# PyTorch implementation of Carlini & Wagner's adversarial attack in the space of audio

**Student :** LEMESLE Yoann
*École Normale Supérieure, Rennes*
yoann.lemesle@ens-rennes.fr

**Supervisor :** AMSALEG Laurent
*LINKMEDIA team, IRISA, Rennes*
Laurent.Amsaleg@irisa.fr

## I. Introduction

The last decade has seen impressive leaps in Machine Learning research thanks to the rise of neural-network-based algorithms. A **neural network** (NN) is a weighted, directed and acyclical graph whose weights $\theta$ can be ajusted to learn any function $f(\mathbf{x})$ mapping between an input space (images, audios...) and output space (classifications, decisions...). To learn such a mapping is to find a particular set of weights $\theta^*$ that makes the network perform as expected (recognizing faces, playing a videogame...). In the typical case of **supervised learning**, $\theta^*$ is found by minimizing a **loss function** $l(f(X), Y)$ that outputs how far the network's predictions $f(X)$ are to the expected outputs $Y$ on an (X,Y) training set.

Neural-based techniques are able to achieve incredible results on tasks like speech transcription or image recognition, leading to an increasing number of real world applications such as Alexa or self driving cars. However, such applications raise security concerns as it has been discovered that neural networks are vulnerable to **adversarial examples**, inputs that look normal to humans while being strongly misclassified.

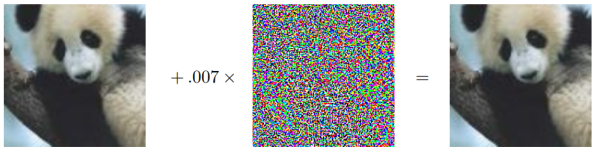



**Figure 1 :** Illustration of the FGSM adversarial attack [2].

When discovered by [1], adversarial examples were first believed to lie in small "pockets" of the input space. However, [2] showed it was possible to make images adversarial by differenciating the loss function with respect to the pixels values then taking a single (small) step towards the direction of the gradient. This attack, known as the **Fast Gradient Sign Method (FGSM)** shows how adversarial examples span across huge subspaces instead of "pockets" and supports the authors **Linearity Hypothesis**. This hypothesis explains the existence of those broad adversarial subspaces by the locally linear behavior of neural networks : when adding tiny perturbations over an entire input, its high dimensionality allows those small changes to add up and have a significant impact on the neurons activations. Since then, other concurrent hypothesis have been explored but no consensus on the exact explanation of adversarial examples exists.

The Linearity Hypothesis, as well as attack and defense methods in general, have been mostly explored in the space of images. However, understanding the adversarial phenomenon requires investigations in other domains such as audio classification. This is what motivates the goals of this internship which were to implement an important audio adversarial attack in PyTorch as well as investigating the specificities of adversarial attacks in the audio space.

## II. Adversarial Attacks on audio inputs

Let $x$ be a **benign input** (correctly classified), an **adversarial attack** is the process of computing a perturbation $\delta$ such that $x' = x + \delta$ is misclassified by a neural network.

As research on adversarial attacks were initially focused on images, recent works have been made in the audio space. Early works were focusing on synthesizing unrecognizable audio that contains hidden and inaudible voice commands [4]–[6] as other works focused on untargeted attacks by computing a perturbation over a benign audio input [7], [8]. Later, [9] were the first to perform a targeted (meaning that the adversarial audio is classified as a target sentence) and nearly imperceptible adversarial attack in the space of audio by optimizing a minimally perceptible perturbation according to the same measurement metrics as used for images ($l_p$ norms, specifically the $l_\infty$ norm that measures the maximum amount any pixel has been changed).

