An Excursion Into Adversarial Robustness

Anurag Pallaprolu
9847112
UC Santa Barbara
Santa Barbara, CA, 93106
apallaprolu@ucsb.edu

Vamshi Chowdary Madala
9921230
UC Santa Barbara
Santa Barbara, CA, 93106
vamshichowdary@ucsb.edu

Abstract

Given enough information about the input/output characteristics of a statistical black-box system, one can construct optimal estimators that make predictions regarding evolving future states. Unfortunately, most of the modern approaches to achieve this goal emphasize on the accuracy of the estimator without establishing the stability of the optimal result. The current exposition aims at surveying the landscape from the oculus of procedural robustness by presenting some state-of-the-art techniques that demonstrate both enhancements and attack patterns tailored to classifier and neural net architectures.

1 Prelude: Are We There Yet?

Supervised Learning (SL) has established itself as the simplest and most logical approach towards solving the problem of state estimation given the observations sampled from a physical system [1]. From straightforward tasks such as curve fitting to moderately complex tasks such as production resource allocation, it has emerged as the dominant strategy for delivering accurate and often real-time predictions for a system under observation. It is also one of the oldest [2, 3] and well studied modes of machine learning thus far.

Given the input space $\mathcal{X}$, the output space $\mathcal{Y}$ and the prediction space $\hat{\mathcal{Y}} = SL(\mathcal{X})$, we consider an appropriate "loss"/"risk" function $\mathcal{L} : \hat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}_+$ that gives us a real number measure of how close a certain prediction $SL(x)$ comes to the corresponding output y. In order to model SL, we also assume that it fundamentally depends upon some parameter θ which lives in the parameter space Θ, and thus $SL : \Theta \times \mathcal{X} \to \mathcal{Y}$. The central dogma of supervised learning revolves around estimating the parameter θ that minimizes the expectation of $\mathcal{L}$ over $\mathcal{X} \times \mathcal{Y}$, that is:

$$\theta^* = arg\,min_{\theta \in \Theta}\, \mathbb{E}_{\mathcal{X},\mathcal{Y}}[\mathcal{L}(SL(x), y)] = arg\,min_{\theta \in \Theta}\, \frac{1}{N} \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \mathcal{L}(SL(x), y)$$

The second equality crucially depends upon the joint probability distribution of the random vector (x, y) to have convergent moments thereby allowing us to use the Law of Large Numbers for equating it to the empirical average. While this is overlooked as a formality in most discussions, it is also well known that such "nice" joint densities are rare in real life scenarios. Minimizing the expected risk tells us something about the output stability only as far as the average case is concerned by the very nature of the expectation operator.

Unsupervised Learning (UL) emerged as a popular alternative to the rather simplified settings of SL techniques by allowing the learning algorithm to select the boundaries of estimation. Consequently, the predictors $UL(\mathcal{X})$ in general take non-analytic representations but continue to assume the existence of a stable multivariate prior that describes the behavior of the system. This fundamental assumption (which also extends into the realm of Reinforcement Learning (RL)) can only be validated based on the input/output samples that have been observed thus far. However these samples

do not give anything concrete about the compliance of the said assumption in the future unless a representative distribution of the same is available. If one were to argue strictly with regards to robustness, by their very definition these future samples develop a significant propensity to turn into outliers: a problem that has been dubbed as "non-observability" [4, 5] but has been grappled before by philosophers as the fallacy of induction [6, 7].

While the situation may appear bleak, we seem to have a solution if we can instead work our way backwards by understanding which data points turn into outliers for the model under consideration. These samples (which originate from $\mathcal{X}$) now obtain a special status and are henceforth called as "adversaries". Therefore it becomes absolutely essential to characterize them to derive any conclusions regarding the robustness of the estimation procedure. [1]

Before we proceed towards an exploration of the input space for finding adversaries, we will setup some more notation which is followed in the rest of the paper. As described earlier, we operate on a "hypothesis class" $H_\theta(\bar{x})$ (general name for the predictor classes SL, UL) which consists of many "hypotheses" $h_\theta : \mathbb{R}^n \to \mathcal{Y}$. Thus, our input space is assumed to be mapped to a vector embedding of n dimensions. The output space is assumed to be a scalar space for now but it too can take vector representations: the most elementary of such being the one-hot embedding (also known as the "logit" vector). We will also be working with the "softmax" loss function for classification, where $\mathcal{Y} = \{1, 2, ..., C\}$ represents the output classes and

$$\mathcal{L} = \sigma(\hat{y}, k) = \log(\sum_j e^{\hat{y}_j}) - \hat{y}_k$$

2 Subtle Adversaries

Let us start by finding ways to extract adversarial examples for SL and UL scenarios. Consider the empirical risk minimization problem stated in the previous section. As commented, due to the expectation integral in the minimization, we smear out the effect of the outlier when we perform the update step during vanilla SGD [9] (we assume α to be the learning rate):

$$\theta_{k+1} = \theta_k - \frac{\alpha}{N} \sum_{i=1}^{N} \nabla_\theta \mathcal{L}(h_\theta(\bar{x}_i), y_i)$$

To find adversaries upon fixing the optimal parameter $\theta = \theta^*$, we must search through $\mathbb{R}^n$ for the vector that maximizes the loss function for each output class $y \in \mathcal{Y}$. If we perform this optimization in an unconstrained manner for a given class y_t we run the risk of ending up with a vector that actually maps to a different class altogether. Or in other words, every $\bar{x}_i \in \mathcal{X},\ i \neq t$ naturally and rightly ends up as an adversarial example for the data point $(\bar{x}_t, y_t)$.

