Leveraging Domain Features for Detecting Adversarial Attacks Against Deep Speech Recognition in Noise

Christian Heider Nielsen and Zheng-Hua Tan*
Department of Electronic Systems, Aalborg University, Denmark

Abstract

In recent years, significant progress has been made in deep model-based automatic speech recognition (ASR), leading to its widespread deployment in the real world. At the same time, adversarial attacks against deep ASR systems are highly successful. Various methods have been proposed to defend ASR systems from these attacks. However, existing classification based methods focus on the design of deep learning models while lacking exploration of domain specific features. This work leverages filter bank-based features to better capture the characteristics of attacks for improved detection. Furthermore, the paper analyses the potentials of using speech and non-speech parts separately in detecting adversarial attacks. In the end, considering adverse environments where ASR systems may be deployed, we study the impact of acoustic noise of various types and signal-to-noise ratios. Extensive experiments show that the inverse filter bank features generally perform better in both clean and noisy environments, the detection is effective using either speech or non-speech part, and the acoustic noise can largely degrade the detection performance.

Index Terms: Adversarial examples, automatic speech recognition, deep learning, filter bank, noise robustness,

1 Introduction

Numerous successful adversarial attack methods on automatic speech recognition (ASR) systems have been proposed [1, 2, 3, 4, 5, 6, 7, 8]. They demonstrate that through small optimised perturbations to an input signal, it is possible to fool an ASR system to produce an alternative targeted result, while the perturbations are largely imperceptible for humans. Prevalence of ASR is on the rise, with already rolled out systems as Google Assistant, Amazon Alexa, Samsung Bixby, Apple Siri and Microsoft Cortana. And with them comes an ever increasing attack surface. The aforementioned systems are potentially responsible for home automation, e.g. turning on and off devices, and for administrative tasks, e.g. making purchases online. Adversarial attacks can make a voice assistant behave maliciously and thus cause a significant threat to the security, privacy, and even safety of its user [9]. Obviously, without validating the request being legitimate will leave security holes in users systems.

The nature of attacks can be targeted or untargeted. While the goal of untargeted attacks is to force the ASR model to produce an incorrect class, targeted attacks aim to make the model output a predetermined class and thus are most dangerous in terms of intrusive behaviour. In this work, we will focus on targeted attacks only.

Strategies for defending adversarial attacks can be categorised as proactive and reactive [10]. In proactive defences, one seeks to build more robust ASR models, e.g. through training with adversarial examples. With reactive approaches, one aims to detect the existence of adversarial attacks at testing time. There exist a number of reactive defence methods. The works in [11, 12, 13, 14] defend against audio adversarial attacks by preprocessing a speech signal prior to passing it onto the ASR system. An unsupervised method with no need for labelled attacks is presented in [15], where the defence is realised using anomalous pattern detection. Rather than detecting adversarial examples,

*Corresponding Author: zt@es.aau.dk

the work in [16] characterises them using temporal dependencies.

In [17], adversarial example detection is formulated as a binary classification problem, in which a convolutional neural network (CNN) is used as the model and Mel-frequency cepstral coefficients (MFCCs) are used as the feature. While MFCC is the most commonly used feature for numerous applications, it remains largely uncontested for this particular defence purpose. Our preliminary study shows that attacks exhibit noticeable energy permutation in high frequency regions, where the resolution of MFCCs is by design sacrificed. Furthermore, it is shown in [18] that cepstral features from learned filter banks that have denser spacing in the high frequency region are more effective in detecting spoofing attacks. We are therefore motivated to contest the MFCC representation with alternative cepstral features that refocus the spectral resolution in different frequency regions.

ASR systems when deployed in the real world will be exposed to adverse acoustic environments. However, detecting adversarial attacks in noise has rarely been investigated in the literature. In this work, we study the impact of acoustic noise on the performance of adversarial attack detection. It is relevant to mention that the success rate of attacking in noisy environments will decrease as briefly shown in [19] and that the noise can change ASR output [14], which is of interest for future study in a systematic way. Furthermore, it is unknown how speech and non-speech parts are useful for adversarial attack detection. This work studies their potentials.

The contribution of this paper is four-fold.

- To our knowledge, this is the first systematic study of various cepstral features for adversarial attack detection and of impact of acoustic noise, speech only, and non-speech only on detection performance. These include MFCC, inverse MFCC (IMFCC), Gammatone frequency cepstral coefficients (GFCC), inverse GFCC (IGFCC) and linear frequency cepstral coefficients (LFCC). These features are not new, but they (except for MFCC) have not been explored for adversarial attack detection. In addition, their use is highly motivated, and significant improvements are obtained using inverse filter banks as in iMFCC and iGFCC. The study of these features further helps understand the characteristics of the adversarial attacks.
- This work systematically studies the impact of adverse environments for the first time and gains much insight.
- The paper analyses the potentials of using speech and non-speech parts separately in detecting attacks.
- A significant effort has been put on making the work reproducible. The source code for reproducing the results with the parameterisation used in this work and a full catalogue of results are made publicly available [1].

