# Understanding Adversarial Attacks and Defenses on Neural Networks

Devvrit Prabal Vashisht Surya S. Dwivedi

April 18, 2020

## 1 ABSTRACT

Recent years has seen tremendous amount of work to understand various aspects about adversarial attacks on neural networks. A lot of to-fro work has been done on generating better defenses and attacks to fool the defenses. On the another hand, a drop in generalization has also been noticed as a result of adversarial training. One such defense mechanism in a recent work is by [Cheng et al., 2020] where the authors solve a "better" choice of min-max optimization problem to get higher robust accuracy. We reason out theoretically why the robustness is increasing. We also improve both robustness and generalization by choosing only a subset of points for adversarial training while keeping the rest of the data clean to improve generalization. We support our work with theoretical intuitions and propose future work plan for the rest of the semester.

(ACKNOWLEDGEMENT: WE WOULD LIKE TO THANK AMIT DESHPANDE (MSR INDIA), MINHAO CHENG (DEPT. OF CS, UCLA), AND NIVED RAJARAMAN (DEPT. OF EECS, UC BERKELEY) FOR VALIDATING OUR ARGUMENTS AND ACTIVE DISCUSSIONS)

## 2 INTRODUCTION

Deep Learning is currently being used extensively in solving problems that were once unattainable. Further, it has become the choice for solving challenging tasks in speech recognition and creating autonomous vehicles. [Szegedy et al., 2014] first discovered that Neural Networks, despite their success, are vulnerable to misclassifying well-designed input called adversarial examples. These adversarial examples are imperceptible to human eyes but can fool a Neural Network with high confidence. Since these models are deployed in safety-critical environment, it becomes important to evaluate various attacks on Neural Networks and investigate further to develop robust defence mechanisms. Adversarial training ([Goodfellow et al., 2014], [Madry et al., 2017]) has become one of the key techniques to develop defenses against adversarial attacks. While the vanilla ERM aims to minimize expected error on the training set, the adversarial training aims to minimize expected error of the worst case loss within

an $\epsilon$ ball of each training point. This $\epsilon$ is a pre-defined perturbation tolerance. For a more thorough review of the literature, we refer the reader to [Akhtar and Mian, 2018], [Chakraborty et al., 2018]. In this report, we focus our interest on a recent defense algorithm proposed by [Cheng et al., 2020] where they solve a variance of the usual min-max optimization problem ([Madry et al., 2017]). The paper highlights the importance of intrinsic robustness particular to each data point. Intuitively, the points close to the decision boundary shouldn't be perturbed as much as the points lying far away form the decision boundary. They also question whether the model should be forced to learn one-hot encoding as per the usual adversarial training method. Intuitively, consider a binary classification. If a point lies on the (optimal) decision boundary, then there's an equal probability of it being labeled a 0 or 1. And hence, a one-warm encoding of $[0.5, 0.5]$ seems more apt than $[1, 0]$. The authors introduce a Custom Adversarial Training (CAT) algorithm, and report performance on Cifar-10 dataset against C&W attack ([Carlini and Wagner, 2017]). We refer the reader to [Cheng et al., 2020] for a more thorough understanding of the work.

Throughout the literature, we notice that almost all the defense methods perturb the entire dataset for adversarial training. While this may help in improving robustness, we believe that a (large-enough) subsample of the perturbed images should perform equally well for adversarial training and making the model robust. Moreover, adversarial training has been shown to affect generalization ([Tsipras et al., 2018], [Raghunathan et al., 2020]). Therefore, training with a mix of a subset of adversarially perturbed points and a lot of clean data should help better generalizations while maintaining robustness.

In this report, we present results of various experiments to understand the working of the CAT algorithm ([Cheng et al., 2020]) and present a novel method to improve both generalisation and robustness. We also mention some of the downsides of the CAT algorithm. We introduce a novel approach to show that using only the points which lie close to the decision boundary to be adversarially perturbed, and keeping the points which are well within the decision boundaries (far from the closest decision boundary) as clean, helps better generalization and also improves robustness. The CAT algorithm, using VGG neural-net, gets 84% clean-test accuracy and 55.8% accuracy on C&W attack ([Carlini and Wagner, 2017]). While our method, using the same hyper-parameters, get 89% clean-test accuracy and 65% accuracy on C&W attack. Using Wide ResNet, the CAT algorithm achieves 91.8% clean-test accuracy and 74.6% accuracy on C&W attack. While our method, using the same hyper-parameters, get 93.4% clean-test accuracy and 80.3% accuracy on the C&W attack. Our method also doesn't introduce any overhead in the running time and takes same time as the CAT algorithm.

