On the Robustness of Deep K-Nearest Neighbors

Chawin Sitawarin, David Wagner
EECS Department, UC Berkeley
{chawins, daw}@berkeley.edu

Abstract—**Despite a large amount of attention on adversarial examples, very few works have demonstrated an effective defense against this threat. We examine Deep k-Nearest Neighbor (DkNN), a proposed defense that combines k-Nearest Neighbor (kNN) and deep learning to improve the model's robustness to adversarial examples. It is challenging to evaluate the robustness of this scheme due to a lack of efficient algorithm for attacking kNN classifiers with large k and high-dimensional data. We propose a heuristic attack that allows us to use gradient descent to find adversarial examples for kNN classifiers, and then apply it to attack the DkNN defense as well. Results suggest that our attack is moderately stronger than any naive attack on kNN and significantly outperforms other attacks on DkNN.**

I. Introduction

Deep learning has recently attained immense popularity from various fields and communities due to its superhuman performance on complicated tasks such as image classification [1], [2], playing complex games [3]–[5], controlling driverless vehicles [6], [7], and medical imaging [8]. Nonetheless, many works have shown that neural networks and other machine learning classifiers are not robust in the face of adversaries (e.g. adversarial examples) [9]–[13] as well as more common cases of distribution shifts [14], [15].

This phenomenon raises a call for more robust and more interpretable neural network models. Many defenses against adversarial examples have been proposed; however, most have been broken by adaptive adversaries [16]–[18]. Only a few defenses provide a significant improvement in robustness on toy datasets like MNIST and CIFAR-10 [19], [20]. One plausible approach to simultaneously combat adversaries and make neural networks more trustworthy is to build interpretable models [21]–[23] or to provide an *explanation* supporting the model's output [24]–[26]. Deep k-Nearest Neighbors (DkNN), recently proposed by Papernot & McDaniel, showed promising results: their evaluation suggests it offers robustness against adversarial examples, interpretability, and other benefits [23].

Nonetheless, adversarial examples are surprisingly difficult to detect when that the adversary has full knowledge of the defense [17]. Among the works that have been beaten, many attempts to distinguish adversarial inputs by statistically inspecting their representation (or activation) from hidden layers of neural networks [27]–[30]. This fact raises some concerns for the robustness of DkNN, which uses kNN on the intermediate representations produced by the neural network.

In this paper, we examine the robustness of DkNN against adversarial examples. We develop a new gradient-based attack on kNN and DkNN. While gradient descent has found great success in attacking neural networks, it is challenging to apply to kNN, as kNN is not differentiable. At a high level, our attack approximates the discrete nature of kNN with a soft threshold (e.g., a sigmoid), making the objective function differentiable. Then, we find a local optimum using gradient descent under an ℓ_p-norm constraint. With this attack, we find that DkNN is vulnerable to adversarial examples with a small perturbation in both ℓ_2 and ℓ_∞ norms. With ℓ_∞-norm of 0.2, our attack manages to reduce the accuracy of a DkNN on MNIST to only 17.44%. Some of the adversarial examples generated with our attack are shown in Fig. 1.

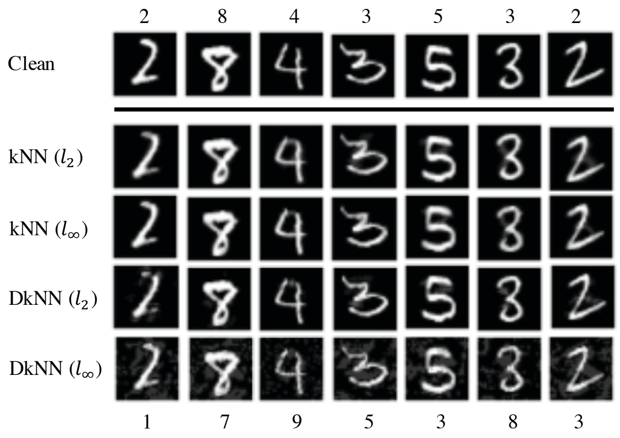

Fig. 1: Adversarial examples generated from the gradient-based attack on kNN and DkNN with ℓ_2- and ℓ_∞-norm constraints. The numbers on top and bottom are predictions of DkNN on the clean and the adversarial samples respectively. For a few adversarial examples, the perturbation might change the human label: some of the adversarial 4's have their top closed, so a human might consider them a 9, and one of the 3's looks close to an 8.