The paper is organized as follows. Section II presents background and analysis of audio adversarial attacks. Methodology and experiment design are provided in Section III. Section IV provides experimental results and discussions. The work is concluded in Section V.

2 Background and Analysis

This section presents both white-box and black-box attacks, analyses spectral representations of attacks, and introduces filter banks applied in this work.

2.1 White-box and black-box attacks

This work considers both white-box and black-box attacks. In white-box attacks, the adversary has full access to the parameters of the victim model, while in black-box attacks, the adversary does not have the access to the model parameters. An examples of state-of-the-art white-box attack methods against ASR systems is the gradient-based Carlini & Wagner method [2]. An example of black-box attack methods is the gradient-free Alzantot method [4] that uses genetic algorithm optimization. Gradient approximation is used to generate black-box audio adversarial examples in [20].

We build our work on the publicly-available adversarial attack data sets provided by our early work in [17]. For white-box attacks, the Carlini & Wagner method [2] is used to attack Baidu DeepSpeech

[1] https://aau-es-ml.github.io/adversarial_speech_filterbank_defence/

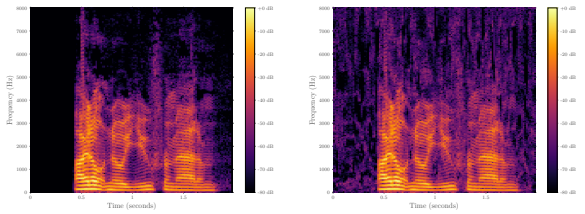


Figure 1: Spectrogram of the utterance "I don't know you from Adam", without (left) and with (right) white-box adversarial attack. x-axis is time in seconds and y-axis is frequency in Hz.

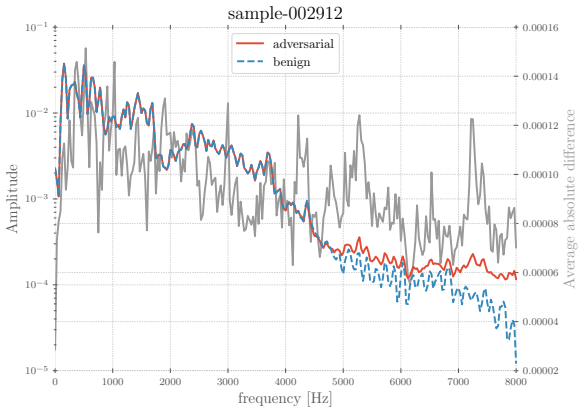


Figure 2: Long term average spectra of both benign and adversarial utterances of "I don't know you from Adam" and the average absolute frame-level difference between them.

model [21]. The DeepSpeech model is trained using the publicly available Mozilla Common Voice data set [22]. For black-box attacks, the Alzantot method [4] is applied to a keyword spotting deep model [23] trained with the publicly available Google Speech Command data set [24]. All data sets are sampled at 16 kHz.

2.2 Spectral analysis

Figure 1 shows the spectrograms of both benign and attacked samples, calculated with a frame length of 32 ms and a frame shift of 16 ms. Inspecting the figure, we hypothesise that for detecting attacks, using filter banks that concentrate resolution in the high frequency regions is beneficial.

Figure 2 shows an example of the long term average spectrum (LTAS) calculated using Welch's method [25] for the benign and adversarial signals, together with the average absolute difference between the two spectra. The difference is computed by taking the spectrograms of both benign and adversarial signals from Fig. 1, subtracting them frame-wise and averaging the absolute residuals. Contrasting the LTAS and the average absolute difference further highlights the importance of high-frequency regions, since the speech signals have much lower energy in high frequency regions and thus perturbations in high frequency regions become *relatively* more prominent.

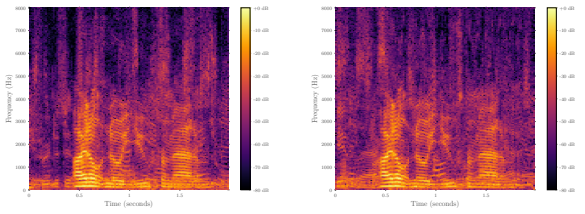


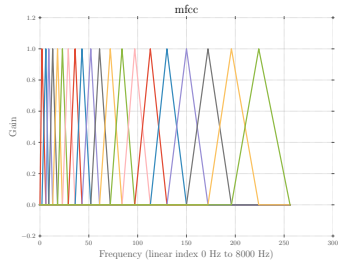
Figure 3: Spectrogram of the utterance "I don't know you from Adam" mixed with babble noise at 10dB SNR, without (left) and with (right) white-box adversarial attack.

2.3 Noisy attack example

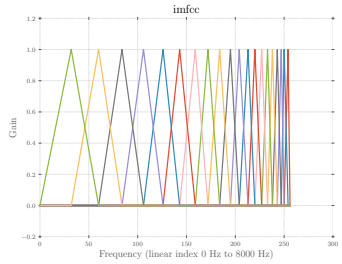
Figure 3 shows two spectra: one for the benign signal and the other for the corresponding attacked signal, both corrupted with babble noise at 10dB SNR. It is clear that the additive noise makes it more difficult to detect attacks.

