Generating Adversarial Examples for Speech Recognition

Dan Iter
Stanford University
daniter@stanford

Jade Huang
Stanford University
jayebird@stanford

Mike Jermann
Stanford University
mjermann@stanford

Abstract

With the proliferation of natural language interfaces on mobile devices and in home personal assistants such as Siri and Alexa, many services and data are becoming available through transcription from a speech recognition system. One major risk factor in this trend is that a malicious adversary may attack this system without the primary user noticing. One way to accomplish this is to use adversarial examples that are perceived one way by a human, but transcribed differently by the Automatic Speech Recognition (ASR) system. For example, a recording that sounds like "hello" to the human ear, but is transcribed as "goodbye" by the ASR system. Recent work has shown that adversarial examples can be created for convolutional neural networks to fool vision recognition systems. We show that similar methods can be applied to neural ASR systems. We show successful results for two methods of generating adversarial examples where we fool a high quality ASR system but the difference in the audio is imperceptible to the human ear. We also present a method for converting the adversarial MFCC features back into audio.

1 Introduction

Recent work has shown that adversarial examples can be created for convolutional neural networks to fool vision recognition systems (Goodfellow et al., 2014), (Szegedy et al., 2014). For example, an adversarial image can be classified by neural networks as "gibbon", but to the human eye, the image clearly appears to be a "panda". Such adversarial examples expose a weakness in neural networks. Adversarial examples are not just the result of overfitting to a particular model or specific selection of a training set. They often remain difficult for a network to classify even with a model trained with multiple hyperparameters. With this in mind, they can be used to improve the quality of existing neural networks through adversarial training.

We show that similar methods can be applied to neural ASR systems, namely the fast gradient sign method (Goodfellow et al., 2014) and the fooling gradient sign method (Karpathy, 2015). We utilize an existing codebase for a state-of-the-art neural ASR system, WaveNet, implemented in Tensorflow (Kim and Park, 2016) for the task of speech-to-text. We integrate our two methods into the code so that we are able to gain access to the computed gradients and modify them accordingly.

As we utilized a pre-trained model of WaveNet whose input were MFCC Features (mainly from VCTKCorpus), we also operated using MFCC Features. After successfully creating adversarial samples that our neural network misclassifies, we convert the features back into audio with an approximate approximation of an inverse MFCC transform, as discussed in 3.4. While this is a lossy transform, the resulting audio is still coherent to the human ear.

The result is that a neural network classifies a set of MFCC features as "Siri call police", but the features when converted back to audio sound like "Please call Stella" to the human ear.

2 Related Work

2.1 Intriguing properties of neural networks

Szegedy et al. presents a somewhat formal definition of an "adversarial examples" in the context of two intriguing properties of neural networks. This concept is presented as part of two main properties

studied in this paper: the complex semantic meaning of individual neural units and the stability of neural networks in response to small perturbations to input. The second property is the one that we exploit to create our own adversarial examples for speech processing.

The main result regarding adversarial examples in Szegedy et al. is that the smoothness assumption, that inputs that are similar will lead to similar outputs, is not true in neural networks. Furthermore, they present an optimization algorithm for finding adversarial examples by adding imperceptibly small perturbations to the input to drastically change the output. The paper claims that this method finds a low probability "pocket" using a box-constrained L-BFGS to approximate the function that minimized the magnitude of the perturbation while still mapping the input to a particular label ℓ (which may not be its true label). It is worth noting that in many neural networks there is a preprocessing step that applies augmentation to the inputs in order to learn a more robust classifier. Unfortunately, the space is much too large for a sampling method to efficiently cover all such "low-probability pockets".

From this paper, we exploit exactly this property of neural networks having low-probability pockets to find adversarial examples in speech such that we can trick the neural network into an incorrect target classification.