Instead, we enforce that the adversary $\hat{x}_t$ lies within a "norm ball" centered at $\bar{x}_t$, but we also stress on the fact that this is not the only "perturbation" hyperspace Δ that is possible for the maximization of the loss function. Mathematically, we have:

$$\hat{x}_t = \bar{x}_t + \bar{\delta}$$
$$\delta \in \Delta = \{ \bar{\tau} \mid ||\bar{\tau}||_p \leq \epsilon \}$$

Of course, we are yet to decide the order of the norm p, and we pick $p = \infty$ for now. This converts our hyperspace description to the following simple form:

[1] There may be some confusion with regards to the Generative Adversarial Network [8] architecture as another solution for the conundrum presented in the earlier paragraph. While GANs end up with the same conclusion with regards to the importance of these adversarial points, there are two fundamental differences: (a) These points are fed-back into the model to improve the training accuracy instead of evaluating any sort of robustness measure and (b) they are generated from the prior data set as the support thereby falling in the same trap as the rest of the UL algorithms.

$$\delta \in \Delta = \{\bar{\tau} \mid max_i \bar{\tau}_i \leq \epsilon \}$$

With this, our recipe for producing authentic adversaries becomes:

$$\hat{x} = \bar{x}_t + arg\, max_{\bar{\delta} \in \Delta} \mathcal{L}(h_{\theta^*}(\bar{x}_t + \bar{\delta}), y_t)$$

We will use this strategy to produce some very startling results while testing the robustness of a specimen classification model.

3 Breaking down ResNet-50[2]

ResNet-50 [10] is known as a standard image classification strategy that reached its height of fame during the ImageNet challenge era [11]. It is a 50 layer deep convolutional neural network that has the resolution of 1000 different image classes. The advantage of this model is that it is "pre-trained" and readily available on standard packages such as PyTorch [12], and we use it to classify the following image of Claude and Betty Shannon [13]:

We convert the image into the appropriate input tensor (8 bit RGB scheme) and then feed it to the ResNet-50 model that we instantiate via Torch. It predicts that the image represents a `suit` with approximately 70% confidence score, which we feel is a pretty strong estimate considering there is a good component of suited people in the image.

We then use this input tensor, and try to maximize the (cross-entropy of) the softmax loss function over the l_∞ norm ball with $\epsilon = 2/255$. Due to the simple nature of the constraint (the maximal entry of the perturbation vector $\bar{\delta}$ is $\leq \epsilon$) we can perform the vanilla SGD on $\bar{\delta} \in \Delta$ and then clip the gradient step onto the interval $[-\epsilon, \epsilon]$ (we will meet a more formal approach of the same in section 5). Once we reach convergence, we can then add $\bar{\delta}$ to the image tensor embedding and feed this back into ResNet-50 again. The resulting label predicted by the model is that of a `miniature-schnauzer`, which is a breed of a dog. What is even more surprising is that the confidence score jumps to 95%.

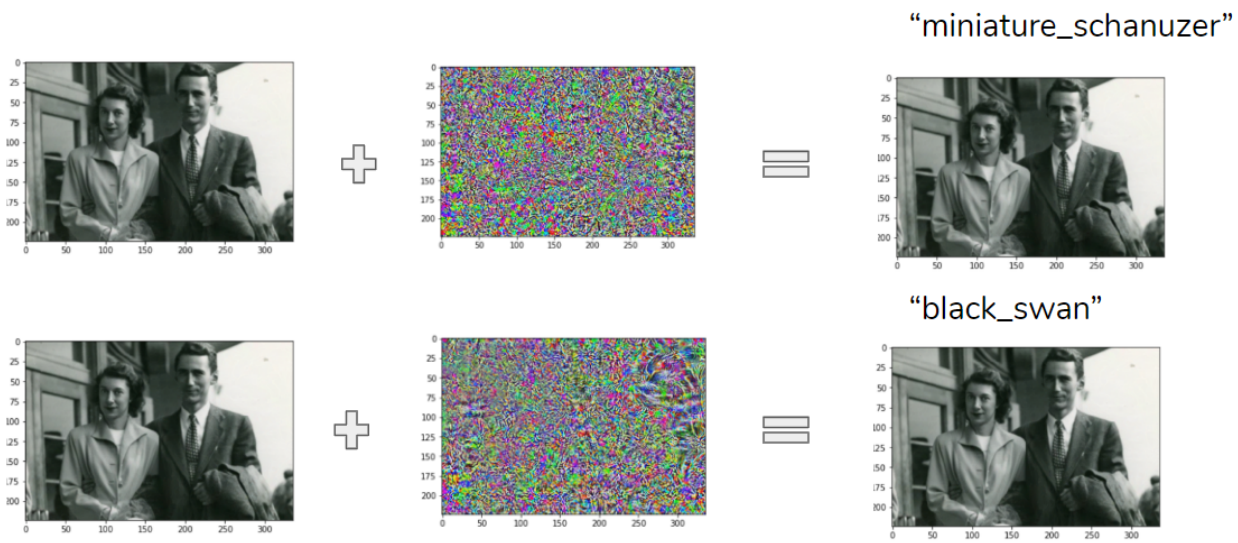
We can take the attack one step further and instead perform the following optimization:

$$\bar{\delta}^* = arg\, min_{\bar{\delta} \in \Delta} [\mathcal{L}(h_{\theta^*}(\bar{x}_t + \bar{\delta}), y_t) - \mathcal{L}(h_{\theta^*}(\bar{x}_t + \bar{\delta}), y_r)]$$

where y_r is a label of our choosing, which, by the nature of the optimization will be the new label that will be predicted by ResNet-50 due to the resulting perturbation. For the sake of irony we selected the label of a `black-swan` as y_r and ran SGD for the above objective, and after adding $\bar{\delta}$ to the input tensor and re-processing the result once again, we get the expected prediction with a whopping

