Speech Coding and Audio Preprocessing for Mitigating and Detecting Audio Adversarial Examples on Automatic Speech Recognition

Krishan Rajaratnam
The College
The University of Chicago
Chicago, Illinois, USA
Email: krajaratnam@uchicago.edu

Basemah Alshemali
Department of Computer Science
University of Colorado
Colorado Springs, Colorado, USA
Email: balshema@uccs.edu
Department of Computer Science
Taibah University
Al-Medina, KSA

Kunal Shah
Department of Biology
University of Florida
Gainesville, Florida, USA
Email: kshah1997@ufl.edu

Jugal Kalita
Department of Computer Science
University of Colorado
Colorado Springs, Colorado, USA
Email: jkalita@uccs.edu

Abstract

An adversarial attack is an exploitative process in which minute changes are made to a natural input, causing that input to be misclassified by a neural model. Due to recent trends in speech processing, this has become a noticeable issue in speech recognition models. In late 2017, an attack was shown to be quite effective against the Speech Commands classification model. Limited-vocabulary classifiers, such as the Speech Commands model, are used quite frequently for managing automated attendants in traditional telephony and voice over IP (VoIP) contexts. As such, this research examines the effectiveness of VoIP speech coding in mitigating audio adversarial attacks when compared to more primitive forms of audio preprocessing and shows that an ensemble defense in tandem with speech coding is more robust than other forms of preprocessing defenses in mitigating adversarial examples. This research also proposes a new metric for evaluating preprocessing defenses against adversarial attacks. Additionally, this research explores using speech coding and various other forms of preprocessing for detecting adversarial examples.

***Index Terms*—adversarial attack, speech recognition, deep learning, audio compression, speech coding**

Introduction

The growing use of deep learning models necessitates that those models be accurate, robust, and secure. However, these models are not without exploitable flaws. Initially applied to computer vision systems (Szegedy et al. 2014), the generation of adversarial examples is a process in which seemingly imperceptible changes are made to an image, with the purpose of inducing a deep learning based classifier to misclassify the image. The effectiveness of such attacks is quite high, often resulting in misclassification rates of above 90% in image classifiers (Goodfellow, Shlens, and Szegedy 2015). Due to the exploitative nature of these attacks, it can be difficult to defend against adversarial examples while maintaining general accuracy.

The generation of adversarial examples is not just limited to image recognition. Although speech recognition traditionally relied heavily on signal processing and hidden Markov models, the gradual growth of computer hardware capabilities and available data has enabled end-to-end neural models to become more popular and even state of the art. As such, speech recognizers that rely heavily on deep learning models are susceptible to adversarial attacks. Recent work has been done on the generation of targeted adversarial examples against a convolutional neural network trained on the widely used Speech Commands dataset (Alzantot, Balaji, and Srivastava 2017) and against Mozilla's implementation of the DeepSpeech end-to-end model (Carlini and Wagner 2018), in both cases generating highly potent and effective adversarial examples that were able to achieve up to a 100% misclassification rate. Due to this trend, the reliability of deep learning models for automatic speech recognition is compromised; there is an urgent need for adequate defense against adversarial examples.

Related Work

The attack against Speech Commands described by Alzantot et al. (Alzantot, Balaji, and Srivastava 2017) is particularly relevant within the realm of telephony, as it could be adapted to fool limited-vocabulary speech classifiers used for automated attendants. This attack produces adversarial examples using a gradient-free genetic algorithm, allowing the attack to penetrate the non-differentiable layers of preprocessing typically used in automatic speech recognition.

Methods of defense against adversarial examples can be divided into two categories: mitigation (i.e. retrieving the original label of an adversarial example) and detection (i.e. declaring a given example as adversarial or benign). This section will discuss audio preprocessing mitigation methods and draw attention to a preprocessing-based ensemble detection method used for detecting adversarial examples in the image space.