The main contributions of this paper are as follows:

1) We propose a gradient-based attack on kNN and DkNN.
2) We evaluate our attack on kNN and DkNN, compare it to other naive approaches as well as the adaptive attack proposed by Papernot & McDaniel, show that our attack performs better than prior attacks, and show that it can find adversarial examples for kNN and DkNN on MNIST.
3) We show that the credibility scores from DkNN models are not effective for detecting our attacks without a significant drop in accuracy on clean images.

DOI 10.1109/SPW.2019.00014

II. Background and Related Work

A. Adversarial Examples

Adversarial examples are a type of an evasion attack against machine learning models at test time. While the robustness of machine learning classifiers in adversarial settings has been studied for a long time [31], [32], the term "adversarial examples" was recently introduced as an attack on deep neural networks by adding very small perturbation to a legitimate sample [10], [11]. Previous works propose algorithms for finding such perturbation under a norm-ball threat model which can be generalized as solving the following optimization problem:

$$x_{adv} = x + \delta^* \quad \text{where } \delta^* = \underset{\delta}{\arg\max} \quad L(x+\delta) \qquad (1)$$
$$\text{such that } \|\delta\|_p \leq d$$

where L is some loss function associated with the correct prediction of a clean sample x by the target neural network. The constraint is used to keep the perturbation small or *imperceptible* to humans. Our attack also uses the norm-ball constraint and an optimization problem of a similar form.

B. Robustness of k-Nearest Neighbors

The kNN classifier is a popular non-parametric classifier that predicts the label of an input by finding its k nearest neighbors in some distance metric such as Euclidean or cosine distance and taking a majority vote from the labels of the neighbors. Wang et al. recently studied the robustness of kNN in an adversarial setting, providing a theoretic bound on the required value of k such that robustness of kNN can approach that of the Bayes Optimal classifier [33]. Since the required value of k is too large in practice, they also propose a robust 1-NN by selectively removing some of the training samples. We did not experiment with this defense as it is limited to a 1-NN algorithm with two classes.

C. Deep k-Nearest Neighbors

DkNN, proposed by Papernot & McDaniel, is a scheme that can be applied to any deep learning model, offering interpretability and robustness through a nearest neighbor search in each of the deep representation layers. Using *inductive conformal prediction*, the model computes, in addition to a prediction, *confidence* and *credibility* scores, which measure the model's assessment of how likely its prediction is to be correct. The goal is that adversarial examples will have low credibility and can thus be easily detected. The credibility is computed by counting the number of neighbors from classes other than the majority; this score is compared to scores seen when classifying samples from a held-out *calibration set*. Papernot & McDaniel evaluate DkNN with an adaptive adversary which is found to be quite unsuccessful. We examine the robustness of DkNN with the stronger attack we propose.

We note that the DkNN proposed by Papernot & McDaniel uses cosine distance, which is equivalent to Euclidean distance given that all samples are normalized to have a unit norm. For the rest of the paper, we tend to omit the normalization for simplicity and less clutter in equations. The implementation and the evaluation, however, use cosine distance as instructed in the original paper.

III. Threat Model

We assume the white-box threat model for attacks on both kNN and DkNN. More precisely, the adversary is assumed to have access to the training set and all parameters of the DkNN neural network. Since a kNN classifier is non-parametric, the training set is, in some sense, equivalent to the weights of parametric models. We also assume that the adversary knows all hyperparameters, namely k, the distance metric used (Euclidean or cosine distance), and additionally the calibration set for DkNN. Though this knowledge is less crucial to the adversary, it allows the adversary to accurately evaluate his/her attack during the optimization resulting in a more effective attack.

For consistent comparisons with previous literature, the adversarial examples must be contained within a norm-ball (ℓ_2 and ℓ_∞) centered at given test samples. We recognize that the ℓ_p-norm constraint may not be representative of human perception nor applicable in many real-world cases.

IV. Attack on k-Nearest Neighbors

A. Notation

We follow notation from Papernot & McDaniel as much as possible. Let z denote a target sample or a clean sample that the adversary uses as a starting point to generate an adversarial example, and y_z its ground-truth label. We denote the perturbed version of z as $\hat{z}$. The training set for both kNN and DkNN is (X, Y) with n samples of dimension d. The classifier's prediction for a sample x is $\mathsf{knn}(x)$.

B. Mean Attack

We first introduce a simple, intuitive attack to serve as a baseline. Let z be a clean sample, y_z its ground-truth class, and $y_{adv} \neq y_z$ be a target class. The attack, which we call the mean attack, works by moving z in the direction towards the mean of all samples in the training set with class y_{adv}. Concretely, we first search for the class $y_{adv} \neq y_z$ such that the mean of training samples with that class is closest to z in Euclidean distance. Let m denote the corresponding mean. We then use binary search to find the smallest $c > 0$ such that $(1-c)z + cm$ is misclassified by the kNN.