[2]This section is based on an experiment elaborated in [14]

99.99% confidence (more than 4σ!). Obviously, we need to check how the perturbed images look to our eyes, and the summary of the observations is shown below:



They look absolutely the same! The noise plots shown above are amplified 50 times for ease of demonstration, which implies that the actual noise images are almost pitch black (since each RGB vector will enforce the bound of $\leq \epsilon$). This is the reason why the resulting images look similar to our eyes, and therefore this problem becomes essential to fix since it undermines the predictive power of the model that was advertised to be efficient. Let us now rethink the problem at its core and see how we could improve our training procedure to ensure the coverage of these adversaries.

4 Min-max Formulation of Robustness

As explained in section 1, the root of the issue lies in the way the vanilla SGD step is applied to the loss function during training. Due to the extension of the expected risk to the empirical risk (via the LLN), we are explicitly averaging out our perturbed image in the spectrum of the training data set. We restate it here for completeness:

$$\theta_{k+1} = \theta_k - \frac{\alpha}{N} \sum_{i=1}^{N} \nabla_\theta \mathcal{L}(h_\theta(\bar{x}_i), y_i)$$

The modification that is to be done is to replace the empirical average risk by the average of the maximum risk possible due to perturbation. Mathematically, our SGD step receives the robustness upgrade as follows:

$$\theta_{k+1} = \theta_k - \frac{\alpha}{N} \sum_{i=1}^{N} \nabla_\theta \max_{\bar{\delta} \in \Delta(\bar{x}_i)} \mathcal{L}(h_\theta(\bar{x}_i + \bar{\delta}), y_i)$$

Or in other words, our global parameter optimization problem now turns into:

$$\theta^* = arg\, min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \max_{\bar{\delta} \in \Delta(\bar{x}_i)} \mathcal{L}(h_\theta(\bar{x}_i + \bar{\delta}), y_i)$$

Visibly enough, we see that if we are to be wary of the subtle adversaries we need to solve the above min-max optimization problem [15, 16]. Such problems are notorious for their combinatorial nature of evaluation, and the problem is exacerbated if the loss function is non-smooth. A

back-of-the-envelope calculation shows that the time complexity of the search algorithm can get as bad as $|\Theta|^{|\Delta|}$ i.e., it is exponential if there are no other scenario-dependent-advantages left.

In [14] the authors present techniques to circumvent solving the min-max problem by showing that sometimes one can actually prove the absence of adversaries in what they dub as the concept of "certifiable robustness". We shall not take that path, but instead conclude this section by solving the min-max problem for a simple hypothesis: a linear binary classifier. To wit, our model reduces the following simple form:

$$h_{\bar{w},b}(\bar{x}) = \bar{w}^T\bar{x} + b$$

Our loss function also concomitantly degenerates to the (negative log likelihood) of the logistic function:

$$\mathcal{L}(h_\theta(\bar{x}), y) = \log(1 + e^{-yh_\theta(\bar{x})})$$

Finally, since we are doing binary classification $y \in \mathcal{Y} = \{-1, +1\}$. The min-max modification to the SGD requires us to calculate the gradient of the result of a maximization problem ($\nabla_\theta \max_{\bar{\delta}\in\Delta(\bar{x}_i)} \mathcal{L}(h_\theta(\bar{x}_i + \bar{\delta}), y_i)$). To this end, we use the simple yet non-trivial theorem due to Danskin [17] which states that if we are successful in finding the $\bar{\delta}^*$ that maximizes the innermost argument, then

$$\nabla_\theta \max_{\bar{\delta}\in\Delta(\bar{x}_i)} \mathcal{L}(h_\theta(\bar{x}_i + \bar{\delta}), y_i) = \nabla_\theta \mathcal{L}(h_\theta(\bar{x}_i + \bar{\delta}^*), y_i)$$

Thus, it becomes imperative to solve the inner maximization first, and to do so we first observe that our loss function has a negative slope everywhere and we can move our maximization on the function as a minimization of its argument instead. That is:

$$\bar{\delta}^* = arg\,min_{||\bar{\delta}||_\infty \leq \epsilon} yh_\theta(\bar{x} + \bar{\delta}) = arg\,min_{||\bar{\delta}||_\infty \leq \epsilon} y\bar{w}^T\bar{\delta}$$

The l_∞ norm simplifies things considerably by noting that if we are able to set $\delta_i = -\epsilon$ for every positive w_i and $\delta_j = -\epsilon$ for every negative w_j we get the worst possible inner product $\bar{w}^T\bar{\delta}$. We can encapsulate this construction of $\bar{\delta}$ in a compact fashion using the "signum" function $sgn(x)$ which simply returns the sign of x. Therefore, we have:

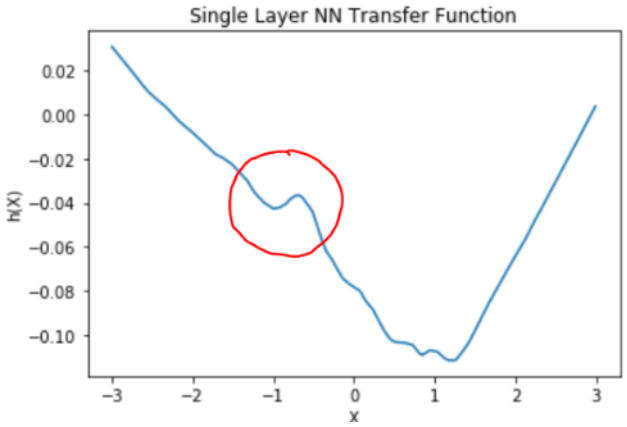
$$\bar{\delta}^* = y\epsilon(-sgn(\bar{w})) \implies min_{||\bar{\delta}||_\infty \leq \epsilon} y\bar{w}^T\bar{\delta} = -\epsilon||\bar{w}||_1$$

We invoke the Danskin theorem and observe that the min-max gradient step now looks like:

$$\theta_{k+1} = \theta_k - \frac{\alpha}{N}\sum_{i=1}^{N} \nabla_\theta \ \log(1 + e^{-yh_\theta(\bar{x}) - \epsilon||\bar{w}||_1})$$

where $\theta = (\bar{w}, \bar{b})$ is the parameter vector over which SGD is being performed for robust minimization. Since the objective is a composition of continuous convex functions, we are assured of a global minimum which thereby "certifies" that the binary classifier is now stable to such norm-ball perturbations.

For more complicated loss function structures this approach becomes infeasible since we cannot ensure the convexity per se. As a preliminary example (when compared to ResNet-50) take the following transfer function ($h_\theta(\bar{x})$, image on the next page) characteristic for a single layer (100 neurons, fully connected, ReLU activated) neural network that is initialized to fit random vector spaces for the input/output embeddings:



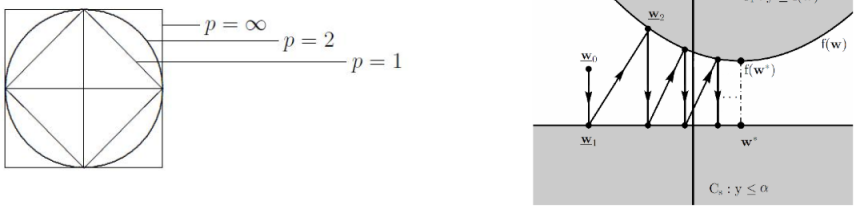
If we end up in the circled concavity, and, for a more realistic scenario, if it was deep enough, the min-max SGD approach will not only be computationally intensive for convergence but the final result may not even be the complete adversarial picture [14]. To handle such non-linear loss functions, we will have to formalize the example presented earlier, and we will do that in the next section as our first complete attack towards mining adversarial examples for a generic hypothesis class.

5 First Attack: Projected Gradient Descent

The process of clipping the extension of the gradient onto the interval $[-\epsilon, \epsilon]$ can be seen as projecting the SGD update step onto the $\epsilon - l_\infty$ ball. The "norm ball" perturbation hyperspaces we work with can be easily checked for convexity. To wit, if we have $\bar{\tau}_1, \bar{\tau}_2 \in \Delta$ and $\lambda_1 + \lambda_2 = 1$ as two non-negative reals, we can use the triangle inequality to see that:

$$||\bar{\tau}_1|| \leq \epsilon, \;\; ||\bar{\tau}_2|| \leq \epsilon \implies ||\lambda_1 \bar{\tau}_1 + \lambda_2 \bar{\tau}_2|| \leq \lambda_1 ||\bar{\tau}_1|| + \lambda_2 ||\bar{\tau}_2|| \leq (\lambda_1 + \lambda_2)\epsilon = \epsilon$$

As a result, the norm balls have nice visual representations: with the 2-norm's radial tendency giving it the familiar appearance of a disk/ring, as shown on the left in the image below:



If we worked with an arbitrary order p for the norm, our clipping/projecting process goes by the more formal name of "Projection Onto Convex Sets" (POCS) type optimization [18, 19]: where we start off at an arbitrary point not necessarily satisfying the constraint, but we force the constraint by iteratively "bouncing" between two convex sets (in our case the other set is at infinity so our "bounces" happen wherever the gradient extension terminates) as shown to the right in the image above (taken directly from [19]). As a consequence, if we halt at the smallest such "bounce displacement" we effectively solve the minimization over the perturbation region.

The POCS approach over the l_∞ ball goes by another formal name: the Fast Gradient Sign Method (FGSM) [20]. Simply put, we perform the adversarial maximization by the vanilla SGD step and the projection by using the signum function as the clipping operator. Mathematically, the SGD step would now have an augmentation:

$$\bar{\delta}_{k+1} = \bar{\delta}_k + \alpha \; clip\{\nabla_{\bar{\delta}_k} \mathcal{L}(h_{\theta^*}(\bar{x} + \bar{\delta}_k), y), [-\epsilon, \epsilon]\}$$
$$\implies \bar{\delta}_{k+1} = \bar{\delta}_k + \alpha\epsilon \; sgn[\nabla_{\bar{\delta}_k} \mathcal{L}(h_{\theta^*}(\bar{x} + \bar{\delta}_k), y)]$$

Thus, the SGD iterations need not go all the way for computing the constrained update step, but instead only need to find the sign of the slope of the tangent plane and multiply it with the radius of the perturbation. This makes FGSM very fast and computationally effective for finding quick adversarial examples (thus justifying its name), but it still depends upon the ease of (automatic) differentiating the loss function. It therefore does not come as a surprise that the above result is the exact solution for the binary classifier (due to the convexity) but it gave us non-perfect confidence scores instead when we tried to break ResNet-50 (which is a deep CNN, thus inherently non-linear).