Audio Preprocessing Defenses

Recent work within computer vision classifiers has shown that some preprocessing, such as JPEG and JPEG2000 image compression (Aydemir, Temizel, and Temizel 2018), cropping and resizing (Graese, Rozsa, and Boult 2016), and pixel deflection (Prakash et al. 2018) have a certain degree of success in defending against adversarial attacks. In a similar vain, preprocessing defenses have also been used for defending against adversarial attacks on speech recognition. Work has shown that using local smoothing, downsampling, and quantization can be somewhat effective in

disrupting adversarial examples produced by the attack of Alzantot et al. (Yang et al. 2018). While quantizing with $q = 256$, Yang et al. were able to achieve their best result of correctly retrieving the original label of 63.8% of the adversarial examples, with a low cost to general model accuracy. As quantization causes the amplitudes of sampled data to be rounded to the closest integer multiple of q, adversarial perturbations with small amplitudes can be disrupted.

Work has also been done in employing audio compression, Hertz shifting, noise reduction, and a low-pass filter (Lemmond and Fitzgibbons 2018) to defend against Carlini and Wagner's attack (Carlini and Wagner 2018) on the DeepSpeech model. The results of Lemmond and Fitzgibbons show that the most promising preprocessing defense tested was the low-pass filter, which achieved a 90.11% success rate in defeating Carlini and Wagner's adversarial examples while maintaining a relatively high general accuracy of 90.91%. This high rate of success may be attributed to the fact that audio samples from human speech are found within relatively lower frequencies, allowing for many of the high-frequency adversarial perturbations to be removed while largely preserving the quality of human speech.

Speech Coding

Although the results of Lemmond and Fitzgibbons seem to suggest that audio compression is outclassed by methods such as low-pass filtering for mitigating adversarial examples, only the Advanced Audio Coding (AAC) and MP3 audio coding standards were discussed. While these compression standards are quite popular and are used in a variety of commercial situations, they are not necessarily optimal for destroying adversarial examples on speech recognition. For teleconferencing and VoIP purposes, speech codecs such as Speex (Valin 2006) and Opus (Valin, Vos, and Terriberry 2012) are more commonly used, as they are able to preserve human speech even through very lossy compression and low bitrates.

In 2002, Valin (Valin 2006) began the Speex project with the goal of providing "a free codec for free speech." Over time, Speex began to grow in popularity, being adopted for many practical VoIP applications, such as TeamSpeak[1] and Twinkle[2]. The codec is based off of the Code Excited Linear Prediction (CELP) algorithm (Schroeder and Atal 1985), which (in a simplified sense) models the vocal tract using a linear prediction model while minimizing the difference with the uncompressed source within a "perceptually weighted domain." In particular, this minimization is accomplished by applying the following weighting filter to the input:

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} \tag{1}$$

where A is a linear prediction filter with γ_1 and γ_2 controlling the shape of the filter. This filter allows for different levels of noise at various frequencies, and seems to be quite useful for destroying adversarial perturbations while preserving human speech. Speex also includes many additional features, such as voice activity detection, denoising, and support of various bandwidths. As this compression seems to exhibit many similar properties to audio preprocessing methods that have shown to be moderately successful in mitigating perturbations, it seems much more suited to the task of defending against adversarial attacks than MP3 or AAC compression.

In 2012, the Opus codec, which is currently used by the widely-used proprietary VoIP application Discord[3], was released as a successor to the Speex codec (Valin, Vos, and Terriberry 2012). It combines the CELT algorithm with SILK, a linear predictive coding algorithm developed by Skype Technologies in 2009[4]. As this is widely considered an improvement to Speex for speech coding, the performance of Opus compression against adversarial attacks is also worth testing.

Ensemble Detection

Preprocessing defenses against adversarial examples can only be effective and practical if they are able to mitigate adversarial examples without greatly compromising general model accuracy. A viable form of preprocessing would disrupt the predictions of adversarial examples more than it would disrupt the predictions of benign examples. In particular, there should ideally be a small difference between the output vectors produced by passing the raw input and preprocessed input through a neural network when the input is benign, but that same difference should be much larger if the input is adversarial. This core idea can be used to apply preprocessing methods to detect adversarial examples, rather than simply mitigating or neutralizing perturbations.