The attack described in [9] acts as a baseline for most of the following works which focused on fixing several issues : (1) this attack is very **slow** (several hours), (2) the perturbations is still **perceptible** and (3) the resulting adversarial audios are **not robust** to the distortions implied by being played over-the-air. [11] adressed (3) by making use of Expectation Over Transforms (EOT) [12], meaning that transformations are applied to the perturbation during the optimization process to mimic the distortions of room-environments. Later, [10] adressed (2) by making use of the psychoacoustic principles in order to add perturbations to parts of the audio that are inaudible to humans. These two techniques were combined by [13] to make an imperceptible and robust targeted attack. Complementary to EOT, [14] proposed to increase the robustness of adversarial examples by reducing the number of perturbed points on the original audios (Sampling Perturbation

Technology). They also adressed (1) by adjusting the weights of the perturbation in order to focus optimization work on "key points", parts of the original audio that have the most effect on the neural network's outputs (Weighted Perturbation Technology).

All these **optimization-based** attacks only work in a white-box setting (meaning that a full access to the neural network's informations is needed) as they require computing gradients through the model. In concrete scenarios this constraint is unrealistic as an attacker might have no knowledge about the model, making black-box attacks necessary. To perform a fully black-box and targeted attack [16] used a **Genetic Algorithm** that creates populations of mutated audios and selects the best one to craft the next generations, making the audios evolve until a good adversarial example can be found.

Despite a lot of progress, optimization and genetic based attacks still require a full observation of the entire input before generating the perturbation, making them impractical in concrete scenarios where real time attacks would be more useful. The problem was adressed by [17] using a **Generative Model**, which means using a neural network to directly generate adversarial examples by only taking the target transcription as an input.

## III. Carlini & Wagner's attack

As mentioned in the previous section, [9] were the first to demonstrate the feasibility of targeted and nearly imperceptible adversarial attack on audio inputs. This white-box, optimization-based attack is applied to Mozilla's implementation of DeepSpeech [18], a speech-to-text system that uses a Recurrent Neural Network.

### *A. Recurrent Neural Networks and Connectionist Temporal Classification*

Let $x_i$ be an **element/frame** from a sequence and $s_i$ an **internal state**, a **Recurrent Neural Network (RNN)** is a function $f(s_i, x_i)$ that outputs $(s_{i+1}, y_i)$ where $y_i$ is an **element/frame classification** and $s_{i+1}$ is the **new interal state** used for the classification of the next element. This way, an RNN is able to process an input sequence of any length by using an internal state (memory) to take into account temporal correlation between the elements.

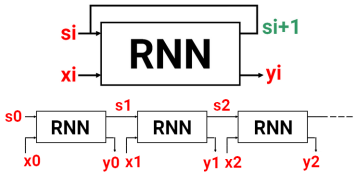



**Figure 2 :** Illustration of a Recurrent Neural Network.

When the task is to map an input sequence to an output sequence of unknown length (like transcription of speech from an audio clip), an RNN can use **Connectionist Temporal Classification (CTC)**. Let $\pi$ be a **sequence/alignment** of tokens (a-z, space and the special token $\epsilon$) and $\mathbf{p}$ a sentence. As defined in Carlini & Wagner's article [9] : *We say that a sequence $\pi$* ***reduces to*** $\mathbf{p}$ *if starting with $\pi$ and making the following two operations (in order) yields* $\mathbf{p}$ *:*

- *Remove all sequentially duplicated tokens.*
- *Remove all $\epsilon$ tokens.*

For example, the alignement HHH$\epsilon$EE$\epsilon$L$\epsilon\epsilon$L$\epsilon$OO reduces to the sentence HELLO.

When given a raw audio input, DeepSpeech will pre-process it into an **input sequence of frame** $\mathbf{x}$ given to an RNN that outputs a **sequence of probability distribution** $\mathbf{y}$. Each probability distribution $\mathbf{y}_i$ gives the probability of each token (a-z, space and $\epsilon$) for the frame $\mathbf{x}_i$. If an alignment $\pi$ has the same lenght as a sequence of probability distributions $\mathbf{y}$, **the probability of** $\pi$ **under** $\mathbf{y}$ is the product of the likelihood of each of its elements :

$$Pr(\pi|\mathbf{y}) = \prod_i \mathbf{y}^i_{\pi^i}$$

The probability of a sentence $\mathbf{p}$ under $\mathbf{y}$ is the sum of the probability of each $\pi$ that reduces to $\mathbf{p}$ :

$$Pr(\mathbf{p}|\mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{p},\mathbf{y})} Pr(\pi|\mathbf{y})$$

