MANIFOLD ASSUMPTION AND DEFENSES AGAINST ADVERSARIAL PERTURBATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

In the adversarial perturbation problem of neural networks, an adversary starts with a neural network model F and a point $\mathbf{x}$ that F classifies correctly, and identifies another point $\mathbf{x}'$, which is *nearby* $\mathbf{x}$, that F classifies *incorrectly*. In this paper we consider a defense method that is based on the *semantics* of F. Our starting point is the common *manifold assumption*, which states that natural data points lie on separate low dimensional manifolds for different classes. We then make a further postulate which states that *(a good model) F is confident on natural points on the manifolds, but has low confidence on points outside of the manifolds,* where a natural measure of "confident behavior" is $\|F(\mathbf{x})\|_\infty$ (i.e. how confident F is about its prediction). Under this postulate, an adversarial example becomes a point that is outside of the low dimensional manifolds which F has learned, but is still close to at least one manifold under some distance metric. Therefore, defending against adversarial perturbations becomes embedding an adversarial point back to the nearest manifold where natural points are drawn from. We propose algorithms to formalize this intuition and perform a preliminary evaluation. Noting that the effectiveness of our method depends on both how well F satisfies the postulate and how effective we can conduct the embedding, we use a model trained recently by Madry et al., as the *base model*, and use *gradient based optimization*, such as the *Carlini-Wagner attack* (but now they are used for defense), as the *embedding procedure*. Our preliminary results are encouraging: The base model wrapped with the embedding procedure achieves almost perfect success rate in defending against attacks that the base model fails on, while retaining the good generalization behavior of the base model.

1 INTRODUCTION

In the adversarial perturbation problem of neural networks, an adversary starts with a neural network model F (we assume that F ends with a softmax layer, which is a common assumption in the literature), and a point $\mathbf{x}$ that F classifies correctly, and identifies another point $\mathbf{x}'$, which is *nearby* $\mathbf{x}$, that F classifies *incorrectly*. Szegedy et al. (2013) first noticed the vulnerability of existing (deep) neural networks to adversarial perturbations, which is a somewhat surprising phenomenon given the great generalization capability of these networks. Since then, a line of research (see, for example, Goodfellow et al. (2014); Papernot et al. (2016b); Miyato et al. (2017); Madry et al. (2017)) has been devoted to harden neural networks against adversarial perturbation. However, while modest progress has been made, until now there is still a large margin in successfully defending against more advanced attacks, such as Carlini-Wagner attacks Carlini & Wagner (2017).

In this paper we propose a defense method that is based on the *semantics* of a neural network. Our starting point is the common manifold assumption (Zhu & Goldberg (2009); man) made in several classic unsupervised and semi-supervised learning algorithms, which states that natural data points lie on separate low dimensional manifolds for different classes. Recent work, for example Basri & Jacobs (2016), has given intriguing evidence that neural networks can efficiently represent low dimensional manifold data. Based on this assumption, we make a further *postulate*, which states that *a good model is confident on natural points on the manifolds, but has low confidence on points outside of the manifolds.* To this end, a natural measure of confidence of a model F at a point $\mathbf{x}$ is to simply measure how confident F is about its prediction, that is, $\|F(\mathbf{x})\|_\infty$. With these assumptions, the adversarial perturbation problem can thus be understood as follows: the manifold assumption

indicates that the natural data points are far from permeating the entire space, thus while we may have learned perfectly for data points from the natural low dimensional manifolds, an adversary may still be able to perturb a point slightly to make it out of the natural manifolds, and render the model's behavior "unpredictable." Moreover, our postulate indicates that this "unpredictable behavior" can be understood as the model having low confidence in its prediction.

Therefore, a natural plan to tackle adversarial perturbation problem is to try to embed an adversarial point, which is distinguishable from natural points as the model has low confidence at it, back to the natural manifolds where the model has higher confidence. We propose two embedding objective functions for this purpose: (1) δ-Most Confident Neighbor objective ($\texttt{MCN}_\delta$), where given a point $\mathbf{x}$, and a radius parameter $\delta > 0$, we compute $\texttt{MCN}_\delta(F, \mathbf{x}) = \arg\max_{\mathbf{z} \in N(x,\delta)} \|F(\mathbf{z})\|_\infty$ where $N(x, \delta)$ denotes the δ-ball around x with respect to some norm $\|\cdot\|$, and (2) p-Nearest Confident Neighbor objective ($\texttt{NCN}_p$), where given a parameter $p \in (0, 1)$, we compute $\texttt{NCN}_p(F, \mathbf{x}) = \arg\min_{\mathbf{z}} \|z - x\|$ subject to $\|F(\mathbf{z})\|_\infty \geq p$. With these two objectives, the end to end prediction thus becomes $F(\texttt{MCN}_\delta(F, x))$ or $F(\texttt{NCN}_p(F, x))$.