Within the field of computer vision, ensembles of preprocessing methods have been used for detecting adversarial examples (Xu, Evans, and Qi 2018). Xu et al. proposed the feature squeezing method for detecting adversarial examples. This method combines smaller "squeezing" methods into an ensemble, and calculates an L_1 score from of the maximum L_1 distance between any pair of output probability vectors produced by passing the raw and squeezed inputs through a deep neural network (DNN). Using feature squeezing, Xu et al. were able to consistently detect over 80% of adversarial examples produced from a variety of attacks.

Methods and Evaluation

The aim of this research can be divided into two parts: using Speex and Opus compression as isolated forms of preprocessing for mitigating adversarial examples, and integrating voice compression with an ensemble defense. The adversarial examples are produced using the gradient-free attack of Alzantot et al., against the same pre-trained

[1]https://teamspeak.com/en/features/overview

[2]https://www.linuxlinks.com/Twinkle/

[3]http://discordapp.com/features

[4]http://www.h-online.com/open/news/item/Skype-publishes-SILK-audio-codec-source-code-955264.html

Speech Commands model (Alzantot, Balaji, and Srivastava 2017).

Speech Commands Dataset and Model

The Speech Commands dataset was first released in 2017 and contains 105,829 labeled utterances of 32 words from 2,618 speakers (Warden 2018). The Speech Commands model is a light-weight model based on a keyword spotting convolutional neural network (CNN) (Sainath and Parada 2015) that achieves a 90% classification accuracy on this dataset. For the purposes of this research, a subset of only 30,799 labeled utterances of 10 words are used, for consistency with previous work regarding the adversarial examples of Alzantot, et al. From this subset, 20 adversarial examples are generated for each nontrivial source-target word pair, for a total of 1800 examples. Each example is generated by implementing the attack with a maximum of 500 iterations.

Voice Compression as a Preprocessing Mitigation Defense

The performance of Opus and Speex compression in destroying adversarial examples are compared with the following forms of preprocessing:

- MP3 Compression,
- AAC Compression,
- Band-pass Filtering, and
- Audio Panning and Lengthening.

While the MP3 and AAC compressions correspond directly to preprocessing defenses in related work described earlier, two of the above preprocessing defenses have not yet been directly tested against audio adversarial examples. The band-pass filter combines the low-pass filter of Lemmond and Fitzgibbons with a high-pass filter, with the purpose of eliminating more adversarial perturbations outside of the frequency range for natural human speech. Audio panning is a form of preprocessing typically used in audio mixing that distributes a signal across stereophonic channels, distorting the channel volumes to mimic the perception of audio coming from an off-centered position. The audio panning and lengthening defense lengthens the audio by 1% after panning to increase the spatial distortion of adversarial perturbations in the signal.

Ensemble Mitigation Defense

Despite the apparent success of isolated preprocessing against certain adversarial examples, it has been shown that attacks aware of the preprocessing defense can optimize examples robust to this(Carlini and Wagner 2018). As such, the use of speech coding alone for mitigating adversarial examples would render the model more insecure and vulnerable to smarter attacks. Therefore, an ensemble of preprocessing methods deployed in tandem with speech coding may be able to provide a more secure defense.

The proposed ensemble involves computing three distinct probability vectors produced by passing the following signals through the pre-trained Speech Commands model after speech coding:

- The decoded signal without additional preprocessing,
- The decoded signal passed through a band-pass filter, and
- The decoded signal panned and lengthened by 1%.

These methods of preprocessing were chosen due to their fundamental differences in how they distort the signal; this may allow for more robust mitigation of adversarial perturbations. The three resultant probability vectors are then added together, with the maximum class being returned as the prediction.