The occurrence of the $1-$norm in the linear classifier adversary is also not a coincidence, and it can be shown [14] that for this specific case, for any $p-$norm ball perturbation space the result will always be $-\epsilon||\bar{w}||_q$ where p, q are Hölder conjugates, i.e., $p^{-1} + q^{-1} = 1$. However, this property is not true for a more generic hypothesis class and the finite norm POCS based SGD maximization is known formally as Projected Gradient Descent (PGD) [21]. It involves splitting the update step into two substeps:

- **Increment**: $\tilde{\bar{\delta}}_{k+1} = \bar{\delta}_k + \alpha \nabla_{\bar{\delta}_k} \mathcal{L}(h_{\theta^*}(\bar{x} + \bar{\delta}_k), y)$
- **Project**: $\bar{\delta}_{k+1} = \mathcal{P}_\Delta(\tilde{\bar{\delta}}_{k+1})$

The projection step obviously depends on the structure Δ but the operation being performed is similar across all orders of the norm ball. A smarter move that reduces the number of iterations is to move in the direction of maximal gradient step which gives us the so-called "steepest-descent" PGD:

- **Increment**: $\tilde{\bar{\delta}}_{k+1} = \bar{\delta}_k + arg\,max_{||\bar{u}|| \leq \alpha} \bar{u}^T \nabla_{\bar{\delta}_k} \mathcal{L}(h_{\theta^*}(\bar{x} + \bar{\delta}_k), y)$
- **Project**: $\bar{\delta}_{k+1} = \mathcal{P}_\Delta(\tilde{\bar{\delta}}_{k+1})$

With this we conclude our most primitive attack strategy and move to a defense mechanism that allows us to effectively distinguish between an actual sample versus an adversarial sample.

6 First Defense: The Odds Are Odd

PGD based attacks try to maximize the confidence of targeted labels with perturbation $\bar{\delta}$ and in doing so, leave the logits or log-odds exposed to random noise perturbations. This was studied in [22] which proposes a defense called "The Odds are Odd": a statistical test using the distribution of a network's log-odd values for detecting adversarial samples. The main assumption is that when logit values for an input are compared with and without adding random noise, these values will remain comparably close if the original input is a clean sample and will differ significantly if the input is an adversarially perturbed sample. Here, the noise $\bar{\delta}$ is sampled at test-time from a fixed distribution (for e.g. a multivariate normal distribution).

So if we have access to the logit values $\bar{y}$ of a network, then consider the difference between these values for the predicted class t and any other class $i \neq t$ for a given sample, we have[3]:

$$\Delta_{t,i}(\bar{x}) = \bar{y}(\bar{x})_i - \bar{y}(\bar{x})_t$$

For each class, the "Odds are Odd" defense computes the logit differences $\Delta_{t,i}(\bar{x})$ with and without the noise added to the input samples:

[3]Note that $\Delta_{t,i}(\bar{x})$ is always a negative scalar and is not to be confused with the perturbation hyperspace Δ

$$\bar{\Delta}_{t,i}(\bar{x}) = \mathbb{E}_{\bar{\delta}\sim\mathcal{N}}\left[\Delta_{t,i}(\bar{x}+\bar{\delta}) - \Delta_{t,i}(\bar{x})\right]$$

To identify an adversarial sample, a threshold $\tau_{t,i}$ is defined which guarantees a maximum false-positive rate (of say 1%), but which can also attain a desired true-positive rate of identifying the adversarial samples. To compute the expectation, the authors of [22] sample multiple independent noise values from the fixed distribution and take the average of these values. So an input is identified as adversarial if

$$\max_{t\neq i} \bar{\Delta}_{t,i}(x) - \tau_{t,i} \geq 0$$

otherwise the input is identified as a clean sample.

7 Second Attack: The Carlini-Wagner Adversaries

Another limitation of PGD which some defenses particularly exploit comes from the fact that we project the perturbation after every step, i.e. PGD performs one step of SGD update, and then clips all the coordinates to be within the norm ball. This approach can work poorly for gradient descent approaches that have a complicated update step (for example, momentum based techniques [27]). When we project onto the convex set, we unexpectedly or rather abruptly change the input to the next iteration of the algorithm.