A notable feature of these defense objectives is that they are *semantic*: They only fail when F has a *confident but wrong* prediction in the neighborhood. However, by our postulate this is infeasible because an adversarial point must be near to at least one natural manifold where F has higher confidence. Therefore, if the natural manifolds are indeed well separated as indicated by the manifold assumption, we can successfully defend the attacks.

Guided by our method, we construct algorithms to compute $\texttt{MCN}_\delta$ and $\texttt{NCN}_p$ and give a preliminary evaluation over CIFAR10. Note that the effectiveness of our method depends on two components: (1) To which degree the base model F satisfies our postulate, and (2) Effectiveness of solving the embedding objectives. As a result, we use a deep residual network recently trained by Madry et al. Madry et al. (2017), as base model, and use gradient based optimization, such as Carlini-Wagner attacks Carlini & Wagner (2017) (which, however, are now used for defense) to solve $\texttt{MCN}_\delta$ or $\texttt{NCN}_p$. Our empirical results are encouraging: (1) It achieves almost perfect success rate in defending against attacks that the base model fail at, and (2) It also retains the good generalization behavior of the base model.

There are two interpretations of the encouraging success of our method: First, from a purely perspective, our results advocate that a good model should have the property that *it is confident only on the structure it has learned (for example the natural manifolds), and does not claim confidence beyond it*. Second, perhaps more importantly, *adversarial perturbation can naturally coexist with good generalization*, and thus may not be surprising after all. In fact, even if one has learned perfectly over the underlying natural manifolds, adversarial perturbation may still exist in abundance, and our method can be viewed as given a *post-processing* to reduce points from outside (but near) the manifolds to the natural manifolds one has learned. This gives rise to intriguing yet challenging questions, such as trying to *learn apriori* how to embed the entire space to manifolds that one is confident about rather than relying on a post-processing embedding procedure, which we leave as future research.

2 Prior Work

Szegedy et al. (2013) first observed the susceptibility of deep neural networks to adversarial perturbations. Since then, a large body of work have been devoted to study hardening neural networks for this problem (a small set of work in this direction is Goodfellow et al. (2014); Papernot et al. (2016b); Miyato et al. (2017)). Simultaneously, another line of work have been devoted to devise more effective or efficient attacks (a small set of work in this direction is Moosavi-Dezfooli et al. (2016); Papernot et al. (2016a); Carlini & Wagner (2017)). Unfortunately, there still seems to be a large margin for the defense methods to defend against more sophisticated attacks, such as Carlini-Wagner attacks Carlini & Wagner (2017). For example, while the recent robust residual network constructed by Madry et al. (2017) achieves encouraging robustness results on MNIST, on CIFAR10 the accuracy against a strong adversary can be as low as 45.8%.

A dominant defense approach is to add to the training objective function an appropriate "robustness component," and hope that the learning algorithm automatically learns to be robust. Our method is fundamentally different and tries to directly exploit the structure a model has learned. Our method

has some remote similarities with a recent proposal by Lu et al. (2017), which tries to exploit features deep in the network (that is "semantics" of a model) to *detect* adversarial examples. However, beyond detecting adversarial attacks, our work takes a step further to compute correct prediction on the adversarial examples. Moreover, our work is built upon the common manifold assumption made in semi-supervised and unsupervised learning, and we are not aware of any previous work that has explored this direction.

We note that the manifold assumption has guided the design of several classic dimensionality reduction algorithm, unsupervised and semi-supervised learning algorithms (for example, see Basri & Jacobs (2016) and references therein). An intriguing recent effort is to demonstrate that neural networks are effective devices to represent low dimensional manifold data (for example, Rifai et al. (2011); Shaham et al. (2015); Basri & Jacobs (2016)). Our work is directly inspired by these work, and tries to argue that if manifold assumption indeed holds then it is somewhat not surprising to see the coexistence of good generalization and adversarial perturbation problem.

3 Preliminaries

We only need minimal background to set up discussions in this paper. Following previous work Carlini & Wagner (2017); Papernot et al. (2016b) we define F to be a neural network after the softmax layer. With this notation, the final classification is then $C_F(\mathbf{x}) = \arg\max_i F(\mathbf{x})_i$ where $F(\mathbf{x})_i$ gives the confidence of the network in classifying $\mathbf{x}$ for class i. We use $Z(\mathbf{x})$ to denote part of F except the softmax layer. That is, $Z(\mathbf{x})$ computes the *logits* to be fed into the softmax function. We use $\mathcal{C}$ to denote the class of all labels.