2.4 Filter banks

Filter banks are widely used to model the frequency selectivity of an auditory system. Traditionally, the design of filter banks is motivated by psychoacoustic experiments. For example, humans are better at discriminating lower frequencies than higher frequencies, and the Mel-scale triangular filter bank exploits the characteristics of human perception of sound by having higher resolution at lower frequencies than at higher frequencies. Other filter banks are equivalent rectangular bandwidth (ERB) scale Gammatone filter bank and linear triangular filter bank. In this work we also apply the inverse variants of these filter banks. Figure 4 illustrates the regular



(a) 20 channel mel-scale filter bank



(b) 20 channel inverse mel-scale filter bank

Figure 4: Mel-scale triangular filter bank and its inverse. x-axis is frequency in Hz and y-axis is gain.

and inverse variant of the Mel-scale triangular filter banks.

2.4.1 Linear triangular filter bank

In a linear triangular filter bank, the gain $H_m(k)$ of the $k'th$ data point of the $m'th$ triangular band filter is as follows [26]

$$H_m(k) = \begin{cases} \frac{k - f_b(m-1)}{f_b(m) - f_b(m-1)} & f_b(m-1) \leq k \leq f(m) \\ \frac{f_b(m+1) - k}{f_b(m+1) - f_b(m)} & f_b(m) \leq k \leq f_b(m+1) \\ 0 & \text{else} \end{cases} \tag{1}$$

where $f_b(m)$ is the boundary frequency point of the $m'th$ filter

$$f_b(m) = f_{low} + m\frac{f_{high} - f_{low}}{M+1} \tag{2}$$

where M is the number of filters, f_{low} and f_{high} are the frequency range $[0, \frac{F_s}{2}]$ and F_s is the sampling frequency in Hz.

2.4.2 Mel-scale triangular filter bank

For a Mel-scale triangular filter bank, we construct a triangular filter bank with the boundary frequencies $f_b(m)$ equally spaced on the Mel-scale

$$f_b(m) = Mel^{-1}(Mel(f_{low}) + m\frac{Mel(f_{high}) - Mel(f_{low})}{M+1}) \tag{3}$$

where $Mel(f) = 1125ln(1 + (f/700)$ and $Mel^{-1}(m) = 700(e^{m/1125} - 1)$.

2.4.3 Gammatone filter bank

For a Gammatone filter bank the center frequencies $f_c(g)$ are equally spaced on the equivalent rectangular bandwidth (ERB) scale $ERB(f) = \frac{f}{9.26} + 24.7$.

Then the gain $H_g(k)$ of the $k'th$ data point of the $g'th$ the gammatone wavelet filter is computed as

$$H_g(k) = \frac{(L-1)!}{(ERB(f_c(g)) + j(2\pi k - 2\pi f_c(g)))^L} \tag{4}$$

where L is the filter order, typically set to $L = 4$, and $f_c(g)$ is the center frequency of the $g'th$ filter

$$f_c(g) = -C + (f_{high} + C)e^{g*log_{10}(\frac{f_{low}+C}{f_{high}+C})/G} \tag{5}$$

where $C = 228.83$, $1 \leq g \leq G$ and G is the number of filters.

2.4.4 Inverse filter banks

Producing the inverse filter bank variant is as simple as reversing the regular filter bank with respect to the frequencies.

3 Methodology and experiment design

3.1 Data sets

The adversarial attack data sets used in this work are from [17], which are publicly available [2]. We will refer to this article for the specifics on the generation of the adversarial attacks. Data set A (white-box) and data set B (black-box) have 1470 and 2660

[2] https://github.com/zhenghuatan/Audio-adversarial-examples

samples, respectively. Data set A contains 620 adversarial samples and 850 benign ones, and Dataset B contains 1252 adversarial samples and 1408 benign ones, which are reasonably balanced data sets. We conduct extensive experiments using data set A for training and data set B for testing, and vice versa, to investigate the robustness of the proposed methods across types of adversarial attacks.

The data sets provided by [17] comprise utterances of varying lengths. In order to use a model with inputs of fixed size, we chop the utterances into blocks of 512 ms with a shift of 512 ms, i.e. no overlap. This is in contrast to [17], where zero-padding is applied to match the longest utterance in the data sets. The issues with zero-padding are that the length of the longest utterance in the data set (rather than the problem domain itself) determines the model size, and computational resources are wasted by adding many zeros. Utterance-level decision can be made easily by later fusion of block level decisions, which is out of the scope of this work.

After block chopping, we split the data into training, validation and test subsets. In order to make the data split unbiased and fully reproducible, we use a hash function-based selection on the utterance filenames, targeting at 70% for training, 10% for validation and 20% for test. In the end, the blocks amount to $(1187, 183, 272)$ adversarial attack blocks and $(1550, 222, 377)$ benign blocks for training, validation, testing splits from data set A. From data set B we end up with $(1260, 180, 360)$ adversarial attack blocks and $(1398, 200, 400)$ benign blocks.