The loss function used to train an RNN under CTC is the negative log likelihood of the target sentence :

$$CTCLoss(f(\mathbf{x}), \mathbf{p}) = -logPr(\mathbf{p}|f(\mathbf{x}))$$

To decode a transcription sentence from a sequence of probability distribution $\mathbf{y}$ is to find the sentence $\mathbf{p}$ that has the maximum probability under $\mathbf{y}$. Because of the exponentially-increasing complexity of this task, decoding is usually done in one of two ways :

- **Greedy Decoding :** Takes the tokens corresponding to the maximum probability of each distribution to form an alignment $\pi$ that is reduced to the final sentence $\mathbf{p}$.
- **Beam Search Decoding :** Simultaneously evaluates the probability of several alignments in order to find the most probable sentence [19].

### *B. Initial Attack Formulation*

Let $\mathbf{x}$ be an audio input, the first objective of this adversarial attack is to find a perturbation $\delta$ such that $x+\delta$ is classified as any desired transcription $\mathbf{p}$ by DeepSpeech. This means finding a perturbation $\delta$ that minimizes $CTCLoss(f(\mathbf{x}+\delta), \mathbf{p})$. The second objective if for the perturbation to be as less perceptible as possible which means that we also need to minimize a measurment of the distortion's loudness. The distortion is measured in Decibels (dB) using the $l_\infty$ metrics, a norm that measures the maximum amount any value has been changed in a vector. Using this metric, the loudness of an audio is :

$$dB(\mathbf{x}) = \max_i 20 \cdot \log_{10}(\mathbf{x}_i)$$

While the loudness of the distortion *relative to the original audio* (the smaller the better) is :

$$dB_{\mathbf{x}}(\delta) = dB(\delta) - dB(\mathbf{x})$$

Let t be the **target sentence**, the initial formulation of the optimization problem could intuitively be formulated as so :

$$\underset{\delta}{\text{minimize}}\ dB_x(\delta) + c \cdot CTCLoss(x+\delta, t)$$

Where $c$ express the relative importance of being adversarial versus being imperceptible. However, the article describes that *when using a $l_\infty$ distortion metric, this optimization process will often oscillate around a solution without converging [20].* To adress this the actual initial formulation is :

$$\begin{aligned}&\underset{\delta}{\text{minimize}}\ |\delta|_2^2 + c \cdot CTCLoss(x+\delta, t)\\ &\text{such that } dB_x(\theta) \leq \tau\end{aligned}$$

Where $\tau$ is an initially large loudness $l_\infty$ threshold that is decreased everytime a solution is found before resuming minimization. Note that the $l_2$ loundess is also minimized.

### C. Improved Loss Function

The authors describe the problem of using CTC Loss as so : *In order to minimize CTC Loss, an optimizer will make* every *aspect of the transcribed phrase more similar to the target phrase. That is, if the target phrase is 'ABCD' and we are already decoding to 'ABCX', minimizing CTC loss will still cause the 'A' to be more 'A'-like, despite the fact that the only important change we require is for the 'X' to be turned into a 'D'.*

This led to the definition of a loss function that outputs the loss of a token wrt its probability distribution in such a way that the loss is zero if it has the maximum likelihood :

$$TokenLoss(y, c) = \max_{y_i \in y} y_i - y_c$$

We can now define an **alignment loss** function that won't be optimized on parts that are already correct :

$$ImprovedLoss(f(\mathbf{x}), \pi) = \sum_i TokenLoss(f(\mathbf{x})^i, \pi_i))$$

As some characters are more difficult to target than others, it would be better to place a different emphasis on each by assigning a different weight $c_i$ to each token $\pi_i$. We now have the improved formulation :

$$\begin{aligned}&\text{minimize}\ |\delta|_2^2 + \sum_i c_i \cdot TokenLoss(f(\mathbf{x})^i, \pi_i))\\ &\text{such that } dB_x(\theta) \leq \tau\end{aligned}$$

However, this forces the attack to target an alignment $\pi$ instead of a sentence which means that we first have to find such a sequence that reduces to the desired transcription. As finding the best possible sequence is too computationally expensive, the authors describe a two-step attack :

1) Find an adversarial example with the initial formulation and extract the corresponding sequence $\pi$ with the Greedy Decoder.
2) Use this sequence $\pi$ as a target for the improved formulation, using the initial adversarial example as a starting point.