4 Postulate and Algorithms

Our starting point is the common *manifold assumption* that guides the design of several unsupervised and semi-supervised learning algorithms, which states that *natural data points of different classes come from separated low-dimensional manifolds* (Zhu & Goldberg (2009); man). Intuitively, we would expect that a good learning algorithm should learn a model that behaves well on these manifolds. On the other hand, what about points that are not on these manifolds? Naturally, we would want that the model behaves *badly* on those points, because those do not belong to data generation. Our postulate gives a concrete version of this intuition and states that:

Postulate: *(A good model) F is confident on natural points drawn from the manifolds, but has low confidence on points outside of the manifolds.*

Essentially, the "confidence" in the postulate provides a means to distinguish between "good" and "bad" behaviors of a model. A natural measure of confidence of a model F at a point $\mathbf{x}$ is to simply measure $\|F(\mathbf{x})\|_\infty$, that is, how confident F is about its prediction. With this measure our postulate thus states the following: (1) $\|F(\mathbf{x})\|_\infty$ is large for a point x that is drawn from the manifolds, and (2) $\|F(\mathbf{x})\|_\infty$ is small otherwise.

We now give a view of adversarial examples under our postulate. The manifold assumption indicates that natural data points are far from permeating the entire space. Therefore adversarial examples can be viewed as points that are "perturbed" to be out of the manifolds that generate natural data points, but they are still close to at least one underlying manifold under certain distance metric. Moreover, with our postulate it is the case that a good model F must be *unconfident* there.

Therefore, a natural plan to tackle adversarial points is to try to embed them back to the manifolds where the model has higher confidence. In particular, we note that there are *two components* for this idea to work: First, we need the model F, which is the *base model* we start with, to be trained in a way to satisfy the postulate so that it will not be confident on "bad" points outside the manifolds. Second, we need an effective, and preferably efficient, *embedding procedure* so that we can go back to the natural manifold.

Objective Functions. We now give two natural objective functions for such embedding:

Definition 1 (**Most Confident Neighbor Objective**). *Let F be a model, $\|\cdot\|$ be a norm, and $\delta > 0$ be a real number. The δ-*Most Confident Neighbor *objective (or simply $\texttt{MCN}_\delta$ objective) computes:*

$$\texttt{MCN}_\delta(F, \mathbf{x}) \equiv \arg\max_{\mathbf{z} \in N(\mathbf{x}, \delta)} \|F(\mathbf{z})\|_\infty. \tag{1}$$

where $N(\mathbf{x}, \delta)$ is the δ-ball around $\mathbf{x}$ with respect to $\|\cdot\|$, and $\|\cdot\|_\infty$ denotes the L_∞-norm.

Definition 2 (**Nearest Confident Neighbor Objective**). *Let F be a model, $\|\cdot\|$ be a norm, and $p \in (0, 1)$ be a real number. The p-*Nearest Confident Neighbor *objective (or simply $\texttt{NCN}_p$ objective) computes:*

$$\texttt{NCN}_p(F, \mathbf{x}) \equiv \arg\min_{\mathbf{z}} \|\, \mathbf{z} - \mathbf{x} \,\| \text{ subject to } \|F(\mathbf{z})\|_\infty \geq p. \tag{2}$$

where $\|\cdot\|_\infty$ denotes the L_∞-norm.

In words, MCN says that one should consider a reasonable δ-neighborhood of a point and view the most confident point in that ball as a good point that lies on a natural manifold; meanwhile, NCN says that one should consider the *nearest* neighbor that achieves certain confidence threshold as a point that lies on a natural manifold. Under our postulate, solutions of these objectives are thus preferable to be used for prediction.

Duality. We note that the optimization problems for 1 and 2 are essentially dual to each other: In the MCN objective, we have a fixed radius of the neighborhood to search and one finds the point that achieves highest confidence. In NCN objective, instead of fixing the neighborhood radius, one starts with a confidence threshold p, and finds the nearest point that achieves that confidence.

In fact, we note that the Lagrange dual of these two objective functions share the same form. To see this, we note that the Lagrange dual problems of 1 and 2 are given as

$$\arg\min_{\mathbf{z}} \big(-\|F(\mathbf{z})\|_\infty + \mu \left(\|\, \mathbf{z} - \mathbf{x} \,\| - \delta\right) \big), \tag{3}$$

and

$$\arg\min_{\mathbf{z}} \big(\|\, \mathbf{z} - \mathbf{x} \,\| - \lambda \left(\|F(\mathbf{z})\|_\infty - p\right) \big) \tag{4}$$

respectively. Therefore, if one sets $\mu = \frac{1}{\lambda}$, then these two problems are equivalent to the following unified optimization problem

$$\arg\min_{\mathbf{z}} \|\, \mathbf{z} - \mathbf{x} \,\| - \lambda \|F(\mathbf{z})\|_\infty \tag{5}$$

Intuitively, this objective encourages small distances of $\mathbf{z}$ from $\mathbf{x}$ while large confidence of F at $\mathbf{z}$.