3.2 Detection model

We use CNNs to classify an input as either being benign or adversarial. The chosen model architecture and hyperparameterisation mostly follows [17]. One change is to have a fixed input size as $(31, 20)$, corresponding to (#frames, #cepstral coefficients), throughout all experiments. Additionally our model has one single output for binary classification. This slightly modified architecture is found in Table 1.

Table 1: Model Architecture

type	size	field	stride	activation
Conv2d	64	$(2,2)$	$(1,1)$	*ReLU*
MaxPool2d	$(1,3)$	$(1,3)$		
Conv2d	64	$(2,2)$	$(1,1)$	*ReLU*
MaxPool2d	$(1,1)$	$(1,1)$		
Conv2d	32	$(2,2)$	$(1,1)$	*ReLU*
MaxPool2d	$(2,2)$	$(2,2)$		
Flatten	896			
Fully Connected	128			*ReLU*
Fully Connected	1			*Sigmoid*

3.3 Detection with various cepstral features

We study a set of filter banks for attack detection aiming at investigating the importance of different frequency regions. In extracting speech features, each block is first segmented into frames using a frame length of 32 ms with a frame shift of 16 ms. Each frame is then processed by taking the discrete Fourier transform, calculating power spectral estimate, applying a filter bank, taking the logarithm of the filter bank energies and finally taking discrete cosine transform, which gives cepstral feature for the frame. This leads to 20 cepstral coefficients for each of the 31 frames in a block. A supervector is a concatenation of the cepstral feature vectors of a multi-frame block.

In this work we consider the following cepstral features depending on different filter banks: Mel-frequency cepstral coefficients, inverse MFCC, Gammatone frequency cepstral coefficients, inverse GFCC, and linear frequency cepstral coefficients.

3.4 Detection using speech and non-speech parts

Attacks may occur in both speech and non-speech regions. To investigate contributions of speech and non-speech parts to attack detection, we split the signals of the white-box data set using the open-source, unsupervised robust voice activity detection (rVAD) method [27] [3], with the default parameterisation of their open source implementation. The extracted speech segments and non-speech segments from a single signal are separately concatenated into one signal for speech and another for non-speech, which each is subsequently chopped into blocks, framed and the cepstrums are computed.

3.5 Detection in noisy environments

To study the impact of noise on detection performance, we consider the various filter bank models

[3] `https://github.com/zhenghuatan/rVAD`

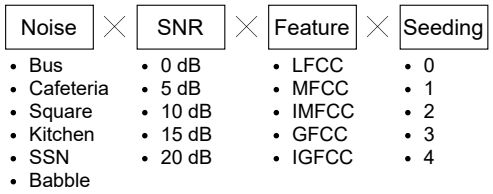


Figure 5: Combinations of models to be trained and evaluated

for a range of noise types and SNRs. Figure 5 shows the combinations of conditions/models in this experiment, leading to quite a number of experiments of different models. The considered noise types are cafeteria noise *PCAFETER*, bus passenger noise *TBUS*, city square noise *SPSQUARE*, kitchen noise *DKITCHEN*, speech shaped noise *ssn* and babble noise *bbl*. The first four are from [28] and the last two from [29]. We chose these to cover possible real world settings where ASR systems are employed and may be exposed to adversarial attacks, and these noise files are readily available online, making the experiments reproducible. We mix noise with speech at $[0, 5, 10, 15, 20]$ dB SNR levels.

We again split a signal into speech and non-speech parts using rVAD [27]. However, this splitting is only for calculating root mean square (RMS) of the speech part. Prior to mixing we scale the noise by the RMS ratio between the speech part of the signal and the noise. We additively mix the scaled noise with the signal, at a target SNR calculated using the speech part. Prior to mixing, noise signals of each noise type are also split into training, validation and test subsets to ensure no leakage among them.

4 Experimental results

In this section, we present the experimental results. For performance evaluation, we use the receiver operator characteristic's area under curve (ROCAUC) metric. In all result tables we will display the mean and standard deviation of 5 independent runs of varying seeds for each experiment. The seeding affects the model parameter initialisation and batch sampling order of the same fixed training, validation and test splits.

4.1 Cepstral features for attack detection

This experiment evaluates the five cepstral features on different white- and black-box attack combinations. To make the results comparable across attack types, we truncate the number of examples in each dataset to be the same as the lesser.

The results are shown in Table 2. First we observe that inverse filter bank features IGFCC and IMFCC perform the best, GFCC and MFCC are inferior, and LFCC is in between. Next, we can see the generalisation performance across data sets with different training and test combinations. Training with black-box and testing with white-box sees a significant performance drop. It is interesting to note that IMFCC and IGFCC perform so well when training on white-box and testing on black-box, but the same is not observed when training on black-box and testing on white-box. Training and testing on black-box shows higher ROCAUC scores than training and testing on white-box, which could be because black-box attacks are more audible and easier to detect than white-box ones. Finally multi-style training combining white-box and black-box helps.