Finally, it's possible to use a modified version of the improved loss function to efficiently target silence by targeting an arbitrary sequence of space characters (a sequence of space-or-blank tokens) :

$$SilenceLoss(\mathbf{x}) = \sum_i \max\left(\max_{t \notin \{"",\epsilon\}} f(x)^i_{t'} - \max_{t \in \{"",\epsilon\}} f(x)^i_t, 0\right)$$

### D. Error in the article

In the article, the token loss function is defined as so :

$$TokenLoss(y, c) = \max\left(y_c - \max_{c' \neq c} y_{c'}, 0\right)$$

The problem is that this formulation actually does the opposite of the desired behavior :

- When $y_c$ is not the most probable item, $y_c - \max y_{c'}$ is negative, therefore the loss is zero when it should be positive
- When $y_c$ is the most probable item, $y_c - \max y_{c'}$ is positive, therefore the loss is positive when it should be zero.

A note will be sent to the authors to notify them of the error.

## IV. PyTorch Implementation

The main objective of the internship was to implement the attack (initialy implementend in TensorFlow) in PyTorch.

### A. PyTorch vs TensorFlow

**TensorFlow** (Google) and **Pytorch** (Facebook) are two open-source Machine Learning frameworks optimized for neural-based algorithm implementation. The first main difference between the two is that TensorFlow only supports **static computational graphs** as PyTorch supports **dynamic graph**. This means that in TensorFlow you first have to define an entire computation graph before running it, as PyTorch allows to modify it at any time. The second difference is the **learning curve** : Pytorch is way more intuitive than TensorFlow, where tasks will often requires many more lines of difficult-to-read code. This results in a PyTorch implementation that actually do more with almost half the number of lines required in the TensorFlow implementation.

### B. Implementation details

The attack has been implementend in PyTorch 1.4.0 using the PyTorch implementation of DeepSpeech2 by [22]. In opposition with Carlini & Wagner's implementation, this one has the following characteristics :

- **Fully functional attacks on batch of audios** where the original implementation requires individual fine-tuning of adversarial examples after the attack.
- **Implementation of the silence loss function and improved attack formulation**, both of which were not implemented in the original available implementation.

However, the following properties were not implemented :

- **The $c_i$ weights :** no details could be found about how to determine their value and the improved loss function is not implemented in their code.
- **Robustness to MP3 compression.**

## V. Results

### A. Methodology

In the original article, the attack has been evaluated by the authors by targeting 10 different incorrect transcriptions on 100 different audio clips, using a learning rate of 10 and 5000 iterations. The audio clips are the first 100 test instances of the Mozilla Common Voice dataset, a multilingual and open source vocal clips dataset. Those instances can directly be found using a link provided by the authors [23].

The PyTorch implementation has been evaluated by targeting 10 different incorrect transcriptions on the first 20 test instances of the Mozilla Common Voice dataset, using the same learning rate of 10 and 5000 iterations. Is is important to note that, as the authors did not precise which target transcriptions lengths they used, it is hard to interpret the difference in the means of distortions between this implementation and the original. The reasons behind such a long execution time are explored in the next section. The results were generated on **IGRIDA**, IRISA's computing grid made of 29 computing nodes (576 cores), 9 GPUs (Nvidia) as well as a shared space of 33 TB for temporarily storing inputs/outputs. For long tasks, it is possible to schedule the launch of "jobs" that will run automatically without requiring to be connected to the platform. It also supports the loading of countless software environments and modules which is especially useful when using a framework that requires specific versions of librairies / drivers.

### B. Carlini & Wagner's Results

The main results of the original implementation are the following :

**Initial Formulation :** 100% success rate, mean perturbation of -31 dB, 95% interval ranged from -15 dB to -45 dB. A longer target phrase causes a bigger distortion with approximately +0.1 dB/characters. Conversely, a longer initial audio results in a lower distortion. Generating a single adversarial examples requires $\sim$1 hour but thanks to the parallel nature of GPUs it is possible to generate 10 examples simultaneously during that time.

