

What Did You Just Say? Toward Detecting Imperceptible Audio Adversarial Attacks

Stanford CS224N Custom Project

John Mistele, Ryan Cao

Department of Computer Science

Stanford University

{jmistele, ryancao}@stanford.edu

Abstract

Adversarial examples, i.e. those which any human would perceive one way but which are confidently mis-perceived by neural networks, have been shown to exist for neural nets in general [1], [2], and in particular ASR (automatic speech recognition) systems. As ASR systems begin to increasingly rely on deep learning models, such systems become increasingly vulnerable to under-the-radar adversarial commands. Building atop attacks such as DolphinAttack [3] and Houdini [4], Carlini and Wagner present a modified version of their iterated CW Attack [5] which achieves a 99.9% success rate against Mozilla’s DeepSpeech transcription network. To this end, we study Carlini’s attack and offer mechanisms for detecting adversarially-attacked audio. We show that attacked examples demonstrate a different distribution of downstream logits in DeepSpeech, giving a first means at distinguishing these examples. We then show that Carlini’s examples are susceptible to minor parameter changes in early-layer feature extraction (MFCC) which do not greatly impact natural, authentic audio; moreover, we show these differences are detectable downstream in DeepSpeech’s transcriptions, yielding a very effective means of distinguishing Carlini’s examples from natural audio, and a means of recovering an approximation of the original transcription.

Project Metadata

• Mentor: Dilara Soylu | External collaborators: None | Sharing Project: N/A

1 Introduction

The vast majority of adversarial attacks [2], [6] and defenses [7], [8] have focused on the vision domain, with seldom an application in the NLP domain [9], largely due to the fact that language is discrete, and changing even a single character is perceptible in a way that changing even many pixels is not. In ASR (automatic speech recognition), however, in which networks are asked to output tokens which transcribe an essentially-continuous frequency signal, there is plenty of room for some of the same techniques to apply – such attacks, indeed, are potentially more worrisome than their visual counterparts. As described by [10], aside from the crude – but ultimately obnoxious, rather than truly harmful – applications seen in [3] of calling 911 or activating Siri, an attacker might broadcast a hidden command over a protest causing all protestors’ phones to text a particular number, allowing tracking of protestors’ identities; on the cybersecurity side, an attacker might navigate a victim’s browser to a particular website and download a malicious payload, giving the attacker an “in” to potentially the whole device.

Given the increasing number of manifestations of ASR – text dictation, hands-free driving commands, Amazon Echo/Google Home, etc. – it is of tantamount importance that we secure such systems, and detect attacked inputs so that we might ignore their intent while recording the audio for further study.

2 Related Work

We offer a quick recap of attacks and defenses as presented in the adversarial audio literature. We begin with a historic attack, the Fast Signed Gradient Method (FSGM).

FSGM

Goodfellow et. al. [2] introduce the fast signed gradient method for convolutional networks over images: given an input image x and true label y , FSGM produces an adversarial example x' , where

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x(J_\theta(x, y))) \quad (1)$$

note that ϵ is a hyperparameter tweaking the amount of perturbation each pixel takes. Intuitively, (1) is simply a single-step gradient ascent over pixels (rather than parameters) which crudely approximates the direction of sharpest loss gain, ideally crossing a decision boundary and causing the classifier to incorrectly classify x' .

Houdini

Houdini [4] takes a task-oriented approach and proposes a surrogate loss function towards a task objective (e.g. IOU for segmentation, CELoss for classification, BLEU for NLT, etc). Let $(x, y) \in X \times Y$ be our training example, and let $g_\theta : X \times Y \rightarrow \mathbb{R}$ be our neural network assigning a score to each (example, label) pair. Finally, let $\hat{y} = \arg \max_{y \in Y} g_\theta(x, y)$ be our network prediction, and let $\ell(\hat{y}, y)$ be our task loss. Houdini proposes generating adversarial example x' using

$$x' = \arg \max_{x': \|x' - x\|_p \leq \epsilon} \ell_H(\theta, x', y) \quad (2)$$

$$\text{where } \ell_H(\theta, x, y) = Pr_{\gamma \sim \mathcal{N}(0,1)}[g_\theta(x, y) - g_\theta(x, \hat{y}) < \gamma] \cdot \ell(\hat{y}, y) \quad (3)$$

The intuition here is that although ℓ is not differentiable, the first term on the RHS of (3) is and allows the network to tradeoff the task loss with its score difference, i.e. confidence, in the correct label being $\hat{y}$.

Cocaine Noodles

The authors of [10] propose using an acoustic feature extractor which takes as input the MFCC (Mel Frequency-Cepstrum Coefficients) computation parameters and raw audio, and produces a salient set of acoustic features in MFCC space which is then inverted back into audio via an inverse MFCC computation. Since MFCC is not injective, noise is necessarily added in the inverse computation, and since only salient acoustic features are input, the resulting acoustic waveform is mangled, making it difficult for a human to understand, yet those same salient features appear to any classifier trained on post-MFCC-processed data, allowing the classifier to correctly output the original transcription.

Carlini-Wagner Audio Attack

In [11], Carlini and Wagner present a white-box audio transcription attack (which we denote the CW audio attack) which is able to generate, from any source audio file, a highly similar audio file (indistinguishable, in most cases, by humans) targeting nearly any other phrase which then gets labeled by Mozilla's DeepSpeech 0.4.1 network [12] as said target 99.9% of the time. The CW audio attack roughly works as follows (we refer readers to [11] for a full description):

Consider the Decibel metric for a sound signal, $dB(x) = \max_i 20 \cdot \log_{10}(x_i)$ (i.e. the a measure of the single "loudest" frame). Moreover, consider the perturbation measure $dB_x(\delta) = dB(\delta) - dB(x)$. Let C be our classifier and t our target transcription. Our objective is the following:

$$\text{minimize } dB_x(\delta) \quad \text{such that } C(x + \delta) = t \quad (4)$$

Which can be reformulated, through an appropriate choice of ℓ , as

$$\text{minimize } dB_x(\delta) + c \cdot \ell(x + \delta, t) \quad (5)$$

where $\ell(x', t) \leq 0 \iff C(x') = t$. (For more details, see [5].) In this case, we use the CTC-Loss, which is formulated thus (here, we point readers to the original paper [13] for details):

$$\ell(x', t) = -\log \sum_{\pi \in \Pi(y, t)} \prod_i t_{\pi_i}^i \quad (6)$$

Where y is the discrete distribution over characters for each frame (as produced by the neural net) and $\pi \in \Pi(p, t)$ is every possible character labeling of the frames in x' which reduces to t (with reduction being a simple function which eliminates all adjacent duplicates and ϵ tokens in a string).

Next, we discuss several detection approaches.

Audiovisual Synchronization

In [14], Ma et. al. present a synchronization scoring framework for detecting adversarial attacks against audiovisual inputs (i.e. video + audio), in which they train a SyncNet model [15], a two-stream network trained using a contrastive loss to literally drive together feature representations from matching audio signals/video frames. They show that a trained SyncNet’s final synchronization score is an excellent indicator of whether a particular video has been attacked.

Acoustic Decoy

In [16], Kwon et. al. propose an audio-modifying approach for detecting adversarial inputs: the modifications, designed to smear adversarial noise but keep the fidelity of any unattacked recording, are applied to the original x to produce x' . They then run the classifier f on both x and x' , and check the ratio between the Levenshtein distance between letters in the output $f(x)$ and those in $f(x')$ and the total number of letters in $f(x)$. Audio modification techniques include a) low-pass filtering, in which high-frequency audio components are zeroed out, b) bit smashing from a 16-bit wave representation to an 8-bit representation, and c) removing audio silence from clips, as determined by a silence classifier.

Temporal Dependency

In [17], Yang et. al. propose a truncate-then-transcribe vs. transcribe-then-truncate method for adversarial audio detection. In particular, their method classifies two pieces of input audio, x and $x[:k]$, and then compares $f(x)[:k]$ to $f(x[:k])$, where f is the ASR software, and $f(x)[:k]$ is the prefix of the whole-sequence transcription which corresponds with $x[:k]$. They show that adversarial attacks on audio files leverage temporal dependencies between audio segments, and dividing the segments tends to ruin such crafted noise, causing a large mismatch in word error rate (their metric of choice) between $f(x)[:k]$ and $f(x[:k])$ for adversarial x , but little difference thus for unattacked x , allowing them to distinguish between the two.

3 Approach

Our approach uses the CW audio attack [11] against Mozilla’s DeepSpeech 0.4.1 model [12], which comes pre-trained on the CommonVoice dataset [18], a collection of over 160k short English audio examples and their transcriptions. We focus, in particular, on two different detection methods which both achieve high accuracy and require no additional fine-tuning or adversarial training on the DeepSpeech model itself. (In fact, we precompute DeepSpeech output logits and train several detection algorithms over these.) Our approach differs from existing detection techniques in that a) to the best of our knowledge, we are the first to approach attack detection via logit classification, and b) we are the first to use a multi-MFCC pass with different parameters to disturb adversarial noise. Our specific approaches are as follows:

Logit Distribution Classification

Our first intuition is that there must be a significant difference between the output distribution of DeepSpeech logits (i.e. features extracted at its final layer, sans argmax) when it is run over an unattacked sample x versus an adversarial sample x' . To this end, we took the following steps:

- **Baseline Experiment:** Given a sample x , call the DeepSpeech output logits $l_x \in \mathbb{R}^{T \times 29}$, where T is the number of audio frames DeepSpeech breaks x into. We train an RBF (radial basis kernel; generalization of monomial power features to infinite dims) binary classifier over the following features:
 - $\text{median}_{t \in T}(\max_{c \in \{0, 28\}}(l_{xt}[c])) \in \mathbb{R}$ – the median over the max logits for each timestep $t \in T$.
 - $\text{median}_{t \in T}(\|(l_x)_{t+1} - (l_x)_t\|_2) \in \mathbb{R}$ – the median over the norm of the per-timestep logit differences.
- **Multi-Layer Perceptron:** We train an MLP over the DeepSpeech logits to predict whether a sample was attacked.
- **Linear Classifier:** We train a two-layer feedforward model over the DeepSpeech logits with the same task.
- **Long Short-Term Memory RNN:** We train an LSTM over a truncated version of the DeepSpeech logits and decode a binary classification from the final hidden state.

Since Carlini’s approach makes minimal changes to the audio to obtain a target transcription, we anticipate that the attacked logit-wise maxima $m_x \in \mathbb{R}^T$ will have different distribution from authentic logit-wise maxima; additionally, we might expect authentic audio to have a more smooth distribution of logits over time, so we also anticipate norms of the time-wise discrete differences $d_x \in \mathbb{R}^{T-1}$ to have different distribution between authentic and attacked examples. Additionally, we believe that the raw distributions between undisturbed x and adversarial x' is such that an MLP or linear model should accurately capture the difference (and validate that this is indeed the case in section 4); a feedforward network is brittle with respect to input size; however, and so we train an LSTM and demonstrate that it, too, works quite well.

Transcription Robustness Classification

We follow the thread of using the DeepSpeech model’s outputs to ascertain if it has been attacked or not. Our second intuition is that, given Carlini back-propagates through the MFCC process to introduce minimal distortion, the attacked example transcriptions are likely brittle to changes in the MFCC window length; however, on authentic examples, we expect DeepSpeech to obtain nearly the same accuracy, as a larger or smaller MFCC window shouldn’t change the rendered features much at all. Thus, we took the following steps:

- **Transcription Generation:** Using the 5000 attacked train and 500 attacked val examples from CommonVoice using CW’s audio attack (see above) and 5000 authentic train and 500 authentic val examples, we re-evaluate the output transcriptions of DeepSpeech on these 11000 examples using MFCC window lengths $W' \in \{1024, 256\}$, and compare to the transcription of DeepSpeech on these 11000 examples using the base MFCC window length $W = 512$.
- **Transcription Robustness Classifiers:** Given a pair of transcriptions $(T_{base}, T_{W'})$ of the same audio using different MFCC window sizes, we may compute the Character Error Rate (CER) and Word Error Rate (WER) between them. Given thresholds b_{cer} and b_{wer} , we obtain classifiers C_{cer} and C_{wer} which classify an example as attacked when $CER(T_{base}, T_{W'}) > b_{cer}$ or $WER(T_{base}, T_{W'}) > b_{wer}$ respectively.

Convolutional Neural Network Classification

To verify our intuition that using DeepSpeech’s own outputs for the classification task has merit, we attempted to identify attacked examples using audio features different from those obtained by DeepSpeech. Thus, we attempted our Logit Distribution Classification approach with features coming from the Convolutional feature extractor of Wav2Vec 2.0, a different ASR architecture [19].

Transcription Repair (Restoration Task)

Following our success with Transcription Robustness Classification, we measure the effectiveness of changing the MFCC window length in our DeepSpeech model on recovering the original transcription

for an adversarial example. We evaluate with Character Error Rate (CER) and Word Error Rate (WER) relative to the authentic audio transcription instead of the ground-truth label, as this better captures the effectiveness of our approach at defending DeepSpeech.

4 Experiments

4.1 Data

We generated our own dataset using a subset of the CommonVoice [18] dataset and the CW audio attack [11] against Mozilla DeepSpeech [12]. CommonVoice is an English audio transcription dataset of over 160k samples of raw spoken audio and transcribed labels, where the task is to reconstruct the original text given the audio. As described above, the CW audio attack is an iterative, optimization-based white-box attack which is able to target an arbitrary phrase given an input audio source, and Mozilla DeepSpeech is an audio transcription network which extracts features from convolution and linear layers over the MFCC spectrogram of an audio file and runs a Bi-LSTM model to classify each audio frame over a pre-determined set of characters.

Audio Dataset Generation: We attacked 5000 train and 500 val examples from CommonVoice using CW’s audio attack with CW’s default hyperparameters (see [11] for details) and 200 iterations of refinement.

- Train set: 10000 total examples (5000 unattacked, 5000 attacked).
- Val set: 1000 total examples (500 unattacked, 500 attacked).

DeepSpeech Logits Dataset Generation: We fed the raw audio dataset through an MFCC preprocessing step (window size = 512) and then through DeepSpeech, generating a set of DeepSpeech logits $l_x \in \mathbb{R}^{T \times 29}$ for each (attacked, unattacked) audio file x . We used the pretrained DeepSpeech v0.4.1 out-of-the-box with no further fine-tuning.

4.2 Evaluation Method

Adversarial Example Classification Task: We use a model’s validation set accuracy and confusion matrix as our primary evaluation methods. In particular, for a trained model f we compute the TP/TN/FP/FN counts of f on our 1000-example validation set, and (as described below) manually examine the few misclassified examples.

Restoration Task: We borrow both the word error rate (WER) and character error rate (CER) metrics from the OCR (optical character recognition) domain to characterize the difference (for a single audio example x) between the original transcription T_{base} under a 512-sample MFCC window and the transcription $T_{\{256,1024\}}$ under a $\{256, 1024\}$ -sample MFCC window. Formally, let T and T' be string transcriptions $\in \Sigma^*$. Let i_w and i_c be the number of insertions (of words and characters, respectively), d_w and d_c the number of deletions, s_w and s_c the number of substitutions, and n_w and n_c the total number of words/characters in the reference text (T , in our case). Note that i, s, d are minimized by computing the Levenshtein distance [20] between T and T' . Then

$$WER(T, T') = \frac{i_w + d_w + s_w}{n_w} \quad CER(T, T') = \frac{i_c + d_c + s_c}{n_c} \quad (7)$$

4.3 Experimental Details

DeepSpeech Logit Statistics RBF Classifier: We fit scikit-learn’s GaussianProcessClassifier to our two-feature vector (as described above). We used an RBF kernel with a kernel size of 1, and kept the default sklearn hyperparameters for fitting.

DeepSpeech Logit Linear Model: We train a single-layer linear model with input dim 32,712 (1128 * 29, the flattened DeepSpeech logit size) and output dim 1, with a single sigmoid activation at the output. We use binary cross-entropy as our loss and Adam with a learning rate of 1e-5, and train for 10 epochs over the DeepSpeech Logits dataset.

DeepSpeech Logit MLP: We train a two-layer feedforward network with an input size of 32,712 (1128 * 29) and a hidden size of 282, with a single ReLU in-between and a sigmoid at the output.

Again, we use binary cross-entropy as our loss, and Adam with a learning rate of $1e-5$, and train for 10 epochs over the DeepSpeech Logits dataset.

DeepSpeech Logit LSTM: We train a 2-layer LSTM model with a hidden dim of 512 and a single linear decoder with input dim 512 and output dim 1, connected to the final hidden state. We use the same training hyperparameters as for the MLP and linear model.

Wav2Vec 2.0 CNN Features Linear Model: We train a linear model over the Wav2Vec CNN features, with an input size of 39,936 ($768 * 52$, the flattened Wav2Vec feature dimension), using the same training hyperparameters as above.

Wav2Vec 2.0 CNN Features LSTM: We train an LSTM model with a hidden dim of 512 and a single linear decoder connected to the final hidden state. We use the same training hyperparameters as above, but train for 30 epochs rather than 10.

Transcription Robustness Classifier: Given a pair of transcriptions $(T_{base}, T_{W'})$ of the same audio using different MFCC window sizes, we may compute the Character Error Rate (CER) and Word Error Rate (WER) between them. Given thresholds b_{cer} and b_{wer} , we obtain classifiers C_{cer} and C_{wer} which classify an example as attacked when $CER(T_{base}, T_{W'}) > b_{cer}$ or $WER(T_{base}, T_{W'}) > b_{wer}$ respectively. Thus, we obtain precision-recall plots for $W' \in \{256, 1024\}$ and CER vs. WER thresholds:

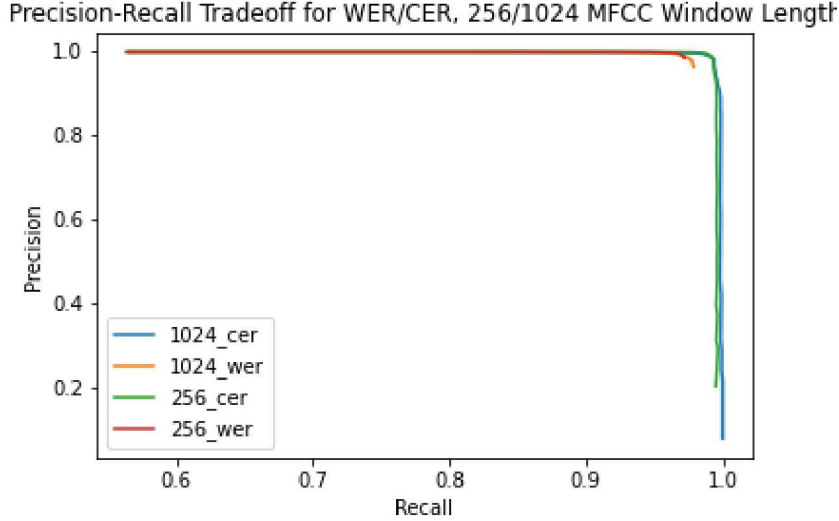


Figure 1: Precision-Recall Tradeoff for thresholds between zero and one on the train set. Accuracies are high enough that there is little difference between our four classifiers; we take the 1024 sample window length and CER metric as our validation classifier, and tune our threshold to maximize train set accuracy.

Transcription Repair (Restoration Task): For each audio clip in the validation set, both natural and attacked, we transcribe the audio using a DeepSpeech model with an MFCC window size of 512 samples (the default, or "No Restoration" setting, which Carlini trains his examples against) as well as window sizes of 256 samples and 1024 samples. For our Restoration Task CER and WER metrics (Table 2), we compare all transcriptions against the default 512 sample MFCC DeepSpeech transcription, listed as "Authentic Samples No Restoration", (which accordingly has 0.0% WER and CER against its own transcriptions).

4.4 Results

Our results are as follows. Note that all accuracies reported in Table 1 are with respect to our 1000-sample validation set.

Takeaways: The RBF baseline performs fairly well, although the features given are admittedly quite crude summary statistics and likely not enough to learn a fine-grained decision boundary. The Wav2Vec classifiers also slightly underperform, as they are trained over the features produced by a

Table 1: Classification Task

Model	Val Accuracy
DeepSpeech Logit Statistics RBF classifier (Baseline)	79.0%
Wav2Vec 2.0 CNN Features Linear Model	71.0%
Wav2Vec 2.0 CNN Features LSTM	80.7%
DeepSpeech Logit Linear Model	98.0%
DeepSpeech Logit MLP	98.2%
DeepSpeech Logit LSTM	99.1%
Transcription Robustness Model	99.3%

wholly different network over the attacked/unattacked examples. Surprisingly, the linear DeepSpeech logit model performs extremely well, and performance only increases when we scale up to an MLP or an LSTM. This strongly points to the conclusion that the features of the attacked model, i.e. DeepSpeech in this case, are genuinely quite indicative of whether the model is being attacked via its top-level logits, and serves as a potent way of detecting such attacks. Finally, the transcription robustness model exceeded all expectations and performed stellarly, revealing that noise-generating, optimization-based attacks such as the CW audio attack are particularly vulnerable to changes in the audio processing steps – we noticed, in particular, that during the attack generation step, the MFCC computation had to be backpropagated through, and estimated that the attack, then, must be sensitive to our choice of MFCC hyperparameters. A valid detection strategy thus would be to process any raw audio under two different choices of MFCC window size (e.g. 512 and 1024), run DeepSpeech on both, and compute the CER between the two transcriptions – a significant difference therein would indicate that the audio has been attacked.

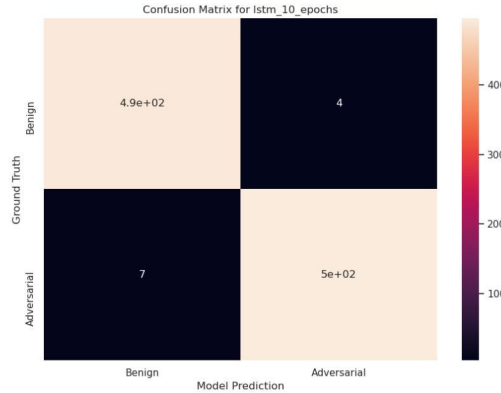


Figure 2: Confusion Matrix for DeepSpeech Logit LSTM

Since our dataset is perfectly balanced, we plot the confusion matrix for the DeepSpeech Logit LSTM (confusion matrices for the other logit models are very similar) and note that for our particular validation set, there is a slight bias towards false negatives.

Restoring Transcriptions: The success of recovering the original transcription on attacked audio is shown in Table 2. We can see changing the window length of the MFCC computation from 512 to 1024 reduces the CER on attacked examples from 86.2% to 52.4%. In some cases, this recovers the transcription perfectly, in others, quite poorly:

Natural DeepSpeech Transcription: i don't care what you charge them with
 Attacked DeepSpeech Transcription: fear estate savings call
 1024 MFCC DeepSpeech Transcription: i don't care what you charge them with

Table 2: Restoration Task

Model	WER	CER
Carlini Samples No Restoration (512-sample MFCC Window)	100.8%*	86.2%
Authentic Samples No Restoration (512-sample MFCC Window)	0.0%	0.0%
Carlini Samples 256-sample MFCC Window	86.3%	56.4%
Carlini Samples 1024-sample MFCC Window	84.2%	52.4%
Authentic Samples 256-sample MFCC Window	28.0%	9.5%
Authentic Samples 1024-sample MFCC Window	27.5%	9.0%

*WER and CER may exceed 100% if the proposed transcription is longer than the reference transcription.

Natural DeepSpeech Transcription: ben i don't get paid

Attacked DeepSpeech Transcription: photos amendment stopping westminster

1024 MFCC DeepSpeech Transcription: ten hunho him even throing wisiter

Note: We considered adding noise and masking the audio in the frequency and time domains, however this yielded no visible improvement in the transcription, so we have omitted these baselines.

5 Analysis

The linear model over just logits performs suspiciously well, leading us to believe that there may be distributional discrepancies between the original transcription and the targeted transcription. In particular, targeted transcriptions were generated by randomly sampling a number of words from [21] proportional to the original audio length. A better approach (ensuring the same character/word/length distribution over transcriptions) would be to simply permute the original transcriptions and use those as targets for different datapoints (e.g. rather than $(x_1, y_1), (x_2, y_2)$ our attacked dataset could look like $(x_1, y_2), (x_2, y_1)$).

We do a deep dive on the Transcription Robustness Model failure cases, and find that they seem to be audio clips for which the *original* transcription is already low-quality or low-confidence. For example, the model incorrectly classified the pair

$$(T_{base}, T_{1024}) = (\text{"by vacies hav beese the ottetent"}, \text{"by vations of pease hear cotect"})$$

as adversarial; these are two transcriptions of authentic audio. However, we see that the original transcription here is already nonsense; moreover the authors listened to the clip and can offer no better transcription than either of these. Thus, further improvements in this approach are gated to some extent by the quality and consistency of the audio and the underlying ASR model.

6 Conclusion

We draw the following conclusions and propose some future directions.

Conclusion 1: The attacked model's own logits provide an excellent indicator of whether an input is adversarial or not. In particular, one can attach an "LSTM head" to the final layer of logits whose sole purpose is to output the probability that the input is an attacked input.

Conclusion 2: The CW audio attack is quite brittle to changes in MFCC hyperparameters, and such optimization-based attacks can be easily detected by a method similar to our Transcription Robustness Model.

Future Work 1: Conclusion 2 begets one straightforward direction: is there an attack which might produce convincing adversarial examples fooling *all* reasonable MFCC window lengths simultaneously?

Future Work 2: Conclusion 1 begets another attack direction: Is there an attack which, *knowing* (in a white-box manner) that the target network harbors an LSTM-style classifier, is able to generate an adversarial example which defeats both the classifier and a human observer?

References

- [1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.
- [2] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.
- [3] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Oct 2017.
- [4] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models, 2017.
- [5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017.
- [6] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses, 2020.
- [7] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser, 2018.
- [8] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *CoRR*, abs/1805.06605, 2018.
- [9] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems, 2017.
- [10] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. Cocaine noodles: Exploiting the gap between human and machine speech recognition. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, Washington, D.C., August 2015. USENIX Association.
- [11] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text, 2018.
- [12] Mozilla. Mozilla deepspeech: <https://github.com/mozilla/deepspeech>.
- [13] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning, ICML '06*, page 369–376, New York, NY, USA, 2006. Association for Computing Machinery.
- [14] Pingchuan Ma, Stavros Petridis, and Maja Pantic. Detecting adversarial attacks on audiovisual speech recognition, 2021.
- [15] Joon Son Chung and Andrew Zisserman. Out of time: automated lip sync in the wild. In *Workshop on Multi-view Lip-reading, ACCV*, 2016.
- [16] Hyun Kwon, Hyunsoo Yoon, and Ki-Woong Park. Acoustic-decoy: Detection of adversarial examples through audio modification on speech recognition system. *Neurocomputing*, 417:357–370, 2020.
- [17] Zhuolin Yang, Bo Li, Pin-Yu Chen, and Dawn Song. Characterizing audio adversarial examples using temporal dependency, 2019.
- [18] 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, 2020.
- [19] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.
- [20] Vladimir Levenshtein. Binary codes with correction of deletions, insertions and substitutions of symbols, 1965.
- [21] Eric Price. <https://www.mit.edu/~ecprice/wordlist.10000>.