2.2 Explaining and Harnessing Adversarial Examples

As Szegedy et al. discovered, neural networks can be vulnerable to adversarial examples. On some datasets like ImageNet, adversarial examples were so close to the original examples that the changes were indistinguishable to the human eye, and frequently misclassified by a variety of classifiers. Here, Goodfellow et al. explain that even a simple linear model can have adversarial examples if its input has sufficient dimensionality. This was in contrast to previous hypotheses that vulnerabilities to adversarial examples were related to the highly non-linear nature of neural networks.

Thus, inspired by this linear view of adversarial examples, Goodfellow et al. hypothesized that neural networks are too linear to resist linear adversarial perturbations. They developed the fast gradient sign method of generating adversarial examples, a "simple, cheap algorithm" and also experimented with a method of rotating an image by a small angle in the direction of the direction of the gradient. Classifiers like a shallow softmax classifier, a maxout network, convolutional maxout network, and logistic regression had an error rate of 80-99% on these adversarial examples.

Goodfellow et al. then examined how to regularize a neural network by training on a mixture of adversarial and clean examples, using an adversarial objective function based on the fast gradient sign method. Before adversarial training, the model used had an error rate of 89.4% on adversarial examples which dropped to 17.9% after training on adversarial examples. However, they found the model was susceptible to becoming resistant to adversarial examples, as the predictions were unfortunately still highly confident on misclassified adversarial examples.

2.3 Practical Black-Box Attacks against Machine Learning

While there is plenty of evidence that neural networks can be fooled with adversarial examples, many of the methods to generate these examples rely on gradient methods that require access to the target network. From the perspective of security, the main threat would be black box attacks on neural networks, because an adversary wouldn't need any knowledge of the internals of the target network. While this is a reach goal for this paper, Papernot et al. provides some inspiration for how black box adversarial attacks could be generated.

Specifically, Papernot et al. describe an algorithm that requires training a simulated network and a artificial dataset to train a network that the adversary has access to in order to approximate the target network. They use forward gradients and saliency maps to guide the generation of their training dataset to bound the amount of sampling that is required to properly explore the manifold of the target network. Ultimately, similar adversarial example generation techniques as mentioned above could be employed on the substitute model. Papernot et al. showed that these examples can also fool the target models with high probability. If we are able to make significant progress on fooling white box networks, we may attempt this technique to also fool a black box ASR system such as Siri, or Google STT.

2.4 Hidden Voice Commands

Carlini et al (Nicholas Carlini and Zhou, 2016) take more of a security angle on the topic of generating adversarial examples for speech recognition and explore how these exploits may be used and how to defend against them. (Nicholas Carlini and Zhou, 2016) describe approaches for both black box and white box attacks, however the major difference to from work is that Carlini et al focus on creating obfuscated examples and they do not operate on end to end neural networks. Obfuscated examples means that the examples sound like random noise rather than normal human perceptible speech, which is an easier task. Interestingly, some of the noise that we hear when reconstructing audios from MFCC features is explicitly used by Carlini et al to try to obfuscate the true message in the audio. Also, the systems targeted by (Nicholas Carlini and Zhou, 2016) are GMM and HMM systems rather than end to end neural networks. This means they can not leverage the gradient methods used by our work.

3 Approach

3.1 Data

We utilized two sources of data: TIDIGITS and VCTK Corpus. The TIDIGITS corpus includes speech data from 326 speakers pronouncing 77 digit sequences (Leonard and Doddington, 1993). The VCTK corpus includes speech data from 109 native speakers of English with various accents (Veaux et al., 2016). Each speaker reads a different set of newspaper sentences. The dataset was originally recorded for the purpose of building HMM-based text-to-speech systems.

Both of our datasets were preprocessed into MFCC features or included preprocessing code to convert `.wav` files into MFCC features for the dataset. We selected the TIDIGITS corpus since it was utilized in our homework and has short utterances. We selected the VCTK Corpus since preprocessing code for the dataset was included with the codebase for a model we are also using.

3.2 Fast Gradient Sign Method

The fast gradient sign method, proposed by (Goodfellow et al., 2014), is a linear perturbation that adds an "imperceptibly small vector" whose "elements are equal to the sign of the elements of the gradient of the cost function with respect to the input", resulting in changes to GoogLeNet's classification of an image.