Isolated Preprocessing for Detection

A simple method for using preprocessing to detect adversarial examples is by checking to see if the prediction produced by the model changes if the input is preprocessed; if the model's prediction of the raw input does not match the prediction of the preprocessed input, it is declared adversarial.

The following preprocessing methods are used in isolation for detecting adversarial examples:

- MP3 Compression,
- AAC Compression,
- Speex Compression,
- Opus Compression,
- Band-pass Filtering, and
- Audio Panning and Lengthening.

These are the same methods of preprocessing that are tested as isolated preprocessing mitigation defenses, as described earlier.

Ensemble Detection Methods

Similarly to the motivation behind incorporating speech coding into a larger preprocessing ensemble mitigation defense, it may be more secure to add some extra complexity to the detection methods discussed earlier. The aforementioned isolated preprocessing detection methods can be combined into larger and more secure ensemble detection methods. This research explores and compares various configurations for combining the isolated preprocessing detection methods into an ensemble.

Majority Voting Ensemble: The simplest method of combining the preprocessing methods together would be by assigning each preprocessing method a vote, and declaring an audio signal as adversarial if a majority of the ensemble declares the signal adversarial. As there are six preprocessing methods that are combined into an ensemble, ties with this discrete voting scheme are possible. To err on the side of security, this procedure will declare a signal as adversarial in the event of a tie.

Learned Threshold Voting Ensemble: The majority voting ensemble declares an audio signal as adversarial if there are at least three votes in favor of it being adversarial. This threshold for deciding how many votes are needed to declare an audio signal as adversarial is arbitrary, and can adapt to different circumstances. A low threshold would result in a high recall in detecting adversarial examples, but would sacrifice precision. A high threshold would result in

a lower recall in detecting adversarial examples, but would yield a higher precision. This ensemble method experiments with using various voting thresholds for detecting adversarial examples on a labeled training set, and chooses the threshold that results in the best precision and recall. To balance both precision and recall, F_1 scores are used for selecting the best threshold, although in practice, one could adjust the F-measure to reflect one's attitude on the relative importances of precision and recall.

L_1 scoring: The previously discussed ensemble voting methods are relatively simple, as they simply examine the model's discrete prediction of the raw and preprocessed inputs for each preprocessing method. Additionally, the voting methods above are indiscriminate and treat each member of the ensemble equally. A more nuanced approach for measuring the differences in predictions between raw and preprocessed inputs is by L_1 scoring the different output logit vectors, similar to how Xu et al. integrated the multiple squeezing methods in their feature squeezing defense. In this method, an ideal threshold L_1 score is learned from training data by finding the threshold of maximum information gain, and test examples that surpass this threshold are declared adversarial. This method uses the maximum L_1 distance to calculate the score, implicitly assigning more importance to preprocessing methods that produce output vectors that are highly different than the output vectors produced by predicting raw signal. As such, this method would theoretically be more sensitive in detecting adversarial examples, but it may also be quite aggressive in declaring signals as adversarial at the risk of falsely declaring benign examples as adversarial.

Tree-based Classification Algorithms: The above ensemble methods discard information of the class-specific variation in the output vector for each preprocessing method, relative to the raw input. In order to preserve this information, a multidimensional vector can be used, with each dimension accounting for the output vector variation for that class. For the tree-based detection methods discussed in this research, a multidimensional vector composed of the summed absolute class-specific differences between the raw input's resultant probability vector and the preprocessed input's resultant probability vector over each method of preprocessing. In particular, the ith dimension of this summed absolute difference (SAD) vector S is calculated as follows:

$$S_i = \sum_{p \in P} |r_i - p_i| \tag{2}$$

where P corresponds to the set of output probability vectors yielded by the methods of preprocessing in the ensemble, and r corresponds to the output probability vector produced by passing the raw signal through the Speech Commands model without any preprocessing.