4.2 Attack detection in speech and in non-speech

This experiment compares performance in both speech and non-speech regions as well as across them to see if a model trained for speech only is able to generalise to non-speech only and vice-versa. To make the results comparable we truncate the size of the speech and non-speech data sets to make them match. For splitting the sample into the speech and non-speech parts, we apply the rVAD method [27], as detailed earlier.

Table 3 shows the five cepstral features on speech and non-speech parts of white-box attack with different training and test set combinations. From the results we observe that adversarial attacks can be detected via both speech and non-speech regions effectively and that inverse filter bank features remain performing the best. It is interesting to note that testing on non-speech always performs better than testing on speech, no matter the model is trained in non-speech, speech or a mixture of speech and non-speech, which indicates non-speech part con-

Table 2: Mean and standard deviation of ROCAUC for scenarios where both training and test tests containing white-box and black-box examples. *w&b* denotes both white-box and black-box attacks.

filter bank train set	test set	MFCC	IMFCC	GFCC	IGFCC	LFCC
white-box						
	white-box	.941 ± .013	.98 ± .003	.915 ± .02	**.981 ± .003**	.961 ± .003
	black-box	.576 ± .072	**.946 ± .01**	.694 ± .04	.932 ± .012	.692 ± .127
black-box						
	white-box	.527 ± .015	**.672 ± .091**	.528 ± .011	.676 ± .034	.625 ± .057
	black-box	.983 ± .005	.991 ± .007	.982 ± .004	**.991 ± .005**	.988 ± .003
w&b-box						
	w&b-box	.966 ± .001	**.989 ± .002**	.948 ± .006	.986 ± .001	.979 ± .001
	white-box	.964 ± .002	**.989 ± .002**	.942 ± .007	.985 ± .002	.976 ± .001
	black-box	.974 ± .004	.99 ± .007	.972 ± .012	**.99 ± .005**	.988 ± .003

Table 3: Mean and standard deviation of ROCAUC for speech and non-speech data sets of white-box attacks. *speech & ns* denotes both speech and non-speech attack datasets.

filter bank train set	test set	MFCC	IMFCC	GFCC	IGFCC	LFCC
non-speech						
	non-speech	.97 ± .016	.993 ± .006	.965 ± .014	**.995 ± .004**	.988 ± .006
	speech	.724 ± .074	.823 ± .06	.661 ± .058	**.851 ± .071**	.746 ± .042
speech						
	speech	.833 ± .051	.944 ± .021	.679 ± .036	**.945 ± .018**	.886 ± .028
	non-speech	.937 ± .017	.987 ± .009	.863 ± .033	**.988 ± .011**	.948 ± .02
speech & ns						
	s&ns	.929 ± .013	**.982 ± .008**	.894 ± .006	.978 ± .005	.941 ± .013
	non-speech	.986 ± .002	**.999 ± .002**	.973 ± .01	.997 ± .001	.98 ± .016
	speech	.903 ± .019	**.975 ± .011**	.858 ± .008	.97 ± .007	.925 ± .015

tains a stronger cue of adversarial attacks. Training on both speech and non-speech and testing on non-speech gives the highest ROCAUC scores for almost all cepstral features.

4.3 Attack detection in noise

In this experiment we evaluate the various filter bank models for a range of noise types and SNRs for white-box attacks.

4.3.1 Noise and SNR specific training

Table 4 presents results for the narrowly trained models using five features for differing noise types and SNRs. We observe a general tendency that using inverse filter bank features outperforms the rest with the only exception being the *kitchen* noise. This could be explained by the fact that the *kitchen* noise has much flatter long term amplitude spectrum, as we found experimentally, which leads to LFCC performing the best. Across all noise types and SNRs, IGFCC performs the best, for which the reason could be that Gammatone filter banks are known to be noise robust. In all cases, noise significantly decreases the detection performance, and in some 0dB cases, the performance is close to random guess.

Table 4: Mean and standard deviation of ROCAUC for data sets of various noise types and various SNR levels using five different cepstral coefficients. White-box attack data set is used. *avg* is calculated across all seedings (of five independent runs) and SNRs, *all* denotes all noise types and *avg_ all* is the average over all noise types and SNRs. Noise type and SNR are matched for both training and testing in all settings, namely a narrowly trained systems is evaluated in this table.

cepstral features		MFCC	IMFCC	GFCC	IGFCC	LFCC
noise type	snr					
bbl						
	0db	.533 ± .047	.654 ± .027	.528 ± .029	**.687 ± .059**	.573 ± .046
	5db	.621 ± .042	.793 ± .032	.576 ± .02	**.806 ± .048**	.702 ± .022
	10db	.706 ± .038	.878 ± .016	.668 ± .04	**.901 ± .011**	.78 ± .02
	15db	.798 ± .017	.921 ± .024	.742 ± .025	**.951 ± .015**	.857 ± .012
	20db	.867 ± .03	.953 ± .009	.838 ± .033	**.955 ± .009**	.901 ± .022
	avg	.705 ± .127	.84 ± .111	.671 ± .118	**.86 ± .109**	.763 ± .121
ssn						
	0db	.54 ± .037	.533 ± .045	.506 ± .012	**.747 ± .018**	.551 ± .03
	5db	.554 ± .032	.611 ± .023	.518 ± .029	**.862 ± .033**	.541 ± .017
	10db	.592 ± .034	.658 ± .038	.546 ± .049	**.926 ± .015**	.69 ± .05
	15db	.73 ± .037	.868 ± .027	.691 ± .016	**.947 ± .003**	.802 ± .037
	20db	.824 ± .045	.937 ± .011	.757 ± .035	**.968 ± .006**	.908 ± .024
	avg	.648 ± .118	.721 ± .16	.603 ± .107	**.89 ± .083**	.699 ± .148
kitchen						
	0db	.541 ± .039	.511 ± .02	**.562 ± .023**	.502 ± .005	.533 ± .023
	5db	**.6 ± .017**	.536 ± .022	.592 ± .018	.548 ± .03	.593 ± .007
	10db	.692 ± .023	.562 ± .013	.652 ± .032	.563 ± .038	**.7 ± .022**
	15db	.799 ± .048	.647 ± .025	.738 ± .04	.653 ± .032	**.808 ± .035**
	20db	.858 ± .013	.791 ± .032	.832 ± .055	.806 ± .05	**.876 ± .035**
	avg	.698 ± .124	.609 ± .106	.675 ± .106	.615 ± .114	**.702 ± .133**
cafeteria						
	0db	.523 ± .027	**.55 ± .045**	.508 ± .011	.513 ± .022	.532 ± .035
	5db	.539 ± .025	**.608 ± .053**	.521 ± .022	.574 ± .053	.581 ± .065
	10db	.605 ± .04	.705 ± .025	.593 ± .052	**.735 ± .019**	.71 ± .04
	15db	.72 ± .021	.808 ± .015	.662 ± .041	**.829 ± .04**	.805 ± .023
	20db	.814 ± .023	.888 ± .025	.816 ± .02	**.9 ± .019**	.866 ± .019
	avg	.64 ± .116	**.712 ± .131**	.62 ± .119	.71 ± .153	.699 ± .135
square						
	0db	.539 ± .066	.647 ± .115	.521 ± .059	**.682 ± .104**	.637 ± .09
	5db	.625 ± .083	.838 ± .009	.607 ± .079	**.843 ± .023**	.784 ± .05
	10db	.732 ± .04	**.91 ± .02**	.763 ± .044	.906 ± .008	.877 ± .02
	15db	.861 ± .02	.935 ± .026	.831 ± .042	**.946 ± .014**	.945 ± .016
	20db	.934 ± .022	**.97 ± .021**	.852 ± .023	.969 ± .01	.969 ± .009
	avg	.738 ± .156	.86 ± .127	.715 ± .141	**.869 ± .114**	.843 ± .131
bus						
	0db	.764 ± .042	.807 ± .038	.695 ± .046	**.816 ± .045**	.784 ± .045
	5db	.865 ± .054	**.928 ± .016**	.798 ± .016	.928 ± .02	.91 ± .027
	10db	.917 ± .025	**.98 ± .012**	.866 ± .019	.964 ± .012	.953 ± .011
	15db	.95 ± .007	**.984 ± .013**	.922 ± .016	.973 ± .007	.973 ± .014
	20db	.933 ± .013	.986 ± .016	.926 ± .023	**.987 ± .008**	.974 ± .02
	avg	.886 ± .075	**.937 ± .072**	.841 ± .092	.933 ± .066	.919 ± .077
all						
	avg_ all	.719 ± .145	.78 ± .162	.688 ± .137	**.813 ± .156**	.771 ± .15

4.3.2 Cross-noise type generalisation

This experiment investigates how the trained models generalise to unseen noise types. First we keep away one single noise type as unseen during training and mix it with test speech blocks. Then we use all of the rest of the noise types for training the model, and the training data with multiple noise types does share the same training speech blocks.

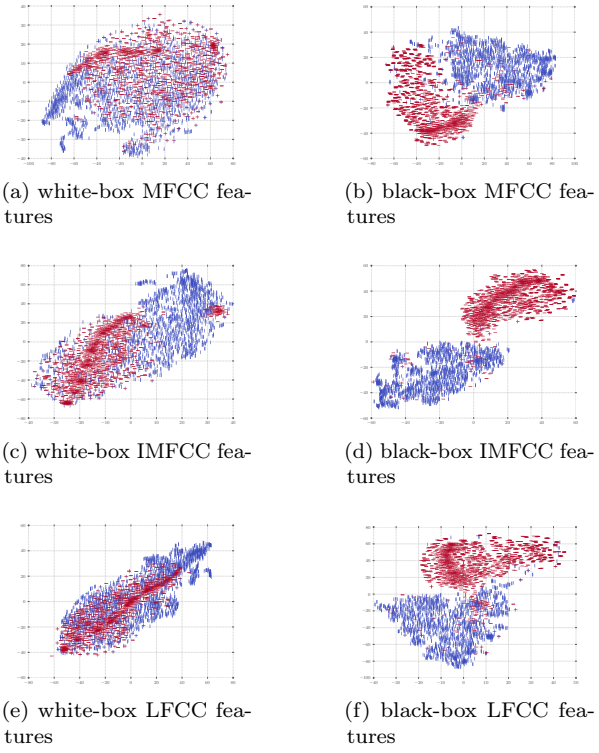
The validation set uses the same noise-types as training, but they have their own validation speech blocks. Note that there is still no overlap between the sample blocks in the training, validation and test sets. Due to space limitation, we show only *bbl* as unknown. Then we also test the *bbl* model on the rest of noise types and the clean model on all noise types. We train models using noisy signals mixed at all SNRs and test the models on the individual SNRs. In the end, we also compute the average performances across SNRs and seedings.