**Improved Formulation :** 100% success rate, mean perturbation of -38 dB.

**Targeting Silence :** Any phrase can be turned into silence with a mean distortion inferior than -45dB. As told by the authors : *"This partially explains why it is easier to construct adversarial examples when starting with longer audio waveforms than shorter ones: because the longer phrase contains more sounds, the adversary can silence the ones that are not required and obtain a subsequence that nearly matches the target. In contrast, for a shorter phrase, the adversary must synthesize new characters that did not exist previously."*

### C. Pytorch Implementation's Results

**Initial Formulation :** The mean distortion is -42.6 dB. We observe the same trends as described in the previous section. When comparing the distortions with the lengths of the target transcriptions, we get a similar result of +0.11 dB/characters and conversely the distortion level seems to decrease when the length of the original audio increases (Figure 3).

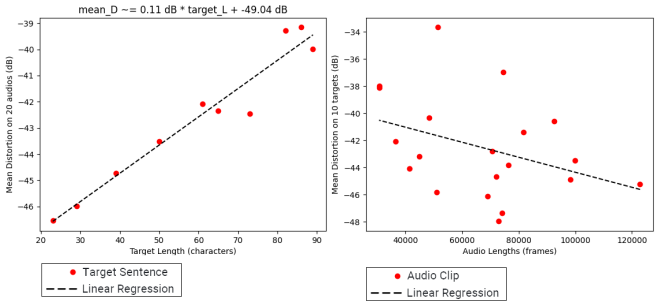



**Figure 3 :** Results for the Initial Formulation.

**Improved Formulation :** The distortion is lower as expected, with a mean of -44.4 dB. We observe the same trends with a slighty faster increasing distortion of +0.14 dB/characters (Figure 4). However, we don't have enough data to conclude that there is an actual difference with the initial formulation.

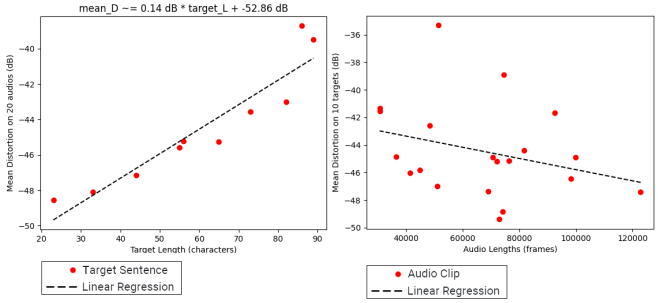



**Figure 4 :** Results for the Improved Formulation.

**Targeting Silence :** When using the silence loss function we obtain a mean perturbation of -49.9 dB, less than previous results as expected. We also observe an inversed trend : distortion increases with audio lengths as there is more sound to hide (Figure 5).

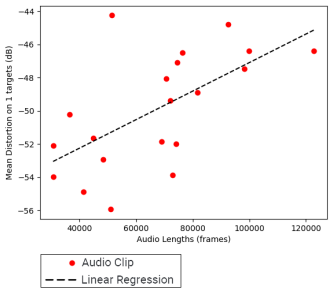



**Figure 5 :** Results for the Silence Formulation.

## VI. Specificities of Adversarial Attacks on RNNs

Adversarial attacks in the space of audio have been proven more difficult than with images. Simple optimisation-based methods can perform really fast targeted attacks on images where similar techniques usually take hours to achieve the same task on audios. It's also harder to produce imperceptible audio adversarial perturbations. Studying the reasons behind such a difference in difficulty could be useful in better understanding the adversarial problem in neural networks.

### A. Specificities of audio inputs

To understand the specificities of adversarial attacks on RNNs, we first need to isolate the difficulties introduced by working with audios.

**Distortion Metric.** It is usually desired for an adversarial distortion to be imperceptible. This is done by trying to minimize a measurement of perceptibility, which means that the choice of metrics should have a big effect on both the final result and optimisation process. While using $l_p$ norms like $l_2$ or $l_\infty$ works well for images, Carlini & Wagner's results show it is not the case with audios. Later works like [10] show how reducing the perceptibility of audio distortions requires the use of more complex metrics like the one based on the psychoacoustic principles, which shows how distortions that are high in numerical value can still be imperceptible.