This vector will preserve information about class-specific variation between the predictions, and will reduce the number of features of the vector inputted to the tree-based classifier down to 12 (which is the same as the number of classes). Considering the relatively small training dataset size (which is discussed in Section 3.4), having less features for tree-based classification may improve performance. However, the 84dimensional vector formed by simply concatenating each output probability vector together would preserve the most amount of information. As such, the use of this concatenated probability (CP) vector for tree-based classification is also tested, even if the dataset isn't large enough for the classification algorithms to effectively handle that large of a vector.

Decision tree-based classification algorithms are well-suited for classifying vectors of features into discrete classes. In this research, three tree-based classification algorithms are employed for using vectors of summed absolute differences for detecting adversarial examples: random forest classification, adaptive boosting, and extreme gradient boosting. Random forest classification functions by constructing many decision trees in an attempt to stave off the possibility of overfitting. Adaptive boosting and extreme gradient boosting are gradient boosting algorithms which function by building an ensemble of weak learners in a stagewise fashion. Each of these tree-based algorithms are used twice in this research: once for using SAD vectors for classification and once for using CP vectors for classification. These tree-based algorithms have had quite high success in applied problems, are possibly well-suited for detecting adversarial examples.

Evaluation

All of the aforementioned preprocessing mitigation defenses are evaluated by their robustness r against the attack and their effect on the general model accuracy a_g. Within the context of this paper, the measurements are defined as such:

$$\begin{aligned} r &= c_{adv} \ / \ n_{adv} \\ a_g &= c_{ben} \ / \ n_{ben} \end{aligned} \tag{3}$$

where c_{adv} represents the number of adversarial examples correctly labeled by the classifier after preprocessing, n_{adv} represents the total number of adversarial examples, c_{ben} represents the number of benign samples correctly labeled by the classifier after preprocessing, and n_{ben} represents the total number of benign samples.

Lemmond and Fitzgibbons noted a tradeoff between general model accuracy and robustness in their comparison of various preprocessing defenses against Carlini and Wagner's attack on DeepSpeech. An inverse correlation between these two quantities would be troublesome; given the widespread use of automatic speech recognition, there is an urgent need for models to be both accurate for overall usability and secure against potentially malicious attacks. Therefore, in order to honestly compare the performances of preprocessing defenses, a metric that acknowledges both model accuracy and robustness must be used.

The F-measure (F_β) was first proposed in 1979 to balance the need for both precision and recall measurements in evaluating binary classifiers (Chinchor 1992). Taking inspiration from this method of combining two measurements to form a single metric, this research proposes the

R-measure (R_β) for responsibly evaluating preprocessing defenses against adversarial examples:

$$R_\beta = (1+\beta^2)\frac{a_g r}{\beta^2 a_g + r}. \quad (4)$$

where β denotes the relative importance assigned to robustness over general model accuracy. While security against attacks is crucial, it should not come at great expense to model usability. As such, $\beta = 1$ is used for calculating R-measures to reflect this attitude of assigning equal importance to both r and a_g.

The detection methods are evaluated by calculating their precision and recall values in detecting adversarial examples. Although it is important to have a high recall in detecting adversarial examples for the sake of security, a low precision in detection would cause the model to decline in usability. This research takes the stance of both security and general model accuracy being equally important. To reflect this attitude, F_1 scores are used to combine the precision and recall measurements with equal consideration.

Results

Mitigation Methods

The robustness, general accuracy, and R_1 scores are evaluated for each of the discussed defenses and are summarized in Table I. From the results, one can see that all of the methods of preprocessing are able to noticeably mitigate adversarial examples produced by the attack. These results are consistent with the findings of Lemmond and Fitzgibbons that MP3 and AAC compression do not perform as well as filtering against adversarial examples. However, we see that Opus and Speex voice compression perform much better than traditional audio compression. Speex compression, in particular, yields noticeably higher robustness and a better R_1 score than the other forms of individual preprocessing, including the quantization method described by Yang, et al. It is also worth noting that all of the defenses only resulted in a slight decline in general model accuracy, going against the notion of a major tradeoff between general model accuracy and robustness conjectured by Lemmond and Fitzgibbons. A decline in general model accuracy would perhaps be more noticeable if these preprocessing defenses were applied on a continuous speech recognition model.