Algorithms. We now give algorithms that instantiate our discussion so far. In particular, we give algorithms to solve MCN and NCN objectives for embedding purpose. To start with, we note that from an algorithmic point of view, our approach essentially tries to wrap around a learned model with a "shell" to handle points that are "slightly" out of the natural manifolds it may have learned. There are many possible shells that can be instantiated for this purpose to exploit the effectiveness-efficiency tradeoff. For example, a valid such shell is to sample in the neighborhood and do a weighted majority vote. We are interested in experimenting with these shells, which may shed light on what a deep network has learned. We thus first give some general definitions. We begin with *model shell*.

Definition 3 (**Model Shell**). *Let F be a model. A model shell G is an algorithm that takes as its input F and a point $\mathbf{x} \in \mathbb{R}^d$ to classify, and outputs a* softmax layer *$G(F, \mathbf{x})$. In this paper we only consider the case where F is a model that is chosen apriori and fixed. In this case, the resulting model for classification is $G(F, \cdot)$, which maps a feature vector to a softmax layer. In this scenario, we say that F is the* base model *of the model shell G, and denote $G(F, \cdot)$ as $G[F]$.*

Model shell captures the intuition that we want to wrap around an existing model with another algorithm that can exploit semantics of the base model. In this paper, we only examine model shells with a restricted structure, which we call *factor model shells*. We define *search factor*,

Definition 4 (**Search Factor**). *A search factor H is an algorithm that takes as its input a model F, and a point $\mathbf{x} \in \mathbb{R}^d$ to classify, and output another point $\mathbf{x}' \in \mathbb{R}^d$. If F is chosen and fixed apriori we say that F is the* base model *of the search factor H, and denote $H(F, \cdot)$ as $H[F]$.*

Definition 5 (**Factor Model Shell**). *A model shell G is said to be a* factor model shell *if G can be written as a composition of $F \circ H$ where H is a search factor. In other words, $G(F, \mathbf{x})$ works as*

$$G(F, \mathbf{x}) \equiv (F \circ H[F])(\mathbf{x}) = F\left(H(F, \mathbf{x})\right).$$

A factor model shell captures our intuitions from MCN or NCN objectives. That is, we first apply a search factor H to find another feature point $\mathbf{x}' = H(F, \mathbf{x})$, and then apply the base model F on $\mathbf{x}'$ to produce a final prediction. As a result, a factor model shell for $\texttt{MCN}_\delta$ objective should compute the final prediction as:

$$(F \circ \texttt{MCN}_\delta)(\mathbf{x}) = F(\texttt{MCN}_\delta(F, \mathbf{x})), \tag{6}$$

and accordingly for $\texttt{NCN}_p$ objective:

$$(F \circ \texttt{NCN}_p)(\mathbf{x}) = F(\texttt{NCN}_p(F, \mathbf{x})) \tag{7}$$

Semantic defenses. We note that exact computations of (6) and (7) provide "semantic defenses:" That is, the final prediction is wrong only if the model has *a confident but wrong* prediction in the neighborhood. Under our postulate this is impossible because an adversarial point must be near to a manifold where the model has high confidence. We note that a line of work shows that a deep network can have high confident predictions on *random noises* Nguyen et al. (2015), we discuss these phenomena later in Section 6.

Non-differentiability of MCN or NCN model shells. We note that there are additional benefits taking the MCN or NCN approach. That is, the resulting factor model shells, $(F \circ \texttt{MCN}_\delta)$ and $(F \circ \texttt{NCN}_p)$ are *non-differentiable*. This will render existing attacks, which are all based on the assumption that *the output softmax layer is differentiable with respect to the input feature vector x*[1] fail to work directly to attack the end-to-end model $F \circ \texttt{MCN}_\delta$ or $F \circ \texttt{NCN}_p$. In particular, we note that $\texttt{MCN}_\delta(F, x)$ and $\texttt{NCN}_p(F, x)$ are non-differentiable given that F is a deep neural network. This, in particular, implies the compositions, namely $F \circ \texttt{MCN}_\delta$ and $F \circ \texttt{NCN}_p$, are non-differentiable as well. This forces attacks to either attack the base model F and then transfer attacks to the model shell, or has to find some differentiable approximations in order to perform attacks, or use derivative-free optimization to attack the model shell directly.