This attack is very simple to carry out and applicable to any classifier. While it is a natural choice for attacking a kNN with Euclidean distance, the attack may perform less well for cosine distance or other distance measures. As our experiments show, the mean attack also produces perturbations that make the resulting adversarial example look, to humans, more like samples from the target class, and thus makes the attack more noticeable. Nonetheless, this attack can be regarded as a simple baseline for measuring the robustness of nearest-neighbor classifiers.

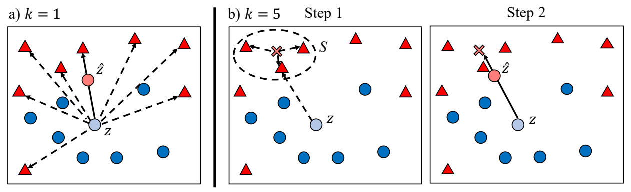

Fig. 2: (a) naive attack for $k = 1$: The target sample z (light blue circle) is moved towards each of the samples from a different class (red triangles). The one that requires the smallest ℓ_2-distance to change the prediction is the optimal adversarial example $\hat{z}$ (pink circle). (b) naive attack for $k > 1$: In the first step, a set S of 3 samples from the different class closest to z are located with a greedy algorithm. The second step involves moving z towards a mean of the samples in S and stops when the prediction changes.

C. Naive Attack

Next, we introduce a second baseline attack that improves slightly on the mean attack. When $k = 1$, a simple algorithm can find the optimal adversarial example in $\mathcal{O}(n)$ time. For each training sample z' of a class other than y_z, the algorithm moves the target sample z in a straight line towards z' until $\mathsf{knn}(\hat{z}) \neq y_z$ (i.e., setting $\hat{z} = (1-c)z + cz'$, we find the smallest $c > 0$ such that $\mathsf{knn}(\hat{z}) \neq y_z$). This produces n candidate adversarial examples, and the algorithm outputs the one that is closest to z. Fig. 2(a) illustrates this algorithm.

This strategy finds the optimal adversarial example when $k = 1$, but when $k > 1$, it is not clear how to find the optimal adversarial example efficiently. Repeating the previous strategy on all sets of k training samples does not guarantee an optimal solution and is inefficient, as its complexity grows exponentially with k. Instead, we propose a computationally cheaper attack that greedily chooses only one set of samples to move towards, as summarized in Fig. 2(b). There are multiple possible heuristics to choose this set. One simple option would be to find the $\lceil \frac{k}{2} \rceil$ nearest neighbors of z whose labels all match but are different from y_z. We instead use a slightly more complex variant: (1) find the nearest neighbor from any class other than y_z, say class y_{adv}, (2) add this sample to an empty set S, and (3) out of all samples with class y_{adv}, iteratively find the nearest sample to the mean of S and add it to S. The final step is repeated until $|S| = \lceil \frac{k}{2} \rceil$. Finally, we move z towards the mean of S until the classifier's prediction differs from y_z.

D. Gradient-Based Attack

Here we introduce our main attack on kNN. On a high-level, it uses a heuristic initialization to choose a set of m samples that are close to the target sample z. Then, a gradient-based optimization is used to move z closer to the ones with the target class y_{adv} and further from the ones with the original class y_z.

We will discuss the choices for the heuristic initialization towards the end of this section. For now, the algorithm can be formulated as the following optimization problem.

$$\hat{\delta} = \underset{\delta}{\arg\min} \sum_{i=1}^{m} w_i \cdot \|x_i - (z+\delta)\|_2^2 \qquad (2)$$
$$\text{such that } \|\delta\|_p \leq \epsilon \text{ and } z + \delta \in [0,1]^d$$

where δ is the perturbation, $\hat{z} = z + \hat{\delta}$ is the adversarial example, $x_1, \ldots, x_m$ are the m training samples selected earlier, and $w_i = 1$ if the label of x_i is y_{adv}, otherwise $w_i = -1$. The first constraint constrains the norm of the perturbation, and the second constraint ensures that the adversarial example lies in a valid input range, which here we assume to be $[0,1]$ for pixel values.

However, Eq. 2 may not achieve what we desire since it treats all x_i equally and does not take into account that for kNN, only the k nearest neighbors contribute to the prediction, while the other training samples are entirely irrelevant. Moreover, the distance to these k neighbors does not matter as long as they are the k closest. In other words, the distance to each of these k neighbors is irrelevant so long as it is under a certain threshold η (where η is the distance to the k-th nearest neighbor). This means that a sample x_i gets a vote if $\|x_i - \hat{z}\|_2 \leq \eta$; otherwise, it gets zero vote. The optimization above does not take this into account.

