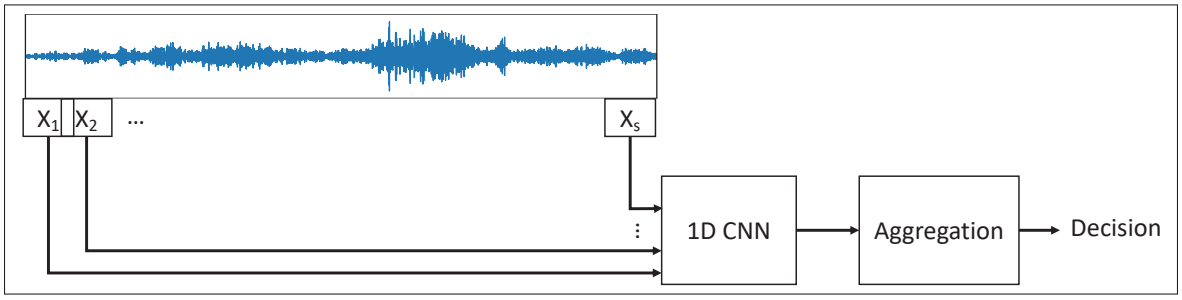


Figure 2.4 Aggregation of the predictions on the audio frames

presented. The method is proposed to address the first research question of the thesis. In the next chapter, the vulnerability of end-to-end models against a specific type of adversarial attack, UAP, is addressed, and two major methodologies are proposed to address the second research question of the thesis.

CHAPTER 3

UNIVERSAL ADVERSARIAL AUDIO PERTURBATIONS

In this chapter, we demonstrate the existence of UAPs, which can fool a family of audio classification architectures, for both targeted and untargeted attack scenarios. The previous studies on UAPs for attacking audio processing systems, however, have focused only on untargeted attacking 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.* (2017), which finds the minimum perturbation that sends examples to the decision boundary. As a part of this method, we show that the decoupled direction and norm (DDN) attack (Rony *et al.*, 2019), which was originally proposed for targeting image processing models, can also be used in an iterative method for targeting audio models. The second method, which is the main contribution of this work, is a novel penalty formulation, which finds targeted and untargeted UAPs. 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. Moreover, we provide a proof that the proposed penalty method theoretically converges to a solution that corresponds to UAPs. Both methods are evaluated on a family of audio classifiers based on deep models for ESC and speech recognition. The experimental results have shown that both proposed methods can attack deep models, which are used as target models, with a high success rate. Section 3.1 of this chapter presents the proposed methods to craft universal audio adversarial perturbations.

3.1 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.* (2017) 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^{sig} be the distribution of audio samples in $\mathbb{R}^N$ 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 (Moosavi-Dezfooli *et al.*, 2017): $\|\mathbf{v}\|_p \leq \xi$ 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 $\mathbb{P}_{\mathbf{x}\sim\mu}(\hat{k}(\mathbf{x} + \mathbf{v}) = y_t) \geq 1 - \delta$.

3.1.1 Iterative Greedy Algorithm

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be a set of m audio files sampled from the distribution μ^{sig}. The greedy algorithm proposed by Moosavi-Dezfooli *et al.* (2017) 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:

$$\begin{aligned} \Delta \mathbf{v}_i \leftarrow \arg\min_{\mathbf{r}} \|\mathbf{r}\|_2 \\ \text{s.t.} \quad \hat{k}\left(\mathbf{x}_i + \mathbf{v} + \mathbf{r}\right) \neq \hat{k}\left(\mathbf{x}_i\right), \\ \text{or} \quad \hat{k}\left(\mathbf{x}_i + \mathbf{v} + \mathbf{r}\right) = y_t. \end{aligned} \tag{3.1}$$

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 (Carlini & Wagner, 2017) or DDN attack (Rony *et al.*, 2019), can be used. Moosavi-Dezfooli *et al.* (2016) 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:

$$\mathcal{P}_{p,\xi}(\mathbf{v}) = \arg\min_{\mathbf{v}'} \|\mathbf{v} - \mathbf{v}'\|_2 \quad \text{s.t.} \quad \|\mathbf{v}'\|_p \leq \xi. \tag{3.2}$$

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

$$\text{ASR}\left(X, \mathbf{v}\right) := \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\{\hat{k}\left(\mathbf{x}_i + \mathbf{v}\right) \neq \hat{k}\left(\mathbf{x}_i\right)\} \geq 1 - \delta, \tag{3.3}$$

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 Equation 3.2. 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.

3.1.2 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 study, 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) (Carlini & Wagner, 2018). The problem of crafting a perturbation in a targeted attack can be reformulated as the following constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & \text{SPL}(\mathbf{v}) \\ \text{s.t.} \quad & y_t = \arg\max_y \mathbb{P}\left(y|\mathbf{x}_i + \mathbf{v}, \theta\right) \\ \text{and} \quad & 0 \leq \mathbf{x}_i + \mathbf{v} \leq 1 \quad \forall i \end{aligned} \tag{3.4}$$

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 Equation 3.4, the pressure level of an audio waveform (noise) can be computed as:

$$\text{SPL}(\mathbf{v}) = 20 \log_{10} P^{sig}(\mathbf{v}), \tag{3.5}$$

where $P^{sig}(\mathbf{v})$ is the root mean square (RMS) of the perturbation signal $\mathbf{v}$ of length N, which is given by:

$$P^{sig}(\mathbf{v}) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} v_n^2}, \tag{3.6}$$

where v_n denotes the n-th component of the array $\mathbf{v}$.

The optimization problem introduced in Equation 3.4 can be solved by a gradient-based algorithm, which however does not enforce the box constraint. Therefore, we need to introduce a new parameter $\mathbf{w}$, which is defined in Equation 3.7 to ensure that the box constraint is satisfied.

This variable change is inspired by the ℓ_2 attack of Carlini and Wagner (Carlini & Wagner, 2017).

$$\mathbf{w}_i = \frac{1}{2}(\tanh(\mathbf{x}'_i + \mathbf{v}') + 1), \tag{3.7}$$

where $\mathbf{x}'_i = \text{arctanh}((2\mathbf{x}_i - 1) * (1 - \epsilon^{uap}))$ and $\mathbf{v}' = \text{arctanh}\,((2\mathbf{v} - 1) * (1 - \epsilon^{uap}))$ are the audio example $\mathbf{x}_i$ and the perturbation vector $\mathbf{v}$ represented in the tanh space, respectively, and ϵ^{uap} is a small constant that depends on the extreme values of the transformed signal that ensures that $\mathbf{x}'_i$ and $\mathbf{v}'$ does not assume infinity values. For instance, ϵ^{uap}=$1e{-}7$ is a suitable value for the datasets used in Section 4.2. The audio example $\mathbf{x}_i$ must be transformed to tanh space and then Equation 3.7 can be used to transform the perturbed data to the valid range of $[0, 1]$. 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. Refer to Appendix I for details. As a result of this transformation, the produced perturbation vector is also in tanh space, and $\mathbf{v}$ can be written as:

$$\mathbf{v} = \frac{\tanh(\mathbf{v}') + 1 - \epsilon^{uap}}{2 - 2\epsilon^{uap}}. \tag{3.8}$$

In order to solve the optimization problem defined in Equation 3.4, we rewrite variable $\mathbf{v}'$ as follows:

$$\mathbf{v}' = \frac{1}{2}\ln\left(\frac{\mathbf{w}_i}{1 - \mathbf{w}_i}\right) - \mathbf{x}'_i. \tag{3.9}$$

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

$$\min_{\mathbf{w}_i} \left\{ \begin{array}{l} L^{uap}(\mathbf{w}_i, t) = \text{SPL}\left(\frac{1}{2}\ln\left(\frac{\mathbf{w}_i}{1-\mathbf{w}_i}\right) - \mathbf{x}_i'\right) + c.G(\mathbf{w}_i, t), \\ G(\mathbf{w}_i, t) = \max\{\max_{j\neq t}\left\{f(\mathbf{w}_i)_j\right\} - f(\mathbf{w}_i)_t, -\kappa\} \end{array} \right\} \tag{3.10}$$

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 level of the attack. For untargeted attacks, we modify the objective function of Equation 3.10 as:

$$\min_{\mathbf{w}_i} \left\{ \begin{array}{l} L^{uap}(\mathbf{w}_i, y_l) = \text{SPL}\left(\frac{1}{2}\ln\left(\frac{\mathbf{w}_i}{1-\mathbf{w}_i}\right) - \mathbf{x}_i'\right) + c.G(\mathbf{w}_i, y_l), \\ G(\mathbf{w}_i, y_l) = \max\{f(\mathbf{w}_i)_{y_l} - \max_{j\neq y_l}\left\{f(\mathbf{w}_i)_j\right\}, -\kappa\} \end{array} \right\} \tag{3.11}$$

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:

$$\begin{aligned} G(\mathbf{w}_i, t) = 0 \quad &\text{if} \quad y_t = \arg\max_y \mathbb{P}(y|\mathbf{w}_i, \theta), \\ G(\mathbf{w}_i, t) > 0 \quad &\text{if} \quad y_t \neq \arg\max_y \mathbb{P}(y|\mathbf{w}_i, \theta), \end{aligned} \tag{3.12}$$

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 (Kingma & Ba, 2014) can be used to minimize the finite-sum loss defined in Equations (3.10) and (3.11). Several other optimization algorithms like AdaGrad (Duchi, Hazan & Singer, 2011), standard gradient descent, gradient descent with Nesterov momentum (Sutskever, Martens, Dahl & Hinton, 2013) and RMSProp (Goodfellow *et al.*, 2016) have also been evaluated but Adam converges in fewer iterations and it produces

relatively similar solutions. Algorithm 3.1 presents the pseudo-code of the proposed penalty method.

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 Equation 3.4[3].

Proof: According to Equation 3.8 and Equation 3.9, $\mathbf{v}^k$ can be defined as:

$$\begin{aligned} \mathbf{v'}^k &= \frac{1}{2}\ln\left(\frac{\mathbf{w}_i^k}{1-\mathbf{w}_i^k}\right) - \mathbf{x}'_i, \\ \mathbf{v}^k &= \frac{\tanh(\mathbf{v'}^k) + 1 - \epsilon^{uap}}{2 - 2\epsilon^{uap}} \end{aligned} \tag{3.13}$$

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 Equation 3.4. Then $\text{SPL}(\mathbf{v}^*) \geq L^{uap}\left(\mathbf{w}_i^k, t\right) \geq \text{SPL}\left(\mathbf{v}^k\right) \forall k$.

Proof of Lemma 1:

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

Proof of Theorem 1. SPL is a monotonically increasing function and continuous. Also, G is a hinge function, which is continuous. L^{uap} is the summation of two continuous functions.

[3] Theorem 1 applies to the context of convex optimization. The neural network is defined as a functional constraint on the optimization problem defined in Equation 3.4. Since neural networks are not convex, a feasible solution to the optimization problem may not be unique and it is not guaranteed to be a global optimum. Moreover, the Theorem 1 is proved based on the assumption defined in Equation 3.12 i.e. κ=0

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

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

If $\mathbf{v}^k$ is a feasible point for the constrained optimization problem defined in Equation 3.4, then, from the definition of function $G(.)$, one can conclude that $\lim_{k\to\infty} c.G(\mathbf{w}_i^k, t) = 0$. Then:

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

$\therefore$ $\bar{\mathbf{v}}$ is a solution of the problem defined in Equation 3.4

In this chapter The methodology for generating UAPs for targeting end-to-end audio classification models were presented. Two methods including an iterative approach and a penalty based optimization solution were presented. The next chapter describes the experimental protocol and the results of the two contributions of this thesis. The end-to-end ESC model and the proposed methods for crafting UAPs are evaluated.

Algorithm 3.1 Penalty method for generating UAP

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 $\mathbf{v}'$

initialize $\mathbf{v}' \leftarrow 0$**,**

while ASR $(X, \mathbf{v}') \leq 1 - \delta$ **do**

  Sample a mini-batch of size S from (X, Y)

  $\mathbf{g} \leftarrow 0$

  for $i \leftarrow 1$ *to* S **do**

    Transform the audio signal to tanh space**:**

    $\mathbf{x}'_i = \text{arctanh}\left((2\mathbf{x}_i - 1) * (1 - \epsilon^{uap})\right)$**,**

    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., Equation 3.10 or Equation 3.11, w.r.t. $\mathbf{w}_i$**:**

    if *targeted attack:* **then**

      $\mathbf{g}^{uap} \leftarrow \mathbf{g}^{uap} + \frac{\partial L^{uap}(\mathbf{w}_i, t)}{\partial \mathbf{w}_i}$

    else

      $\mathbf{g}^{uap} \leftarrow \mathbf{g}^{uap} + \frac{\partial L^{uap}(\mathbf{w}_i, y_i)}{\partial \mathbf{w}_i}$

    end if

  end for

  Compute update $\Delta\mathbf{v}'$ using $\mathbf{g}$ according to Adam update rule (Kingma & Ba, 2014)

  apply update**:** $\mathbf{v}' \leftarrow \mathbf{v}' + \Delta\mathbf{v}'$

end while

return $\mathbf{v}'$

CHAPTER 4

EXPERIMENTAL PROTOCOL AND RESULTS

This chapter presents the experimental protocol and results on the two major contributions of this thesis, such as the end-to-end model for ESC and the UAP. For each contribution, the datasets and the evaluation methodology of the proposed methods are presented. The results are also discussed and are compared with the state-of-the-art methodologies.

This chapter is organized as follows: Section 4.1 presents the experimental results and analysis on an end-to-end ESC model using a 1D convolutional neural network. The methods for fine-tuning the model, evaluating the models on different audio lengths, and architecture enhancement methods are also presented in this section. Finally, results on evaluating the model for ESC are also discussed in Section 4.1.4. Section 4.2 presents the dataset, the target models used to evaluate the proposed methods for UAP generation. The experimental results on two bench-marking datasets for UAP evaluation are presented in Section 4.2.1.

4.1 Experimental results on end-to-end ESC model

The proposed end-to-end 1D CNN for ESC was evaluated on the UrbanSound8k dataset (Salamon *et al.*, 2014). This dataset consists of 8,732 audio clips summing up to 7.3 hours of audio recordings. The maximum duration of audio clips is four seconds. The classes and the number of samples in each class are: "Air conditioner (AI): 1000", "Car horn (CA): 429", "Children playing (CH): 1000", "Dog bark (DO): 1000", "Drilling (DR): 1000", "Engine (EN) idling: 1000", "Gunshot (GU): 374", "Jackhammer (JA): 1000", "Siren (SI): 929", "Street music (ST): 1000". The original audio clips are recorded at different sample rates. For the experiments presented in this study, they have been downsampled to 16 kHz in order to unify the shape of the input signal for the 1D CNN.

4.1.1 Fine-tuning the 1D CNN architecture

The number of convolutional layers plays a key role in detecting high-level concepts. The number of convolutional layers for the base model shown in Figure 2.2 was determined in an exploratory experiment using the audio files of the UrbanSound8k dataset. The audio files were segmented into 16,000 samples and successive frames have 50% of overlapping. Ten percent of the dataset was used as validation set and 10% percent of the dataset was also used as test set. Each network was trained with 80% of the dataset up to 100 epochs with batch sizes of 100 samples. The accuracy achieved by the 1D CNN with one to four convolutional layers on test set was 69%, 75%, 79% and 80%, respectively. Four convolutional layers is the upper limit since the minimal dimension of the feature map has been reached at this layer. The same procedure was also adopted to find the best number of convolution layers and their parameters for the other configurations derived from the base model described in Table 2.1.

4.1.2 Evaluation on different audio lengths

All experiments reported in this subsection used a 10-fold cross-validation procedure to produce a fair comparison with the results reported by Salamon *et al.* (2014). One of the nine training folds is used as validation set for optimizing the parameters of the network to achieve the best accuracy. A batch size of 100 samples was used for training the CNNs and they were trained up to 100 epochs with early stopping. The Adadelta (Zeiler, 2012) optimizer with the default learning rate of 1.0 was used. Adadelta has been chosen because this method dynamically adapts the learning rate during the optimization process.

First, the proposed end-to-end 1D CNN is evaluated on different audio lengths to assess the impact of the input length on the classification performance. Next, the full audio recordings of UrbanSound8k dataset, which has 59,999 frames ($\approx$ three seconds), were segmented into shorter frames using a sliding window and considering different overlapping percentages (0%, 25%, 50%, and 75%). The architecture shown in Figure 2.2 was adapted according to the parameters

described in Table 2.1, leading to audio frames of 1,600 (≈ 100 msec), 8,000 (≈ 500 msec), 16,000 (≈ 1 second) and 32,000 samples (≈ 2 seconds).