Solving objective functions by Gradient-based optimization. We now give one concrete search factor to solve (1) and (2). To start with, we note that the optimization for solving $\texttt{MCN}_\delta$ (1) is nothing but for each label $t \in \mathcal{C}$ we try to find

$$\mathbf{z}_t = \operatorname*{arg\,max}_{\mathbf{z} \in N(\mathbf{x}, \delta)} F(\mathbf{z})_t \tag{8}$$

and then compute $\mathbf{z}^* = \arg\max_{\mathbf{z}_t} F(\mathbf{z}_t)_t$.

Now, however, we can solve (8) using any preferred gradient-based optimization (for example, projected gradient descent Nocedal & Wright (2006)). This thus gives the following factor model shell:

[1] For example, let us examine one objective function used in Carlini-Wagner Carlini & Wagner (2017), which is one of the strongest attacks known. They use an objective function of the form (f_5 on pp. 6 of their paper Carlini & Wagner (2017)):

$$f(\mathbf{x}') = -\log(2F(\mathbf{x}')_t - 2)$$

where $F(\mathbf{x}')_t$ denote the coordinate of $F(\mathbf{x}')$ at label t. This objective is differentiable in $\mathbf{x}'$ as F is differentiable in $\mathbf{x}'$. We note that such a first-order assumption is natural for neural networks because neural networks are typically composition of differentiable functions. With first order information, attacks can "relatively accurately" follow the gradient direction to modify the image feature so as to increase the confidence of the model in *incorrect labels* and thus produce adversarial perturbations.

Algorithm 1 Solving $\texttt{MCN}_\delta$ objective.

Input: $\mathbf{x}$ a feature vector, $\delta > 0$ a real parameter, a base model F, any gradient-based optimization algorithm $\mathcal{O}$ to solve the constrained optimization problem defined in (8).

function MCNOracleShell($\mathbf{x}, \delta, F$)
  for $t \in \mathcal{C}$ **do**
    $\mathbf{z}_t \leftarrow \mathcal{O}(\mathbf{x}, F, t)$
  return $\arg\max_{t \in \mathcal{C}} F(\mathbf{z}_t)$

Remark 1 (**Turning an adversarial attack to a defense**). *In particular, we note that solving (8) is similar to* a targeted adversarial attack where we want to modify $\mathbf{x}$ so as to increase the confidence in label t. *Therefore one can instantiate $\mathcal{O}$ as* any adversarial attack. *Specifically, one can any strong attack proposed in the Carlini-Wagner Carlini & Wagner (2017), denoted by* CW, *which takes as input a feature vector* $\mathbf{x}$, *a model F, and a label t, and output $\mathbf{z}_t$ that* $\arg\max_i F(\mathrm{CW}(\mathbf{x}, F, t)) = t$, *and use it as $\mathcal{O}$.*

Remark 2 (**Local vs. global optimal**). *We note that gradient-based optimization may only find local optimal in the neighborhood. However, global optimal may not be needed for our method to work, as long as local minima are good. For example, let l be the correct class, and denote*

$$\mathcal{S}_l = \{z : z \in N(x, \delta), z \text{ is a local maximal of } \|F(x)\|_\infty, \|F(z)\|_\infty = F(z)_l\}.$$

That is $\mathcal{S}_l$ contains all the local maximal for label l. Moreover, define

$$\mathcal{S}_{-l} = \{z : z \in N(x, \delta), z \text{ is a local maximal of } \|F(x)\|_\infty, \|F(z)\|_\infty \neq F(z)_l\}.$$

That is $\mathcal{S}_{-l}$ contains all the local maximal that is not for label l, then local optimal works as soon as $\inf_{z \in \mathcal{S}_l} \|F(z)\|_\infty > \sup_{y \in \mathcal{S}_{-l}} \|F(y)\|_\infty$.

We can also take a similar path to solve $\texttt{NCN}_p$ objective with standard numeric optimization. Note that $\|F(\mathbf{z})\|_\infty \geq p$ is equivalent to $\exists t, F(\mathbf{z})_t \geq p$. Therefore, we can solve

$$\mathbf{z}_t = \arg\min_{F(\mathbf{z})_t \geq p} \| \mathbf{z} - \mathbf{x} \| \tag{9}$$

for every t, and then compute $\mathbf{z}^* = \arg\min_{\mathbf{z}_t} \| \mathbf{z}_t - \mathbf{x} \|$. By a similar reasoning as the MCN case above, $\mathbf{z}^*$ is the solution to problem 2. Note that problem 9 is a non-convex optimization problem since $F(\mathbf{z})_t$ is non-convex and thus it remains hard to find a global optimal solution. Nevertheless, it makes sense to find local solutions. For example, a natural approach is to turn solving the original constrained optimization problem into solving a series of unconstrained optimization. More specifically, let us define the quadratic penalty function

$$Q(\mathbf{z}, \alpha, t) = \| \mathbf{z} - \mathbf{x} \| + \frac{\alpha}{2} \left([F(\mathbf{z}) - p]^- \right)^2 \tag{10}$$

where $[y]^-$ denotes $\max\{y, 0\}$. Then we can solve the unconstrained optimization $\min Q(\mathbf{z}, \alpha, t)$ for a series of different α, and terminates once we find a satisfying solution. The intuition is that i) solving the unconstrained optimization is easy, ii) by increasing the coefficient α, we can force the minimizer of the penalty function closer to the feasible region $F(\mathbf{z}) \geq p$, , and iii) for well chosen α, the minimizer of the penalty function is close enough to that of problem 9. This is well known as the augmented Lagrangian method Bertsekas (1996). There are other classic constrained optimization algorithms such as interior methods and sequential quadratic programming Nocedal & Wright (2006). More recently, there is also some work Bienstock & Michalka (2014) specifically focusing on optimizing convex objective over non-convex constraints, into which problem 9 falls as well. In short, there are quite a few methods can be exploited to solve problem 9. Having that being said, the algorithm to solve the original $\texttt{NCN}_p$ is summarized below as Algorithm 2.

Algorithm 2 Solving $\texttt{NCN}_p$ objective.

Input: $\mathbf{x}$ a feature vector, $p > 0$ a real parameter, a base model F, any gradient-based optimization algorithm $\mathcal{O}$ to solve the constrained optimization problem defined in (9).

function NCNOracleShell($\mathbf{x}, p, F$)
  for $t \in \mathcal{C}$ **do**
    $\mathbf{z}_t \leftarrow \mathcal{O}(\mathbf{x}, F, t)$
  return $\mathbf{z}^* = \arg\min_{\mathbf{z}_t} \| \mathbf{z}_t - \mathbf{x} \|$

5 PRELIMINARY EVALUATION

In this section we perform a preliminary empirical evaluation of our approach against strong adversarial attacks. Note that there are three basic components in a defense with our method: (1) the *base model* we choose, (2) the *embedding procedure* we choose to wrap around the base model, and (3) the *attack* we choose to challenge the defense. As a result, there are three key empirical questions we aim to answer:

1. *How susceptible is the chosen base model to the chosen attack?* Clearly, we want that the chosen attack is strong enough to break even a "good" base model on many points.
2. *How effective does our approach improve robustness of the base model?* Ideally, we want to see a significant improvement of robustness for a good base model.
3. *Will our approach change the generalization behavior of the base model?* Since our method will change the way how the model predicts, we need to justify if there is little or no change in generalization.

A summary of our findings are the following:

1. We choose a recent robust model trained by Madry et al. Madry et al. (2017) as the base model, and use a strong attack by Carlini Wagner (denoted by CW attack) Carlini & Wagner (2017) as the attack method. We use CIFAR10 Krizhevsky (2009) as the dataset to evaluate the robustness and generalization. As reported by the paper, Madry et al.'s model is fairly susceptible to CW attack over CIFAR10, and the accuracy against CW attack can be as low as 45.8%. Our evaluation confirms this susceptibility and *indicates a significant room for improving robustness*.
2. For model shell, we choose $\mathrm{MCNOracleShell}$, and use again CW attack as indicated by our discussion in Remark 1. We find that this embedding procedure *significantly improves the robustness of the base model* over the small batch of points in our evaluation. In fact, CarliniWagerShell retains robustness for points that are already robust with the base model, but also successfully defend all attacks that can succeed on the base model. Therefore, it reduces the attack success rate from 30% to 0%.
3. Finally, *applying* CW *attack as the embedding procedure incurs almost no effect on generalization.* In fact, there is only one such point that the embedding procedure changes the prediction from correct to wrong. We note that the model has very low confidence ($\sim 50\%$, which is much lower than other correct predictions) on the correct prediction, which may indicate that this image is difficult to classify. We give more discussions on this point later in this section.

Limitations of our experiments. Unfortunately due to the complexity of the Madry et al.'s residual network model, our embedding procedure (which is CW attack in our experiments) takes a long time to compute $\mathbf{z}_t$ in the $\mathrm{MCNOracleShell}$ (Algorithm (1)). Therefore in the current evaluation we only sample a small batch of points to test. Specifically, we generate a random permutation of images in the test set, go one by one through the permutation, and *test our method only on test images that the model correctly classifies*. In our experiments, we have evaluated our method on 30 points which the base model classifies correctly.