TABLE I
PERFORMANCE OF PREPROCESSING DEFENSES

Preprocessing Defense	Robustness	General Model Accuracy	R_1 Score
No Defense	7.1%	**90.3%**	0.132
MP3 Compression	53.5%	89.4%	0.669
AAC Compression	59.6%	89.6%	0.716
Quantization-256[a]	63.8%	89.0%	0.743
Band-Pass Filtering	68.7%	89.7%	0.778
Audio Panning[b]	69.1%	89.7%	0.781
Opus Compression	65.0%	90.1%	0.755
Speex Compression	76.8%	89.8%	0.828
Ensemble Defense	**77.4%**	89.7%	**0.831**

[a]Taken from Yang et al.
[b]Includes a lengthening of 1% after the panning.

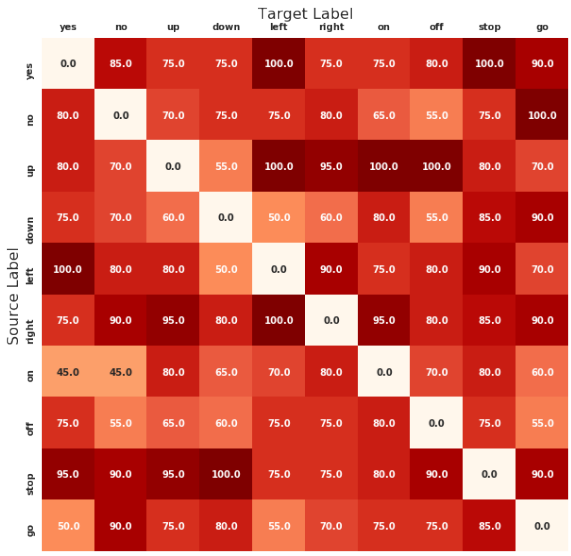

Fig. 1. A heat map depicting the robustness measurements (in percentages) of the ensemble defense to specific targeted attacks. The diagonal of zeroes correspond to trivial source-target pairs for which no adversarial examples were generated.

The ensemble defense achieved better robustness and brought down the targeted attack success rate to a mere 2.9%. The full defense ultimately incorporated just Speex as the speech codec, as islolated Speex compression generally outperformed isolated Opus compression for defending against adversarial examples and achieved a higher R_1 score; additionally, there were no specific targeted examples where Opus compression outperformed Speex compression. As Speex compression was able to preserve human voice quite well, the general model accuracy was not noticeably compromised when using the ensemble defense. The robustness of the full mitigation defense is detailed in Fig. 1.

Detection Methods

The results of the isolated preprocessing detection methods described in summarized in Table II. Measurements indicate that all of the methods are capable of detecting adversarial examples produced by the attack with varying rates of success. These results are also consistent with the findings of Lemmond and Fitzgibbons in that MP3 compression performs adequately at best when compared with the other methods. AAC and Opus compression perform notably better, but are not able to achieve as high of a recall as Speex compression (which also yields the highest F_1 score).

Although the use of bandpass filtering for detecting adversarial examples is extremely precise, it yields a remarkably low recall, which suggests it is a bit too passive with its declaration of adversariality. As many of these preprocessing methods distort audio signals in fundamentally different ways, the overall high precision (and lower recall) measurements of each of the individual preprocessing suggest that some of the ensemble methods may be more effective in detecting adversarial examples.

TABLE II
PERFORMANCE OF ISOLATED PREPROCESSING DETECTION METHODS

Preprocessing Defense	Precision	Recall	F_1 Score
MP3 Compression	93.7%	70.7%	0.806
AAC Compression	95.0%	81.2%	0.876
Band-Pass Filtering	**97.3%**	40.6%	0.573
Audio Panning[a]	95.8%	82.4%	0.886
Opus Compression	94.5%	81.8%	0.877
Speex Compression	93.7%	**88.5%**	**0.910**

[a]Includes a lengthening of 1% after the panning.