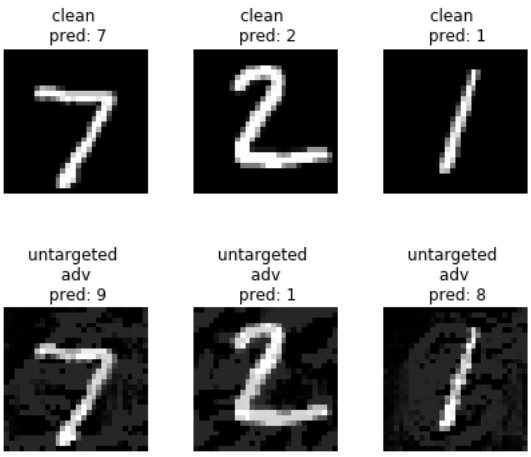
Carlini and Wagner propose in [23] a barrage of attacks to target this vulnerability. They state that instead of maximizing just the loss function $\mathcal{L}\left(h_\theta(\bar{x}+\bar{\delta}), y\right)$ and then using projection to constrain $\bar{\delta}$, we can directly find the smallest successful adversarial perturbation such that

$$\max_{\bar{x}'}\left[\mathcal{L}\left(h_\theta\left(\bar{x}'\right), y\right) - \lambda\left\|\bar{x}' - \bar{x}\right\|_p\right]$$

is achieved, where the order of the norm p can be varied depending on sparsity. We constrain $\bar{\delta}$ by normalization to ensure that the perturbed samples remain in the range of the original input samples. For example, if the original sample $\bar{x} \in [0,1]^n$, then $\hat{x} = \bar{x} + \bar{\delta} \in [0,1]^n$ by transforming $\bar{\delta}$ to $\bar{\kappa}$ according to the following relation:

$$\bar{\delta} = \frac{1}{2}(\tanh(\bar{\kappa}) + 1) - \bar{x}$$

An example of the C&W attacked samples are shown in the figure below:



In practice, this attack requires a smaller perturbation bound ϵ than that is required for PGD based attacks, thereby leading to quicker convergence. This is due to the fact that the perturbation $\bar{\delta}$ is not clipped after every iteration thereby speeding up the gradient step, but as a consequence restricting how far SGD can explore the perturbation space. The attack also makes it harder for defenses like the first one since the attacked samples do not rely on just the confidence or logit values but also force the perturbation to be in the neighbourhood of the sample hoping that adding random noise is not going to vary the logit values vastly.

8 Second Defense: k-Winners Take All

The PGD and Carlini-Wagner based attacks are methods of optimizing the min-max formulation that we discussed in section 4. Solving the min-max problem relies on computing the gradient of the loss function with respect to the input tensor, or in other words, crucially relies on the sensitivity of the loss function. A natural defense therefore, against such optimization, is to include non-differentiable and/or stochastic components into the network which can mask the gradients and render the adversarial search fruitless. One such recent defense is known as "k-Winners Take All" [24] which replaces the standard ReLU activation function in a deep neural network with a discontinuous k-Winners-Take-All (k-WTA) function:

$$[\phi_k(\bar{y})]_j = \begin{cases} y_j & \text{if } y_j \in \{k - largest - in - y\} \\ 0 & \text{o. } w \end{cases}$$

The parameter k is computed by multiplying the dimensionality N of the layer's output with a constant sparsity factor $\gamma \in [0, 1]$ thereby introducing both gradient masking and randomization into the network. Also k-WTA is different from the previous "The Odds are Odd" defense in such a way that it makes it harder to create adversarial samples instead of identifying that a particular sample has been perturbed adversarially.

k-WTA ties the network parameters and their corresponding activation maps produced after applying the discontinuous function at every layer. The amount of perturbation needed for a given sample will suffice if it changes these activation maps for that particular sample significantly (which can be interpreted as the amount of perturbation needed for successful adversarial sample creation). This is argued by a theorem proposed in their paper which we state directly:

Theorem 1 (Xiao et al., (24)) *Given any input $x \in \mathbb{R}^m$ and some β, and $\forall x' \in \mathbb{R}^m$ such that*

$$\frac{\mathrm{d}^2(\boldsymbol{x}, \boldsymbol{x}')}{\|\boldsymbol{x}\|_2^2} \geq \beta$$

If the following condition

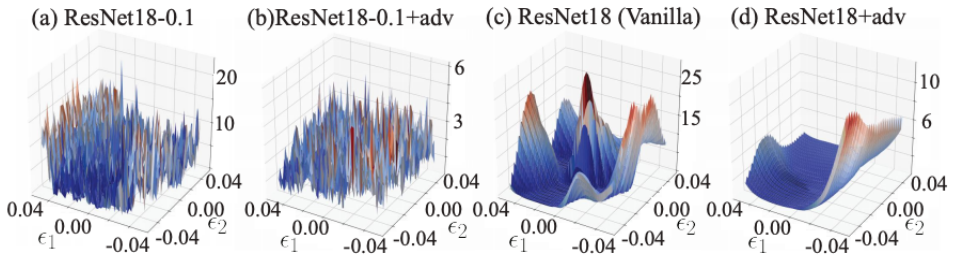
$$l \geq \Omega\left(\left(\frac{m}{\gamma} \cdot \frac{1}{\beta}\right) \cdot \log\left(\frac{m}{\gamma} \cdot \frac{1}{\beta}\right)\right)$$

is satisfied, then with a probability at least $1 - 2^{-m}$ we have

$$\mathcal{A}(\mathrm{W}\boldsymbol{x} + \boldsymbol{b}) \neq \mathcal{A}(\mathrm{W}\boldsymbol{x}' + \boldsymbol{b})$$

Here l is the width of the layer, and γ is the sparsity ratio defined earlier.