While this is a definite limitation of our current experiments, we believe that our method has shown some early promise in tackling the important problem of adversarial perturbation. We have made our model and code publicly available in an anonymous repository (cod) for better visibility (in fact our method is also easy to set up once a base model is ready). We hope that our work can trigger more thoughts in exploiting manifold assumption to enhance model robustness.

Finally, we believe that the efficiency issue of an embedding procedure will be eventually resolved once we get the right robustness method. So far we have been trying to control the parameters of CW attack to accelerate the embedding and it already gives significant improvements over the vanilla version (without sacrificing robustness and generalization). Also, there are some faster attack methods that seem to give more reasonable running time, which we plan to integrate into our code. All in all, accelerating embedding while retaining good robustness and generalization seems to be interesting avenue to pursue. We now give more details of our experiments.

Experimental Setup We use CIFAR10 dataset Krizhevsky (2009) for our experiments. CIFAR10 dataset consists of 60000 color images of objects where 50000 are for training and 10000 are for testing. Each image is of size 32×32 pixels, and there are ten labels $\{0, 1, \ldots, 9\}$ for classification. We use a model recently trained by Madry et al. (2017) as base model. We port their model to `Keras` so that the original Carlini-Wagner attack code (cw-) can be directly applied. We do so by using the same architecture and copy the weights of the trained model from Madry et al. (mad).

As we mentioned briefly before, we use Carlini-Wager attacks (CW attack) for two purposes: (1) *as an attack to challenge the defense.* In this case, we use the original attack for attacks that are as strong as possible. We apply *non-targeted* attacks (namely changing classification is the purpose) with L_∞ norm (i.e. try to find a perturbation of minimal L_∞ norm). We enforce a norm bound $\varepsilon = \frac{8}{255}$ and a perturbation is valid if and only if it both changes the classification and its L_∞ norm is within the bound. Note that in the original work of Madry et al. (2017) the norm bound used is $\varepsilon = 8.0$. We use the current bound because we normalize the images when applying the CW attack. (2) *as an embedding procedure for instantiating* MCNOracleShell. We call the resulting shell CarliniWagnerShell[2].

Experimental Methodology We first compute a random permutation of the test images, and then go through the permutation one by one. For one image, we discard it if the base model makes a wrong prediction. Otherwise, we do the following tests: (1) Susceptibility of *the base model* to CW attack at this point. The success/failure of the CW attack on each point is recorded for later comparisons. (2) Robustness. There are two cases: (i) the model is already robust to CW attack where then we expect CarlinWagnerShell retains this robustness, and (ii) the model is vulnerable to CW attack, where we then expect CarliniWagnerShell to harden the model. (3) Generalization. We also measure whether CarliniWagnerShell produces correct prediction on the point that the base model predicts correctly. This reflects whether the model shell retains the generalization.

Experimental results. Among 30 points which the base model classifies correctly, we find the base model is susceptible to CW for 9 of them (thus the attack success rate is 30%). When applying CarliniWagnerShell, we find that the points that were robust with base model remain robust. Moreover, CarliniWagnerShell also successfully defends all the attacks for those 9 vulnerable points where the base model is not robust, thus the attack success rate is reduced to *zero* (we defend all attacks without introducing new vulnerability). Finally, for all these points (where base model is correct), CarliniWagnerShell also makes correct predictions *except for one of them*. Thus the accuracy we retain is 96.67%. These results are summarized in 1.

Table 1: Summary of the experimental results

Base model	CarliniWagnerShell-equipped model	
Attack success rate	Attack success rate	Retained accuracy
30%	0%	96.67%

The wrong data point. We now describe the data point that the model shell gives wrong prediction while the base model is correct. We note that the model gives low confidence ($\sim 50\%$) on the original image (where the correct label is a "cat"), and it has significantly higher confidence ($\sim 75\%$) on the perturbed image (classified as a "frog"). The results are collected in Figure 1, with a frog image for comparison. In this case, it seems that it is the vagueness of the signals/data that leads to a natural difficulty to separate these two images into different manifolds.

6 Discussion

We now discuss ramifications of our postulate, algorithms and encouraging experimental results.

Our technical contributions. Our first technical contribution is the proposal to directly exploit model semantics to defend against adversarial perturbation for neural networks. Our proposal is fundamentally different from previous proposals, which are based on modifying the training objectives, and hoping that the learning algorithm will "learn" to be robust using the new objectives.