TABLE III
PERFORMANCE OF ENSEMBLE DETECTION METHODS

Ensemble Detection Method	Precision	Recall	F_1 Score
Majority Voting	**96.1%**	88.1%	0.919
Learned Threshold Voting	93.5%	91.2%	**0.924**
L_1 Scoring	76.9%	92.4%	0.840
Random Forest Classification (SAD Vector)	79.3%	87.0%	0.830
Random Forest Classification (CP Vector)	86.7%	**94.4%**	0.904
Adaptive Boosting (SAD Vector)	83.5%	81.8%	0.827
Adaptive Boosting (CP Vector)	86.7%	93.0%	0.898
Extreme Gradient Boosting (SAD Vector)	83.0%	84.2%	0.836
Extreme Gradient Boosting (CP Vector)	88.3%	**94.4%**	0.913

[a]Includes a lengthening of 1% after the panning.

The results of the ensemble detection methods are summarized in Table III. The voting methods performed quite well and achieved the two highest F_1 scores of all the methods discussed in this paper. This may be attributed to the high precisions and low recalls of the individual preprocessing methods described in Table II; the relatively strict voting threshold of votes needed for an adversarial declaration capitalizes on the high precision of each of the methods and is able to increase recall. The majority voting method especially benefited from the high precisions of its constituents and yielded an extremely high precision of 96.1%. The Learned Threshold Voting method was able to learn a lower voting threshold of only two votes needed for an adversarial declaration. As such, this method was able to yield a notably higher recall than what was achieved through majority voting, but at a noticeable cost to precision. As the Learned Threshold Voting method still retained a fairly high precision, it achieved the overall highest F_1 score of any of the other preprocessing methods.

The L_1 Scoring method was able to achieve higher recall than either of the two voting methods, perhaps due to its aggressive nature. However, this was achieved at the cost of precision, which evidently lowered the F_1 score.

Although tree-based classification algorithms can be quite powerful in a variety of situations, the tree-based methods were not able to perform as well as the voting methods in detecting adversarial examples using SAD vectors. This may be because the SAD vectors fed into the tree algorithms discarded important voter-specific information. In particular, the vector of summed absolute differences effectively anonymizes the voters in the ensemble; it inherently considers each member of the ensemble equally.

This discarded information proved to be quite crucial for effectively detecting adversarial examples, as the tree-based classification methods performed significantly better with CP vectors (which are highly conservative). In particular, the extreme gradient boosting and adaptive boosting classification algorithms were able to yield the highest recall values for detecting adversarial examples out of all of the detection methods discussed in this research. Considering that the tree-based classification methods performed significantly better with the voter-specific information available in the CP vector, it is worth noting that the Learned Threshold Voting method, which yielded a higher F_1 score than any tree-based classification method, does not use voter-specific information; each vote carries equal weight towards breaking the learned threshold. As such, it may be possible that the tree-based classification methods outperform the Learned Threshold Voting method on larger datasets, as it could be that this training dataset was not sufficiently large enough for learning how to optimally use an 84dimensional vector for classification. However, given the heavy reliance of training data that the tree-based classification methods exhibit, they are likely not as well-suited for flexibly handling different types of attacks as the voting methods.

Future Work

Although the results suggest that a ensemble defenses incorporating speech coding are effective in both mitigating and detecting adversarial examples produced by the unmodified algorithm of Alzantot et al., it does not necessarily show that this defense is secure against more complex attacks. Although an ensemble defense may provide marginal security over isolated preprocessing, recent work has shown adaptive attacks on image classifiers are able to bypass ensembles of weak defenses (He et al. 2017); this work could be applied to attack speech recognition models. Future work can be done to adapt speech coding into a stronger defense that can withstand these types of

adaptive adversarial examples, or at least cause the attacks to become more perceptible.