## 3 Motivation

We study the CAT algorithm and visualise how it is pushing the decision boundary to become as close as possible to the optimal decision boundary (we say a boundary as optimal which achieves the optimal clean and robust accuracy). In order to do so, for every $20^{th}$ epoch, we plot how $\epsilon_i$ (this is defined in [Cheng et al., 2020] as the custom perturbation tolerance $\epsilon$ for each point) changes. The results are in Figure 3.1, where $x-$axis represents $\epsilon_i$ and $y-$axis represents the number of points with this $\epsilon_i$. While $\epsilon_i$ is a continuous variable, the CAT algorithms approximates it by putting it in certain buckets of multiples of $\eta$ (a hyperparameter which we set as $\eta = 0.0025$ for our experiments)

We notice a change in the graph at epochs = $80, 140, 180$. This is due to the reduction in learning rate of SGD in the CAT algorithm. More importantly, we notice that in the start most of the points are close to the decision boundary. That is, the distance to the nearest decision boundary is small for most of

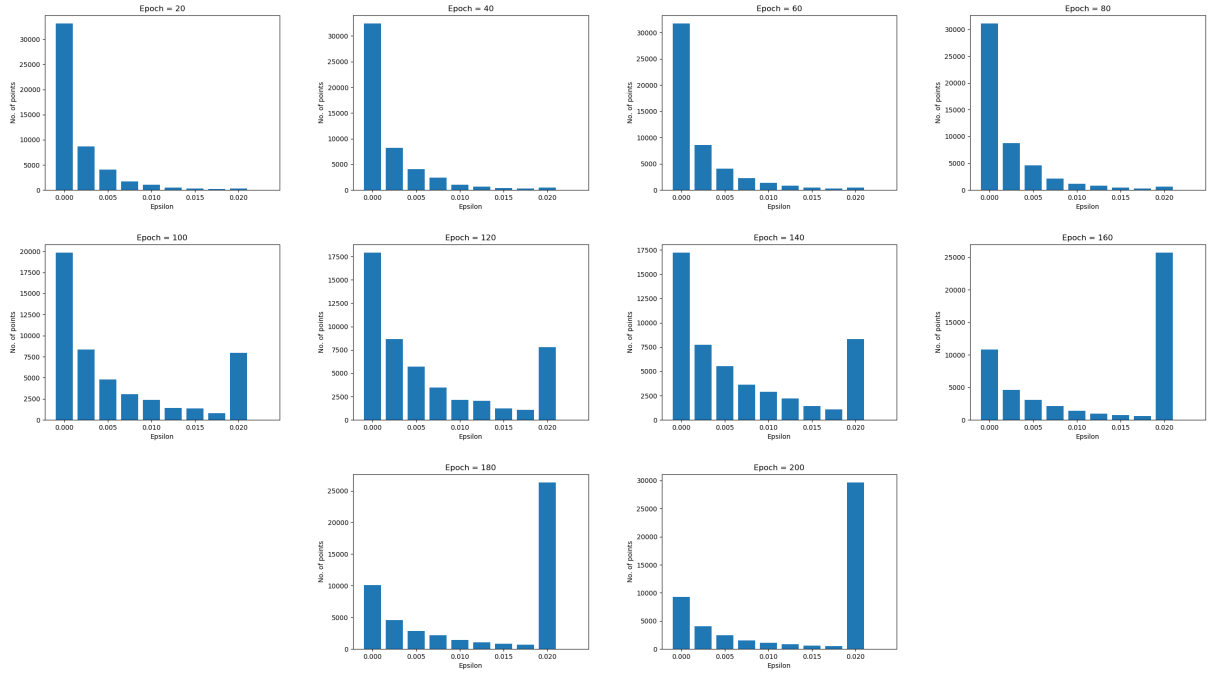


Figure 3.1: Epsilons evolving throughout the training.