The results are shown in Table 5. It is noted that training on multiple noise types (widely trained) and testing on a single one outperforms the opposite and the model trained with clean data. Inverse filter bank features remain to be superior.

4.4 Cepstral feature projection

We perform a projection of each supervector of cepstral features, from $31 * 20 = 620 \rightarrow 2$, to visualise if any low dimensional structure exists. We limit the number of random samples in the projection to 3500 blocks, from the training subset of white- and black-box adversarial attack data sets. For the projection method we use T-distributed Stochastic Neighbour Embedding(TSNE) [30]. The results for MFCC, IMFCC and LFCC features are shown in Figure 6. Similar behaviour is as well observed for GFCC and IGFCC features, which is not shown due to space limit.

Surprisingly although there is only a small discrepancy between the white-box and black-box attack classification performances, it is much easier to find dichotomic projections of the black-box attacks than those of white-box attacks, i.e. projected data of black-box attacks are much more separated from those of benign than they are for white-box attacks.



(a) white-box MFCC features

(b) black-box MFCC features

(c) white-box IMFCC features

(d) black-box IMFCC features

(e) white-box LFCC features

(f) black-box LFCC features

Figure 6: 2D TSNE projection of 3500 data points, blue |'s denotes benign data points while red -'s denotes adversarial data points. Left side is the white-box attacks and the right is the black-box attacks.

5 Conclusion

We investigated the effect of filter banks on detecting white-box and black-box adversarial attacks in original signals, signals with additive noise, speech only and non-speech only parts. Our experiments show a clear tendency that the inverse filter bank based features outperform their counterparts and the linearly spaced filter bank based feature. Consistently through near to all trials we observe that inverse Mel-frequency cepstral coefficient and inverse Gammatone frequency cepstral coefficients based models are showing higher precision and recall measures than the rest of cepstral features spaces, indicating that in order to better detect adversarial attacks one should prefer the inverse filter banks over their counterparts.

A similar observation is obtained when testing

Table 5: Mean and standard deviation of ROCAUC for testing on unknown noise data sets. *clean* is a training set consisting clean white-box attacks, and *all* is all noise types and SNRs

filter bank		MFCC	IMFCC	GFCC	IGFCC	LFCC
train set	test set, snr					
bbl_all_snr						
	rest, 0db	.553 ± .031	.59 ± .02	.516 ± .02	**.629 ± .009**	.602 ± .018
	5db	.584 ± .045	.639 ± .03	.523 ± .031	**.684 ± .014**	.643 ± .02
	10db	.621 ± .064	.703 ± .037	.536 ± .048	**.746 ± .013**	.691 ± .024
	15db	.66 ± .086	.77 ± .033	.548 ± .066	**.813 ± .007**	.748 ± .031
	20db	.698 ± .104	.821 ± .024	.557 ± .076	**.861 ± .01**	.807 ± .031
	avg	.623 ± .088	.705 ± .089	.536 ± .055	**.747 ± .085**	.698 ± .077
rest_all_snr						
	bbl, 0db	.56 ± .019	**.635 ± .019**	.568 ± .01	.63 ± .01	.576 ± .013
	5db	.605 ± .015	.727 ± .018	.615 ± .029	**.744 ± .012**	.609 ± .026
	10db	.673 ± .025	.829 ± .017	.669 ± .023	**.846 ± .015**	.697 ± .03
	15db	.765 ± .028	.895 ± .023	.765 ± .033	**.923 ± .016**	.804 ± .041
	20db	.861 ± .025	.94 ± .017	.844 ± .023	**.954 ± .009**	.882 ± .028
	avg	.693 ± .111	.805 ± .113	.692 ± .103	**.819 ± .12**	.713 ± .119
clean						
	all, all	.732 ± .132	.719 ± .159	.682 ± .118	.741 ± .156	**.753 ± .149**

on noisy signals. One exception is the *kitchen* noise type, which has highly flat long term average spectrum, and thus linear frequency cepstral feature performs the best. We also show that adversarial detection is effective in both speech only and non-speech only parts.

Future work includes investigating the use of learnt features for detecting adversarial examples, as learnt filter bank cepstral coefficients have shown to be highly effective for voice spoofing detection [31].