We show how to adjust the optimization to model this aspect of kNN classifiers. The function that maps $\hat{z}$ to 0 or 1 according to whether x_i gets a vote is not a continuous function and it has zero gradient where it is differentiable, so it poses challenges for gradient-based optimization. To circumvent this problem, we approximate the threshold with a sigmoid function, $\sigma(x) = \frac{1}{1+e^{-\alpha x}}$ where α is a hyperparameter that controls "steepness" (or an inverse of *temperature*) of the sigmoid. As $\alpha \to \infty$, the sigmoid exactly represents the Heaviside step function, i.e., a hard threshold. This lets us adjust Eq. 2 to incorporate the considerations above, as follows:

$$\hat{\delta} = \underset{\delta}{\arg\min} \sum_{i=1}^{m} w_i \cdot \sigma\big(\|x_i - (z+\delta)\|_2 - \eta\big) \qquad (3)$$
$$\text{such that } \|\delta\|_p \leq \epsilon \text{ and } z + \delta \in [0,1]^d$$

Ideally, η should be recomputed at every optimization step, but this requires finding k nearest neighbors at each step, which is computationally expensive. Instead, we fix the value of η by taking the average distance, over all training samples, from each sample to its k-th nearest neighbor.

Choosing the initial m samples. There is no single correct way to initialize the set of m samples. We empirically found that choosing all of them from the same class y_{adv}, and choosing the m training samples of that class that are closest to z, works reasonably well. We choose y_{adv} by computing the distance from z to the mean of all samples of class y, for each y, and taking the class y that minimizes this distance. Other heuristics might well perform better; we did not attempt to explore alternatives in depth, as this simple heuristic sufficed in our experiments. The choice of the attack parameter m affects the attack success rate. A larger m means we consider more training samples which make the kNN more likely to be fooled, but it is also more expensive to compute and may produce larger distortion. In principle, one could recompute the set of m samples periodically as the optimization progresses, but for our experiments, we select them only once in the beginning.

For $p = \infty$, we use a change of variable as introduced by Carlini & Wagner [34] to provide pixel-wise box constraints that simultaneously satisfy both of the optimization constraints in Eq. 3. More precisely, the i-th pixel of the adversarial example is written as $\hat{z}_i = \frac{1}{2}(\tanh(v_i) + 1) \cdot (b_u - b_l) + b_l$ where b_u and b_l are the upper and the lower bound of that pixel respectively. v becomes the variable that we optimize over, but for simplicity, we omit it from Eq. 3. In the case of $p = 2$, this change of variables enforces the second constraint. The first constraint is relaxed and added to the objective function as a penalty term:

$$\begin{aligned}\hat{\delta} \ &= \arg\min_{\delta} \ \sum_{i=1}^{m} w_i \cdot \sigma\big(\|x_i - (z+\delta)\|_2 - \eta\big) \\ &\qquad\qquad + c \cdot \max\left\{0,\ \|\delta\|_2^2 - \epsilon^2\right\} \\ \text{such that} \quad & z + \delta \in [0,1]^d\end{aligned} \tag{4}$$

To find an appropriate value for c, we use a binary search for five steps. If the attack succeeds, c is increased; otherwise, c is decreased.

V. Attack on Deep k-Nearest Neighbors

A. Notation

Let $\mathsf{dknn}(x)$ denote DkNN's prediction for a sample x. The prediction of the l-layer neural network part of the DkNN is denoted as $f(x)$, and the output from the λ-th layer as $f_\lambda(x)$ where $\lambda \in \{1, 2, ..., l\}$. The calibration set (X^c, Y^c) is used to calculate the empirical p-value as well as the credibility and confidence.

B. Mean Attack

The mean attack for DkNN is exactly the same as for kNN without any modification as the attack does not depend on the choice of classifiers.

C. Baseline Attack

We use the adaptive attack evaluated by Papernot & McDaniel as a baseline. Given a target sample z, we try to minimize the distance between its representation at the first layer and that of a *guide sample* x_g, a sample from a different class whose representation is closest to $f_1(z)$. For the ℓ_∞-norm constraint, the attack can be written as:

$$\begin{aligned}\hat{\delta} \ &= \arg\min_{\delta} \ \ \|f_1(x_g) - f_1(z+\delta)\|_2^2 \\ \text{such that} \quad & \|\delta\|_\infty \leq \epsilon \text{ and } z + \delta \in [0,1]^d\end{aligned} \tag{5}$$

The optimization is solved with L-BFGS-B optimizer as suggested in Sabour et al. [35]. For completeness, we will also evaluate the attack with a ℓ_2 constraint, using the same relaxation as Eq. 4.