the points. While by the end of the algorithm, most of the points are quite far away from the decision boundary. Though the above figure represents the trend for all the points, we notice a very similar trend for each class. That is, we plotted the same graph for each class and noted the very same trend. If a point is quite within the decision boundary for a model, then even if we adversarially perturb the data point, most likely it'll be labelled correctly. Hence, using them for adversarial training might not be very helpful. In fact, perhaps using them clean will help in generalization. One can reason out this from the fact that training on just clean data points help better generalization. So keeping a lot of clean points even when using a few others as adversarially perturbed, should help in generalization. We support our this argument more formally later when analysing it theoretically. Also, the CAT algorithm treats the perturbed images of points originally well within the decision boundary as noise (the label considered for the perturbed image is close to a random labelling, if the point was actually quite far from the decision boundary), and so near to decision boundary points are the main candidates in deciding the decision boundary. The inclusion of noise in itself can affect generalization. We notice that as most of the points get far from the decision boundary, a lot of points become somewhat like a noise by the end of the CAT algorithm iterations. Hence, treating them as clean helps better the generalization. Considering only nearby points for adversarial training also helps in robustness, which we explain later in Section 6.3

## 4 Algorithm

In each epoch of the CAT algorithm, we calculate $\epsilon_i$ for each point. Consider a given epoch. Let $\epsilon_m = max\, \epsilon_i$ in this particular epoch. Then, in this epoch, we adversarially perturb all the points which have $\epsilon_i < \epsilon_m$. And for the rest of the points, we let them stay clean. Now using this set of perturbed points (and corresponding one-warm encoding), and the clean points (with one-hot encoding), we run the

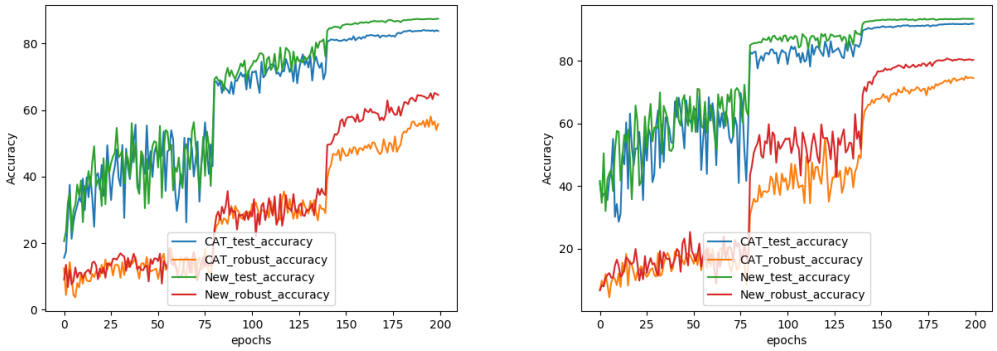


Figure 5.1: Comparison of accuracy vs epochs between CAT and our method, using VGG (on left) and Wide ResNet (on right)

usual CAT iteration to train the model. We make no other change in the algorithm.

# 5 Results

We compare the vanilla CAT algorithm with the hyperparameters as follows. We set the number of iterations in adversarial attack to be 10 for all methods. We use the CAT-MIX variation of the CAT algorithm, for best robust results. $\epsilon_{max} = 0.03, \eta = 0.0025, c = 10, \kappa = 50,$ momentum 0.9, weight decay $5 * 10^{-4}$. There's also a decay of the learning rate to 90% at the $80^{th}, 140^{th},$ and $180^{th}$. The plot for final accuracy of the vanilla CAT algorithm compared to our technique is in Figure 5.1
Table 5.1 gives an empirical comparison:

Table 5.1: Empirical comparison between CAT and our method

| *Methods* | *Clean Accuracy* | *C&W Accuracy* |
|---|---|---|
| Natural training | **95.93%** | 0% |
| CAT-MIX, VGG | 83.8% | 55.82% |
| Subsampling+ CAT-MIX, VGG **(Our)** | 87.45% | 65% |
| CAT-MIX, Wide ResNet | 91.81% | 74.6% |
| Subsampling+ CAT-MIX, Wide ResNet **(Our)** | **93.41%** | **80.3%** |

# 6 Theoretical Intuition

We divide this section into several parts, covering individual small questions in each of them.

## 6.1 Understanding the optimization problem

While the authors of CAT [Cheng et al., 2020] give intuition over the increase in generalization, we didn't find enough intuition to understand the increase in robustness. Hence, we address this issue here, reasoning out why CAT algorithm given an increase in robustness.