**Several degrees of non-linearity.** One of the main differences between images and audios is the need for a **pre-processing** step to divide the audio into frames, which introduces a first degree of non-linearity when attacking raw audios, while directly attacking the processed audio is shown to be easier [21]. A second degree of non-linearity is introduced by the need for **decoding** (computing the probability of a sentence given an alignement of probability distributions) which forces an attacker to minimize a non-linear function like the CTC Loss of a target sentence. The fact that the improved loss function in Carlini & Wagner's article yields better results shows the difficulty of differenciating through the decoding step.

### B. Investigating specificities of RNNs

In order to be easily trainable, neural networks are specifically designed to have linear properties, allowing efficient computing of gradients. According to the Linearity Hypothesis, these linear properties are enough to explain the existence of broad adversarial subspaces in the input-output space [2]. Using the Fast Gradient Sign Method, [2] demonstrated the linear properties of Convolutional Neural Networks (CNNs, used in the space of images). By removing the difficulties introduced by the distortion constraint, pre-processing and decoding, the same could be done on RNNs.

In the image space, a CNN will output a single probability distribution for a given input, where the output's class is simply the index of maximum probability within the distribution. In the audio space, an input is divided into a sequence of frames for which an RNN will output a sequence of probability distributions. To each distribution can be associated a class (in this case : a-z, space, $\epsilon$) corresponding to the most likely token. To test the Linearity Hypothesis on RNNs, we can use the FGSM to perform adversarial attacks on pre-processed audio in order to directly modify the class of the outputed probability distributions :

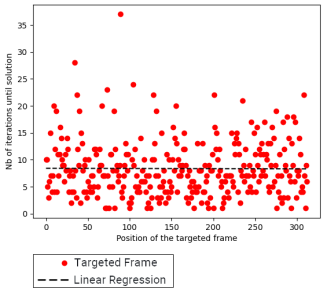
$$x' = x + \alpha * sign(\nabla_x L(\{f(x)^i\}_{i \in T}, \{y_i\}_{i \in T}))$$

Where T is a set of indexes, $f(x)^i$ the $i^{th}$ distribution (associated with the $i^{th}$ frame of x), $y_i$ the target class of the $i^{th}$ distribution and L the Cross-Entropy Loss function.

### C. Experiments

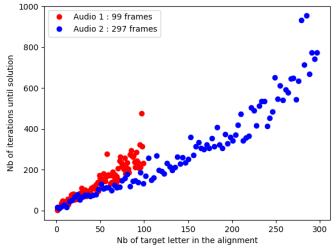
**Untargeted FGSM on alignements.** Setting $\alpha$ to a positive value results in a maximization of the loss function, allowing untargeted attacks. When performing such an attack on the whole alignment (= trying to misclassify the class of each distribution), a single-step FGSM with an $\alpha$ of 0.1 is enough to misclassify any frame with a mean confidence of 95%. This shows how the adversarial subspace is still huge with RNNs.

**Targeted FGSM on single frames.** Setting $\alpha$ to a negative value results in a minimization of the loss function, allowing targeted attacks. When performing such an attack on a **single** frame, a 10/20-step FGSM with an $\alpha$ of -0.025 is enough to target any class. Furthermore, performing this attack with a random target class on frames that are distant from the beginning of the audio doesn't seem to increase the mean number of required iterations:




DeepSpeech uses LSTMs (Long Short Time Memory), a type of RNN that is capable of conserving a relatively long-term memory. As memory is important for the classification of a frame, this result shows that long-term memory in LSTMs may not introduce any level of non linearity.