The process of segmenting the audio signal into frames and aggregating the classifier's predictions for all frames, resembles the process of aggregating the prediction of an ensemble of classifiers. In this process, the most important parts of the audio signal contribute more to the final decision, while the noisy or outlier frames have their importance averaged during the aggregation process. Table 4.1 shows the best results achieved with different frame sizes, window overlapping and combination rules on the UrbanSound8k dataset in terms of mean accuracy. For the classification of each test sample of the original dataset, the predictions for each audio segment are combined using either the majority voting or the sum rule (Kittler, Hatef, Duin & Matas, 1998). Table 4.1 shows that the 16,000-input architecture achieved the highest accuracy, which is the same accuracy achieved by 1D CNN with 59,999 inputs, even if it has almost twice fewer parameters than that network. Furthermore, the 8,000-input architecture achieved a mean accuracy close to that, even if it has almost three times fewer parameters. If we increase the input size from one second to two or more seconds, besides increasing the number of parameters of the models, we reduce the number of audio segments, which may affect the training of such models due to the reduced amount of data. For this reason, we do not observe any improvement for audio segments beyond one second (16,000 frames). On the other hand, for the 1,600-input architecture, the mean accuracy is about 6% lower than the best architectures. This is an indication that short audio segments do not contain enough information to train properly the 1D CNN. However, this behaviour may be particular for the UrbanSound8k dataset and it cannot be generalized to other audio classification tasks or datasets. Table 4.1 also shows the computational time per epoch for training the networks with a subset 10,000 audio segments. The input size has also a direct relationship with training time, as more operations need to be done for larger inputs. Therefore, the 16,000-input CNN provides the best tradeoff between the number of parameters of network, computational time and mean accuracy.

The box-plot of Figure 4.1 also shows that the 16,000-input 1D CNN is the best choice since it provides the highest median; the interquartile range is the smallest one; and there is no outlier.

Table 4.1 Mean accuracy and standard deviation on the UrbanSound8k dataset over the 10 folds for the different architectures having as input the full audio (59,999) or segmented audio with different window widths and 50% overlapping

Input Dimension	**Combination Rule**	**Mean±SD Accuracy**	**# of Parameters**	**Computational Time (Sec)**
59,999	NA	**83%**±1.3%	421,146	3.917
32,000	Maj Voting	82%±0.9%	322,842	3.325
16,000	Sum Rule	**83%**±1.3%	256,538	1.863
8,000	Sum Rule	80%±1.9%	116,890	1.073
1,600	Sum Rule	77%±3.0%	394,906	0.648

NA: Not applicable

Furthermore, such an architecture has the same mean accuracy, but almost half of the number of parameters than the second-best choice, the 50,999-input 1D CNN. Therefore, the 16,000-input 1D CNN is preferable over other architectures, as it presents the best trade-off between the number of parameters and accuracy.

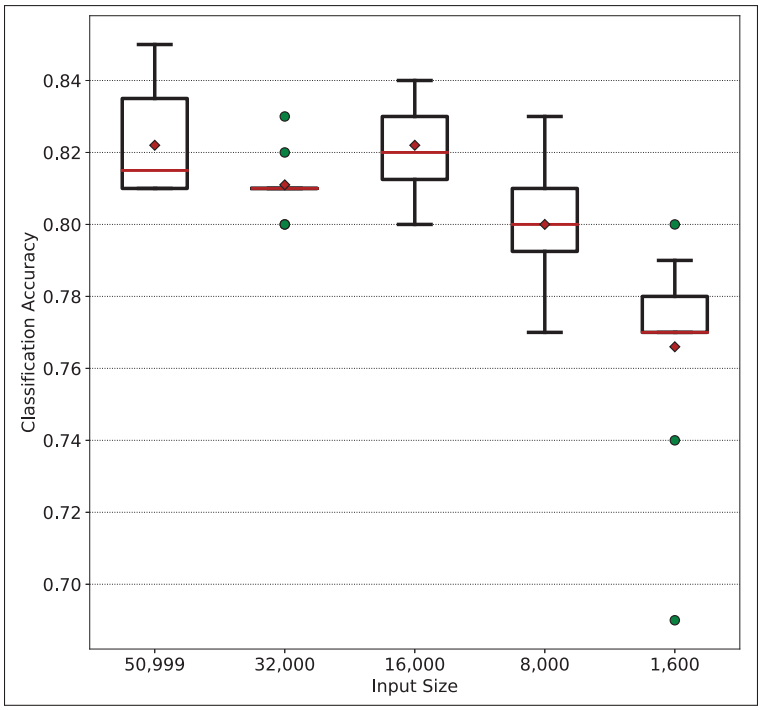


Figure 4.1 The box plot for the five different input sizes on UrbanSound8k dataset

In order to have a better insight into the behavior of the convolutional filters learned by the proposed 1D CNN, the Fourier transform of some filters was computed and their frequency responses are shown in Figure 4.2. These filters were randomly initialized and trained for the specific task. Their parameters, such as central frequency, bandwidth, gain/attenuation, were learned directly from the data to minimize a loss function. The learned filters combine different (mainly band-pass and band-reject) filters with selective attenuation levels for different frequency levels. The filters of the first layers (CL1 and CL2) do not exhibit dominant frequencies and are quite noisy. On the other hand, the filters learned at the deeper layers (CL3 and CL4) are more regular filters, i.e., they have a well-defined frequency response that is closer to ideal filters. However, the resolution of the Fourier transform of the deeper layers is lower than in the initial layers because they are smaller than the initial ones. This analysis lead us to propose some enhancements to the proposed approach as an attempt to improve the response of the filters learned by the network.

4.1.3 Architecture enhancement

Three enhancements to the proposed approach are evaluated: (i) replacing the Hamming sliding window with a rectangular window because the Hamming window smooths the signal and reduces the energy of the beginning and end of the audio frame and this may cause a loss of information; (ii) augmenting the amount of training data slightly by increasing the window overlapping during the audio segmentation; (iii) initializing the first convolutional layer as a Gammatone filter-bank as described in Section 2.1, and make this layer non-trainable.

Table 4.2 summarizes the three proposed enhancements and their impact on the mean accuracy. The rectangular window leads to a slight improvement of 2% in the mean accuracy. Increasing the overlapping from 50% to 75% led to another 2% of improvement in the mean accuracy. Finally, initializing the first layer of such a 1D CNN with a Gammatone filter-bank, also contributed to improve the mean accuracy in 2%, even if the number of parameters doubles due to increase of the number of filters in such a layer. This also increases the training time. An important remark is that all these enhancements have also improved the performance of most of the other 1D CNN

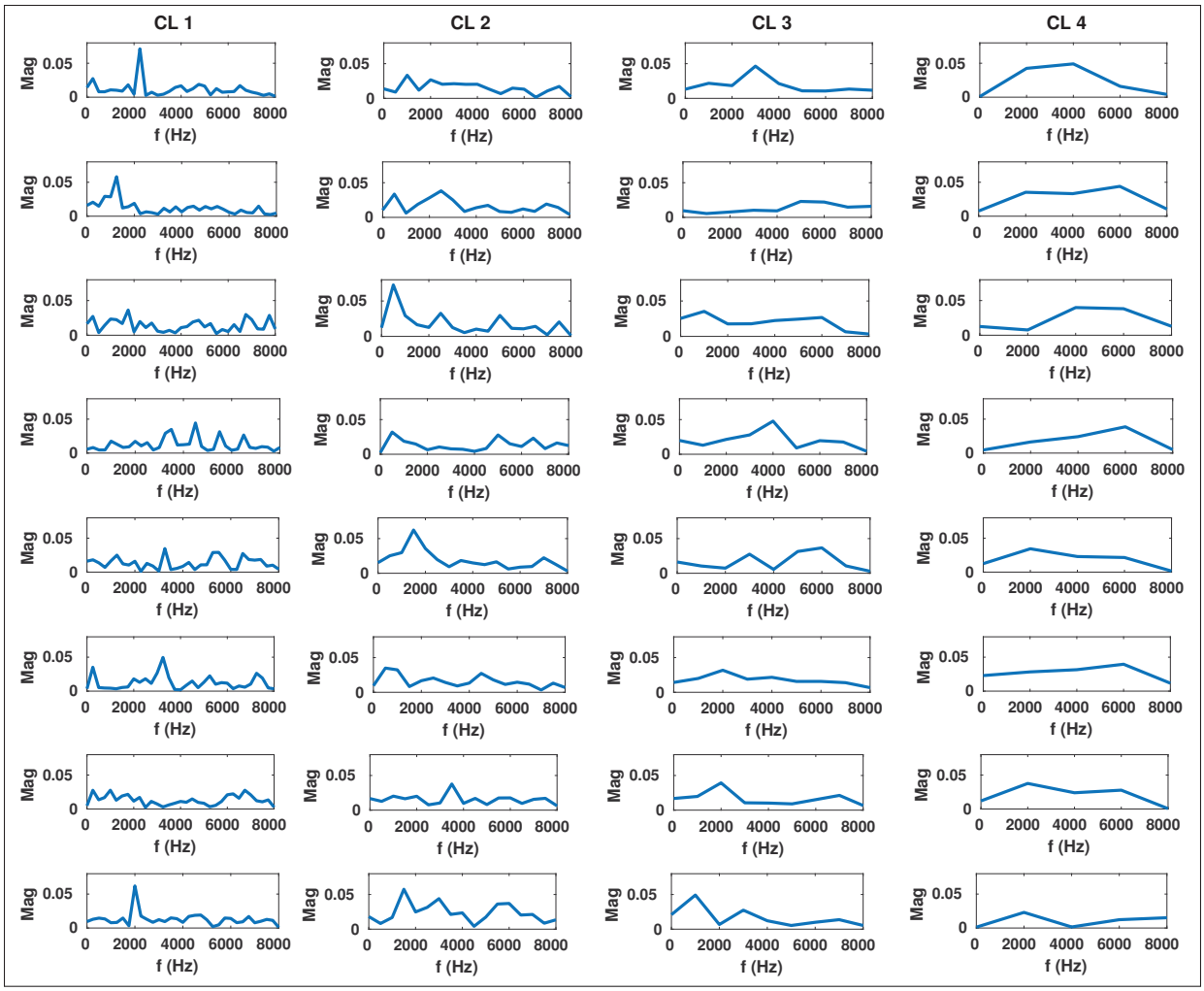


Figure 4.2 Fourier transform of randomly selected filters from the four convolutional layers (CLs) of the proposed 16,000-input 1D CNN shown in Figure 2.2

architectures presented in Table 2.1. In spite of that, the 16,000-input 1D CNN remains the one with the highest mean accuracy.

Figure 4.3 shows the Fourier transform of some of the filters of the enhanced model with a non-trainable Gammatone filter-bank. Similar to the filters of the original model (Figure 4.2), the filters of the deepest layers (CL3 and CL4) have a well-defined frequency response. Filters of the intermediate layer (CL2) still do not exhibit dominant frequency levels. Even though the minor changes in the responses of the intermediate and deeper filters, the Gammatone filters of the first layer were useful to improve the mean accuracy of the proposed 1D CNN.

Table 4.2 Enhancement of 1D CNN model for ESC. Improvements in the mean accuracy for the 16,000-input 1D CNN on the UrbanSound8k dataset

CL1 Initialization	**Window**	**Overlapping**	**Combination Rule**	**Mean Accuracy**	**# of Parameters**	**Computational Time (Sec)**
Randomly	Hamming	50%	Sum Rule	83%	256,538	1.863
Randomly	Rectangular	50%	Sum Rule	85%	256,538	1.863
Gammatone	Rectangular	50%	Sum Rule	87%	550,506	6.099
Randomly	Rectangular	75%	Sum Rule	87%	256,538	1.863
Gammatone	Rectangular	75%	Sum Rule	**89%**	550,506	6.099

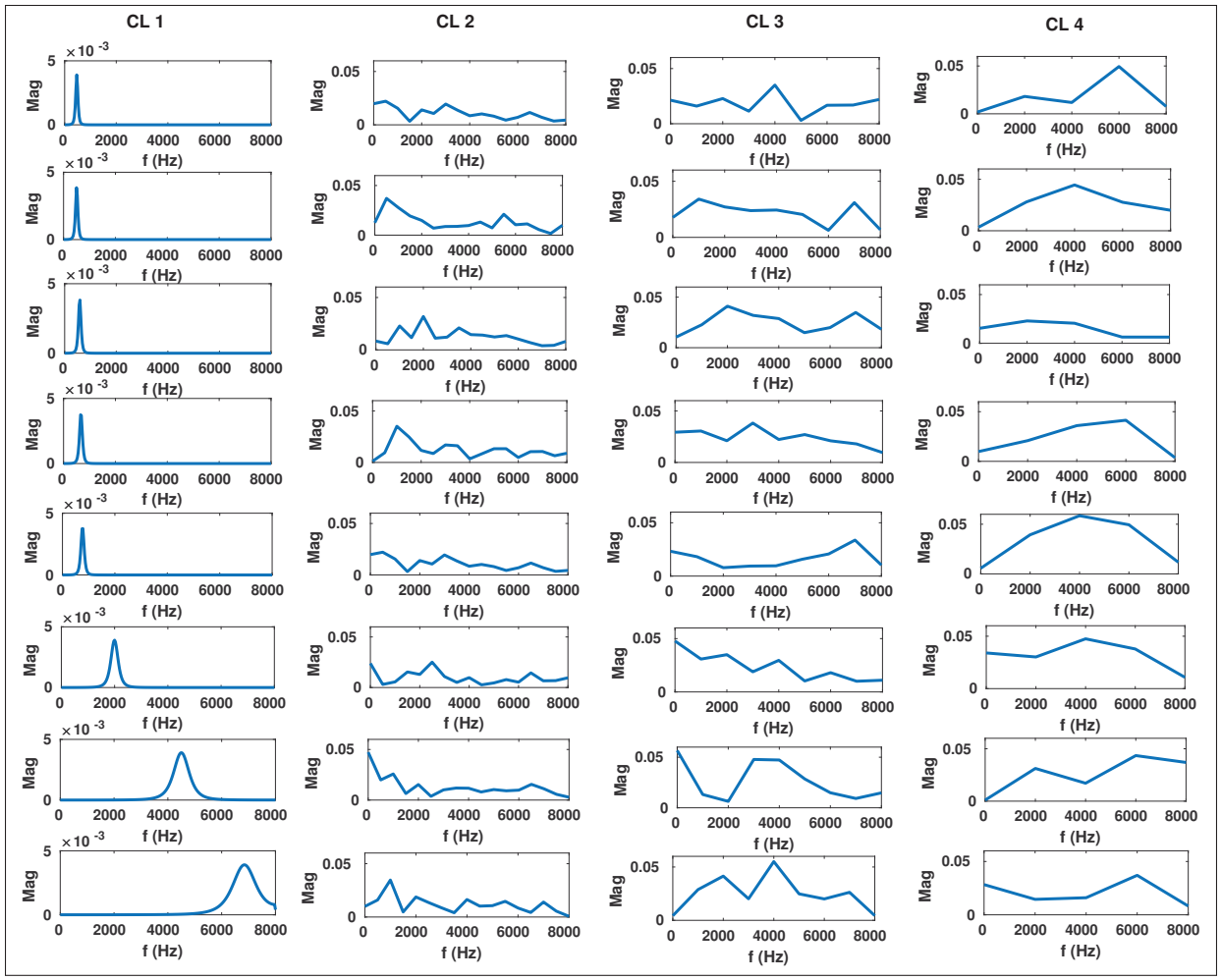


Figure 4.3 Fourier transform of randomly selected filters from the four convolutional layers (CLs) of the 16,000-input 1D CNN with Gammatone filter-bank in the first convolutional layer of the network

4.1.4 Discussion

Table 4.3 shows the mean classification accuracy achieved by the proposed 1D CNN as well as the results achieved by other state-of-the-art approaches described in the literature. The proposed 1D CNN achieved a mean accuracy of 89% with a standard deviation of only 0.9% across the 10 folds. The proposed 1D CNN, the RawNet (Li, Yao, Hu, Liu, Yao & Hu, 2018), the EnvNet-v2 (Tokozume *et al.*, 2018) and the M18 CNN (Dai *et al.*, 2017) are end-to-end architectures, which learn the representation directly from the audio waveform. DS-CNN is a combinational model, which uses both raw audio signal and 2D representations as input to a CNN. All other approaches in Table 4.3 use 2D representations of the audio signal as input. As it is shown in Table 4.3, the proposed approach has lower number of parameters than most of the state-of-the-art approaches described in the literature and therefore it requires a relatively few number of samples for appropriate training. Furthermore, it is shown that the proposed algorithm outperforms all other approaches which use raw audio signal as input to the CNN. Therefore, the proposed approach is a quite suitable candidate to be used in ensemble models as described by Li *et al.* (2018). Figure 4.4 also compares the proposed 1D CNN with other approaches on UrbanSound8k dataset for environmental sound classification using a boxplot generated from the accuracy scores of 10 folds. Note that for some models the information about the accuracy scores of 10 folds was not available. So, only mean accuracy of the models are reported.