Additionally, this paper only discusses two speech codecs, both of which are popular among VoIP applications. Speech codecs that are used more commonly in cellular communication applications are generally better equipped to handle noise; as such, the use of those codecs for mitigating adversarial perturbations is a realm for future work.

Furthermore, this paper merely focuses on mitigating and destroying adversarial examples through preprocessing, rather than detecting adversarial examples. As such, another area for future work is incorporating the current defense with common adversarial detection techniques found in image processing.

Future Direction of This Research

Although there are many avenues for continue work, the future direction of this research will likely be towards re-implementing the attack of Alzantot, et al. to be aware of the preprocessing defense, and evaluating that new attack on the defense. While this new attack should be able to penetrate much of the defense, ideally these aware attacks would contain more perceptible perturbations. The next phase of research will focus on analyzing the perceptibility of the perturbations within examples, and see if it is true that the preprocessing-aware attacks produce "noisier" examples to penetrate the defense.

Conclusion

This paper showed that speech coding commonly used in VoIP applications is currently fairly effective in mitigating the single-word targeted adversarial attacks of Alzantot, et al. This paper also proposed a more secure ensemble defense in tandem with speech coding, and compared this defense with isolated preprocessing defenses, using a newly defined metric to balance robustness and general model accuracy. While these defenses would not be extremely secure against more adaptive attacks, this research aimed ultimately to further discussion of defenses against adversarial examples within the audio domain: a field in desperate need of more literature.

Acknowledgements

I would like to thank Terrance Boult for general advice and interesting ideas regarding the research, and the UCCS LINC and VAST labs for providing a supportive community for conducting this work. This work is supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in the material are those of the author(s) and do not necessarily represent the views of the National Science Foundation.

References

Alzantot, M.; Balaji, B.; and Srivastava, M. 2017. Did you hear that? adversarial examples against automatic speech recognition. In *31st Conference on Neural Information Processing Systems (NIPS)*.

Aydemir, A. E.; Temizel, A.; and Temizel, T. T. 2018. The effects of JPEG and JPEG2000 compression on attacks using adversarial examples. *arXiv preprint* (1803.10418).

Carlini, N., and Wagner, D. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *1st IEEE Workshop on Deep Learning and Security*.

Chinchor, N. 1992. MUC-4 evaluation metrics. In *4th Conference on Message Understanding*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Graese, A.; Rozsa, A.; and Boult, T. E. 2016. Assessing threat of adversarial examples on deep neural networks. In *15th IEEE International Conference on Machine Learning and Applications (ICMLA)*.

He, W.; Wei, J.; Chen, X.; Carlini, N.; and Song, D. 2017. Adversarial example defense: Ensembles of weak defenses are not strong. In *11th USENIX Workshop on Offensive Technologies, WOOT 2017*.

Lemmond, D., and Fitzgibbons, R. 2018. Adversarial examples in audio. CS4860 Final Project Report, University of Colorado, Colorado Springs Spring 2018.

Prakash, A.; Moran, N.; Garber, S.; DiLillo, A.; and Storer, J. 2018. Deflecting adversarial attacks with pixel deflection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sainath, T. N., and Parada, C. 2015. Convolutional neural networks for small-footprint keyword spotting. In *INTERSPEECH*.

Schroeder, M. R., and Atal, B. S. 1985. Code-excited linear prediction (CELP): High-quality speech at very low bit rates. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 937–940.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Valin, J.-M.; Vos, K.; and Terriberry, T. B. 2012. Definition of the Opus audio codec. RFC 6716.

Valin, J.-M. 2006. Speex: A free codec for free speech. In *Proceedings of linux.conf.au*.

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

Xu, W.; Evans, D.; and Qi, Y. 2018. Feature squeezing: Detecting adversarial examples in deep neural networks. In *2018 Network and Distributed System Security Symposium (NDSS18)*.

Yang, Z.; Li, B.; Chen, P.-Y.; and Song, D. 2018. Towards mitigating audio adversarial perturbations.