**Targeted FGSM of several frames.** Performing targeted attacks on more the one frame shows that (1) difficulty increases with the number of targeted frames and (2), for the same number of targeted frames, difficulty decreases for longer audios :



These characteristics may be caused by the effect of memory on the size of adversarial subspaces. As an RNN takes a frame as well as an internal state (memory vector) as inputs, the classification is affected by the values of previous frames. Targeting the classification of many frames constraints their values which has the effect of constraining the space of possible memory inputs for later frames. The more targeted frame there is, the more constrained is the memory space which could explain it would be harder to target the classification of later frame. However, for the same number of targeted frames, a longer audio implies more "free" frames and thus a less constrained memory space, resulting in easier attack.

## VII. Conclusion

Recent works show how adversarial attacks in the space of audio are harder than with images. This difference in difficulty can be explained by both the specificities of audio (perceptibility metrics, pre-processing, sequence decoding) and RNNs (effect of memory on the adversarial subspaces of frames). State-of-the art attacks then require the use of creative methods to achieve imperceptibility (psychoacoustic principles), fast execution time (generative models, Weighted Perturbation Technology...) and robustness (Sampling Perturbation Technology, Expectation Over Transforms with over-the-air distortions...) but remain fundamentally close to the methods used in the space of images, as the Linearity Hypothesis seems to hold with RNNs. However, more investigations need to be done on the specificities of adversarial subspaces with RNNs and how it should affect attack and defense strategies as well as our comprehension of those intriguing properties of neural networks.

## References


[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014

[3] Serban, Alexandru & Poll, Erik. (2018). Adversarial Examples - A Complete Characterisation of the Phenomenon.

[4] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou. Hidden voice commands. In 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, 2016.

[5] L. Song and P. Mittal. Inaudible voice commands. arXiv preprint arXiv:1708.07238, 2017.

[6] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu. Dolphinatack: Inaudible voice commands. CCS, 2017.

[7] Y. Gong and C. Poellabauer. Crafting adversarial examples for speech paralinguistics applications. arXiv preprint arXiv:1711.03280, 2017

[8] C. Kereliuk, B. L. Sturm, and J. Larsen. Deep learning and music adversaries. IEEE Transactions on Multimedia, 17(11):2059–2071, 2015

[9] Carlini, N. and Wagner, D. A. Audio adversarial examples: Targeted attacks on speech-to-text. 2018 IEEE Security and Privacy Workshops (SPW), pp. 1–7, 2018.

[10] L. Schonherr, K. Kohls, S. Zeiler, T. Holz, and D. Kolossa, "Ad- versarial attacks against automatic speech recognition systems via psychoacoustic hiding," in 26th Annual Network and Distributed System Security Symposium (NDSS). The Internet Society, 2019.

[11] Yakura, H., and Sakuma, J. 2018. Robust audio adversarial example for a physical attack. arXiv preprint arXiv:1810.11793.

[12] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. arXiv preprint arXiv:1707.07397, 2017

[13] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in 36th International Conference on Machine Learning (ICML). PMLR, 2019, pp. 5231–5240.

[14] Liu, Xiao-lei et al. "Weighted-Sampling Audio Adversarial Example Attack." AAAI (2020).

[15] Liu, Yanpei et al. "Delving into Transferable Adversarial Examples and Black-box Attacks." ArXiv abs/1611.02770 (2017): n. pag.

[16] Taori, Rohan et al. "Targeted Adversarial Examples for Black Box Audio Systems." 2019 IEEE Security and Privacy Workshops (SPW) (2019): 15-20.

[17] Xie, Yi et al. "Enabling Fast and Universal Audio Adversarial Attack Using Generative Model." ArXiv abs/2004.12261 (2020): n. pag.

[18] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al. Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567, 2014.

[19] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning, pages 369–376. ACM, 2006

[20] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In Security and Privacy (SP), 2017 IEEE Symposium on, pages 39–57. IEEE, 2017.

[21] Hu, Shengshan & Shang, Xingcan & Qin, Zhan & Li, Minghui & Wang, Qian & Wang, Cong. (2019). Adversarial Examples for Automatic Speech Recognition: Attacks and Countermeasures. IEEE Communications Magazine. PP. 1-7. 10.1109/MCOM.2019.1900006.

[22] https://github.com/SeanNaren/deepspeech.pytorch

[23] https://nicholas.carlini.com/code/audio_adversarial_examples/commonvoice_subset.tgz