The proposed approach also does not require any signal processing module for feature extraction from audio signal. Therefore, it is suitable to be used in mobile or embedded devices. Moreover, as mentioned by Boddapati, Petef, Rasmusson & Lundberg (2017), the operation of generating 2D representations from audio signal is time-consuming. For instance, producing spectrograms of ESC-50 (Piczak, 2015a) dataset which consists of 2,000 samples takes five minutes. Generating corresponding MFCC features also takes five minutes. In addition to that, producing CRP representations takes 24 hours. Also, 2D representations can not be computed on GPU due to lack of suitable libraries. This issue makes models based on 2D representations impractical for real-time applications.

Salamon & Bello (2017) proposed several data augmentation techniques for training the SB-CNN model such as *"Time Stretching"*, *"Pitch Shifting"*, *"Dynamic Range Compression"* and adding *"Background Noise"*. In this research, all of the techniques, with the parameter setup proposed by Salamon & Bello (2017), and the combination of them were used to augment the samples of the dataset. However, all of them had a detrimental effect on the accuracy of the proposed model for ESC in this research. For the sake of brevity, the results for the augmented dataset are not reported.

Moreover, approaches based on 2D representations are much more vulnerable to adversarial attacks which can easily fool these models. As it is shown by Esmaeilpour, Cardinal & Koerich (2019), the models based on 2D representations can be easily fooled by adversarial attacks originally designed to fool image processing models. They have also pointed out that generalizing the current attacks to raw audio signals is not feasible because of the high-dimensionality of raw audio signals.

Table 4.3 Mean accuracy of different approaches on the UrbanSound8k dataset

Approach	**Representation**	**Mean Accuracy**	**# of Parameters**
TSCNN-DS (Su, Zhang, Wang & Madani, 2019)	2D	97%	15.9 M
GoogLeNet (Boddapati *et al.*, 2017)	2D	93%	6.7 M
MelNet (Li *et al.*, 2018)	2D	90%	211 k
SB-CNN (DA) (Salamon & Bello, 2017)	2D	79%	241 k
SKM (DA) (Salamon & Bello, 2015)	2D	76%	NA
SKM (Salamon & Bello, 2015)	2D	74%	NA
PiczakCNN (Piczak, 2015b)	2D	73%	26 M
SB-CNN (Salamon & Bello, 2017)	2D	73%	241 k
VGG (Pons & Serra, 2018)	2D	70%	77 M
DS-CNN (Li *et al.*, 2018)	1D-2D	92%	NA
Proposed 1D CNN Gamma	**1D**	**89%**	**550 k**
RawNet (Li *et al.*, 2018)	1D	87%	377 k
Proposed 1D CNN Rand	1D	87%	256 k
EnvNet-v2 (Tokozume *et al.*, 2018)	1D	78%	101 M
M18 CNN (Dai *et al.*, 2017)	1D	72%	3.7 M

NA: Not available. DA: With data augmentation.

Figure 4.5 shows the confusion matrix of the proposed end-to-end 1D CNN on the UrbanSound8k dataset. Values along the diagonal indicate the number of samples classified correctly for each

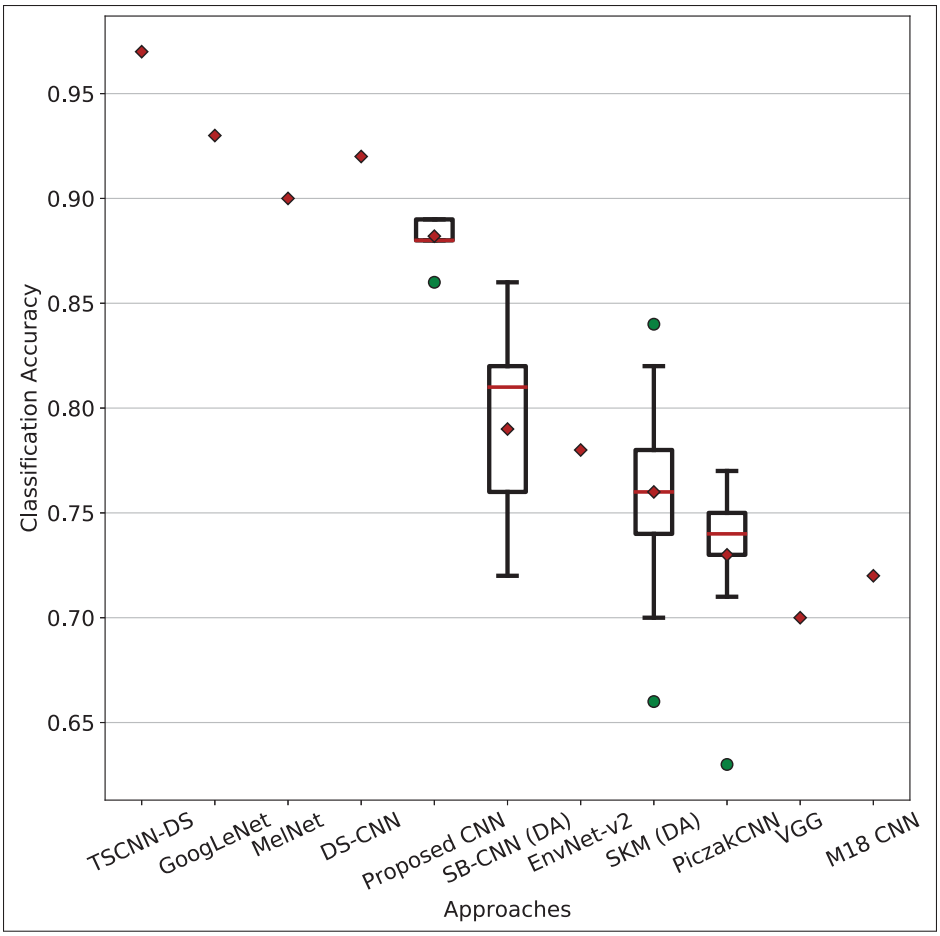


Figure 4.4 Classification accuracy of the proposed 1D CNN as well as the results obtained by other state-of-the-art approaches Adapted from (Salamon & Bello, 2017)

specific class. It shows that the ST and CH classes are the most challenging classes for the CNN. However, EN and GU classes are well separated by the proposed CNN.

4.1.4.1 Filter response

The magnitude responses of the convolutional filters of the first layer of the proposed 1D CNN are shown in Figure 4.6. To obtain a better image representation of the frequency response, the number of kernels in the first layer has been increased to 64 (compared to 16 in the one used in the experiments). Note that this configuration led to a slight decrease in the classification

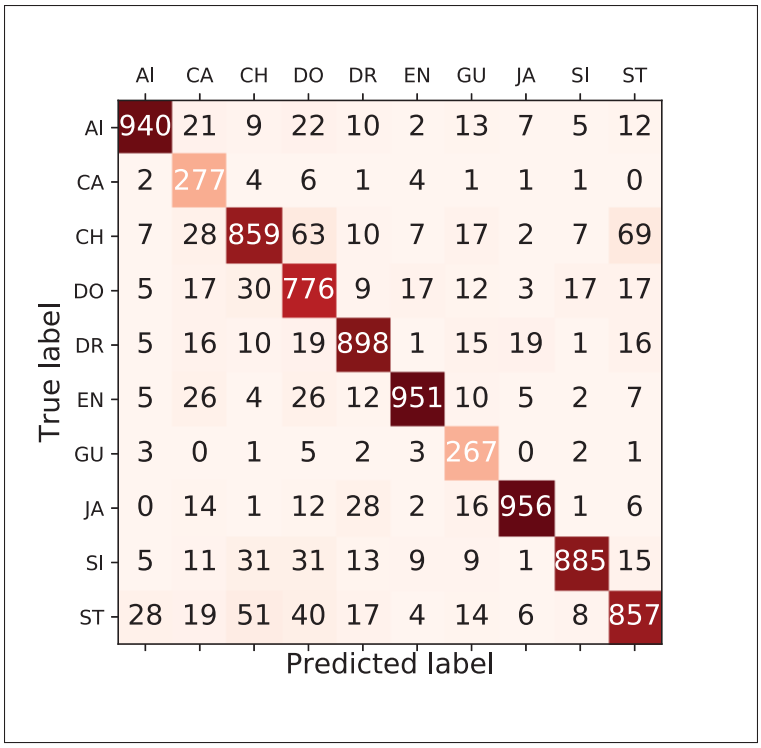


Figure 4.5 Confusion matrix for the proposed end-to-end 1D CNN

accuracy. Figures 4.6(a) and 4.6(b) show the response of the filters after convergence and the response of the kernels sorted based on their central frequencies, respectively. The central frequency of each kernel is computed by computing the Fast Fourier transform of the filter and by selecting the frequency bin with the highest peak. Each row in the image is created by feeding the network with a sinusoidal wave with a specific frequency. For such an aim, sinusoidal waves in the range of 1 Hz to 8 kHz, with a step of 100 Hz, have been used. The feature map of the first convolutional layer is first obtained and then, it is computed the average of the feature map along the time axis. Figure 4.6(c) shows the output of 64 Gammatone filters used as band-pass filters.

From Figure 4.6(b), it can be seen that the learned filters have a logarithmic response similar to band-pass filters created using cardinal sinusoidal functions. In addition, this behavior is also similar to how humans perceive sounds, which is also logarithmic (Roederer, 2008). A similar

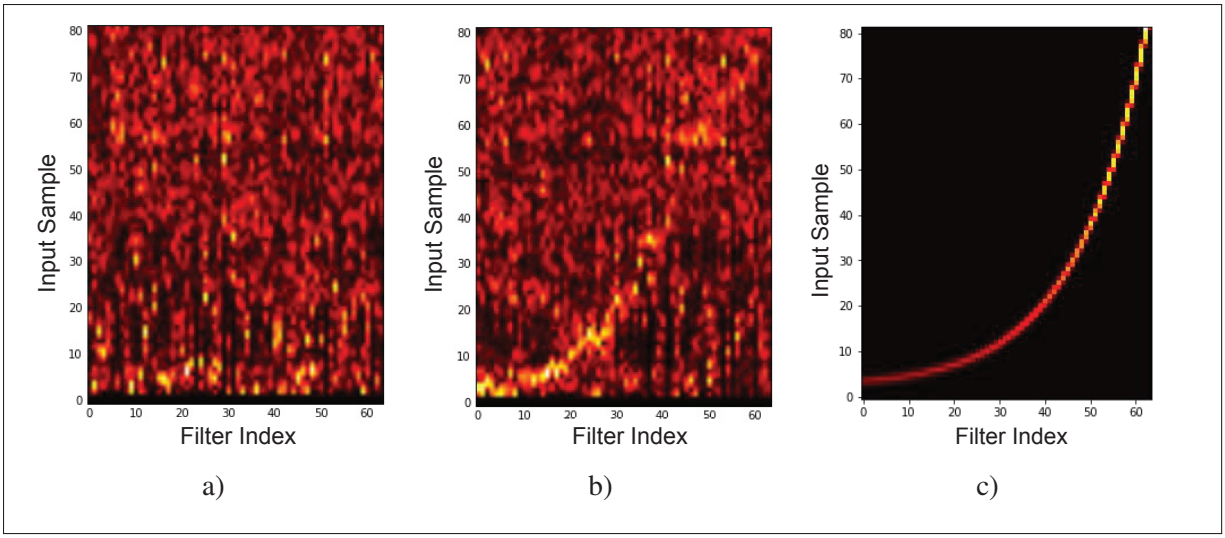


Figure 4.6 Magnitude response of the convolutional filters of the first layer of the end-to-end 1D CNN: (a) response of the filters after convergence; (b) response of the kernels after sorting based on their central frequency; (c) frequency response of band-pass filters. Center frequency of filters are selected according to constant Q transform rules (Brown, 1991)

behavior has also been observed in other end-to-end systems for audio processing tasks (Hoshen *et al.*, 2015; Sainath *et al.*, 2015; Tokozume & Harada, 2017).

In this section, an end-to-end 1D CNN for ESC has been proposed. The architecture of the network consists of three to five convolutional layers, depending on the length of the audio signal. Instead of using handcrafted static filter-banks such as those used to extract MFCC features, the proposed 1D CNN learns the filters directly from the audio waveform. The proposed approach was evaluated on a dataset of 8,732 audio samples and the experimental results have shown that the proposed end-to-end approach learns several relevant filter representations which allows it to outperform other state-of-the-art approaches based on 2D representations and 2D CNNs. Furthermore, the proposed end-to-end 1D architecture has fewer parameters than most of the other CNN architectures for ESC, while achieving mean accuracy that is between 11.24% and 27.14% higher than such 2D architectures.

This section presented the experiments and the evaluation results of the proposed end-to-end model based on 1D CNN for environmental sound classification. The next section explains the experimental protocol for evaluating the UAPs for attacking a family of audio processing models. The results will be discussed and the strengths and downsides of the proposed methods will be mentioned.

4.2 Experimental results on UAPs

The proposed methods on UAP generation are evaluated on two audio tasks: ESC and speech command recognition. For ESC, we have used the complete audio recordings of UrbanSound8k dataset (Salamon *et al.*, 2014) downsampled to 16 kHz for training and evaluating the models and generating adversarial perturbations. The dataset was split into training (80%), validation (10%) and test (10%) set. The target models are trained and validated by using training and validation sets from scratch. No pre-trained model is used. For generating perturbations, 1,000 samples of the training set were randomly selected, and for the penalty-based method a mini-batch size of 100 samples is used. The perturbations were evaluated on the whole test set (874 samples).

We have used the Speech Command dataset for speech command recognition, consisting of 61.83 hours of audio sampled at 16 kHz (Warden, 2018) and categorized into 32 classes. The training set consists of 17.8 hours of speech command recordings split into 64,271 audio clips of one second. The test set consists of 158,537 audio samples corresponding to 44.03 hours of speech commands. This dataset was used as the benchmark dataset for TensorFlow Speech Recognition (TFSR) challenge in 2017[4]. This challenge was based on the principles of open science so, data, source code, and evaluation methods of over 1,300 participant teams are publicly available for commercial and non-commercial usage. So, as it is shown in this study, it is quite straightforward for the adversary to attack the model.