It's a well known fact that given a function to be optimized over the original distribution, the best training approach is to do ERM, optimising the same function over training dataset. Consider how the attacks are framed (as per [Carlini and Wagner, 2017]):

$$\begin{aligned} &\text{minimize} && D(x, x+\delta) \\ &\text{such that} && C(x+\delta) \neq y \\ & && x+\delta \in [0,1]^n \end{aligned} \tag{6.1}$$

The function to be optimized upon is the distance $D(x, x+\delta)$. It could be $l_\infty, l_2, l_1$, Wasserstein, etc. The first condition says that the output class is not the actual real class. The last constraint says that the perturbed image should be a valid image. Notice that we assumed here that the model is fixed, hence there's no $\theta$ term like $C(\theta, x+\delta)$. Now note that for a fixed $\theta$, the function $D(x, x+\delta)$ minima will be obtained when the perturbed image $x+\delta$ is the closest point of $x$ to the decision boundary in the appropriate distance metric, whichever we are considering. Hence, it makes sense to train the model also using $\epsilon_i$ as suggested in previous works ([Balaji et al., 2019], [Cheng et al., 2020]), and not having a fixed $\epsilon$ which most of the papers had been trying to, originally suggested by [Madry et al., 2017] among others. Now, consider the following scenario. The attacker gives a set of $\epsilon_i$, meaning that he/she is going to perturb $x_i$ point with distance at most $\epsilon_i$. Then the problem we need to optimize is the following. Assume $\epsilon_x$ for point $x$ same as $\epsilon_i$ for point $x_i$.

$$\min_{\theta} \mathbb{E}[\max_{x' \in \mathscr{B}(x,\epsilon_x)} l(f_\theta(x'), y)] \tag{6.2}$$

Which if we write the ERM of, is:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{x'_i \in \mathscr{B}(x_i,\epsilon_i)} l(f_\theta(x'_i), y_i) \tag{6.3}$$

Also notice that given a $\theta$, the attacker gets $\epsilon_i$ by calculating the distance of point $x_i$ to the nearest decision boundary. Hence, getting the set of $\epsilon_i$ can be formulated as:

$$\epsilon_i = \underset{\epsilon}{argmin}\ \exists x'_i \in \mathscr{B}(x_i, \epsilon) \text{ satisfying } f_\theta(x'_i) \neq y_i \tag{6.4}$$

The CAT paper solves exactly these two set of equations iteratively (refer to Section 3.3 of [Cheng et al., 2020]).

Note that if the attacker was perturbing every point by the same epsilon, the best method to train would had been the vanilla min-max problem as mentioned by [Madry et al., 2017]. Because that is the ERM that one needs to optimize over to get the best robustness in such kind of attack using same uniform $\epsilon$. But because attack doesn't happen with fixed $\epsilon_i$ for all points, hence doing so will be an overkill. Doing so would be solving an optimisation problem of some other target function rather than the actual one. But because the CAT algorithm tries to solve the correct ERM, hence it gets much better robustness.

## 6.2 Downsides and limitations of the CAT algorithm

**Limitation-1**: Consider a given iteration of CAT algorithm where given $\epsilon_i$, we need to compute the min step. That is, we need to solve eq.6.3. Notice that in order to solve eq.6.3, we don't need any other information except $\epsilon_i(s)$. But the CAT algorithm finds an approximation to $x'_i \in \mathscr{B}(x_i, \epsilon_i)$ by using the previous iteration's $\theta'$ and perturbing $x_i$ to the nearest decision boundary of this $\theta'$. Basically, it doesn't solve the problem optimally, as it's probably a hard problem, and tries to approximate it by depending on the previous iteration's $\theta'$ and thereby not getting to the actual optimal solution. Specifically, consider a very simple scenario as follows. We're going to consider the setting in $l_2$ distance and $l_2$ perturbations. A similar example could easily be generated for $l_\infty$ perturbations also. Given two data points in $2-D$, say $x_1 = (3,3), y_1 = 1$ and $x_2 = (-3,-3), y_2 = -1$. The optimal decision boundary is the perpendicular bisector of $x_1$ and $x_2$. But let's assume that we somehow land up with current decision boundary being the $y-$axis. We'll analyze the next iteration of the CAT algorithm compared to the optimally solving the next iteration. We get $\epsilon_1 = \epsilon_2 = 3$, which is the distance to the decision boundary. So given $\epsilon_1 = \epsilon_2 = 3$, the optimal way of perturbing the points to get the *worst best margin*, that is, the optimal $x'_i$ solving