Let θ be the parameters of a model, $\mathbf{x}$ the input to the model, y the targets associated with $\mathbf{x}$, and $J(\theta, \mathbf{x}, y)$ the cost used to train the neural network. Then we linearize the cost function around the current value of θ, obtaining an optimal max-norm constrained perturbation of

$$\eta = \epsilon sign(\Delta J(\theta, \mathbf{x}, y)) \quad (1)$$

To implement the Fast Gradient Sign Method (FGSM), we utilized code by Goodfellow (Papernot et al., 2016) called cleverhans, a Python library to benchmark machine learning systems' vulnerability to adversarial examples. His repository includes implementations of methods for generating adversarial examples such as the fast gradient method, which are also written in Tensorflow.

We pieced together and modified code from DeepMind's WaveNet (Kim and Park, 2016) and Goodfellow's cleverhans in order to create a network that utilizes the FGSM to generate adversarial examples. We utilized a pretrained model for WaveNet which was trained on VCTKCorpus, thus not performing any training of parameter weights. In the place of validation, we ran our modified network which generates adversarial examples using the fast gradient sign method. As later discussed in 4.1, we experimented with iteratively applying FGSM to an example to create an adversarial example. For one iteration, we would input an example to FGSM which would produce a linearly perturbed adversarial example. For more than one iteration, we would iteratively linearly perturb the adversarial example based on its previous iteration's result.

In section 4.1, we show that our adversarial MFCC features result in predictions that range from misspellings to entirely different meanings. Moreover, the adversarial MFCC features converted back to audio sound nearly identical to the audio from the original MFCC features.

3.3 Fooling Gradient Method

The Fooling Gradient Method is a simple method that is referenced in a number of adversarial example generation projects including (Szegedy et al., 2014), (Karpathy, 2015) and even assignment 3 in Stanford's CS231N (Convolutional Neural Networks for Visual Recognition). In the standard setup for training a neural network, a loss is com-

puted between the network's prediction and the target output. A gradient is computed to minimize this loss. Finally, using backpropagation, the gradient is computed with respect to the parameters of the network and is used to update the parameters with gradient descent. To fool a network, the gradient is computed with respect to the input rather than the parameters and the same gradient descent technique is applied to iteratively modify the input by small deltas that will minimize the loss. The key differences are that the gradients are applied only to the input and the target is the incorrect output that you are targeting to fool the network.

This method was implemented in the Speech-to-Text-WaveNet (Kim and Park, 2016) code base and attacks their pretrained model which can correctly transcribe English with high quality. One key implementation detail is that Tensorflow does not allow modifying input since these are Placeholder objects, and only Variable objects can be modified. Therefore, we add a variable that we call "Noise" that is added to the input before it is fed to the rest of the network. The noise is initialized to zero and then updated by calling `minimize()` on the `GradientDescentOptimize` and only passing the noise variable in as a parameter. We did some search over hyperparameters such as learning rate to find reasonable values for generating our adversarial examples. Small values for the learning rate may take a very long time to converge to the target adversarial example while very large learning rates may increase the noise added to the reconstruction unnecessarily. In section 4.2 we show that we can produce a new `.wav` file that is incorrectly classified by the network but the augmentation is imperceptible to humans.

3.4 Inverse MFCC

To ascertain the similarity between the original audio and their adversarial reconstructions, we needed to obtain audio from cepstral domain representations. This is challenging because phase information is discarded when when calculating the MFCCs, making this a lossy conversion. While achieving a perfect replica of the original is unrealistic, our goal was to approximate an inverse MFCC transform that achieved reasonable reconstruction quality (as per human judgment) without degrading speech recognition accuracy.