SNR is used as a metric to measure the level of noise concerning the legitimate signal. This metric, which is also measured in *dB*, is used for measuring the level of the perturbation of the

[4] https://www.kaggle.com/c/tensorflow-speech-recognition-challenge

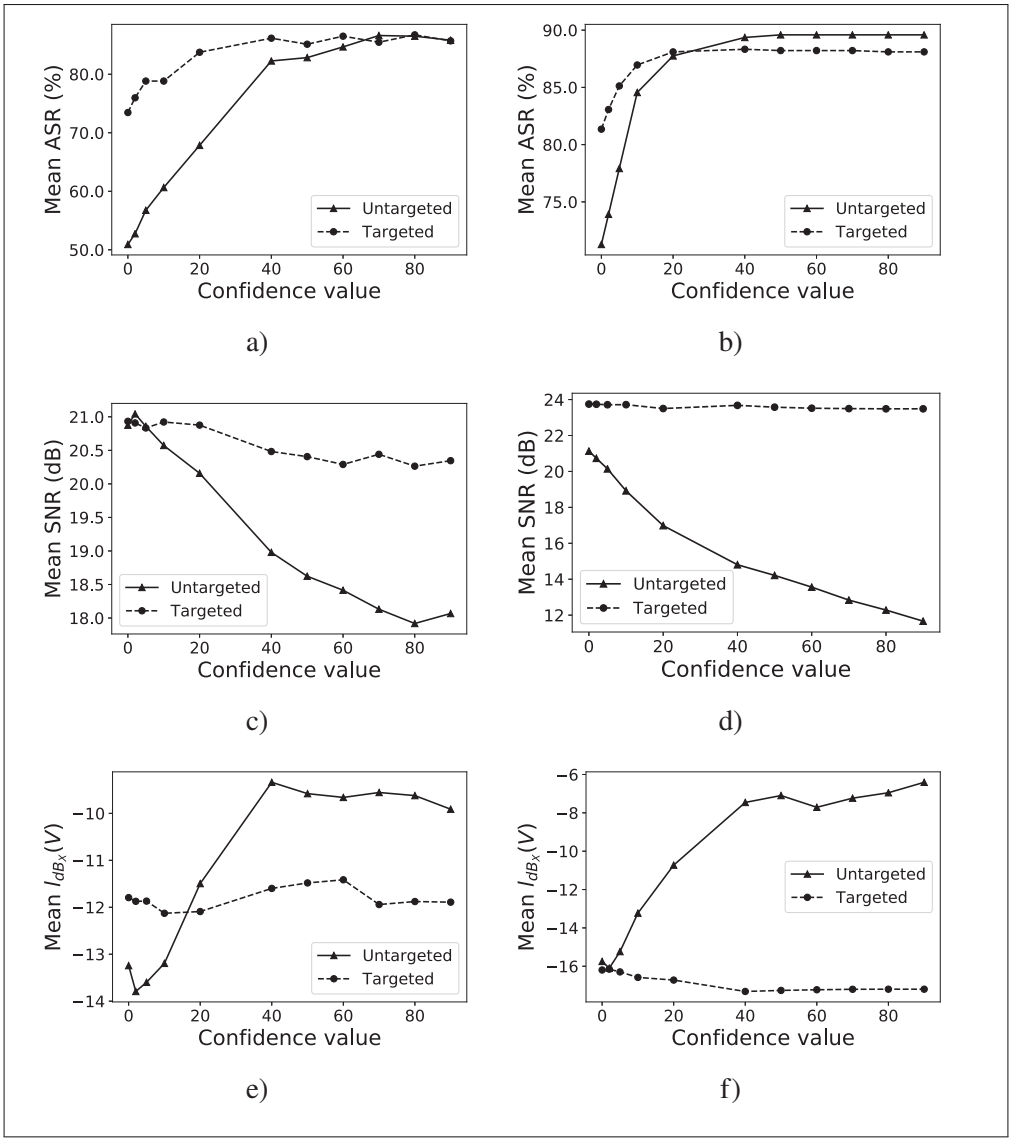


Figure 4.7 Effect of different confidence values on the mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ for targeted and untargeted attacks. (a, c, e): ENVnet-V2 (Tokozume *et al.*, 2018) used as target model. (b, d, f): 1D CNN Gamma presented in Chapter 2 used as target model

signal after adding the universal perturbation. This measure is also used in previous works for evaluating the quality of the generated adversarial audio attacks (Kereliuk *et al.*, 2015; Du *et al.*, 2020), and it is defined as:

$$\mathrm{SNR}(\mathbf{x}, \mathbf{v}) = 20 \log_{10} \frac{P^{sig}(\mathbf{x})}{P^{sig}(\mathbf{v})}, \tag{4.1}$$

where $P^{sig}(.)$ is the power of the signal defined in Eq. (3.6). A high SNR indicates that a low noise level is added to the audio sample by the universal adversarial perturbation. Additionally, the relative loudness of perturbation with respect to audio sample, measured in *dB*, is also applied as:

$$l_{\mathrm{dB_x}}(\mathbf{v}) = l_{\mathrm{dB}}(\mathbf{v}) - l_{\mathrm{dB}}(\mathbf{x}) \tag{4.2}$$

where $l_{dB}(\mathbf{x}) = \max_n(20\log_{10}(x_n))$ and x_n denotes the n-th component of the array $\mathbf{x}$. This measure is similar to the ℓ_∞ norm in the image domain. Smaller values indicate quieter distortions. This measure is also applied in recent studies for quality assessment of adversarial attacks against audio classification models (Neekhara *et al.*, 2019; Carlini & Wagner, 2018; Yang *et al.*, 2019).

We have chosen a family of diverse end-to-end architectures as our target models. This selection is based on selecting architectures that might learn representations directly from the audio signal. We briefly describe the architecture of each model as follows. 1D CNN Rand, 1D CNN Gamma, ENVnet-V2, SincNet and SincNet+VGG19 are just used for ESC while the SpchCMD model is used for speech command recognition. A detailed description of the 1D CNN Rand model is presented in Table 2.1 of Chapter 2. The description of the other architectures can also be found in Appendix IV.

- **1D CNN Rand:** This model is presented in Chapter 2 and it is based on 1D CNN for ESC. The weights of all of the layers are initialized randomly.
- **1D CNN Gamma:** This model is also presented in Chapter 2 and it is similar to 1D CNN Rand. The only difference is that it employs a non-trainable Gammatone filter-bank in its first layer.
- **ENVnet-V2 (Tokozume *et al.*, 2018):** The architecture for sound recognition was slightly modified to make it compatible with the input size of the downsampled audio samples of the 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 features in time and the frequency domains using five

two-dimensional CLs. Two FC layers and an output layer with a softmax activation function complete the network.

- **SincNet (Ravanelli & Bengio, 2018):** The end-to-end architecture for sound processing extracts meaningful features from the audio signal at its first layer. Several sinc functions are used as band-pass filters in this model, 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 (Ravanelli & Bengio, 2018):** This model uses sinc filters to extract features from the raw audio signal as SincNet (Ravanelli & Bengio, 2018). After a one-dimensional maxpooling layer, the output is stacked along the time axis to form a 2D representation. This time-frequency representation is used as the input to a VGG19 network (Simonyan & Zisserman, 2015) followed by an FC layer and an output layer with softmax activation for classification. This time-frequency representation resembles a spectrogram representation of the audio signal.
- **SpchCMD**: This architecture was proposed by the winner of the TFSR challenge. The model is based on a CNN, which uses one-dimensional CLs and several depth-wise CLs for extracting useful information from several chunks of raw audio. The model also uses an attention layer and a global average PL. According to the challenge rules, the model must handle silence signal and unknown samples from other proposed classes in the Speech Command dataset (Warden, 2018). Therefore, the softmax layer has 32 outputs.

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 (Rony *et al.*, 2019). This attack is designed to find small perturbations that fool the model efficiently. The difference with DeepFool (Moosavi-Dezfooli *et al.*, 2016) 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 set to 40 and 10 for crafting untargeted and targeted perturbations, respectively. For both methods, based on our initial experiments, we found that these values are appropriate to produce fine quality perturbed samples within a reasonable number of iterations. Figure 4.7 shows the effect of different confidence values on mean ASR, mean SNR and mean $l_{dB_{\mathbf{x}}}(\mathbf{v})$ for targeted and untargeted attacks on the ENVnet-V2 (Tokozume *et al.*, 2018) and 1D CNN Gamma, presented in Chapter 2, models 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"*. Figure 4.7 shows that the ASR increases as the confidence value increases. However, the SNR also decreases in the same way. For untargeted attacks, it has a more detrimental effect on the mean $l_{dB_{\mathbf{x}}}(\mathbf{v})$ than the targeted scenario. For both iterative and penalty methods, we set the desired fooling rate on perturbed training samples to δ=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.

4.2.1 Results

Table 4.4 shows the results of the iterative and penalty methods against the five ESC 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 and mean $l_{\mathrm{dB}_{\mathbf{x}}}(\mathbf{v})$ of the perturbed samples of the test set. For crafting the perturbation vector, 1,000 randomly chosen samples of the training set are used. 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. For the targeted scenario, the penalty method produces better results in terms of mean $l_{\mathrm{dB}_{\mathbf{x}}}(\mathbf{v})$ for attacking SincNet and SincNet+VGG19 models. The iterative method, however, works slightly better for attacking the ENVnet-V2 model. For the rest of the models, the difference is negligible. The projection operation on ℓ_∞ ball used in the iterative method is like clipping the perturbation vector. Since the maximum amplitude of the perturbation vector surpasses the limit (ξ= 0.12)

while producing the perturbation, the projection operation causes the method to achieve the same mean $l_{\mathrm{dB_x}}(\mathbf{v})$ for almost all of the models in this attacking scenario. The penalty method also produces quieter perturbations for the untargeted scenario in terms of mean $l_{\mathrm{dB_x}}(\mathbf{v})$ for attacking 1D CNN Gamma, SincNet and SincNet+VGG19 models. The crafted universal perturbations produced from the training 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 are reported in Appendix III. For a better assessment of the perturbations produced by both proposed methods, several randomly chosen examples of perturbed audio samples are presented in Appendix V and are available for listening from the website[5].

Table 4.5 shows the results achieved by iterative and penalty methods for targeting the SpchCMD model. In order to find the universal perturbation vector, 3,000 samples from the training set were randomly selected. All samples of the test set were used to evaluate the effectiveness of the perturbation vector. For targeting this model, the perturbation generated by the penalty method is also projected around the ℓ_2 ball according to Equation 3.2 with radius ξ=6. Based on the preliminary experiments for this task, this projection produces slightly better results in terms of mean SNR and mean $l_{\mathrm{dB_x}}(\mathbf{v})$. Both methods produce similar results for the targeted attack scenario in terms of in terms of ASR and SNR. However, the penalty method generates higher-quality perturbation vectors in terms of mean $l_{\mathrm{dB_x}}(\mathbf{v})$. For untargeted attacks, the penalty method produces better results in terms of mean ASR and mean $l_{\mathrm{dB_x}}(\mathbf{v})$ and both methods produce similar perturbations in terms of SNR. Roughly speaking, SNRs higher than 20 are considered acceptable ones. As mentioned by Yang *et al.* (2019), perturbed audio samples that have $l_{\mathrm{dB_x}}(\mathbf{v})$ ranging from -15 dB to -45 dB are tolerable to human ears. From Tables 4.4 and 4.5, the mean SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ of the perturbed samples are mostly in this range. Neekhara *et al.* (2019) also reported relatively similar results (between -29.82 dB and -41.86 dB) for attacking a speech-to-text model in an untargeted scenario. The model predictions after adding the perturbations were submitted to the evaluation system of the TFSR challenge to evaluate the

[5] https://sajabdoli.netlify.app/publication/uap/

Table 4.4 Mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ for attacking environmntal sound classification systems on training and test sets for untargeted and targeted perturbations. Higher ASRs are in boldface

Panel (A): Targeted attack					
		Training Set	**Test Set**		
Method	**Model**	**ASR**	**ASR**	**SNR (dB)**	$l_{\mathbf{dB_x}}(\mathbf{v})$
Iterative	1D CNN Rand	0.926	0.672	25.321	-18.416
	1D CNN Gamma	0.945	0.795	23.587	-18.416
	ENVnet-V2	0.916	0.767	22.734	-18.416
	SincNet	1.000	0.899	28.668	-21.044
	SincNet+VGG19	0.985	0.872	26.203	-18.416
Penalty	1D CNN Rand	0.917	**0.854**	23.468	-16.762
	1D CNN Gamma	0.913	**0.888**	22.835	-17.375
	ENVnet-V2	0.922	**0.877**	21.832	-14.198
	SincNet	0.962	**0.971**	30.411	-31.916
	SincNet+VGG19	0.916	**0.898**	26.736	-21.059
Panel (B): Untargeted attack					
		Training Set	**Test Set**		
Method	**Model**	**ASR**	**ASR**	**SNR (dB)**	$l_{\mathbf{dB_x}}(\mathbf{v})$
Iterative	1D CNN Rand	0.911	0.412	25.244	-14.258
	1D CNN Gamma	0.904	0.737	22.922	-13.979
	ENVnet-V2	0.910	0.669	24.960	-13.979
	SincNet	0.915	0.886	24.025	-18.959
	SincNet+VGG19	0.924	0.838	26.362	-14.007
Penalty	1D CNN Rand	0.900	**0.876**	20.350	-13.371
	1D CNN Gamma	0.901	**0.858**	20.551	-18.133
	ENVnet-V2	0.900	**0.831**	18.727	-10.490
	SincNet	0.900	**0.919**	29.972	-27.494
	SincNet+VGG19	0.902	**0.865**	23.555	-17.759

impact of the perturbations on the model accuracy. Table 4.5 shows that the accuracy of the winner speech recognition model (Model Acc.) has dropped to near of random guessing for the perturbations produced by both proposed algorithms. The success rates reported in Tables 4.4 and 4.5 are also statistically significant. Refer to Appendix II for details.

We now consider the influence of the number of training data points on the quality of the universal perturbations. Figure 4.8 shows the ASR and mean SNR achieved on the test set with different number of data points, and considering two target models. The untargeted attack is evaluated on SincNet and the targeted attack is evaluated on the 1D CNN Gamma model.

Table 4.5 Mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ for targeting speech recognition model (SpchCMD) on training and test sets for untargeted and targeted perturbations . Higher ASRs are in boldface

	Targeted Attack				**Untargeted Attack**				
	Training Set	**Test Set**			**Training Set**	**Test Set**			
Method	**ASR**	**ASR**	**SNR (dB)**	$l_{\mathrm{dB_x}}(\mathbf{v})$	**ASR**	**ASR**	**SNR (dB)**	$l_{\mathrm{dB_x}}(\mathbf{v})$	**Model Acc.**
Iterative	0.902	**0.855**	27.437	-18.924	0.901	0.834	28.524	-18.418	0.192
Penalty	0.903	0.850	26.728	-24.575	0.910	**0.875**	26.716	-21.462	0.191

For targeted attacks, the target class is *"Gun shot"*. For both targeted and untargeted scenarios, the penalty method produces better ASR when the perturbations are crafted with fewer data points. The iterative method produces perturbations with a slightly better mean SNR than those produced by the penalty method for the untargeted scenario. However, this difference is perceptually negligible. When the number of data points is limited, the penalty method also produces better results in terms of mean $l_{\mathrm{dB_x}}(\mathbf{v})$. However, the iterative method produces slightly better results in terms of mean $l_{\mathrm{dB_x}}(\mathbf{v})$ when more data points are available (e.g. more than 50 samples). For the targeted attack scenario, when the number of data points is limited (e.g. lower than 100), 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. For such a scenario, the penalty method also produces quieter perturbations in terms of mean $l_{\mathrm{dB_x}}(\mathbf{v})$. 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, the attacker can't obtain higher ASRs at the expense of having a universal perturbation with slightly higher SPL, especially when the number of samples is limited. However, in the penalty method, the algorithm can 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 Equations 3.10 and 3.11. 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 can also 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 Equations 3.10 and 3.11 are minimized iteratively using Algorithm 3.1 for targeted and untargeted attacks, respectively. We set the penalty coefficient c=0.2 and the confidence value κ=90 for both untargeted and targeted attack scenarios for this experiment. The algorithm is executed for 19 iterations and the perturbation produced is used to perturb all audio samples of the test set, which are then used to fool the 1D CNN Gamma model.

Table 4.6 shows the results of the untargeted attack in terms of ASR and SNR on the perturbed samples of the test set. Figure 4.9 shows the results of the targeted attack in terms of SNR, ASR and mean $l_{\mathrm{dB_x}}(\mathbf{v})$, 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 0.698 and 0.602 are achieved for untargeted and targeted attack scenarios, respectively. Moreover, mean SNR of 18.489 dB and 19.690 dB are also achieved for untargeted and targeted attack scenarios, respectively. Mean $l_{\mathrm{dB_x}}(\mathbf{v})$ of -15.844 dB and -15.571 dB are also achieved for untargeted and targeted attack scenarios, respectively. Similar results were also obtained for all other target models.

Table 4.6 Results of the untargeted attack in terms of mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ on the test set. The perturbation is generated by a single randomly selected sample of each class. The proposed penalty method is used to generate attacks against the 1D CNN Gamma model, presented in Chapter 2

	Available Class										
	AI	**CA**	**CH**	**DO**	**DR**	**EN**	**GU**	**JA**	**SI**	**ST**	**Mean**
ASR	0.641	0.722	0.730	0.732	0.720	0.561	0.688	0.796	0.677	0.713	0.698
SNR	17.718	18.317	19.597	18.813	16.699	19.039	17.908	20.046	18.014	18.743	18.489
$l_{\mathrm{dB_x}}(\mathbf{v})$	-15.130	-16.574	-16.322	-16.052	-15.048	-16.963	-15.672	-15.510	-15.088	-16.086	-15.844

Finally, Table 4.7 shows the transferability of perturbed audio samples of the test set of UrbanSound8k dataset generated by one model to other models. Generating transferable adversarial perturbations among the proposed models is a challenging task (Abdullah *et al.*, 2020; Subramanian, Benetos, Xu, McDonald & Sandler, 2019). Abdullah *et al.* (2020) have

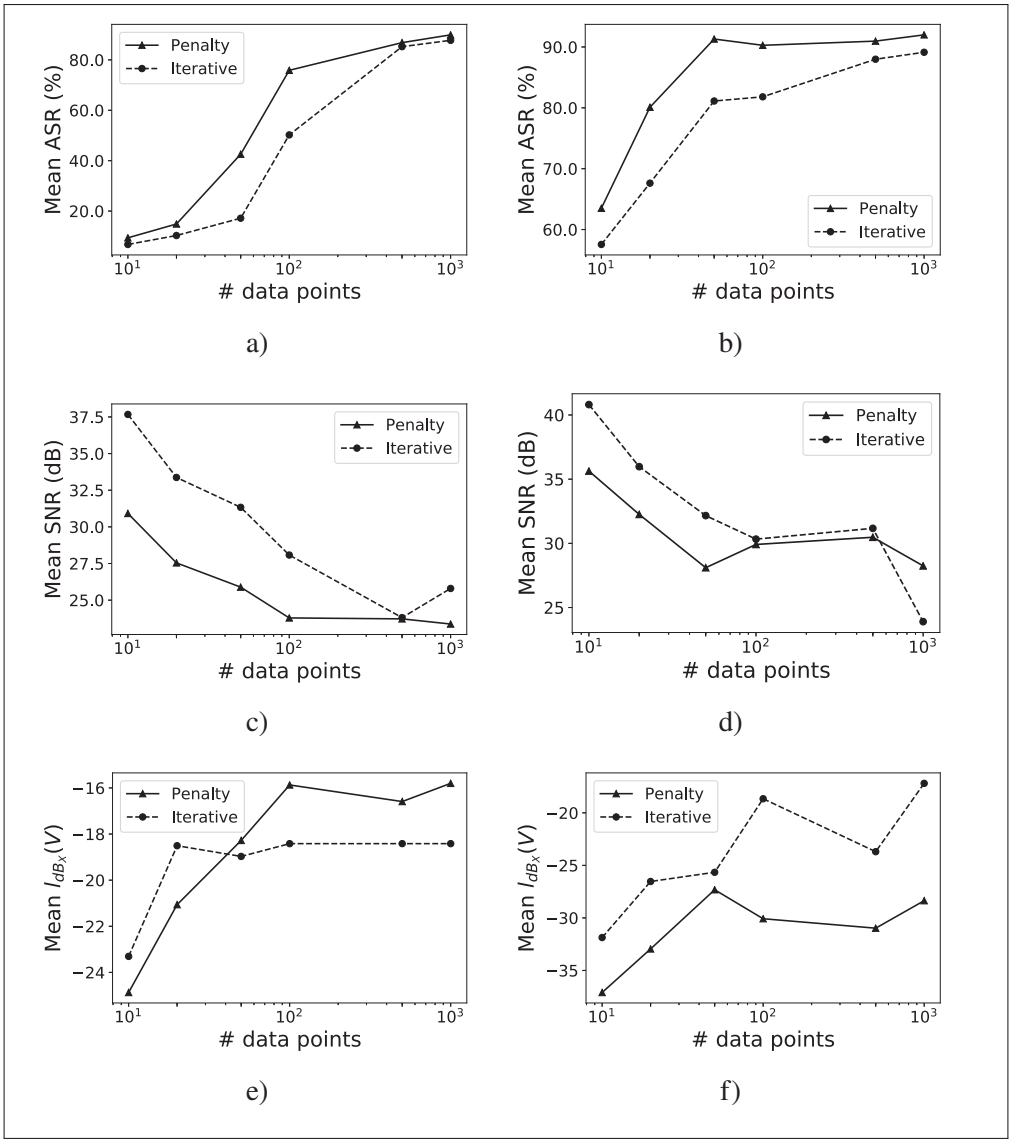


Figure 4.8 Effect of the number of data points on the mean values of ASR, SNR and $l_{\text{dB}_\mathbf{x}}(\mathbf{v})$ for the test set. (a, c, e): Targeted attack on the 1D CNN Gamma model, presented in Chapter 2. (b, d, f): Untargeted attack on the SincNet model (Ravanelli & Bengio, 2018)

shown that transferability is challenging for most of the current audio attacks. This problem is still more challenging to gradient-based optimization attacks. Liu, Chen, Liu & Song (2017) introduced an ensemble-based approach that seems to be a promising research direction to produce more transferable audio examples.

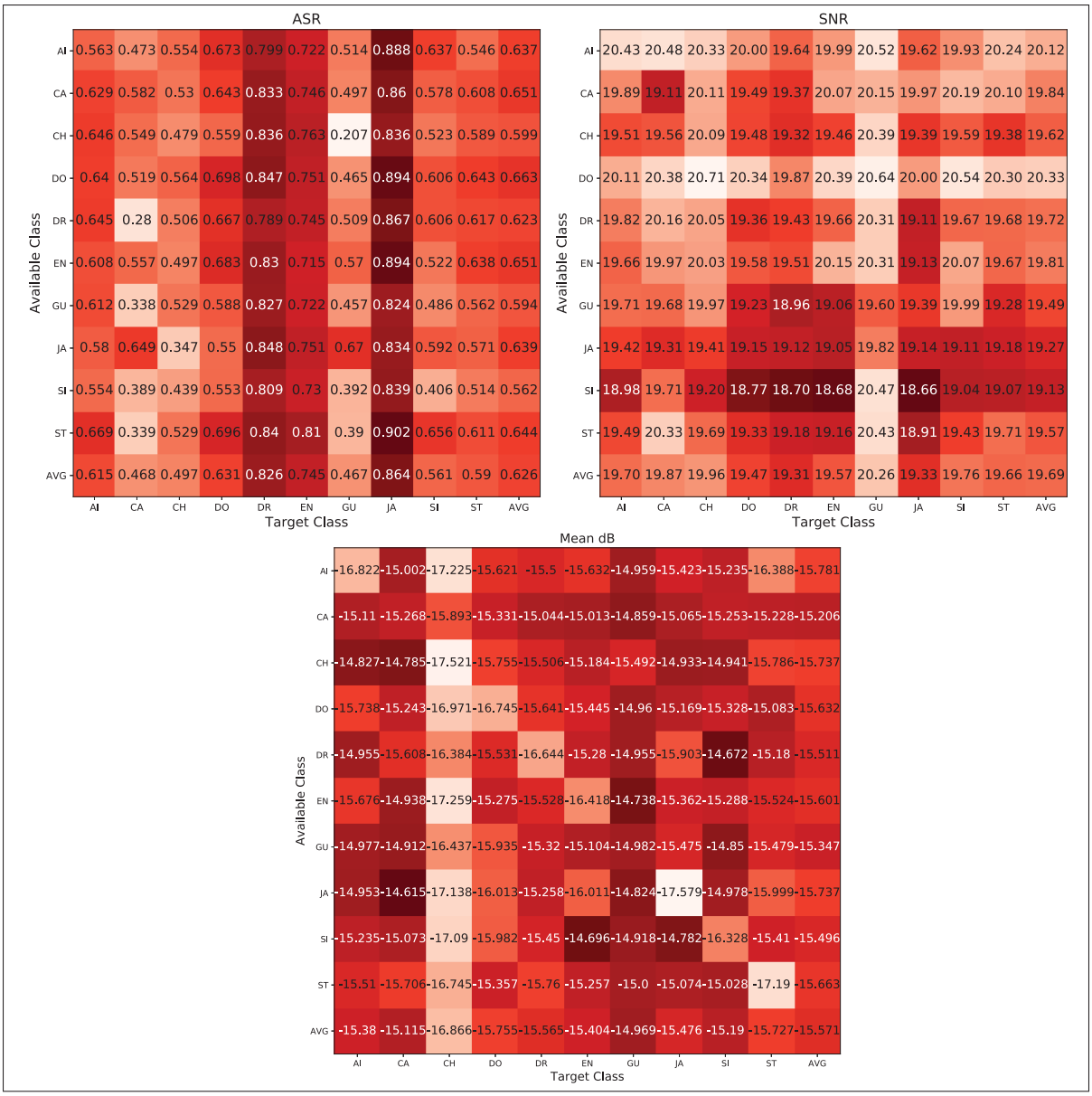


Figure 4.9 Targeted mean values of ASR (a), SNR (b) and $l_{\mathrm{dB_x}}(\mathbf{v})$ (c) on the test set. Proposed penalty method is used for crafting the perturbation. The perturbation is generated by a single randomly selected sample of each class in order to attack the 1D CNN Gamma model, presented in Chapter 2, for a specific target class

In this chapter, the proposed methods for ESC and the proposed methods for generating targeted and untargeted UAPs for attacking audio classification models were evaluated and comprehensive

Table 4.7 Transferability of adversarial samples generated by both iterative and penalty methods between pairs of the models for untargeted and targeted attacking scenarios. The cell (i, j) indicates the ASR of the adversarial sample generated for model i (row) evaluated over model j (column)

Panel (A): Untargeted attack						
Method	**Model**	**1D CNN Rand**	**1D CNN Gamma**	**ENVnet-V2**	**SincNet**	**SincNet+VGG19**
Iterative	1D CNN Rand	N/A	0.371	0.175	0.300	0.200
	1D CNN Gamma	0.340	N/A	0.245	0.627	0.530
	ENVnet-V2	0.269	0.339	N/A	0.304	0.310
	SincNet	0.164	0.207	0.152	N/A	0.237
	SincNet+VGG19	0.176	0.191	0.134	0.228	N/A
Penalty	1D CNN Rand	N/A	0.558	0.342	0.675	0.388
	1D CNN Gamma	0.423	N/A	0.295	0.698	0.672
	ENVnet-V2	0.390	0.503	N/A	0.490	0.479
	SincNet	0.142	0.170	0.103	N/A	0.185
	SincNet+VGG19	0.253	0.192	0.194	0.340	N/A
Panel (B): Targeted attack						
Method	**Model**	**1D CNN Rand**	**1D CNN Gamma**	**ENVnet-V2**	**SincNet**	**SincNet+VGG19**
Iterative	1D CNN Rand	N/A	0.352	0.186	0.534	0.232
	1D CNN Gamma	0.285	N/A	0.210	0.476	0.423
	ENVnet-V2	0.292	0.340	N/A	0.327	0.356
	SincNet	0.126	0.131	0.105	N/A	0.215
	SincNet+VGG19	0.185	0.149	0.172	0.237	N/A
Penalty	1D CNN Rand	N/A	0.119	0.192	0.130	0.109
	1D CNN Gamma	0.130	N/A	0.117	0.251	0.134
	ENVnet-V2	0.127	0.153	N/A	0.146	0.159
	SincNet	0.103	0.106	0.103	N/A	0.129
	SincNet+VGG19	0.102	0.113	0.099	0.142	N/A

experiments were conducted to assess the methods' strengths and weaknesses. The following section concludes this thesis and research directions for future work will be presented.

CONCLUSION AND RECOMMENDATIONS

This research aimed to identify the possibility of utilizing an end-to-end deep learning algorithm for effective representation learning for audio waveforms while addressing the vulnerability of such an end-to-end model against adversarial perturbations. Current advances on end-to-end audio processing and the results on the proposed model for ESC validate the idea of using an end-to-end model for audio processing. Moreover, recent studies on adversarial machine learning for audio processing and our study on UAPs shed light on the vulnerability of such models to adversarial attacks.

Based on the extensive review of current approaches for audio processing, Most of the research studies are based on audio features, such as MFCCs, or representations, such as spectrograms, as input to deep models. However, using fully end-to-end models for representation learning is recently becoming a popular audio classification topic. Accordingly, the proposed end-to-end model for ESC was presented to validate the idea of using an end-to-end model for audio classification. Such a model produces comparable results to its counterparts that use the 2D representation of the audio signal as input. The security of such end-to-end models is also an important issue that must be considered for developing and deploying such models. Based on an extensive survey on current advances on adversarial machine learning for audio processing models designed for various tasks like SID, SR, or classification, these models are vulnerable to adversarial perturbations. As we also demonstrated in this research, a family of audio classification models, including the proposed end-to-end model for ESC, can be attacked by UAPs that a single perturbation vector can infect most of the audio samples of the dataset.

Based on these conclusions, practitioners should consider testing and deploying various end-to-end models in physical environments. Such models can be translated into a Tensorflow lite model

[6] and be tested on a Rasberry pi [7] platform (Morehead et al., 2019). Different environmental setups with different background noise levels can be proposed for evaluation. The effect of environmental reverberations on the performance of such models should also be evaluated.

For instance, the proposed method in this research has the potential to be used as a gunshot spotter. However, similar sounds like the sounds of fireworks might be problematic for the model to come up with false positive predictions (Martin, 2020). Further study is needed to assess the reliability of the proposed model for distinguishing between gunshot sounds and very similar sounds like fireworks (ShotSpotter, 2017). Additionally, the use of ESC models in real-life applications should also address social concerns. The high false-positive rates from the classifier may pose a potential threat to specific social groups if the audio sensors are disposed of widely in particular neighborhoods. Wrong decisions by ESC models may lead to false arrests (Feathers, 2021) by police. Moreover, authorities may alter the alerts generated by the technology to justify the arrests (Guariglia, 2021). The privacy concerns of audio sensors should be considered and appropriately answered. The sensors need to monitor the environmental sounds to provide the predictions constantly. Appropriate limitations and measures on recording the sounds should be well established and regulated (Guariglia, 2021).

Moreover, using a combinational model based on fusion techniques that use both the raw audio signal and the 2D representations of the signal is an interesting research direction for ESC. Based on the results discussed in this study, such techniques could be leveraged to improve the overall accuracy since the networks based on raw audio waveforms and 2D representations tend to learn different representations. This could be done by using a combinational model with two CNNs. A CNN based on 1D convolutional layers for feature learning from the audio signal and another CNN based on 2D convolutional layers for representation learning from 2D representations like spectrograms can be used. The proposed approach used the sum rule and the majority voting for

[6] https://www.tensorflow.org/lite

[7] https://www.raspberrypi.org/

aggregating the decision of the model for different frames. In addition to that, using the attention mechanism (Bahdanau, Cho & Bengio, 2014) for magnifying some of the frames of the input audio signal is an interesting research direction. Conducting a comprehensive study on novel architectures like SincNet (Ravanelli & Bengio, 2018) for ESC, where some of the parameters of the first layer filter-bank are learned during the process of training, could also be considered as a promising future work.

Results discussed in this research illustrate that the end-to-end audio classification models are pretty vulnerable to adversarial attacks. Based on this conclusion and considering current advances in adversarial machine learning, the industry and practitioners should consider how these machine learning models could be tricked, evaded, or misled by adversarial attacks. The industry should be aware of the techniques that are used for different types of attacks for various goals like model/data poisoning and evasion attacks, and mitigation approaches like protecting the training data, model's architecture and its weights should be applied (Kumar et al., 2020).

Despite that, the political dimensions of adversarial machine learning on such models should be considered. The machine learning model should indeed be safe against adversarial attacks. But, it should be stated that safe from whom? (Albert, Penney, Schneier & Siva Kumar, 2020). Consider this scenario that the authoritarian government uses powerful spyware that is covertly installed on mobile phones to break into the devices of the activists, civil society groups, journalists, and dissidents. The spyware enables the victim's mobile phone microphone to record and detect the audio scene by an acoustic scene classification model that the authoritarian government might use to track and localize the dissidents' whereabouts. On the other side, the activists may use adversarial attacks, like Dolphin attack (Zhang *et al.*, 2017), against the audio processing model to escape the tracking system. In this case, efforts to protect the machine learning models against adversarial attacks may lead to serious harms against activists by authoritarian and radical governments (Albert *et al.*, 2020). In this scenario, the practitioners

and industries should consider committing to human rights policies when it comes to selling, deploying, and securing such audio processing models (Albert *et al.*, 2020).

This research demonstrates the vulnerability of end-to-end audio classification models to adversarial perturbations, in particular UAPs. Still, two major obstacles with evasion attacks on such models should be considered in future studies. The first one is the lack of transferability of the attacks, especially for optimization-based methods. For reporting the successful adversarial attacks, the transferability of such attacks must be evaluated and reported. Further research is needed to determine the causes of this challenge. Moreover, as we emphasized in this research, we believe that using an ensemble-based approach could be a starting point to generate more effective transferable attacks (Liu *et al.*, 2017). The second issue is that most attacks are not entirely effective in the physical environment. We observed this issue while we failed to transfer some UAP samples through an iPhone 6s mobile phone speaker to ESC and speech recognition models using the Mac-book pro microphone. The final goal of the adversary is to generate effective samples that are effective to be transferred on the air. Finally, proposing a defensive mechanism against such attacks is an important research direction. Adversarial training is a popular method for vaccinating image processing models against evasion attacks (Madry, Makelov, Schmidt, Tsipras & Vladu, 2018). This methodology could also be applied for a defensive mechanism against audio adversarial perturbation and, in particular audio UAPs.

APPENDIX I

MATHEMATICAL SOLUTIONS

1. Solving $\mathbf{w}_i$ for $\mathbf{v}'$

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

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

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

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 $\mathbf{u}_i$:

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned} e^{\mathbf{b}_i} &= \sqrt{\frac{\mathbf{w}_i}{1-\mathbf{w}_i}} \\ \Rightarrow e^{\mathbf{b}_i} &= \left(\frac{\mathbf{w}_i}{1-\mathbf{w}_i}\right)^{\frac{1}{2}} \\ \Rightarrow \ln\left(e^{\mathbf{b}_i}\right) &= \ln\left(\frac{\mathbf{w}_i}{1-\mathbf{w}_i}\right)^{\frac{1}{2}} \\ \Rightarrow \mathbf{b}_i &= \frac{1}{2}\ln\left(\frac{\mathbf{w}_i}{1-\mathbf{w}_i}\right). \end{aligned}$$

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

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

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.

2. Solving $\mathbf{v}'$ for $\mathbf{v}$

$$\mathbf{v}' = \operatorname{arctanh}\left((2\mathbf{v}-1) * (1-\epsilon)\right).$$

Substituting $(2\mathbf{v} - 1) * (1 - \epsilon)$ by $\mathbf{s}$:

$$\begin{aligned} \mathbf{v}' &= \text{arctanh}\,(\mathbf{s}) \\ \Rightarrow \tanh(\mathbf{v}') &= \tanh(\text{arctanh}(\mathbf{s})) \\ \Rightarrow \tanh(\mathbf{v}') &= \mathbf{s}. \end{aligned}$$

By substituting back $\mathbf{s}$ by $(2\mathbf{v} - 1) * (1 - \epsilon)$ we have:

$$\begin{aligned} \tanh(\mathbf{v}') &= (2\mathbf{v} - 1) * (1 - \epsilon) \\ \Rightarrow \tanh(\mathbf{v}') &= 2\mathbf{v} - 2\mathbf{v}\epsilon - 1 + \epsilon \\ \Rightarrow \tanh(\mathbf{v}') &= \mathbf{v}(2 - 2\epsilon) - 1 + \epsilon \\ \frac{\tanh(\mathbf{v}') + 1 - \epsilon}{2 - 2\epsilon} &= \mathbf{v} \end{aligned}$$

APPENDIX II

STATISTICAL TEST FOR UNIVERSAL ADVERSARIAL AUDIO PERTURBATION METHODS

Since the ASRs reported in this paper are the proportion of the successfully attacked samples by the two methods (iterative and penalty), the test of hypothesis concerning two proportions can be applied here (Johnson, Miller & Freund, 2017). Compare the ASRs obtained by two proposed methods (iterative and penalty) presented in Table 4.4 and Table 4.5 and let $\widehat{p_l}$ and $\widehat{p_h}$ be the ASR of the method with lower ASR and the ASR of the method with higher ASR, respectively. The statistical test (Johnson *et al.*, 2017) is given by :

$$Z = \frac{\widehat{p_l} - \widehat{p_h}}{\sqrt{2\widehat{p}(1-\hat{p})/m}}, \quad \widehat{p} = \frac{(\widehat{X_l} + \widehat{X_h})}{2m}. \tag{A II-1}$$

Where $\widehat{X_l}$ and $\widehat{X_h}$ are the number of successfully attacked samples by the method with lower ASR and by the method with higher ASR, respectively. Parameter m is also the number of samples in the test set. The intention is to prove the ASR of the two methods *i.e.*, we want to establish that: $\widehat{p_l} < \widehat{p_h}$ so,

- $H_0 : \widehat{p_l} = \widehat{p_h}$ (Null hypothesis)
- $H_a : \widehat{p_l} < \widehat{p_h}$ (Alternative hypothesis).

The null hypothesis is rejected if $Z < -z_\alpha$, where z_α is critical value obtained from a standard normal distribution that is related to level of significance, α. If the condition is true we can reject H_0 and accept H_a. Table II-1 shows $\widehat{p_l}$ and $\widehat{p_h}$ for all of the target models for each attacking scenario as well as the corresponding Z values. The ASR of the method which produces higher ASR is considered as $\widehat{X_h}$ and vice versa. From the standard normal distribution function (Johnson *et al.*, 2017), we know that $z_{0.057} = 1.58$. Since all of the Z values in Table II-1 are below -1.58 ($Z < -z_{0.057}$), we can reject the null hypothesis at least with 94.3% significance level (1-α) for all of the target models. In other words, the method corresponding to ASR of $\widehat{p_h}$ is more effective with 94.3% significance level. For most of the models, the attacking method

with higher ASR, is more effective with much higher significance level. We can conclude that results are statistically significant.

Table-A II-1 Statistical test of the two attacking methods. $\widehat{p_l}$ and $\widehat{p_h}$ for all of the target models for each attacking scenario as well as the corresponding Z values. $\widehat{p_l}$ and $\widehat{p_h}$ are derived from ASRs reported in Table 4.4 and Table 4.5

Attacking scenario	**Model**	$\widehat{p_l}$	$\widehat{p_h}$	Z
Targeted	1D CNN Rand	0.672	0.854	-8.946
	1D CNN Gamma	0.795	0.888	-5.323
	ENVnet-V2	0.767	0.877	-6.011
	SincNet	0.899	0.971	-6.105
	SincNet+VGG19	0.872	0.898	-1.703
	SpchCMD	0.850	0.855	-3.969
Untargeted	1D CNN Rand	0.412	0.876	-20.257
	1D CNN Gamma	0.737	0.858	-6.294
	ENVnet-V2	0.669	0.831	-7.820
	SincNet	0.886	0.919	-2.325
	SincNet+VGG19	0.838	0.865	-1.587
	SpchCMD	0.834	0.875	-32.737

APPENDIX III

DETAILED TARGETED ATTACK RESULTS BY UNIVERSAL ADVERSARIAL AUDIO PERTURBATIONS

Tables III-1 to III-6 show the detailed mean vlaues of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ on the target models in the targeted attack scenario for training and test sets. For Tables III-1 to III-5 the measures are reported for each specific target class of UrbanSound8k (Salamon *et al.*, 2014) and for the Table III-6, the measures are reported for each specific target class of speech commands dataset (Warden, 2018). The target classes of Urbansound8K dataset (Salamon *et al.*, 2014) 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).

Table-A III-1 Mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ for targeting each label of UrbanSound8k (Salamon *et al.*, 2014) dataset. The target model is 1D CNN Rand. Higher ASRs are in boldface

Method		**Target Classes**									
		AI	**CA**	**CH**	**DO**	**DR**	**EN**	**GU**	**JA**	**SI**	**ST**
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
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416
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
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-18.848	-15.553	-15.421	-14.980	-19.406	-18.046	-16.627	-16.752	-16.394	-15.590

Table-A III-2 Mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ for targeting each label of UrbanSound8k (Salamon *et al.*, 2014) dataset. The target model is 1D CNN Gamma. Higher ASRs are in boldface

Method		**Target Classes**									
		AI	**CA**	**CH**	**DO**	**DR**	**EN**	**GU**	**JA**	**SI**	**ST**
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
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416
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
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-15.560	-15.113	-16.543	-16.950	-21.505	-17.312	-15.828	-21.380	-16.329	-17.229

Table-A III-3 Mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ for targeting each label of UrbanSound8k (Salamon *et al.*, 2014) dataset. The target model is ENVnet-V2. Higher ASRs are in boldface

Method		Target Classes									
		AI	**CA**	**CH**	**DO**	**DR**	**EN**	**GU**	**JA**	**SI**	**ST**
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
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416
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
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-13.571	-14.281	-13.874	-12.977	-15.444	-14.541	-12.176	-15.025	-15.190	-14.896

Table-A III-4 Mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ for targeting each label of UrbanSound8k (Salamon *et al.*, 2014) dataset. The target model is SincNet. Higher ASRs are in boldface

Method		Target Classes									
		AI	**CA**	**CH**	**DO**	**DR**	**EN**	**GU**	**JA**	**SI**	**ST**
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
	Mean $l_{\mathrm{dB_x}}(\mathbf{v})$	-20.1006	-18.416	-23.405	-24.369	-19.411	-20.230	-18.416	-18.416	-24.250	-23.427
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
	Mean $l_{\mathrm{dB_x}}(\mathbf{v})$	-35.161	-25.740	-35.174	-35.117	-29.397	-29.583	-29.214	-29.408	-35.149	-35.219

Table-A III-5 Mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ for targeting each label of UrbanSound8k (Salamon *et al.*, 2014) dataset. The target model is SincNet+VGG. Higher ASRs are in boldface

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
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416	-18.416
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
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-20.024	-21.601	-23.594	-21.293	-23.944	-18.213	-14.776	-22.494	-22.045	-22.607

Table-A III-6 Mean values of ASR, SNR and $l_{\mathrm{dB_x}}(\mathbf{v})$ for targeting each label of Speech Commands dataset (Warden, 2018). The target model is SpchCMD. Higher ASRs are in boldface

Method		Target Classes																														
		silence	**unknown**	**sheila**	**nine**	**stop**	**bed**	**four**	**six**	**down**	**bird**	**marvin**	**cat**	**off**	**right**	**seven**	**eight**	**up**	**three**	**happy**	**go**	**zero**	**on**	**wow**	**dog**	**yes**	**five**	**one**	**tree**	**house**	**two**	**left**	**no**
Iterative	ASR training set	0.903	0.902	0.903	0.901	0.908	0.902	0.900	0.902	0.904	0.906	0.901	0.902	0.901	0.902	0.903	0.902	0.902	0.900	0.900	0.901	0.900	0.903	0.900	0.903	0.905	0.904	0.904	0.901	0.905	0.904	0.900	0.900
	ASR test set	**0.875**	0.893	**0.850**	**0.868**	**0.859**	**0.857**	**0.871**	0.829	0.845	**0.861**	0.841	**0.840**	**0.843**	**0.863**	0.838	0.872	**0.849**	**0.871**	0.835	**0.857**	**0.864**	**0.863**	**0.855**	**0.857**	**0.854**	0.840	**0.870**	0.838	**0.860**	0.840	**0.856**	0.835
	SNR (dB) test set	21.559	27.747	31.240	26.656	32.116	25.345	26.726	29.381	30.795	25.127	26.984	28.068	25.640	27.637	29.073	25.217	25.775	27.901	27.404	28.173	29.630	27.074	26.043	26.052	28.267	26.428	27.789	28.529	26.800	28.596	26.782	26.203
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-18.416	-18.416	-20.546	-18.416	-22.206	-18.416	-18.416	-19.350	-22.887	-18.416	-18.416	-18.529	-18.416	-18.418	-20.292	-18.416	-18.416	-19.338	-18.416	-18.416	-19.459	-18.416	-18.416	-18.416	-18.421	-18.416	-18.417	-18.450	-18.416	-18.840	-18.416	-18.416
Penalty	ASR training set	0.900	0.901	0.902	0.905	0.901	0.906	0.903	0.901	0.904	0.900	0.901	0.901	0.901	0.909	0.901	0.900	0.901	0.913	0.900	0.907	0.914	0.901	0.903	0.902	0.907	0.901	0.903	0.903	0.902	0.901	0.903	0.902
	ASR test set	0.859	**0.910**	0.849	0.867	0.858	0.835	0.838	**0.856**	**0.864**	0.821	**0.861**	0.830	0.832	0.837	**0.854**	**0.883**	0.829	0.867	**0.855**	0.848	0.837	0.857	0.823	0.825	0.843	**0.850**	0.857	**0.849**	0.860	**0.849**	0.851	**0.842**
	SNR (dB) test set	26.707	26.693	26.716	26.763	26.714	26.706	26.722	26.728	26.741	26.744	26.729	26.716	26.765	26.720	26.713	26.758	26.729	26.750	26.726	26.738	26.721	26.761	26.712	26.727	26.727	26.735	26.716	26.722	26.728	26.722	26.730	26.769
	$l_{\mathrm{dB_x}}(\mathbf{v})$	-19.767	-19.913	-24.018	-20.482	-22.100	-20.514	-22.881	-21.749	-21.484	-20.034	-20.996	-23.751	-20.333	-23.642	-20.663	-20.971	-17.679	-22.893	-19.322	-22.805	-23.252	-19.994	-22.219	-21.555	-22.725	-20.931	-23.301	-23.257	-20.861	-20.510	-20.723	-22.056

APPENDIX IV

TARGET MODELS FOR UNIVERSAL ADVERSARIAL AUDIO PERTURBATIONS EVALUATION

In this study six types of models are targeted with UAPs. For training all models, categorical crossentropy is used as loss function. Adadelta (Zeiler, 2012) is used for optimizing the parameters of the models proposed for ESC. For training the speech command classification model RMSprop is also used as the optimization method. In this section, we present the complete description of 1D CNN Gamma, ENVnet-V2, SincNet, SincNet+VGG19 and SpchCMD. Refer to Table 2.1 of Chapter 2 for detailed description of 1D CNN Rand model.

1. 1D CNN Gamma

This model is similar to 1D CNN Rand except that a Gammatone filter-bank is used for initialization of the filters of the first layer of this model as described in Chapter 2. Table IV-1 shows the configuration of this model. The Gammatone filters are kept frozen and they are not trained during the back-propagation 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 (Ioffe & Szegedy, 2015).

2. ENVnet-V2

Table IV-2 shows the architecture of ENVnet-V2 (Tokozume *et al.*, 2018). This model extracts short-time frequency features from audio waveforms by using two one-dimensional CLs 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 CLs each with 32 filters. After the CLs, a two-dimensional max-pooling layer is used. After that, two other two-dimensional CLs followed by a max-pooling layer are used and finally another two-dimensional CL with 128 filters is used. After using two FC layers with 4,096 neurons, a

Table-A IV-1 1D CNN Gamma

Layer	**Ksize**	**Stride**	**number 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)

softmax layer is applied for classification. Drop-out with probability of 0.5 is applied to FC layers (Srivastava *et al.*, 2014). Relu is used as the activation function for all of the layers.

Table-A IV-2 ENVnet-V2 architecture

Layer	**Ksize**	**Stride**	**number 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)

3. SincNet

Table IV-3 shows the architecture of SincNet (Ravanelli & Bengio, 2018). 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 CLs with 80 and 60 filters are applied. Layer normalization (Lei Ba, Kiros & Hinton, 2016) is used after each CL. After each CL, 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 (Srivastava *et al.*, 2014). Batch normalization (Ioffe & Szegedy, 2015) is used after FC layers. In this model, all hidden layers use leaky-ReLU (Maas, Hannun & Ng, 2013) non-linearity.

Table-A IV-3 SincNet architecture

Layer	**Ksize**	**Stride**	**number 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)

4. SincNet+VGG19

Table IV-4 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 (Ravanelli & Bengio, 2018). After applying one-dimensional max-pooling layer of size 218 with stride of one, and layer normalization (Lei Ba *et al.*, 2016), the output is stacked along time axis to form a 2D representation. This time-frequency representation is used as input to a VGG19 (Simonyan & Zisserman, 2015) network followed by a FC layer and softmax layer for classification. The parameters of the VGG19 are the same as described in (Simonyan & Zisserman, 2015) 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.

Table-A IV-4 SincNet+VGG19 architecture

Layer	**Ksize**	**Stride**	**number 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 (Simonyan & Zisserman, 2015)	-	-	-	(4096)
FC	-	-	10	(10)

5. SpchCMD

Table IV-5 shows the architecture of SpchCMD model. The model receives the raw audio input of size of 16,000. The model generates 40 patches of overlapped audio chunks of size of 800. After that, a one-dimensional convolution layer with 64 filters is used. Eleven depth-wise one-dimensional convolution layers with appropriate number of feature maps are then used to extract suitable information from the representations from the last layer. Relu is used as the activation function for all convolution layers. Bach normalization is also used after each convolution layers. After an attention layer and also a global average pooling layer a FC and softmax layer is used to classify the input samples into appropriate classes. This model is trained based on the available recipe from the Github page of the winner of the challenge [8]. A batch consists of 384 sample from the dataset is used where up to 40% of the batch are samples from the test set which are annotated based on pseudo labeling. An ensemble of the three best performed models during the initial experiments on the test is used for the labeling where, the three models predict the same label for the sample. Up to 50% percent of the samples are also augmented by the use of time shifting with the minimum and maximum range of (-2000, 0) moreover, up to 15% of the samples are also silent signals.

[8] https://github.com/see--/speech_recognition

Table-A IV-5 SpchCMD architecture

Layer	**Ksize**	**Stride**	**number of filters**	**Data shape**
InputLayer	-	-	-	(16,000)
Time_slice_stack	40	20	-	(800, 40)
Conv1D	3	2	64	(399, 64)
DeptwiseConv1D	3	1	128	(397, 128)
DeptwiseConv1D	3	1	192	(199, 192)
DeptwiseConv1D	3	1	192	(197, 192)
DeptwiseConv1D	3	1	256	(99, 256)
DeptwiseConv1D	3	1	256	(97, 256)
DeptwiseConv1D	3	1	320	(49, 320)
DeptwiseConv1D	3	1	320	(47, 320)
DeptwiseConv1D	3	1	384	(24, 384)
DeptwiseConv1D	3	1	384	(22, 384)
DeptwiseConv1D	3	1	448	(11, 448)
DeptwiseConv1D	3	1	448	(9, 448)
AttentionLayer	-	-	448	(9, 448)
GolobalAvgPooling	-	-	448	(448)
FC	-	-	32	(32)

APPENDIX V

AUDIO EXAMPLES ATTACKED BY UNIVERSAL ADVERSARIAL PERTURBATIONS

Several randomly chosen examples of legitimate and perturbed audio samples of Urbansound8k dataset (Salamon *et al.*, 2014) and Speech commands (Warden, 2018) datasets are also presented. The audio samples are perturbed based on two presented methods in this study. Targeted and untargeted perturbations are considered. Table V-1 shows a list of the samples of audio samples of Urbansound8k dataset (Salamon *et al.*, 2014) and Table V-2 also shows the list of audio samples of Speech Commands (Warden, 2018) dataset . Methodology of crafting the samples, target models, SNR of the perturbed samples, $l_{\mathrm{dB}_{\mathbf{x}}}(\mathbf{v})$ of the perturbed samples, detected class of the sample by each model as well as the true class of the samples are also presented.

Table-A V-1 List of examples of perturbed audio samples, Methodology of crafting the samples, SNR of the perturbed samples, $dB_{\mathbf{x}}(\mathbf{v})$ 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 (Salamon *et al.*, 2014). N/A: Not Applicable

Sample	Detected Class	True Class	Target Model	Method	Targeted/Untargeted	SNR	$l_{\mathbf{dB}_{\mathbf{x}}}(\mathbf{v})$
DR_0_org.wav	Drilling	Drilling	SINCNet	N/A	N/A	N/A	N/A
DR_0_pert_itr.wav	Gun shot	Drilling	SINCNet	Iterative	Targeted	27.025	-18.416
DR_0_pert_pen.wav	Gun shot	Drilling	SINCNet	penalty	Targeted	28.040	-29.264
SI_0_org.wav	Siren	Siren	SINCNet	N/A	N/A	N/A	N/A
SI_0_pert_itr.wav	Airconditioner	Siren	SINCNet	Iterative	Targeted	29.244	-20.101
SI_0_pert_pen.wav	Airconditioner	Siren	SINCNet	penalty	Targeted	33.766	-35.153
CH_0_org.wav	Children playing	Children playing	SINCNet	N/A	N/A	N/A	N/A
CH_0_pert_itr.wav	Dog bark	Children playing	SINCNet	Iterative	Targeted	31.784	-24.369
CH_0_pert_pen.wav	Dog bark	Children playing	SINCNet	penalty	Targeted	33.207	-35.154
ST_0_org.wav	Street music	Street music	SINCNet	N/A	N/A	N/A	N/A
ST_0_pert_itr.wav	Airconditioner	Street music	SINCNet	Iterative	Targeted	28.994	-20.101
ST_0_pert_pen.wav	Airconditioner	Street music	SINCNet	penalty	Targeted	33.539	-35.126
DO_0_org.wav	Dog bark	Dog bark	SINCNet	N/A	N/A	N/A	N/A
DO_0_pert_itr.wav	Drilling	Dog bark	SINCNet	Iterative	Targeted	28.861	-19.410
DO_0_pert_pen.wav	Drilling	Dog bark	SINCNet	penalty	Targeted	28.040	-29.306
EN_0_org.wav	Engine idling	Engine idling	SINCNet+ VGG19	N/A	N/A	N/A	N/A
EN_0_pert_itr.wav	Drilling	Engine idling	SINCNet+ VGG19	Iterative	untargeted	26.378	-14.005
EN_0_pert_pen.wav	Children playing	Engine idling	SINCNet+ VGG19	penalty	untargeted	23.444	-17.651
CA_0_org.wav	Car horn	Car horn	SINCNet+ VGG19	N/A	N/A	N/A	N/A
CA_0_pert_itr.wav	Drilling	Car horn	SINCNet+ VGG19	Iterative	untargeted	26.175	-14.005
CA_0_pert_pen	Street music	Car horn	SINCNet+ VGG19	penalty	untargeted	23.341	-17.588
AI_0_org.wav	Airconditioner	Airconditioner	SINCNet+ VGG19	N/A	N/A	N/A	N/A
AI_0_pert_itr.wav	Jackhammer	Airconditioner	SINCNet+ VGG19	Iterative	untargeted	25.759	-14.005
AI_0_pert_pen.wav	Children playing	Airconditioner	SINCNet+ VGG19	penalty	untargeted	23.073	-17.535
SI_1_org.wav	Siren	Siren	SINCNet+ VGG19	N/A	N/A	N/A	N/A
SI_1_pert_itr.wav	Jackhammer	Siren	SINCNet+ VGG19	Iterative	untargeted	26.334	-14.005
SI_1_pert_pen.wav	Children playing	Siren	SINCNet+ VGG19	penalty	untargeted	23.423	-17.907
DR_1_org.wav	Drilling	Drilling	SINCNet+ VGG19	N/A	N/A	N/A	N/A
DR_1_pert_itr.wav	Jackhammer	Drilling	SINCNet+ VGG19	Iterative	untargeted	27.040	-14.005
DR_1_pert_pen.wav	Children playing	Drilling	SINCNet+ VGG19	penalty	untargeted	24.555	-17.688

Table-A V-2 List of examples of perturbed audio samples, Methodology of crafting the samples, SNR of the perturbed samples, $dB_{\mathbf{x}}(\mathbf{v})$ of the perturbed samples and also detected class of the sample by SpchCMD model and the true class of the samples. The audio files belong to Speech Commands dataset (Warden, 2018). N/A: Not Applicable

Sample	Detected Class	True Class	Method	Targeted/Untargeted	SNR	$l_{\mathbf{dB_x}}(\mathbf{v})$
No_0_org.wav	no	no	N/A	N/A	N/A	N/A
No_0_pert_pen.wav	bed	no	penalty	Targeted	29.190	-20.958
No_0_pert_itr.wav	bed	no	iterative	Targeted	27.323	-18.416
Up_0_org.wav	up	up	N/A	N/A	N/A	N/A
Up_0_pert_pen.wav	unknown	up	Penalty	Targeted	24.773	-20.319
Up_0_pert_itr.wav	unknown	up	iterative	Targeted	25.385	-18.416
Five_0_org.wav	five	five	N/A	N/A	N/A	N/A
Five_0_pert_pen.wav	silence	five	penalty	Targeted	25.475	-20.218
Five_0_pert_itr.wav	silence	five	iterative	Targeted	20.293	-18.416
Down_0_org.wav	down	down	N/A	N/A	N/A	N/A
Down_0_pert_pen.wav	dog	down	penalty	targeted	31.342	-23.030
Down_0_pert_itr.wav	dog	down	iterative	targeted	29.130	-18.416
One_0_org.wav	one	one	N/A	N/A	N/A	N/A
One_0_pert_pen.wav	seven	one	penalty	targeted	27.276	-21.299
One_0_pert_itr.wav	seven	one	iterative	targeted	29.351	-20.292
Right_0_org.wav	right	right	N/A	N/A	N/A	N/A
Right_0_pert_pen.wav	five	right	Penalty	Untargeted	25.784	-24.836
Right_0_pert_itr.wav	six	right	iterative	Untargeted	27.580	-18.416
On_0_org.wav	on	on	N/A	N/A	N/A	N/A
On_0_pert_pen.wav	sheila	on	Penalty	Untargeted	25.126	-25.037
On_0_pert_itr.wav	stop	on	iterative	Untargeted	26.609	-18.416
Eight_0_org.wav	eight	eight	N/A	N/A	N/A	N/A
Eight_0_pert_pen.wav	sheila	eight	penalty	Untargeted	25.921	-24.822
Eight_0_pert_itr.wav	six	eight	iterative	Untargeted	27.764	-18.416
Two_0_org.wav	two	two	N/A	N/A	N/A	N/A
Two_0_pert_pen.wav	sheila	two	penalty	targeted	26.627	-24.813
Two_0_pert_itr.wav	sheila	two	iterative	Untargeted	28.200	-18.416
No_1_org.wav	no	no	N/A	N/A	N/A	N/A
No_1_pert_pen.wav	sheila	no	penalty	Untargeted	27.316	-24.840
No_1_pert_itr.wav	sheila	no	iterative	Untargeted	28.845	-18.416

BIBLIOGRAPHY

Abdullah, H., Warren, K., Bindschaedler, V., Papernot, N. & Traynor, P. (2020). The Faults in our ASRs: An Overview of Attacks against Automatic Speech Recognition and Speaker Identification Systems. *arXiv preprint 2007.06622*.

Abdullah, H. et al. (2019a). Hear" No Evil", See" Kenansville": Efficient and Transferable Black-Box Attacks on Speech Recognition and Voice Identification Systems. *arXiv preprint arXiv:1910.05262*.

Abdullah, H. et al. (2019b). Practical hidden voice attacks against speech and speaker recognition systems. *arXiv preprint arXiv:1904.05734*.

Akhtar, N. & Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6, 14410–14430.

Albert, K., Penney, J., Schneier, B. & Siva Kumar, R. S. (2020). Politics of adversarial machine learning. *Towards Trustworthy ML: Rethinking Security and Privacy for ML Workshop, Eighth International Conference on Learning Representations (ICLR)*.

Allen, J. B. & Berkley, D. A. (1979). Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, 65(4), 943–950.

Alzantot, M., Balaji, B. & Srivastava, M. (2018). Did you hear that? adversarial examples against automatic speech recognition. *arXiv preprint 1801.00554*.

Amodei, D. et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. *International conference on machine learning*, pp. 173–182.

Ardila, R. et al. (2020). Common Voice: A Massively-Multilingual Speech Corpus. *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pp. 4211–4215.

Athalye, A., Carlini, N. & Wagner, D. (2018a). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*.

Athalye, A., Engstrom, L., Ilyas, A. & Kwok, K. (2018b). Synthesizing Robust Adversarial Examples. *International conference on machine learning*, pp. 284–293.

Aytar, Y., Vondrick, C. & Torralba, A. (2016). Soundnet: Learning sound representations from unlabeled video. *Advances in Neural Information Processing Systems*, pp. 892–900.

Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Benetos, E., Lafay, G., Lagrange, M. & Plumbley, M. (2016). Detection of overlapping acoustic events using a temporally-constrained probabilistic model. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6450–6454.

Benzi, K., Defferrard, M., Vandergheynst, P. & Bresson, X. (2016). FMA: A Dataset For Music Analysis. *arXiv preprint arXiv:1612.01840*.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B. & Lamere, P. (2011). The Million Song Dataset. *International Society for Music Information Retrieval Conference*, 2(9), 10.

Biggio, B. & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84, 317–331.

Bisot, V., Serizel, R., Essid, S. & Richard, G. (2016). Acoustic scene classification with matrix factorization for unsupervised feature learning. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6445–6449.

Boddapati, V., Petef, A., Rasmusson, J. & Lundberg, L. (2017). Classifying environmental sounds using image recognition networks. *Procedia computer science*, 112, 2048–2056.

Bosch, J. J., Janer, J., Fuhrmann, F. & Herrera, P. (2012). A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals. *International Society for Music Information Retrieval Conference*, pp. 559–564.

Brown, J. C. (1991). Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1), 425–434.

Campbell, J. P. (1995). Testing with the YOHO CD-ROM voice verification corpus. *International Conference on Acoustics, Speech, and Signal Processing*, 1, 341–344.

Carlini, N. & Wagner, D. (2017). Towards evaluating the robustness of neural networks. *IEEE Symposium on Security and Privacy*, pp. 39–57.

Carlini, N. & Wagner, D. (2018). Audio adversarial examples: Targeted attacks on speech-to-text. *IEEE Security and Privacy Workshop*, pp. 1–7.

Carlini, N. et al. (2016). Hidden voice commands. *25th USENIX Security Symposium*, pp. 513–530.

Chen, Y. et al. (2020). Devil's whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices. *29th USENIX Security Symposium*.

Choi, K., Fazekas, G. & Sandler, M. (2016). Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*.

Chu, S., Narayanan, S. & Kuo, C. (2009). Environmental sound recognition with time–frequency audio features. *IEEE Transaction on Audio, Speech, and Language Processing*, 17(6), 1142–1158.

Cisse, M. M., Adi, Y., Neverova, N. & Keshet, J. (2017). Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. *Advances in neural information processing systems*, 30, 6977–6987.

Costa, Y. M., Oliveira, L. S. & Silla, C. N. (2017). An evaluation of Convolutional Neural Networks for music classification using spectrograms. *Applied Soft Computing*, 52, 28–38. doi: 10.1016/j.asoc.2016.12.024.

Costa, Y., Oliveira, L., Koerich, A., Gouyon, F. & Martins, J. (2012). Music genre classification using LBP textural features. *Signal Processing*, 92(11), 2723–2737.

Dai, W., Dai, C., Qu, S., Li, J. & Das, S. (2016). Very deep convolutional neural networks for raw waveforms. *arXiv preprint arXiv:1610.00087*.

Dai, W., Dai, C., Qu, S., Li, J. & Das, S. (2017). Very Deep Convolutional Neural Networks for Raw Waveforms. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 421–425.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dieleman, S. & Schrauwen, B. (2014). End-to-end learning for music audio. *International Conference on Acoustics, Speech, and Signal Processing*, pp. 6964–6968.

Dieleman, S., Brakel, P. & Schrauwen, B. (2011). Audio-based music classification with a pretrained convolutional network. *International Society for Music Information Retrieval Conference*, pp. 669–674.

Dong, Y. et al. (2018). Boosting adversarial attacks with momentum. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193.

Dosovitskiy, A. et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Du, T., Ji, S., Li, J., Gu, Q., Wang, T. & Beyah, R. (2020). SirenAttack: Generating Adversarial Audio for End-to-End Acoustic Systems. *15th ACM Asia Conference on Computer and*

Communications Security, pp. 357–369.

Duchi, J., Hazan, E. & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul), 2121–2159.

EducationFirst. (2019). 1,000 Most Common US English Words, Website: https://www.ef.edu/english-resources/english-vocabulary/top-1000-words/ (Accessed Dec. 15, 2021).

Ellis, D. P. W. (2009). Gammatone-like spectrograms, Website: http://www.ee.columbia.edu/ dpwe/resources/matlab/gammatonegram/ (Accessed Dec. 15, 2021).

Emiya, V., Badeau, R. & David, B. (2009). Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6), 1643–1654.

Esmaeilpour, M., Cardinal, P. & Koerich, A. L. (2019). A robust approach for securing audio classification against adversarial attacks. *IEEE Transaction on Information Forensics and Security*, 15, 2147–2159.

Facebook. (2019). Wit.ai Natural Language for Developers, Website: https://wit.ai (Accessed Dec. 15, 2021).