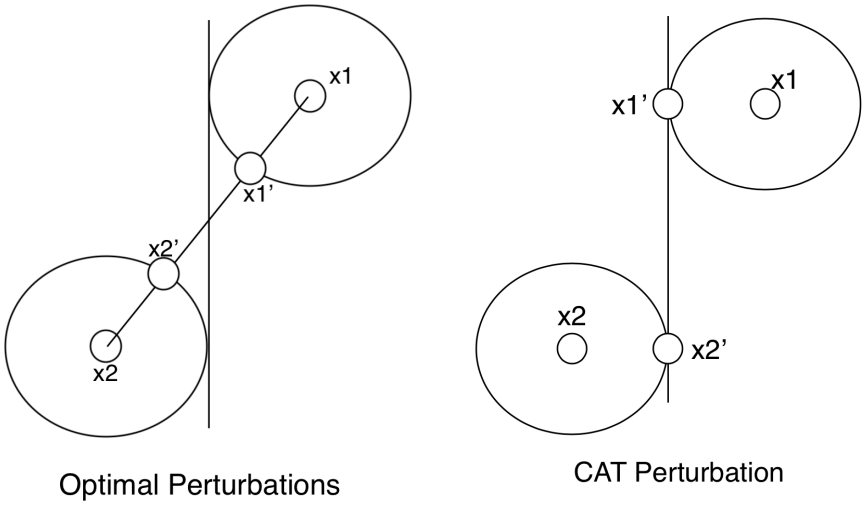


Figure 6.1: Solving eq.6.3. Optimal perturbation (on left), what CAT does (on right)

eq.6.3 would be to perturb $x_1$ and $x_2$ towards each other along the $y = x$ line, where the circle of radius 3 cuts this line with $x_1$ and $x_2$ respectively being the center. Refer to the figure 6.1. This is because then the resulting decision boundary, perpendicular bisector of $x_1', x_2'$, would have least distance between these perturbed points. Note that we get the actual optimal decision boundary too in this case, as perpendicular bisector of $x_1, x_2$ and $x_1', x_2'$ is same for this case of perturbing optimally. But the CAT algorithm will perturb images to $x_1' = (0,3)$ and $x_2' = (0,-3)$. And the decision boundary calculated in this step would be the $x-$axis, having distance of 6 units between the points. Which is much more than the distance attained between $x_1', x_2'$ by solving eq.6.3 optimally. Moreover, in the next iteration, the CAT algorithm will perturb $x_1$ and $x_2$ to $x_1' = (3,0)$ and $x_2' = (-3,0)$, and so we will again get the $y - axis$ as the next $\theta$. And the decision boundary will keep oscillating between $x - axis$ and $y - axis$. One might think that if we rather perturb the points by $0.9\epsilon$ towards the decision boundary, the algorithm may converge. But it doesn't, we ran a small simulation and with any $\alpha\epsilon$ with $\alpha > 0$, the decision boundary keeps oscillating never converging to any boundary.
Thus, we see that the approximations used by the CAT algorithm could make the ERM (eq. 6.3) perform quite bad and not be a good approximation to the actual expectation (eq. 6.2).

**Limitation-2**: Another downside of the CAT algorithm is that it treats the perturbed points, that were originally far away, as noise during training (refer to Section 3.4 eq. 5 of [Cheng et al., 2020]). And as the epochs increase, number of points far away also increases, thereby a lot of points possibly being considered as noise. A lot of noise isn't helpful in training and can affect generalization, whether robust generalization or clean data generalization. Hence, this could be another limitation of the CAT algorithm. An alternative option could have been to weigh the points according to how far are they from the decision boundary. For example, consider an ERM as follows:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{x_i' \in \mathscr{B}(x_i, \epsilon_i)} w_i \, l(f_\theta(x_i'), y_i) \tag{6.5}$$

Where now we introduce $w_i$ as the weight for the point $x_i$, inversely proportional on its $\epsilon_i$. We haven't yet played with this kind of setting and plan to if possible within time constraints

**Limitation-3**: The ERM that CAT optimizes upon tries to get an optimal solution to the generalization eq. 6.2. This generalization comes from the way we have defined attack (eq. 6.1). If we use some other way of attack, there might be a chance that CAT fails. For example, one such alternative attack is by using Wasserstein distance [Wong et al., 2019]. Hence, one needs to test the algorithm on other attacks as well.

### 6.3 Our method gives better robustness

When we have an optimization problem as in eq.6.2, we solve it using ERM, which is a different optimization problem but we hope that the ERM would be a good approximation to the generalization. There exist many upper bound on the difference between the average loss by ERM and the generalization, like bounded by terms including square root of VC Dimension, Rademachaer complexity etc., and inversely proportional to the number of points to train on (the $n$ term in eq. 6.3). When usually we optimize over the same function in generalization (the loss function in the expectation as in eq. 6.2) and the ERM (eq. 6.3), the upper bound gets mainly controlled by the number of points we have to train on. But in adversarial training, that's not the case. Even the function to optimize upon is approximated, and hence now we need to take care of two factors - the number of points, as well as how good the approximation is of $l(f_\theta(x'), y)$.

Now consider the example in Section 6.2. Let the points were at, say, $x_1 = (10^9, 10^9)$ and $x_1 = (-10^9, -10^9)$. Basically, very far from each other. Then, the $\epsilon_1 = \epsilon_2 = 10^9$. And in the ideal case they would have been perturbed to $x_1', x_2'$ where distance between $x_1', x_2'$ would have been almost negligible (both would be almost at the origin). And this would have been the best perturbation possible. But the points, due to the approximation of the CAT algorithm, lie at a distance of $2 * 10^9$ units apart from each other, which is way more than the optimal negligible distance. Therefore, as this example shows, the approximation is much more bad for points which lie far away from the decision boundary, compared to how bad the approximation is for points lying close to the decision boundary.

One may argue that the far away points are given random labellings, but then including noise also isn't helpful. On the other hand, if the points is away from the decision boundary but not far away enough, so that it gets almost a one-hot encoding, then the above observation comes into play that because it's farther than rest of the points, it's approximation is also comparatively worse. Overall, we are left out with a choice to make, to have a good balance between $n$, the number of points perturbed and used for adversarial training, and the approximation to the loss function $l(f_\theta(x'), y)$. If we take far away points in the ERM, then $n$ increases but the approximation is bad. Therefore, one has to try different choices and see where the tradeoff works the best. In the case of CIFAR-10, and the CAT algorithm, not considering points which are farthest away from the decision boundary seems to be a better tradeoff than considering all the points together for adversarial training. And hence we get better robustness.

### 6.4 Better Generalization

Natural training, which refers to the vanilla training solely on the clean training set, gives best generalization. This motivated us to rather keep as many points clean as possible while not affecting the robustness. The points which lie close to the boundary are most useful for deciding the decision boundary, and we also notice that with increasing epochs, a lot of points are far away from the decision boundary. Hence, keeping these far-away points clean does not decrease robustness (in fact increases it as discussed in Section 6.3) and also increases generalization. Consider the following definition, taken

from Definition 3.1 of [Cheng et al., 2020]. Let $h_\theta(x) : \mathbb{R}^n \to [0,1]^K$ as the prediction probability for the $K$ classes. Then, define the bilateral margin as:

**Definition 1.** ***(Bilateral margin)*** *Define the bilateral perturbed network output by* $H_\theta(x, \delta^i, \delta^o)$

$$H_\theta(x, \delta^i, \delta^o) = h_\theta(x + \delta^i ||x||) + \delta^o ||x + \delta^i ||x||||$$

*The bilateral margin is now defined as the minimum norm of* $(\delta^i, \delta^o)$ *required to cause the classifier to make false predictions:*

$$m_F(x, y) = \min_{\delta^i, \delta^o} \sqrt{||\delta^i||^2 + ||\delta^o||^2}$$
$$\text{subject to} \quad \max_{y'} H_\theta(x, \delta^i, \delta^o)_{y'} \neq y$$

We can see that there's an improvement in generalization by the following theorem, taken from Theorem 2.1 of [Wei and Ma, 2019]:

**Theorem 1.** *Suppose the parameter space* $\Theta$ *we optimize over has covering number that scales as* $log(\mathcal{N}_{||.||_{op}})(\eta, \Theta) \leq \lfloor C^2/\eta^2 \rfloor$ *for some complexity* $C$*. Then with probability* $1 - \delta$ *over the draw of the training data, any classifier* $f_\theta, \theta \in \Theta$ *which achieves training error* $0$ *satisfies:*

$$\mathbb{E}[\mathbb{1}(f_\theta(x) \neq y)] \lesssim \frac{C log^2 n}{\sqrt{n}} \sqrt{\frac{1}{n} \sum_{i=1}^{n} \frac{1}{m_f(x_i, y_i)}} + \zeta$$

*where* $\zeta$ *is of small order* $O(\frac{1}{n} log(1/\delta))$

Note that the upper bound is dominated by points which have small margin. Hence, the authors of [Cheng et al., 2020] argue that they keep the points which are nearby as one-hot so that they achieve high-enough margin and the upper bound is low. But notice that the CAT algorithm perturbs all the points to close to the decision boundary. If we rather keep the points which are far from the decision boundary unperturbed, then nearby points will have even more control over deciding the decision boundary and pushing them farther away. Hence, the upper bound gets even better. One can see this happening by comparing the final bar-graph of number of points vs $\epsilon$ using the CAT algorithm, and our approach, shown in figure 6.2.
The number of points which have $\epsilon = 0$ is more than 10000 for the CAT algorithm, while it's less than 10000 using our method. Similarly, number of points having $\epsilon = 0.02$ is higher with our approach compared to the CAT algorithm. Hence, we get a comparatively better upper bound and better generalization too.

## 7 Future Action Plan

1. To try to solve a weighted version of the ERM as in eq. 6.5
2. The ERM that CAT optimizes upon tries to get an optimal solution to the generalization eq. 6.2. This generalization comes from the way we have defined attack (eq. 6.1). If we use some other way of attack, there might be a chance that CAT fails. For example, one such alternative attack is by using Wasserstein distance [Wong et al., 2019]. We would like to test CAT on this attack as well.

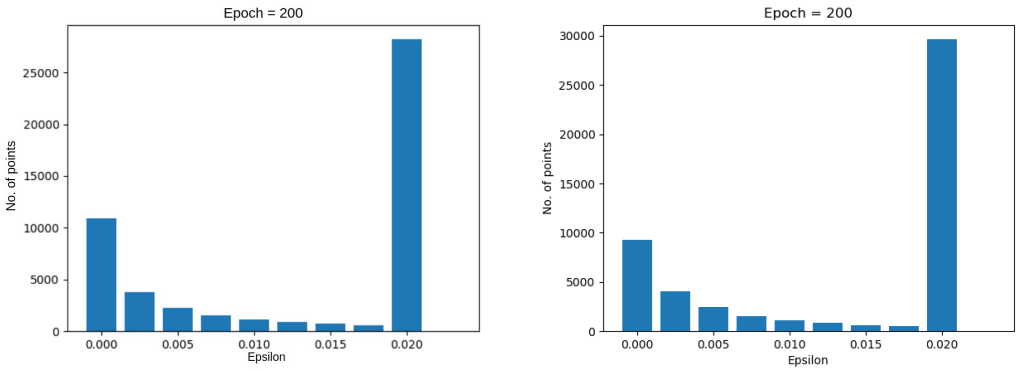


Figure 6.2: Comparison of number of points with $\epsilon = 0$ and $\epsilon = 0.02$ (max) between (left) CAT and (right) our method

## REFERENCES

[Akhtar and Mian, 2018] Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey.

[Balaji et al., 2019] Balaji, Y., Goldstein, T., and Hoffman, J. (2019). Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets.

[Carlini and Wagner, 2017] Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*.

[Chakraborty et al., 2018] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. (2018). Adversarial attacks and defences: A survey.

[Cheng et al., 2020] Cheng, M., Lei, Q., Chen, P.-Y., Dhillon, I., and Hsieh, C.-J. (2020). Cat: Customized adversarial training for improved robustness.

[Goodfellow et al., 2014] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples.

[Madry et al., 2017] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks.

[Raghunathan et al., 2020] Raghunathan, A., Xie, S. M., Yang, F., Duchi, J., and Liang, P. (2020). Understanding and mitigating the tradeoff between robustness and accuracy.

[Szegedy et al., 2014] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations*.

[Tsipras et al., 2018] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2018). Robustness may be at odds with accuracy.

[Wei and Ma, 2019] Wei, C. and Ma, T. (2019). Improved sample complexities for deep networks and robust classification via an all-layer margin.

[Wong et al., 2019] Wong, E., Schmidt, F. R., and Kolter, J. Z. (2019). Wasserstein adversarial examples via projected sinkhorn iterations.