At a high-level, extracting MFCC features entails converting the audio to the frequency domain via a discrete Fourier transform, applying a Mel Filter Bank, and then performing an inverse fourier transform to convert the signal back to the time domain, at which point the coefficients consist of mostly uncorrelated values which separate out the vocal characteristics of the glottis and the vocal tract. By harnessing the concepts outlined in Chazan et al (Chazan et al., 2000) and adapting code from the librosa python library[1] (McFee et al., 2017), we implemented an approximate inverse cepstral transformation function. To do so, we applied a discrete Fourier transform to the MFCC input, after which the inverses of the Mel Bank Filter and log transform were applied. The resulting amplitudes were combined with the phases from a "white noise" waveform which served as phase since the original phase information was discarded. This reconstructed frequency domain signal was then converted back to audio with an inverse discrete Fourier Transform, reconstructing the audio signals.

Prediction on Original	Prediction on Adversarial	Target
iofe	iove	above
a nomenents	a nomenens	phenomenon
foro	fro	frog
token	tolken	token
ardshu	ardsh	arch
ther rizen	ther rien	horizon
reflect on	reflection	reflection
tht universa	the universa	universal
americol	amiericl	miracle

Table 1: Results after one iteration of FGSM on single words

4 Experiments

4.1 Fast Gradient Sign Results

We experimented with the number of iterations we applied FGSM to our examples, namely one (Table 1), five (Table 2), and ten (Table 3). We also experimented with the length of our examples, from one-word examples that we spliced manually from VCTKCorpus examples using Praat to full sentences from the same corpus.

As the number of iterations of applying FGSM increased, so did the number of examples that re-

[1] https://github.com/librosa/librosa/issues/424

Prediction on Original	Prediction on Adversarial	Target
iofe	uabove	above
tht universa	the universa	universal
token	tolken	token
magin	imadgined	imagine
stor	store	store
reflect on	reflection	reflection
americol	amierical	miracle
ray bo	raybo	rainbow
foro	fr	frog
plasti	plastic	plastic
ther rizen	ther rison	horizon
ardshu	arhp	arch
a nomenents	a nomenens	phenomenon
piece	peace	peas

Table 2: Results after five iterations of FGSM on single words

Prediction on Original	Prediction on Adversarial	Target
iofe	above	above
bob	bup	bob
reflect on	reflection	reflection
a nomenents	anomenons	phenomenon
token	hoken	token
magin	imadgined	imagine
americol	amierical	miracle
ardshu	arhep	arch
plasti	plastic	plastic
tht universa	the universa	universal
ther rizen	their rison	horizon
foro	fr	frog
piece	peace	peas
stor	store	store
five	fife	five

Table 3: Results after ten iterations of FGSM on single words

Target	After one iteration	After five iterations	After ten iterations
im not looking for a breakthrough	im not **woking** for a **break frough**	im not **loking** for a **break through**	im not **loking** for a **great through**
we have an election in eight days	-	we have an election in **ah** days	we have an election in **a** days
he leaves a wife a son and a daughter	he leaves a wife a son **under dauhter**	-	he leaves a **life** a son **under dauhter**

Table 4: Comparing predictions on adversarial examples after variable iterations on sentences from VCTKCorpus. A "-" indicates that the adversarial example was exactly the same as the original example in that iteration.

sulted in a different prediction by at least one character between original examples and their corresponding adversarial examples. After one iteration, 56.25% examples were changed; after five iterations, 87.5% examples; and after ten iterations, 93% examples.

Noting that sometimes the prediction on the original is also flawed, the predictions on the adversarial examples range from misspellings such as "tolken" (adversarial) vs. "token" (target), and completely different words like "ther rien" (adversarial) vs. "horizon" (target). Surprisingly, sometimes the prediction for the adversarial example is more accurate than the prediction for the original example, resulting in the same value as the target whereas the prediction for the original example is misspelled. This has some interesting implications in that perhaps these linear perturbations sometimes move the prediction closer to the truth, and could possibly help networks train better if taken into consideration.

Next, we moved onto creating adversarial examples from sentences in the VCTKCorpus. We obtained similar results with sentences as we did with single words, though here (Table 4), we noticed that the prediction on some adversarial examples resulted in several completely different words. For example (Table 4), with the example "im not looking for a breakthrough", after one iteration, "breakthrough" had become "break frough". After five iterations, it became "break through", oddly correctly its misspelling but maintaining the whitespace. And after ten iterations, it finally became something completely different: "great through". While this is not entirely a coherent sentence, we were able to transform "breakthrough" into "great through".