Fayek, H. (2016). Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between, Website: http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html (Acessed April 2017).

Feathers, T. (2021). Gunshot-Detecting Tech Is Summoning Armed Police to Black Neighborhoods Website: https://www.vice.com/en/article/88nd3z/gunshot-detecting-tech-is-summoning-armed-police-to-black-neighborhoods (Accessed Dec. 15, 2021).

Fonseca, E., Favory, X., Pons, J., Font, F. & Serra, X. (2020). FSD50k: an open dataset of human-labeled sound events. *arXiv preprint arXiv:2010.00475*.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G. & Pallett, D. S. (1993). DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon technical report n*, 93.

Geiger, J. & Helwani, K. (2015). Improving event detection for audio surveillance using gabor filterbank features. *23rd European Signal Processing Conference*, pp. 714–718.

Gemmeke, J. F. et al. (2017). Audio set: An ontology and human-labeled dataset for audio events. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780.

Girdhar, R., Carreira, J., Doersch, C. & Zisserman, A. (2019, June). Video Action Transformer Network. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I. J., Shlens, J. & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Goodfellow, I. J., Shlens, J. & Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. *International Conference on Learning Representations (ICLR)*.

Google. (2019). Google Cloud Speech-to-Text API, Website: https://cloud.google.com/speech-to-text (Accessed Dec. 15, 2021).

Grosse, K., Trost, T. A., Mosbach, M., Backes, M. & Klakow, D. (2019). Adversarial Initialization – when your network performs the way I want. *arXiv preprint 1902.03020*.

Guariglia, M. (2021). It's Time for Police to Stop Using ShotSpotter Website: https://www.eff.org/deeplinks/2021/07/its-time-police-stop-using-shotspotter (Accessed Dec. 06, 2021).

Han, Y., Kim, J., Lee, K., Han, Y., Kim, J. & Lee, K. (2017). Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(1), 208–221.

Hannun, A. et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint 1412.5567*.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hershey, S. et al. (2017). CNN architectures for large-scale audio classification. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135.

Hertel, L., Phan, H. & Mertins, A. (2016). Comparing time and frequency domain for audio event recognition using deep learning. *International Joint Conference on Neural Networks (IJCNN)*, pp. 3407–3411.

Hoshen, Y., Weiss, R. J. & Wilson, K. W. (2015). Speech acoustic modeling from raw multichannel waveforms. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4624–4628.

Huang, C.-Z. A. et al. (2018). Music transformer. *arXiv preprint arXiv:1809.04281*.

Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning (ICML)*, pp. 448–456.

Jankowski, C., Kalyanswamy, A., Basson, S. & Spitz, J. (1990). NTIMIT: A phonetically balanced, continuous speech, telephone bandwidth speech database. *International Conference on Acoustics, Speech, and Signal Processing*, pp. 109–112.

Jensen, H. A., Rasmussen, B. & Ekholm, O. (2019). Neighbour noise annoyance is associated with various mental and physical health symptoms: Results from a nationwide study among individuals living in multi-storey housing. *BMC public health*, 19(1), 1508.

Johnson, R., Miller, I. & Freund, J. (2017). *Miller and Freund's Probability and Statistics for Engineers, Global Edition*. Pearson Education Limited.

Jurafsky, D. & Martin, J. H. (2014). *Speech and language processing*. Pearson London.

Kereliuk, C., Sturm, B. L. & Larsen, J. (2015). Deep learning and music adversaries. *IEEE Transaction on Multimedia*, 17(11), 2059–2071.

Kingma, D. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kittler, J., Hatef, M., Duin, R. P. & Matas, J. (1998). On combining classifiers. *IEEE Transactions on pattern analysis and machine intelligence*, 20(3), 226–239.

Koerich, K. M., Esmaeilpour, M., Abdoli, S., Britto Jr., A. S. & Koerich, A. L. (2020). Cross-Representation Transferability of Adversarial Attacks: From Spectrograms to Audio Waveforms. *International Joint Conference on Neural Networks (IJCNN)*, pp. 1-7.

Kong, Q., Cao, Y., Iqbal, T., Wang, Y., Wang, W. & Plumbley, M. D. (2020). Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, 2880–2894.

Kreuk, F., Adi, Y., Cisse, M. & Keshet, J. (2018). Fooling end-to-end speaker verification with adversarial examples. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1962–1966.

Krizhevsky, A., Hinton, G. et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 1097–1105.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84–90.

Kumar, S. et al. (2020). Adversarial Machine Learning-Industry Perspectives. *2020 IEEE Security and Privacy Workshops (SPW)*, pp. 69-75. doi: 10.1109/SPW50608.2020.00028.

Kurakin, A., Goodfellow, I. & Bengio, S. (2017). Adversarial examples in the physical world. *International Conference on Learning Representations (ICLR)*.

Laffitte, P., Wang, Y., Sodoyer, D. & Girin, L. (2019). Assessing the performances of different neural network architectures for the detection of screams and shouts in public transportation. *Expert Systems with Applications*, 117, 29–41.

Lamere, P. et al. (2003). Design of the CMU Sphinx-4 decoder. *Eighth European Conference on Speech Communication and Technology*.

Lavechin, M., Bousbib, R., Bredin, H., Dupoux, E. & Cristia, A. (2020). An open-source voice type classifier for child-centered daylong recordings. *arXiv preprint arXiv:2005.12656*.

Law, E. & Von Ahn, L. (2009). Input-agreement: a new mechanism for collecting data using human computation games. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1197–1206.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

Lei Ba, J., Kiros, J. R. & Hinton, G. E. (2016). Layer normalization. *arXiv preprint 1607.06450*.

Li, F.-F. (2021). CS231n: Convolutional Neural Networks for Visual Recognition (Stanford University course), Website: https://cs231n.github.io/convolutional-networks/ (Accessed Dec. 15, 2021).

Li, S., Yao, Y., Hu, J., Liu, G., Yao, X. & Hu, J. (2018). An ensemble stacked convolutional neural network model for environmental event sound recognition. *Applied Sciences*, 8(7), 1152.

Li, T. L., Chan, A. B. & Chun, A. (2010). Automatic musical pattern feature extraction using convolutional neural network. *International Conference Data Mining and Applications*.

Liu, Y., Chen, X., Liu, C. & Song, D. (2017). Delving into Transferable Adversarial Examples and Black-box Attacks. *International Conference on Learning Representations (ICLR)*.

Ludeña-Choez, J. & Gallardo-Antolín, A. (2016). Acoustic Event Classification using spectral band selection and Non-Negative Matrix Factorization-based features. *Expert Systems with Applications*, 46, 77–86.

Maas, A. L., Hannun, A. Y. & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *International Conference on Machine Learning (ICML)*, 30(1), 3.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks. *International Conference on Learning Representations (ICLR)*.

Martin, A. (2020). Fireworks or gunshots? Here's how to tell the difference. Website: https://chicago.suntimes.com/2020/6/30/21303442/fireworks-gunshots-difference-chicago (Accessed Dec. 06, 2021).

Mesaros, A., Heittola, T., Dikmen, O. & Virtanen, T. (2015). Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 151–155.

Microsoft. (2019). Azure speaker identification api. Consulted at https://azure.microsoft.com/en-us/services/cognitive-servic/speaker-recognition/.

Millet, J. & Zeghidour, N. (2019). Learning to detect dysarthria from raw speech. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5831–5835.

Moosavi-Dezfooli, S.-M., Fawzi, A. & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O. & Frossard, P. (2017). Universal adversarial perturbations. *IEEE Conf Comp Vis Patt Recog*, pp. 1765–1773.

Morehead, A. et al. (2019). Low Cost Gunshot Detection using Deep Learning on the Raspberry Pi. *2019 IEEE International Conference on Big Data (Big Data)*, pp. 3038–3044.

Mozilla. (2017). Project deep speech, Website: https://github.com/mozilla/DeepSpeech (Accessed Dec. 15, 2021).

Mulimani, M. & Koolagudi, S. G. (2019). Segmentation and characterization of acoustic event spectrograms using singular value decomposition. *Expert Systems with Applications*, 120, 413–425.

Mydlarz, C., Salamon, J. & Bello, J. (2017). The implementation of low-cost urban acoustic monitoring devices. *Applied Acoustics*, 117, 207–218.

Neekhara, P. et al. (2019). Universal Adversarial Perturbations for Speech Recognition Systems. *Annual Conference International Speech Communication Association (INTERSPEECH)*, pp. 481–485.

Orekondy, T., Schiele, B. & Fritz, M. (2019). Knockoff nets: Stealing functionality of black-box models. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4954–4963.

Panayotov, V., Chen, G., Povey, D. & Khudanpur, S. (2015). Librispeech: an asr corpus based on public domain audio books. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210.

Papernot, N., McDaniel, P., Wu, X., Jha, S. & Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597.

Papernot, N. et al. (2017). Practical black-box attacks against machine learning. *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519.

Payton, T. (2015). Google Speech API: Information and Guidelines. Consulted at http://blog.travispayton.com/wp-content/uploads/2014/03/Google-Speech-API.pdf.

Peck, J., Roels, J., Goossens, B. & Saeys, Y. (2017). Lower bounds on the robustness to adversarial perturbations. *Advances in Neural Information Processing Systems*, pp. 804–813.

Piczak, K. J. (2015a). ESC: Dataset for environmental sound classification. *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018.

Piczak, K. (2015b). Environmental sound classification with convolutional neural networks. *25th International Workshop on Machine Learning for Signal Processing*, pp. 1–6.

Pons, J. & Serra, X. (2018). Randomly weighted CNNs for (music) audio classification. *arXiv preprint*. Consulted at https://arxiv.org/pdf/1805.00237.pdf.

Povey, D. et al. (2011). The Kaldi Speech Recognition Toolkit. *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*.

Qin, Y., Carlini, N., Cottrell, G. W., Goodfellow, I. J. & Raffel, C. (2019). Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. *International Conference on Machine Learning (ICML)*, pp. 5231–5240.

Radhakrishnan, R., Divakaran, A. & Smaragdis, A. (2005). Audio analysis for surveillance applications. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 158–161.

Ravanelli, M. & Bengio, Y. (2018). Speaker Recognition from raw waveform with SincNet. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 1021–1028.

Roederer, J. G. (2008). The physics and psychophysics of music: an introduction. Springer Science & Business Media.

Rony, J. et al. (2019). Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4322–4330.

Rudzicz, F. et al. (2008). Towards a comparative database of dysarthric articulation. *Proc. 8th Int. Seminar Speech Production (ISSP'08)*.

Saeed, A., Grangier, D. & Zeghidour, N. (2021). Contrastive learning of general-purpose audio representations. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3875–3879.

Sainath, T. N., Weiss, R. J., Senior, A., Wilson, K. W. & Vinyals, O. (2015). Learning the speech front-end with raw waveform CLDNNs. *Annual Conference International Speech Communication Association (INTERSPEECH)*.

Sak, H., Senior, A., Rao, K., Beaufays, F. & Schalkwyk, J. (2015). Google voice search: faster and more accurate. *Google Research Blog, 2015, http://googleresearch. blogspot. ch/2015/09/google-voice-search-faster-and-more. html*.

Salamon, J. & Bello, J. (2015). Unsupervised feature learning for urban sound classification. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 171–175.

Salamon, J. & Bello, J. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3), 279–283.

Salamon, J., Jacoby, C. & Bello, J. (2014). A Dataset and Taxonomy for Urban Sound Research. *22nd ACM International Conference on Multimedia*, pp. 1041–1044.

Schalkwyk, J. et al. (2010). "Your word is my command": Google search by voice: a case study. In *Advances in speech recognition* (pp. 61–90). Springer.

Scheibler, R., Bezzam, E. & Dokmanić, I. (2018). Pyroomacoustics: A python package for audio room simulation and array processing algorithms. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 351–355.

Schönherr, L., Kohls, K., Zeiler, S., Holz, T. & Kolossa, D. (2018). Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *arXiv preprint arXiv:1808.05665*.

Shafahi, A., Huang, W. R., Studer, C., Feizi, S. & Goldstein, T. (2019). Are adversarial examples inevitable? *International Conference on Learning Representations (ICLR)*.

Shen, J., et al. (2019). Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*.

ShotSpotter. (2017). How Does ShotSpotter Differentiate Between Gunshots And Fireworks? Website: https://www.shotspotter.com/news/how-does-shotspotter-differentiate-between-gunshots-and-fireworks/ (Accessed Dec. 06, 2021).

Sigtia, S., Stark, A., Krstulović, S. & Plumbley, M. (2016a). Automatic environmental sound recognition: Performance versus computational cost. *IEEE/ACM Transactionon Audio, Speech, and Language Processing*, 24(11), 2096–2107.

Sigtia, S., Benetos, E. & Dixon, S. (2016b). An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(5), 927–939.

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Simonyan, K. & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations (ICLR)*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1), 1929–1958.

Stowell, D., Giannoulis, D., Benetos, E., Lagrange, M. & Plumbley, M. (2015). Detection and classification of acoustic scenes and events. *IEEE Transaction Multimedia*, 17(10), 1733–1746.

Stowell, D. & Plumbley, M. D. (2014). Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *PeerJ*, 2, e488. doi: https://doi.org/10.7717/peerj.488.

Su, Y., Zhang, K., Wang, J. & Madani, K. (2019). Environment Sound Classification Using a Two-Stream CNN Based on Decision-Level Fusion. *Sensors*, 19(7), 1733.

Subramanian, V., Benetos, E., Xu, N., McDonald, S. & Sandler, M. (2019). Adversarial attacks in sound event classification. *arXiv preprint 1907.02477*.

Sun, C., Myers, A., Vondrick, C., Murphy, K. & Schmid, C. (2019). Videobert: A joint model for video and language representation learning. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7464–7473.

Sutskever, I., Martens, J., Dahl, G. & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. *International Conference on Machine Learning (ICML)*, pp. 1139–1147.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826.

Szegedy, C. et al. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Tan, M. & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning (ICML)*, pp. 6105–6114.

Taori, R., Kamsetty, A., Chu, B. & Vemuri, N. (2019). Targeted adversarial examples for black box audio systems. *2019 IEEE Security and Privacy Workshops (SPW)*, pp. 15–20.

Tokozume, Y. & Harada, T. (2017). Learning environmental sounds with end-to-end convolutional neural network. *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 2721–2725.

Tokozume, Y., Ushiku, Y. & Harada, T. (2018). Learning from Between-class Examples for Deep Sound Recognition. *Sixth International Conference on Learning Representations (ICLR)*.

Twilio. (2019). Twilio - Communication APIs for SMS, Voice, Video and Authentication, Website: https://www.twilio.com (Accessed Dec. 15, 2021).

Tzanetakis, G. & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5), 293–302.

Vaidya, T., Zhang, Y., Sherr, M. & Shields, C. (2015). Cocaine noodles: exploiting the gap between human and machine speech recognition. *9th Workshop on Offensive Technologies*.

Verma, P. & Berger, J. (2021). Audio Transformers: Transformer Architectures For Large Scale Audio Understanding. Adieu Convolutions. *arXiv preprint arXiv:2105.00335*.

Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint 1804.03209*.

Xie, J. & Zhu, M. (2019). Investigation of acoustic and visual features for acoustic scene classification. *Expert Systems with Applications*, 126, 20–29.

Yakura, H. & Sakuma, J. (2019). Robust Audio Adversarial Example for a Physical Attack. *28th International joint Conference on Artificial Intelligence*, pp. 5334–5341.

Yang, Z., Li, B., Chen, P.-Y. & Song, D. (2019). Characterizing audio adversarial examples using temporal dependency. *7th International Conference on Learning Representations (ICLR)*.

Zeghidour, N., Usunier, N., Synnaeve, G., Collobert, R. & Dupoux, E. (2018). End-to-end speech recognition from the raw waveform. *arXiv preprint*.

Zeghidour, N., Teboul, O., Quitry, F. d. C. & Tagliasacchi, M. (2021). LEAF: A Learnable Frontend for Audio Classification. *arXiv preprint arXiv:2101.08596*.

Zeiler, M. (2012). ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, G. et al. (2017). DolphinAttack: Inaudible Voice Commands. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, (CCS '17), 103–117. doi: 10.1145/3133956.3134052.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A. & Oliva, A. (2014). Learning deep features for scene recognition using places database. *Advances in neural information processing systems*,

pp. 487–495.

Zhu, Z., Enge, J. H. & Hannun, A. (2016). Learning multiscale features directly from waveforms. *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 1305–1309.