This theorem states that if the layer width l is large enough, then a small β and thus a small perpendicular perturbation distance $d(\bar{x}, \bar{x}')$ is enough to trigger a change of the activation pattern for the given sample. This can be observed in the loss landscape of k-WTA networks plotted against the input space and subsequently comparing with the regular ReLU activated networks as shown in the image on the next page, taken directly from [24]:



Training these networks may be difficult due to the rough landscape of the loss function, and the network might not perform better on the regular clean training and test set samples. However the authors argue that this is not the case with k-WTA networks with the following theorem from [24]:

Theorem 2 (Xiao et al., (24)) *Consider N data points $x_1, x_2, \dots x_N \in \mathbb{R}^m$. Suppose $\forall i \neq j$,*

$$\frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|_2} \neq \frac{\boldsymbol{x}_j}{\|\boldsymbol{x}_j\|_2}$$

If l is sufficiently large, then with a high probability we have $\forall i \neq j$,

$$\mathcal{A}\left(\mathrm{W}\boldsymbol{x}_i + \boldsymbol{b}\right) \cap \mathcal{A}\left(\mathrm{W}\boldsymbol{x}_j + \boldsymbol{b}\right) = \varnothing$$

Thus, if the network is sufficiently wide, then for any $i \neq j$, activation of $\bar{x}_i$ is almost different from that of $\bar{x}_j$. The network parameters for predicting the classes or network outputs for $\bar{x}_i$ and $\bar{x}_j$ can be done independently and this does not require triggering a change in their individual activations.

9 Third Attack Phase 1: Backward Pass Differentiable Approximation (BPDA)

Gradient masking defenses such as k-WTA rely on the inability of automatic differentiation (eg. `autograd` by PyTorch [29]) to reliably compute gradients of the non-differentiable components of the network due to the following reasons:

- Obfuscation/Masking of gradients or
- Randomizing the gradient evaluation itself

The authors of [25] propose that these defenses can be brought down using approximation techniques when computing the adversarial search loss optimization. For obfuscated gradients, they propose to approximate derivatives by computing the forward pass normally and computing the backward pass using a differentiable approximation of the function, a process that is known formally as the Backward Pass Differentiable Approximation (BPDA) [25]. To wit, let f be a n-layer network:

$$f(\bar{x}) = f_n \circ f_{n-1} \circ f_{n-2} \dots \circ f_1(\bar{x})$$

where layer f_i is non-differentiable or poses difficulties while computing its gradient. If it is possible to find a differentiable function $g(\bar{x}) \approx f_i(\bar{x})$, and if one is using an automatic differentiation framework [29], we can compute the forward pass through f_i but when back-propagating through the network, we replace ∇f_i by ∇g.

Many non-differentiable defenses can therefore be expressed as follows: given a pre-trained classifier f, construct a preprocessor g and let the secured classifier be defined as:

$$\hat{f}(\bar{x}) = f \circ g(\bar{x}); \quad g(\bar{x}) \approx \bar{x}$$

i.e. in practice, most of the existing defenses rely on a non-differentiable approximate of the identity

map. Thus, we can approximate the difficult derivative using the derivative of the identity function at the point $\hat{x}$ as follows:

$$\nabla_{\bar{x}} f \circ g(\bar{x})|_{\bar{x}=\hat{x}} \approx \nabla_{\bar{x}} f(\bar{x})|_{\bar{x}=\hat{x}}$$

We perform forward propagation through the neural network as usual, but on the backward pass, we replace ∇g with the identity function and this technique goes by the formal name: "Straight-Through" estimation [25].

10 Third Attack Phase 2: Expectation Over Transformation

The paper [26] proposes the technique Expectation over Transformation (EOT) as an attack on the stochastic gradient setup that is introduced by some the defenses. It is used to compute gradients during back-propagation for networks with randomized components. It has been shown that adversarial attacks fail when input samples, in this case the images, undergo transformations like angle, depth or view-point changes. EOT is used in these circumstances to effectively make the adversarial samples robust for such inputs.

Given a network with a randomized component or a component which applies a random transformation to the input $f_r(\bar{x})$, we can estimate its gradient using

$$\nabla_{\bar{x}} \mathbb{E}_r \left[f_r(\bar{x}) \right] = \mathbb{E}_r \left[\nabla_{\bar{x}} f_r(\bar{x}) \right] \approx \frac{\sum_{i=1}^{N} \nabla_{\bar{x}} f_{r_i}(\bar{x})}{N}$$

where r_i are independent draws of the function's randomness. Much like the BPDA technique, these approximated gradients can be computed safely during the back-propagation thus once again making our adversarial samples subtle.

11 Conclusion: The Cat and Mouse Game

After looking at all these attacks and defenses, we must always refer back to our min-max formulation introduced in section 4, which we restate here for the sake of continuity:

$$\theta^* = arg\, min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \max_{\bar{\delta} \in \Delta(\bar{x}_i)} \mathcal{L}(h_\theta(\bar{x}_i + \bar{\delta}), y_i)$$

The adversarial attacks are always trying to push the inner maximization so that outer minimization cannot be achieved and the defenses are catching up by doing just the opposite. Almost all attacks are characterized by two significant features:

- The norm ball perturbation space Δ that they consider (eg. l_2, l_0, l_∞ et cetera.) and
- the method they use for optimizing over that norm ball (SGD, C&W Regularization et cetera).