Likewise for the example "he leaves a wife a son and a daughter", after one iteration, "daugh-

ter" had become misspelled as "dauhter". The phrase "and a" became "under"–which is not altogether unreasonable, as "and a" and "under" can sound similar. After ten iterations, "wife" became "life". Again, while this is not entirely a coherent sentence, the meaning of "he leaves a life" is completely different than "he leaves a wife".

Also interesting is that while we applied FGSM to the entire example, some words were left untouched while others did change or grew misspellings. What's heartening is that this method did not result in absolute gobbledygook, as simple and cheap it may be as its authors dubbed it.

The increasingly linear-like relationship between number of examples resulting in a different prediction between original and adversarial examples observed with single words was also observed here. After one iteration, 39.84% examples were changed; after five iterations, 60.15% examples; and after ten iterations, 64.06% examples.

These results along with the results on single words suggest to us that the fast gradient sign method, while effective on fooling classifiers on the subject of visual recognition, is also successful for fooling classifiers on the subject of transcription. Simple linear perturbations when repeatedly applied to MFCC features are able to result in different and somewhat coherent predictions that differ from the predictions on the original examples.

4.2 Fooling Gradient Results

The goal of the Fooling Gradient method is to generate adversarial examples. By this, we mean that when we feed the MFCC features for this example into our ASR system, we want the system to output an incorrect *target* prediction. For example, we may start with an audio of a person saying "hello". This would correctly be transcribed by the ASR system to "hello". Then after applying the fooling method, the ASR will transcribe the input as "goodbye" event though when you convert the MFCC features to an audio, a human still clearly hears "hello". To create the adversarial examples, we iteratively apply the Fooling Gradient method until it converges, meaning the ASR system predicts exactly our adversarial target.

We evaluate the results of the Fooling Gradient method on a small set of utterances from the VCTK corpus. We first showed that we were able to apply the method to the CS224S Homework 3 code to fool our simple TIMIT network. We omit this result because it is trivial. To demonstrate the method on shorter utterances, we also used Praat to cut examples from the VCTK corpus into single words (since the shortest utterance in VCTK is 3 words). We also show successfully generated adversarial examples on longer utterances (3 words).

Original Prediction	Target	Num Iteration
rainbow	thunder	3000
i was rubbish	i was awesome	2500
please call stella	siri call police	800

Table 5: Adversarial examples and number of iterations to converge.

In table 5 we see that we can completely change the prediction of the ASR system for these utterances. In the case of converting "please call stella" to "siri call police", we show that this can be used as a malicious attack on the system. Interestingly, the longest utterance actually took the fewest iterations to converge. As a point of reference, generating each of these examples took less than 15 minutes on a modern laptop. Most importantly, the resulting adversarial examples were MFCC features that we could convert back to audio files. When playing back the audio files, the adversarial examples sounded exactly the same as the original reconstructions, meaning the changes are imperceptible to humans (because they are mostly small deltas and white noise). It is worth noting again that reconstructing MFCCs to audio does introduce some white noise, so no reconstruction sounds exactly like the original, but the words are still clearly perceptible to humans.

Interestingly, we did not observe a correlation between the length of the utterance and the number of iterations required to converge to an adversarial example. This means that generating longer adversarial examples does not suffer from any scale issues. Also, we noticed that certain transformations seemed to be easier than others. For example, it took fewer iterations to generate an adversarial example of "cat" for the utterance "frog" than to generate the example for target "dog". This is surprising because "frog" is much closer to "dog" than it is to "cat".