[2]In our actual CarliniWagnerShell implementation, we slightly tweak the parameters for CW attacks so that the CW attack used as a defense mechanism is weaker, but faster than the CW attack attacking the base model. This only makes our results *stronger* because we only make the defense weaker.

(a) Original cat image with unconfident prediction "cat"

(b) Adversarially Perturbed cat image with a confident but wrong prediction "frog"

(c) A frog image

Figure 1: The point (a) where our CarliniWagnerShell makes a wrong prediction. We show the original image (a), the adversarially perturbed image (b), and a frog image for comparison (c).

Our second technical contribution is concrete algorithms for the proposed semantics defenses and an empirical evaluation of these algorithms which shows early promise. Most importantly, we show that when a "good" model, such as the residual network trained by Madry et al., is armed with our proposed embedding procedure, it achieves both good generalization and almost perfect robustness.

Interpretations of our results. It is clear that we cannot expect our method to work for arbitrary models (as we need assumptions about what a model has learned). On the other hand, we interpret our results as suggesting that *for good models*, adversarial perturbation can *naturally coexist with good generalization*. In fact, *even if one has learned perfect behaviors over natural manifolds, adversarial perturbation may still exist in abundance*. In hindsight this is somewhat not surprising: if we only train a learner using certain universe, then we cannot expect the learner to have good behavior outside of the universe, and it is easy to fool the learner if it is easy to produce points that are outside of the learner's universe. In this sense, our work can be viewed as a post-processing reduction to reduce points that are "slightly" out of the universe back to it.

We also note that if one is only interested in robustness, then the manifold assumption is not necessary. For robustness all we want from a "good model" is that *it is never confidently wrong on points that it does not learn.* For example, consider the following "good model": If a data point is from the training set, then the model outputs the correct label with confidence 1, otherwise it outputs a uniform distribution over labels. In other words, this model learns nothing but fitting the training set. We note that in this case, the adversarial perturbation problem is *only well defined* around the training points, and, our defense method still works, which is 1-nearest neighbor search among the training points. As a result, the model is still perfectly robust with our method if training points are well separated. Clearly, this model will never generalize, but is only good in the sense that it satisfy our needs for "robustness."

As a result of the discussion so far, we feel that a somewhat surprising part of our work is that it does suggest that one can simultaneously achieve generalization and robustness with our method. This suggests that deep neural networks may have indeed effectively learned some underlying geometric structure, while claiming "not enough confidence" outside the structure learned.

Highly confident predictions on random noises. We note that several work shows that neural networks can have highly confident predictions on random noises (for example, Nguyen et al. (2015)). In view of our work this is again somewhat not surprising after all: These points are essentially ones that are *far from* the universe the learner is asked to learn, and so we do not expect that the learning result can make any sense of those points (and thus it is valid to have "divergent" behaviors there). In fact, we note that the adversarial perturbation problem is not well defined even near those points.

REFERENCES

Anonymous code release for adversarial perturbation defense - GitHub. https://github.com/adversarial-perturbation-defense/anonymous-adversarial-perturbation-defense.

Carlini-Wagner attacks - GitHub. https://github.com/carlini/nn_robust_attacks.

CIFAR10 Adversarial Examples Challenge - GitHub. https://github.com/MadryLab/cifar10_challenge.

What is the manifold assumption in semi-supervised learning. https://stats.stackexchange.com/questions/66939/what-is-the-manifold-assumption-in-semi-supervised-learning.

Ronen Basri and David W. Jacobs. Efficient representation of low-dimensional manifolds using deep networks. *CoRR*, abs/1602.04723, 2016. URL http://arxiv.org/abs/1602.04723.

Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. 1996.

Daniel Bienstock and Alexander Michalka. Cutting-planes for optimization of convex functions over nonconvex sets. *SIAM Journal on Optimization*, 24(2):643–677, 2014.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 39–57, 2017.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Jiajun Lu, Theerasit Issaranon, and David A. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. *CoRR*, abs/1704.00103, 2017. URL http://arxiv.org/abs/1704.00103.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *CoRR*, abs/1704.03976, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2574–2582, 2016.

Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 427–436, 2015.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization, second edition*. World Scientific, 2006.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proceedings of the 1st IEEE European Symposium on Security and Privacy*, Saarbrucken, Germany, 2016a. IEEE.

Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pp. 582–597, 2016b.

Salah Rifai, Yann Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pp. 2294–2302, 2011.

Uri Shaham, Alexander Cloninger, and Ronald R. Coifman. Provable approximation properties for deep neural networks. *CoRR*, abs/1509.07385, 2015. URL http://arxiv.org/abs/1509.07385.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.