We have seen from the FGSM and PGD attacks as to how vanilla SGD can be used to create adversarially perturbed samples by following the inner maximization. Simple defenses can use the outer minimization to combat this by performing further alternate minimization and maximization steps to train a network on these adversarial samples. We then observed how "The Odds are Odd" approach introduced random noise into the samples to combat against these attacks, effectively leveraging PGD's deficiency in stabilizing the inner maximization to cover the entire adversarial sample space. Then come the Carlini & Wagner attacks which regularize the maximization such that adversarial samples are bound to the neighbourhood of the original sample. Adding random noise now does not knock out the adversarial sample and we also speed up our inner optimization. k-WTA defense was then presented which brought in the notion of non-differentiable activation functions thereby seemingly rendering PGD and C&W attacks useless. However, BPDA and EOT show that these obfuscated gradients only give a false sense of security and can be attacked easily with simple

approximations to calculating the gradients during back-propagation. To conclude this discussion, in general, given sufficiently many iterations and large enough perturbation bound ϵ, PGD methods are empirically the most effective and quicker methods for optimizing over virtually any norm ball. It has been shown in [28] that all recent defenses proposed can be defeated this way.

12 Epilogue

We feel that there is still a long way to go in achieving robust neural networks against any generic adversarial sample. Nevertheless we also think it is very interesting to watch this "cat-and-mouse" game! We thank Prof. Ramtin Pedarsani, UCSB, for giving us an opportunity to study the current adversarial robustness landscape as part of the ECE 284 course conducted in Spring 2020 at UC Santa Barbara.

References

[1] Vapnik V., *The Nature of Statistical Learning Theory*, Springer-Verlag, 2nd Edition, 2000.

[2] Bayes, T., *An Essay towards solving a Problem in the Doctrine of Chance*, Philosophical Transactions, Volume 53, pp. 370–418, 1763.

[3] Legendre, A. M., *Nouvelles méthodes pour la détermination des orbites des cométes*, 1805.

[4] Taleb N., Pilpel A., *I problemi epistemologici del risk management (On The Unfortunate Problem of the Nonobservability of the Probability Distribution)*, Economia del rischio, Milan, 2004.

[5] Levi I., *The Enterprise of Knowledge*, MIT Press, 1980.

[6] Hume D., *A Treatise of Human Nature*, Oxford University Press, 1739.

[7] Hume D., *An Enquiry Concerning Human Understanding*, Oxford University Press, 1748.

[8] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.,; Bengio, Y., *Generative Adversarial Networks*, Proceedings of the International Conference on Neural Information Processing Systems 2014, pp. 2672–2680, 2014.

[9] Bottou, L., *Online Algorithms and Stochastic Approximations*, Cambridge University Press, 1998.

[10] He K., Zhang X., Ren S., Sun J., *Deep Residual Learning for Image Recognition*, arXiv:1512.03385, 2015.

[11] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., Berg A., and Fei-Fei L., *ImageNet Large Scale Visual Recognition Challenge*, International Journal of Computer Vision, 2015.

[12] `torchvision.models.resnet50`, Torch Vision Documentation, PyTorch Team, https://pytorch.org/docs/stable/torchvision/models.html, 2019.

[13] Original image due to: Andrew and Peggy Shannon, published in Soni J., Goodman R., *Betty Shannon, Unsung Mathematical Genius*, Scientific American, 24 July 2017.

[14] Madry A., Kolter Z., *Adversarial Robustness: Theory and Practice*, Tutorial Session, NIPS 2018.

[15] Von Neumann, J., *Zur Theorie der Gesellschaftsspiele*, Mathematische Annalen, Vol. 100, pp. 295–320, 1928.

[16] Parthasarathy T., *On Games over the unit square*, Society of Industrial and Applied Mathematics, Volume 19, Number 2, 1970.

[17] Başar T., Bernhard P., *H^∞-Optimal Control and Related Minimax Design Problems*, pp. 383-389, Birkhäuser Basel, 2008.

[18] Bregman L., *The Relaxation Method of Finding the Common Point of Convex Sets and Its Application to the Solution of Problems in Convex Programming*, Computational Mathematics and Mathematical Physics, Vol. 7, No. 3, pp. 200-217, Leningrad, 1967.

[19] Tofighi M., Kose K., Cetin A., *Denoising Using Projection Onto Convex Sets (POCS) Based Framework*, arXiv:1309.00700v1, 2013.

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

[21] Rockafellar R. T., *Monotone operators and the proximal point algorithm*, SIAM Journal on Control and Optimization, Vol. 14(5), pp. 877–898, 1976.

[22] Roth, K., Kilcher, Y., and Hofmann, T. *The odds are odd: A statistical test for detecting adversarial examples*, International Conference on Machine Learning, 2019.

[23] Carlini, N., Wagner, D., *Towards Evaluating the Robustness of Neural Networks*, arXiv:1608.04644, 2017.

[24] Xiao, C., Zhong, P., and Zheng, C., *Resisting Adversarial Attacks by k-Winners-Take-All*, International Conference on Learning Representations, 2020.

[25] Athalye, A., Carlini, N., Wagner, D., *Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples*, International Conference on Machine Learning, 2018.

[26] Athalye, A., Engstrom, L., Ilyas, A., Kwok, K., *Synthesizing Robust Adversarial Examples*, International Conference on Machine Learning, 2017.

[27] Botev A., Lever G., Barber D., *Nesterov's Accelerated Gradient and Momentum as approximations to Regularised Update Descent*, arXiv:1607.01981, 2016.

[28] Tramer F., Carlini N., Brendel W., and Madry A., *On Adaptive Attacks to Adversarial Example Defenses*, arXiv:2002.08347, 2020.

[29] Paszke A., Gross S., Chintala S., Chanan G., Yang E., DeVito Z., Lin Z., Demsaison A., Antiga L., Lerer A., *Automatic differentiation in PyTorch*, 31st Conference on Neural Information Processing Systems, 2017.