Another interesting observation is that different steps along the way to converging to a target transcription take an unexpected number of iterations. For example, we show that we can convert from "rainbow" to "thunder" in 3,000 iterations. However, converting from "rainbow" to "orinder" only takes 300 iterations (which is just an intermedi-

ate transcription during the generation of "thunder"). The network then predicts "oender" from iteration 500 to 1,400. And then "toender" until iteration 2,600. While we can not explain these unusual patterns in the intermediate transcriptions during generation, it demonstrates a complex relationship between the small noise we are applying to the MFCC features and the various effects on the output.

4.3 Audio Reconstruction

Samples of reconstructed original and adversarial samples can be accessed via github[2]. Overall, the inverse MFCC resulted in identifiable audio samples. As seen in Figure 1, reconstructed audio using a low number of MFCC features (20) still resulted in a waveform that somewhat captured the original signal (albeit with more noise and sound artifacts).

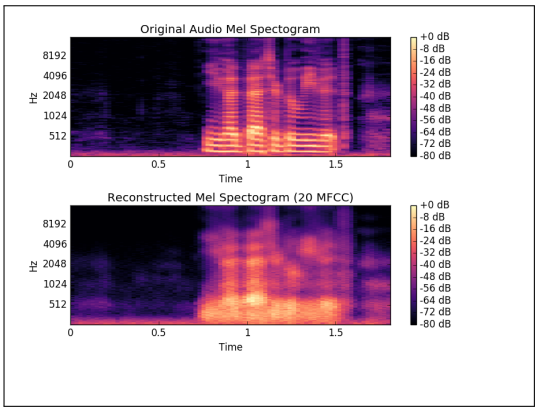

Figure 1: Audio of sentence "You had better believe it" - Original and Reconstructed

The quality improved as more MFCC features were included (since more of the signal was captured), at the expense of increasing system complexity. This can be seen in Figure 2, in which the 100 MFCC reconstruction has much less noise in its spectrum than the 20 MFCC reconstruction.

Despite this, we chose to bias towards high speech recognizer accuracy and to use 20 MFCC. For both methods, this resulted in reconstructed signals for which the original audio was detectable, sound quality was substantially lowered, and the adversarial and original audios were essentially indistinguishable to the human ear, as seen in Figures 3 and 4.

Investigations into how much perturbation the input vector could receive without dramatically

[2]https://github.com/daniter-cu/AdversarialSpeech

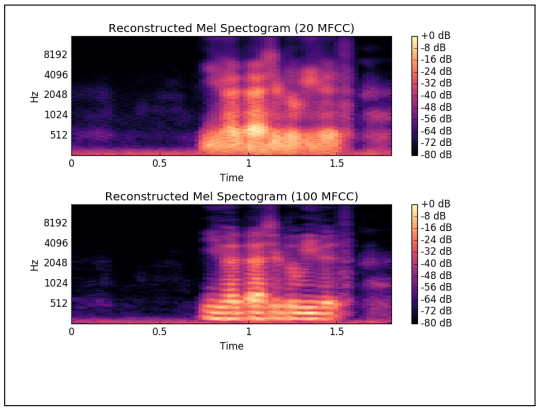

Figure 2: Audio of sentence "You had better believe it" - Reconstructions with different numbers of MFCCs

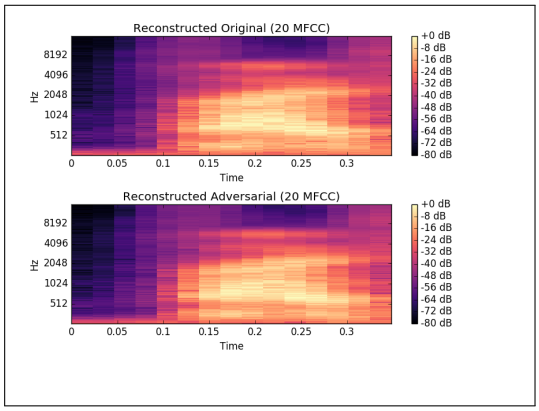

Figure 3: Reconstructed audio from MFCC for "five" and adversarial example "fife", using Fast Gradient Sign Method

changing the audio found that these are fairly robust. For example, adding a random noise vector to the MFCC with an average magnitude approximately 25% of the original signal still resulted in identifiable audio, albeit with noticeable distortions.