D. Gradient-Based Attack

The baseline attack relies on an assumption that if $f_1(\hat{z})$ is close to $f_1(x_g)$, then $f_\lambda(\hat{z})$ will also be close to $f_\lambda(x_g)$ for $2 \leq \lambda \leq l$, resulting in both $\hat{z}$ and x_g having a similar set of neighbors for all of the layers as well as the final prediction. However, while this assumption makes intuitive sense, it can be excessively strict for generating adversarial examples. The adversary only needs a large fraction of the neighbors of $\hat{z}$ to be of class y_{adv}. By extending the gradient-based attack on kNN, we formulate an analogous optimization problem for attacking DkNN as follows:

$$\begin{aligned}\hat{\delta} &= \arg\min_{\delta} \sum_{i=1}^{m} \sum_{\lambda=1}^{l} \ w_i \cdot \sigma\big(\|f_\lambda(x_i) - f_\lambda(z+\delta)\|_2 - \eta_\lambda\big) \\ &\text{such that } \|\delta\|_p \leq \epsilon \text{ and } z + \delta \in [0,1]^d\end{aligned} \tag{6}$$

The m samples are chosen similarly to the attack on kNN. In the interest of space, we omit the formulation for the ℓ_2 constraint as it is also analogous to Eq. 4.

VI. Experimental Setup

We reimplement DkNN from Papernot & McDaniel with the same hyperparameters, including the network architecture and the value of $k = 75$. We evaluate our attacks on the MNIST dataset [36] as past research suggests that finding adversarial examples on other tasks is even easier. 60,000 samples are used as the training samples for kNN, DkNN, as well as the neural network part of DkNN. 750 samples (75 from each digit) are held out as the calibration set, leaving 9,250 test samples for evaluating the accuracy and the robustness of the classifiers against the attacks. Similarly to Papernot & McDaniel, for a quick nearest neighbor search on DkNN, we use a locality-sensitive hash (LSH) from the FALCONN Python library, which is based off cross-polytope LSH by Andoni et al. [37]. kNN uses an exact neighbor search without any approximation. The kNN and the DkNN have an accuracy of 95.74% and 98.83% on the clean test set, respectively. The neural network alone has an accuracy of 99.24%.

All of the attacks are evaluated under both ℓ_2- and ℓ_∞-norm constraints, except for the naive attack on kNN and the mean attacks. For simplicity, we only evaluate untargeted attacks. Both the mean and the naive attacks use only five binary search steps. For the other attacks, we use 400 iterations of gradient updates and five steps of binary search on the ℓ_2-penalty constant. The Adam optimizer is used in the gradient-based attack, and to save computation time, we only check for the termination condition (i.e., whether $\hat{z}$ is misclassified) three times at iterations 320, 360, and 400, instead of at every step.

TABLE I: Evaluation of all the attacks on kNN.

Attacks	Accuracy	Mean Distortion in ℓ_2
Clean Samples	0.9574	-
Mean Attack	**0.0589**	**8.611**
Naive Attack	0.7834	8.599
Gradient Attack (ℓ_2)	**0.0989**	**6.565**
Gradient Attack (ℓ_∞)	0.8514	5.282

TABLE II: Evaluation of all the attacks on DkNN.

Attacks	Accuracy	Mean Dist.	Mean Cred.
Clean Samples	0.9883	-	0.6642
Mean Attack	0.1313	4.408	0.0172
Baseline Attack (ℓ_2)	0.1602	3.459	0.0185
Baseline Attack ($\ell_\infty = 0.2$)	0.8891	2.660	0.0807
Baseline Attack (fixed ℓ_2)	0.5004	3.435	0.1385
Gradient Attack (ℓ_2)	**0.0000**	**2.164**	0.0482
Gradient Attack ($\ell_\infty = 0.2$)	0.1744	3.476	0.1037
Gradient Attack (fixed ℓ_2)	0.0059	3.375	**0.3758**

We made minimal effort to select hyperparameters. We fix the steepness α of the sigmoid at 4, and for DkNN, we arbitrarily choose the initial m samples to be the k training samples with class y_{adv} whose first-layer representation is closest to that of z. For the ℓ_2-norm attacks, ϵ is simply chosen to be 0 with the constant c being 1. This choice of penalty generally allows the optimization to find adversarial examples most of the time but may result in unnecessarily large perturbations. To set a more strict constraint, one could set ϵ to a desired threshold and c to a very large number.

VII. Results

A. k-Nearest Neighbors

Table I displays the accuracy and mean ℓ_2 distortion of the successful adversarial examples for kNN. As expected, the mean attack is very good at finding adversarial examples but the perturbation is large and the adversarial examples sometimes introduce anomalies that may be noticeable to humans. Surprisingly, the naive attack performs much more poorly compared to the mean attack, indicating that the heuristic used to choose the set of target samples can significantly affect the attack success rate. The gradient-based attack with the ℓ_2-norm performs well and is on par with the mean attack while having considerably smaller mean distortion. On the other hand, the gradient attack with ℓ_∞-norm of 0.2 is mostly unsuccessful. We speculate this might be because $\epsilon = 0.2$ is too small and the ℓ_∞-norm is an ineffective choice of norm as kNN relies on Euclidean distance in the pixel space for prediction.

B. Deep k-Nearest Neighbors

Table II compares the accuracy, mean ℓ_2 distortion, and mean credibility of the successful adversarial examples for DkNN between the three attacks. Our novel gradient-based attack outperforms the baseline as well as the mean attack by a significant margin. With an ℓ_∞-norm constraint of 0.2, the gradient attack reduces the classifier's accuracy much further compared to the baseline. With an ℓ_2-norm constraint, our gradient attack also performs better with smaller perturbation. Although the mean attack reduces the accuracy even lower than the gradient attack with ℓ_∞-norm of 0.2, it has lower mean credibility and the perturbation is also considerably larger and more visible to humans.

Unlike an ℓ_∞ constraint, which is strictly enforced by the change of variables trick, an ℓ_2 constraint is written as a penalty term with only a tunable weighting constant. To compare the baseline and the gradient attacks under a similar ℓ_2-norm, we arbitrarily set ϵ to be the mean ℓ_2-norm of the ℓ_∞ gradient attack (3.476) and the constant c to be just high enough that the optimization still finds successful attacks with a minimal violation on the constraint $\|\epsilon\|_2 \leq 3.476$. We report the results for both attacks in Table II on the "fixed ℓ_2" rows. The gradient attack, when given a large ℓ_2 budget, can increase the credibility significantly and reduce the accuracy to almost zero (0.6%). In contrast, the baseline attack can only find adversarial examples for about 50% of the samples under the same ℓ_2 constraint.

Fig. 3 shows a clean sample and its adversarial versions generated by all of the attacks along with their five nearest neighbors at each of the four layers of representation. On the first column, all of the 20 neighbors of the clean sample have the correct class (a six). On the other hand, the majority of neighbors of the adversarial examples are of the incorrect class (a five) with an exception of the first layer whose neighbors generally still come from the correct class. The other common property of all the attacks is that almost every neighbor in the final layer has the adversarial class.

Note that the ℓ_2-attacks, both the baseline and the gradient-based attack, often perturb the sample in a semantically meaningful manner. Most are subtle, but some are quite prominent. For instance, the input of the third column from the left in Fig. 3 is perturbed by slightly removing the connected line that distinguishes between a five and a six, making the adversarial example appear somewhat ambiguous to humans. In contrast, the ℓ_∞ adversarial examples usually spread the perturbation over the entire image without changing its semantic meaning in a way that is noticeable to humans.

For the ℓ_∞-norm constraint, as we increase ϵ, the accuracy of DkNN drops further and hits zero at $\epsilon = 0.3$, as shown in Fig. 4(a), whereas increasing ϵ on the baseline attack reduces accuracy at a much slower rate.

Fig. 4(b) displays the mean credibility of successful adversarial examples generated from the baseline and the gradient attacks. As expected, as we increase ϵ, the mean credibility also increases for both attacks because the adversarial example can move closer to training samples from the target class. The gradient-based attack increases the mean credibility at a much faster rate than the baseline potentially because its objective function indirectly corresponds to the credibility as it takes into account m training samples instead of one like the baseline. In the next section, we discuss the possibility of detecting adversarial examples by setting a threshold on the credibility score.

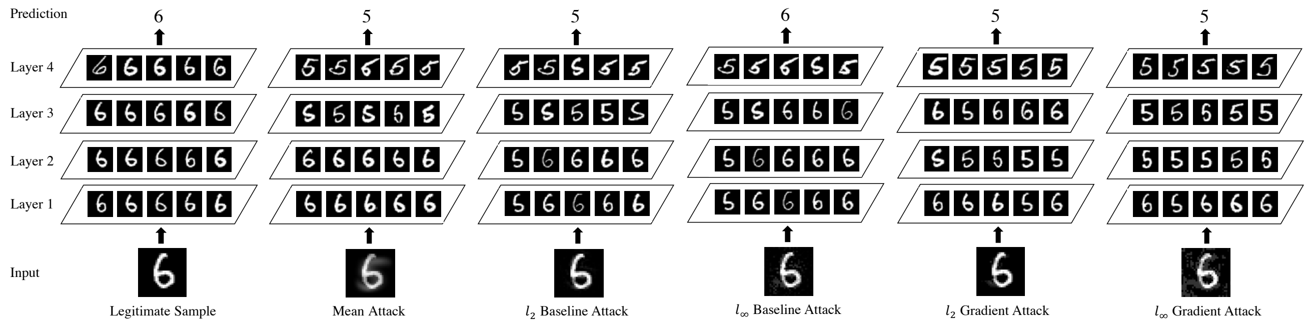

Fig. 3: Each column shows five nearest neighbors for each of the four deep representation spaces of DkNN. From left to right, the inputs are a randomly chosen legitimate sample, its ℓ_2 and ℓ_∞ baseline attacks, and its ℓ_2 and ℓ_∞ gradient attacks. For the ℓ_∞-norm constraint, ϵ is 0.2. The legitimate sample is correctly predicted by the DkNN, and all of the attacks succeed in changing the prediction from a six to a five, except for the ℓ_∞ baseline attack.

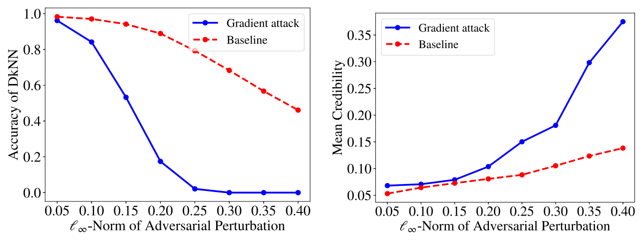

Fig. 4: (a) Accuracy and (b) mean credibility of DkNN under the baseline attack and our gradient-based attack at different ℓ_∞-norm constraints.

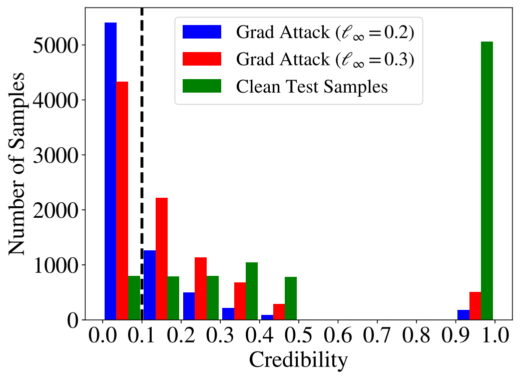

Fig. 5: Histogram of credibility of the clean test samples and the adversarial examples generated from the gradient-based attack with the ℓ_∞-norm constraint of 0.2 and 0.3. The black dashed vertical line indicates credibility of 0.1.

VIII. Discussion

A. Credibility Threshold

Papernot & McDaniel argues that the credibility output by DkNN is a well-calibrated metric for detecting adversarial examples. In Fig. 5, we show the distribution of the credibility for the clean test set and for adversarial examples generated from the gradient-based attack with two different ℓ_∞-norms. Most of the test samples (around 55%) have credibility between 0.9 and 1. On the other hand, the majority of the adversarial examples have credibility less than 0.1, suggesting that setting a threshold on credibility can potentially filter out most of the adversarial examples. However, doing so comes at a cost of lowering accuracy on legitimate samples. Choosing a credibility threshold of 0.1 reduces accuracy on the test set to 91.15%, which is already very low for MNIST, and with this threshold, 28% and 43% of the adversarial examples with ℓ_∞-norm of 0.2 and 0.3 respectively still pass the threshold and would not be detected. It is also important to note that our attack is not designed to maximize the credibility. Rather, it is designed to find adversarial examples with minimal distortion. Simple parameter fine-tuning, e.g. a larger m, more iterations, and a smaller η, might all help increase the credibility.

Our experiments suggest that DkNN's credibility may not be sufficient for eliminating adversarial examples, but it is still a more robust metric for detecting adversarial examples than a softmax score of typical neural networks. Unfortunately, thresholding the credibility hurts accuracy on legitimate examples significantly even for a simple task like MNIST. According to Papernot & McDaniel, the SVHN and GTSRB datasets both have a larger fraction of legitimate samples with low credibility than MNIST, making a credibility threshold even less attractive. Experiments with the ImageNet dataset, deeper networks, choosing which layers to use, and pruning DkNN for robustness are all interesting directions for future works.

IX. Conclusion

We propose two heuristic attacks and a gradient-based attack on kNN and use them to attack DkNN. We found that our gradient attack performs better than the baseline: it generates adversarial examples with a higher success rate but lower distortion on both ℓ_2 and ℓ_∞ norms. Our work suggests that DkNN is vulnerable to adversarial examples in a white-box adversarial setting. Nonetheless, DkNN still holds promise as a direction for providing significant robustness against adversarial attacks as well as interpretability of deep neural networks.

X. Acknowledgements

This work was supported by the Hewlett Foundation through the Center for Long-Term Cybersecurity and by generous gifts from Huawei and Google.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017. [Online]. Available: http://arxiv.org/abs/1712.01815

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[7] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[8] G. J. S. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *CoRR*, vol. abs/1702.05747, 2017. [Online]. Available: http://arxiv.org/abs/1702.05747

[9] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases*, H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 387–402.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013. [Online]. Available: http://arxiv.org/abs/1312.6199

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[12] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *arXiv preprint arXiv:1511.04599*, 2015.

[13] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 427–436.

[14] L. Engstrom, D. Tsipras, L. Schmidt, and A. Madry, "A rotation and a translation suffice: Fooling cnns with simple transformations," *CoRR*, vol. abs/1712.02779, 2017. [Online]. Available: http://arxiv.org/abs/1712.02779

[15] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and surface variations," *CoRR*, vol. abs/1807.01697, 2018. [Online]. Available: http://arxiv.org/abs/1807.01697

[16] N. Carlini and D. Wagner, "Defensive distillation is not robust to adversarial examples," *arXiv preprint arXiv:1607.04311*, 2016.

[17] ——, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. New York, NY, USA: ACM, 2017, pp. 3–14. [Online]. Available: http://doi.acm.org/10.1145/3128572.3140444

[18] A. Athalye, N. Carlini, and D. A. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *CoRR*, vol. abs/1802.00420, 2018. [Online]. Available: http://arxiv.org/abs/1802.00420

[19] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *CoRR*, vol. abs/1706.06083, 2017. [Online]. Available: http://arxiv.org/abs/1706.06083

[20] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *CoRR*, vol. abs/1704.01155, 2017. [Online]. Available: http://arxiv.org/abs/1704.01155

[21] J. Kim and J. F. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," *CoRR*, vol. abs/1703.10631, 2017. [Online]. Available: http://arxiv.org/abs/1703.10631

[22] Q. Zhang, Y. N. Wu, and S. Zhu, "Interpretable convolutional neural networks," *CoRR*, vol. abs/1710.00935, 2017. [Online]. Available: http://arxiv.org/abs/1710.00935

[23] N. Papernot and P. D. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," *CoRR*, vol. abs/1803.04765, 2018. [Online]. Available: http://arxiv.org/abs/1803.04765

[24] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *CoRR*, vol. abs/1312.6034, 2013. [Online]. Available: http://arxiv.org/abs/1312.6034

[25] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," *CoRR*, vol. abs/1602.04938, 2016. [Online]. Available: http://arxiv.org/abs/1602.04938

[26] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "Lemna: Explaining deep learning based security applications," in *ACM Conference on Computer and Communications Security*, 2018.

[27] D. Hendrycks and K. Gimpel, "Visible progress on adversarial images and a new saliency map," *CoRR*, vol. abs/1608.00530, 2016. [Online]. Available: http://arxiv.org/abs/1608.00530

[28] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," *CoRR*, vol. abs/1612.07767, 2016. [Online]. Available: http://arxiv.org/abs/1612.07767

[29] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *CoRR*, vol. abs/1703.00410, 2017.

[30] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. D. McDaniel, "On the (statistical) detection of adversarial examples," *CoRR*, vol. abs/1702.06280, 2017. [Online]. Available: http://arxiv.org/abs/1702.06280

[31] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, 2006, pp. 16–25.

[32] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and Artificial Intelligence*. ACM, 2011, pp. 43–58.

[33] Y. Wang, S. Jha, and K. Chaudhuri, "Analyzing the robustness of nearest neighbors to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 5133–5142. [Online]. Available: http://proceedings.mlr.press/v80/wang18c.html

[34] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017.

[35] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," *CoRR*, vol. abs/1511.05122, 2015. [Online]. Available: http://arxiv.org/abs/1511.05122

[36] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[37] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, "Practical and optimal lsh for angular distance," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 1225–1233. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969239.2969376