6 Acknowledgements

We acknowledge the students Amalie V. Petersen, Jacob T. Lassen and Sebastian B. Schiøler at Aalborg University for conducting initial experiments.

References

[1] Dan Iter, Jade Huang, and Mike Jermann, "Generating adversarial examples for speech recognition," *Stanford Technical Report*, 2017.

[2] Nicholas Carlini and David Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 1–7.

[3] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa, "Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding," *arXiv preprint arXiv:1808.05665*, 2018.

[4] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava, "Did you hear that? adversarial examples against automatic speech recognition," *arXiv preprint arXiv:1801.00554*, 2018.

[5] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet, "Houdini: Fooling deep structured prediction models," *arXiv preprint arXiv:1707.05373*, 2017.

[6] Felix Kreuk, Yossi Adi, Moustapha Cisse, and Joseph Keshet, "Fooling end-to-end speaker verification with adversarial examples," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 1962–1966.

[7] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *International conference on machine learning*. PMLR, 2019, pp. 5231–5240.

[8] Rohan Taori, Amog Kamsetty, Brenton Chu, and Nikita Vemuri, "Targeted adversarial examples for black box audio systems," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 15–20.

[9] Chen Yan, Xiaoyu Ji, Kai Wang, Qinhong Jiang, Zizhi Jin, and Wenyuan Xu, "A survey on voice assistant security: Attacks and countermeasures," *ACM Computing Surveys (CSUR)*, 2022.

[10] Shengshan Hu, Xingcan Shang, Zhan Qin, Minghui Li, Qian Wang, and Cong Wang, "Adversarial examples for automatic speech recognition: Attacks and countermeasures," *IEEE Communications Magazine*, vol. 57, no. 10, pp. 120–126, 2019.

[11] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Li Chen, Michael E Kounavis, and Duen Horng Chau, "Adagio: Interactive experimentation with adversarial attack and defense for audio," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 677–681.

[12] Krishan Rajaratnam, Kunal Shah, and Jugal Kalita, "Isolated and ensemble audio preprocessing methods for detecting adversarial examples against automatic speech recognition," *arXiv preprint arXiv:1809.04397*, 2018.

[13] Vinod Subramanian, Emmanouil Benetos, and Mark Sandler, "Robustness of adversarial attacks in sound event classification," in *Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019, p. 239.

[14] Krishan Rajaratnam and Jugal Kalita, "Noise flooding for detecting audio adversarial examples against automatic speech recognition," in *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2018, pp. 197–201.

[15] Victor Akinwande, Celia Cintas, Skyler Speakman, and Srihari Sridharan, "Identifying audio adversarial examples via anomalous pattern detection," *arXiv preprint arXiv:2002.05463*, 2020.

[16] Zhuolin Yang, Bo Li, Pin-Yu Chen, and Dawn Song, "Characterizing audio adversarial examples using temporal dependency," *arXiv preprint arXiv:1809.10875*, 2018.

[17] Saeid Samizade, Zheng-Hua Tan, Chao Shen, and Xiaohong Guan, "Adversarial example detection by classification for deep speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3102–3106.

[18] Hong Yu, Zheng-Hua Tan, Zhanyu Ma, Rainer Martin, and Jun Guo, "Spoofing detection in automatic speaker verification systems using dnn classifiers and dynamic acoustic features," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4633–4644, 2017.

[19] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter, "Commandersong: A systematic approach for practical adversarial voice recognition," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 49–64.

[20] Po-Hao Huang, Honggang Yu, Max Panoff, and Ting-Chi Wang, "Generation of black-box audio adversarial examples based on gradient approximation and autoencoders," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2022.

[21] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur Yi Li, Hairong Liu, Sanjeev Satheesh, Anuroop Sriram, and Zhenyao Zhu, "Exploring neural transducers for end-to-end speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 206–213.

[22] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[23] Tara N Sainath and Carolina Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[24] Pete Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[25] Peter Welch, "The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on audio and electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.

[26] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, and Raj Reddy, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall PTR, USA, 1st edition, 2001.

[27] Zheng Hua Tan, Achintya kr Sarkar, and Najim Dehak, "rvad: An unsupervised segment-based robust voice activity detection method," *Computer Speech and Language*, vol. 59, pp. 1–21, 2020, `https://github.com/zhenghuatan/rVAD/`.

[28] Joachim Thiemann, Nobutaka Ito, and Emmanuel Vincent, "Demand: a collection of multi-channel recordings of acoustic noise in diverse environments," in *Proc. Meetings Acoust*, 2013, pp. 1–6.

[29] Morten Kolboek, Zheng-Hua Tan, and Jesper Jensen, "Speech enhancement using long short-term memory based recurrent neural networks for noise robust speaker verification," in *2016 IEEE spoken language technology workshop (SLT)*. IEEE, 2016, pp. 305–311.

[30] Laurens Van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[31] Hong Yu, Zheng-Hua Tan, Yiming Zhang, Zhanyu Ma, and Jun Guo, "Dnn filter bank cepstral coefficients for spoofing detection," *Ieee Access*, vol. 5, pp. 4779–4787, 2017.