5 Future Work

One major point of improvement we can focus on in the future is to improve the audio reconstructions. There are two ways we can accomplish this. One, we can employ more advanced techniques for reconstruction that have proved better results. Chazan et al (Chazan et al., 2000) published a paper with a publicly available demo that shows that MFCC features can be converted back to high quality audio. An alternative approach would be omit the MFCC featurization step completely and apply these attacks directly to an end to end neural

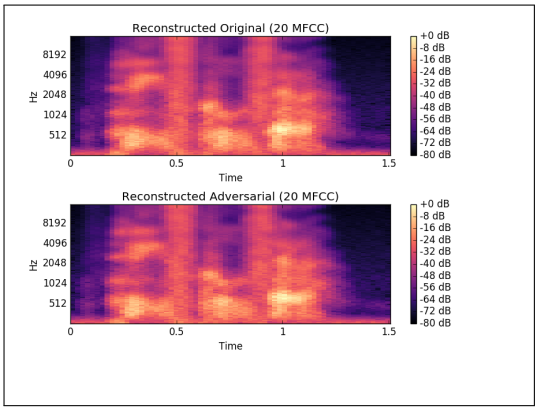

Figure 4: Reconstructed audio from MFCC for "Please Call Stella" and adversarial example "Siri Call Police", using Fooling Gradient Method

model that takes raw audio as input. This should allow the back propagation of the gradients all the way to the raw audio bits and should also produce more high quality "reconstructions".

This work completely focuses on white box attacks. Arguably a more interesting and challenging extension would be to generate adversarial examples for black box systems. In other words, generating similar examples to those that we generate in this work that also fool main stream systems like Siri and Google VTT for which we do not have access to models.

While our work focused on creating adversarial MFCC features and relied on an inverse MFCC transform in order to hear the resulting audio, other work may instead operate directly on the image of the waveform or spectrogram. This way, the issue of a lossy transform will not be an impeding factor.

6 Conclusion

We are successfully able to fool WaveNet with adversarial examples created using the fast gradient sign method as well as the fooling gradient method, showing that these methods used in the realm of image recognition are also effective in speech recognition. The resulting reconstructions of the adversarial MFCC features sound identical to the original to the human ear (when accounting for the noise introduced by the inverse MFCC transform), but WaveNet predicts either misspellings or entirely different words.

In the future, one might take our results and use it and use it to attack systems that may be deployed in private homes. However, these results also provide some insight into defense vectors that can be considered when building more robust systems. Others may take our results and use adversarial examples to improve the performance of ASR systems.

References

Dan Chazan, Ron Hoory, Gilad Cohen, and Meir Zibulski. 2000. Speech reconstruction from mel frequency cepstral coefficients and pitch frequency. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. IEEE, volume 3, pages 1299–1302.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* .

Andrej Karpathy. 2015. Breaking linear classifiers on imagenet. http://karpathy.github.io/2015/03/30/breaking-convnets/.

Kim and Park. 2016. Speech-to-text-wavenet. https://github.com/buriburisuri/.

R. Gary Leonard and George Doddington. 1993. Tidigits. https://catalog.ldc.upenn.edu/ldc93s10.

Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, Dan Ellis, Fabian-Robert Stoter, Douglas Repetto, Simon Waloschek, CJ Carr, Seth Kranzler, Keunwoo Choi, Petr Viktorin, Joao Felipe Santos, Adrian Holovaty, Waldir Pimenta, and Hojin Lee. 2017. librosa 0.5.0. https://doi.org/10.5281/zenodo.293021.

Pratyush Mishra Tavish Vaidya Yuankai Zhang Micah Sherr Clay Shields David Wagner Nicholas Carlini and Wenchao Zhou. 2016. Hidden voice commands. *Usenix Security* .

Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, and Patrick McDaniel. 2016. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768* .

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. *arXiv:1602.02697* .

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Dumitru Erhan Joan Bruna, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* .

Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al. 2016. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit .