Towards Breaking Deep k-Nearest Neighbours

Vedika Agarwal
2571203
Saarland University
s8veagar@stud.uni-saarland.de

Khushboo Mehra
2576512
Saarland University
s8khmehr@stud.uni-saarland.de

Chirag Bhuvaneshwara
2571703
Saarland University
s8chbhuv@stud.uni-saarland.de

Abstract

Deep neural networks (DNNs) have performed really well in wide-ranging applications from image recognition to machine translation and many more. However, these models often break in adversarial settings thereby questioning the robustness of the model. Furthermore these models are criticized for their general inability to rationalize its predictions. This calls for robust and interpretable models. Papernot et al.[6] take a first step in this direction and introduce the Deep k-Nearest Neighbors (DkNN). This hybrid classifier combines the k-nearest neighbors algorithm with representations of the data learned by each layer of the DNN. They show that the model is robust to popular gradient-based adversarial attacks. In this report, created adversarial examples that target the kNN classifiers underlying the DkNN. and found that the model is sensitive to certain attacks and can the defence be broken in both black-box and white-box settings.

1. Introduction

Human interpretability is not always given importance when designing machine learning algorithms but in [6], Papernot et al. combine the idea of neural networks and nearest neighbours approach to evaluate the prediction, credibility and confidence of a model. To calculate these metrics the data must be first divided into training set, testing set and calibration set. The DNN is trained on the training set. The layer outputs of the DNN are obtained for the training set and indexed using the Locality-Sensitive Hashing (LSH) method in [1]. Papernot et al. [6] employs the calibration set to obtain the expected k training point neighbors for the calibration data that is known to belong to the same distribution as the training set. Each of the layer outputs of the points from the testing set are found to be similar to k number of layer outputs of the training datapoints.

When adversary attacks the model, it changes the distribution of the features and the DNN is fooled. Here since we are comparing the k-neighbours found at every layer, the robustness of the DkNN algorithm increases. Adversarial attacks might change the layer output for a particular test input only for certain layers of a DNN model. This is used by the above comparison to signify if the test points are similar to the calibration points giving the credibility metric. The prediction metric is defined for a test input as the most frequently appearing neighbor at all the layers of a DNN. The confidence metric is to signify how confident the DkNN algorithm is and it is defined as the inverse of the label with the second highest probability of occurrence for a test input. Since we aim to break the model, we first reproduce the model and attack it by using adversarial examples generated using Fast Gradient Sign Method [2], Jacobian-based Saliency Map Attack [10] and Spatial Transformation Method [11]. Lastly, we also see how the calibration set could be modified and sensitivity of the model accuracy to the values in the calibration set. We use MNIST dataset [4] for our experimentation and report the accuracies in the report.

2. Deep k-Nearest Neighbour

In this section, we present an overview of the adversarial defence introduced by Papernot et. al. [6]. We will briefly discuss how it differs from other defence techniques, its performance and particular advantages.

2.1. Model Overview

The Deep k-Nearest Neighbours (DkNN) is a hybrid classifier that applies the k-nearest neighbours algorithm on the high-dimensional representations learnt by the hidden layers of a deep neural network (DNN). Compared to existing adversarial defence techniques, this approach is different in that it addresses the real reason deep models perform poorly on adversaries. Small changes in the adversarial inputs (which are often not visible to the human eye) are gradually magnified by the non-linearilites applied in each layer. The authors show that DNNs misclassify when a hidden layer transforms the input representation into one that is closer to the incorrect class. Instead, a DkNN finds the neighbouring training points in the embedding space of each layer. When prediction between layers changes, the labels of the nearest neighbours can be analysed to get the true label. This is crucial to the defence in two ways. First, it enforces conformity of test predictions w.r.t. training data, and second, it ensures consistency of the final output with each intermediary layer. Thus, the DkNN algorithm removes exploitable degrees of freedom available to gradient based attacks and exploits the structure of deep learning to achieve a more robust and interpretable model.

2.2. Evaluation Metrics

Softmax probability is not a reliable indicator of prediction confidence. In practice, a DNN is often more confident when it predicts the wrong class [8]. Papernot et. al. propose new metrics for assessing model integrity based on *non-conformity*, which is a measure of how different a labelled input is from previous observations. The idea behind using this is that classification must be supported by training data. The new evaluation metrics are explained below.

2.2.1 Credibility

A measure of whether the prediction agrees with the nearest neighbours in every layer. Credibility is calculated using the the empirical p-value of the prediction, defined as the fraction of non-conformity measure for calibration data that is larger than that of the test case. Low credibility implies less evidence, which enables the model to identify adversarial input.

2.2.2 Confidence

This score is an indicator of how likely the prediction is to be correct. It is defined as the non-conformity of the second most likely prediction.

3. Method

3.1. DkNN Implementation

We implement the model as described by Papernot et. al. [6]. The model details for DNN used in the algorithm are given in the Table 1. Parameters for convolution are in the following order: filters, kernel shape, strides, and padding.

Layer	Layer Paramters
1. Conv 1	64 filters, (8x8), (2x2), same
2. Conv 2	128 filters, (6x6), (2x2), valid
3. Conv 3	128 filters, (5x5), (1x1), valid
4. Linear	10 units

Table 1. DNN Architecture details

The algorithm used for DkNN is built up on this DNN. Given a test input *x*: we run the input *x* through the DNN to obtain the layer-wise representations (for each layer in the DNN). For each of these representations, a nearest neighbour classifier based on locality-sensitive hashing is used to find the *k* training points whose representations at the layer are closest to the one of the test input. In our experimentation, we use $k = 75$ as given in [6]. Following which, for each layer, we find the labels of *k* inputs with reference to the training set. Finally all these *y* labels found are used to compute the prediction of DkNN according to the framework of conformal prediction. The entire algorithm is depicted in Figure 1.

Algorithm 1 – Deep k-Nearest Neighbor.

Input: training data (X, Y), calibration data (X^c, Y^c)
Input: trained neural network f with l layers
Input: number k of neighbors
Input: test input z

// Compute layer-wise k nearest neighbors for test input z
for each layer $\lambda \in 1$ .. l **do**
  $\Gamma \leftarrow k$ points in X closest to z found w/ LSH tables
  $\Omega_\lambda \leftarrow \{Y_i : i \in \Gamma\}$ ▷ Labels of k inputs found
end for
// Compute prediction, confidence and credibility
$A = \{\alpha(x, y) : (x, y) \in (X^c, Y^c)\}$ ▷ Calibration
for each label $j \in 1..n$ **do**
  $\alpha(z, j) \leftarrow \sum_{\lambda \in 1..l} |i \in \Omega_\lambda : i \neq j|$ ▷ Nonconformity
  $p_j(z) = \frac{|\{\alpha \in A : \alpha \geq \alpha(z,j)\}|}{|A|}$ ▷ empirical p-value
end for
prediction $\leftarrow \arg\max_{j \in 1..n} p_j(z)$
confidence $\leftarrow 1 - \max_{j \in 1..n, j \neq prediction} p_j(z)$
credibility $\leftarrow \max_{j \in 1..n} p_j(z)$
return prediction, confidence, credibility

Figure 1. **Deep k-Nearest Neighbour Algorithm [6]**

The DNN accuracy is 0.9895 whereas the DkNN gives an accuracy of 0.9889 with better interpretabilty. Next we

report the accuracies obtained by both of the models using FGSM[2] and Basic Iterative Method (BIM)[3]. The numbers are reported in 4. We observe that the DkNN does really well when attacked by FGSM and BIM as compared to the DNN. We note that these numbers are not same as the ones reported in [6]. This might be due to the different parameters these attacks take into account and the authors had not specified all the parameters for the attacks.

3.2. Adversarial attacks on DkNN

The authors showed that standard gradient based attacks (except iterative attacks) do not work on the DkNN model. In order to break the defence, we need to target the underlying kNN classifiers. One way of approaching this is to force the kNN to have many labels from the wrong class. This can be achieved by encoding characteristics of the wrong class in such a way that the adversarial input's hidden layer representation (especially for lower layers) closely aligns with that of the nearest neighbours from the wrong class. This would ensure that the credibility remains reasonably high for the misclassifications. Another approach for adversarial attacks is where the alpha scores from the calibration set are accessible to the attacker. This can be used to find the second most likely prediction. By directly targeting the nonconformity scores, credibility can remain high even when the prediction is not supported by training data.

4. Experiments

In order to break the model, we conduct a series of different experiments. We attack the model in both black-box and white-box settings. The following subsections include the details of the attacks made. We mainly focus on different adversarial perturbations and how could these be used in a way that it also affects the Locality Sensitive Hashing method used to predict the nearest neighbours. We also study the influence of the calibration set, A on the model prediction.

Attack	Attack Paramters	DNN	DkNN
FGSM	$\epsilon = 0.25$	0.011	0.3938
BIM	$\epsilon = 0.25, \alpha = 0.01, i = 100$	0.0081	0.0248
STM		0.0102	0.0007
JSMA1	targets: random	0.0842	0.5581
JSMA2	targets: from DkNN	0.2032	0.6043

Table 2. **Adversarial example classification accuracy for DNN and DkNN. Any parameters not listed above were unchanged from default implementation in [5].**

4.1. Spatial Transformation

We evaluate the DkNN's robustness to black-box attacks, using an targeted adversarial attack based on spatial transformation[11]. Instead of directly manipulating pixel values, the authors generated samples generated by optimising the per-pixel displacement field vector. They showed that examples produced using this method are perceptually realistic and are more difficult to defend with existing defense system. To carry out the attack, we use the implementation provided in cleverhans library[5], with the default parameters.

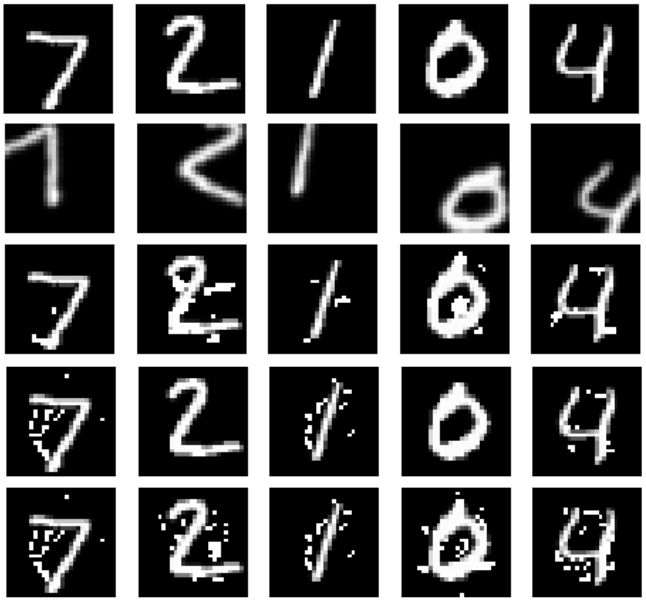

Figure 2. Examples of original (row 1) and adversarial images generated on MNIST dataset for each experiment: STM (row 3), JSMA with random target class (row 3), JSMA with nearest (row 4) and farthest (row 5) neighbour as target class.

Result

The attack is able to break the defence on the DkNN, with an accuracy of 0.07%, which is marginally lower compared to the DNN alone (Table 3). Visualisation of the adversarial examples shows smooth image deformation (row 2 in Fig. 2). Although the adversaries look quite different from the benign input, the true label is easily discernible to the human eye.

4.2. Breaking the model by varying A

The DkNN accuracy is dependent on prediction which is dependent on the closest labels predicted by the network and the calibration set **A**. The set **A** stores the nonconformal scores with respect to true labels for a hold out set of 750 from the test set which is not used for evaluation. So for a given test input z, I calculate the non-conformal scores for every class (*0 to 9 here* against the closest labels predicted. Following which, for every class, a prediction score is assigned by checking the times the label score was lesser than the the scores in A. Here we observe that

the highest-likely class has the least non-conformity, hence there are many values in A which are bigger than the actual class score- and has a bigger number assigned to it, which in turn calls for the prediction being made. In our experimentation we find that *A* has all the values ranging from 0 to 292 with a mean value of 17.29. So we enforce all the values in A to these three values and report the results in the Table3

A values	DkNN Accuracy
1. standard, A	0.9889
2. max(A)	0.7504
3. mean(A)	0.5848
4. min(A)	0.1801

Table 3. Model Sensitivity to A

4.3. Leveraging kNN for gradient based attacks

We experiment whether information about the nearest neighbours provided from the kNN classifiers can be leveraged to construct adversaries that use gradient based optimisation. We use a targeted, white-box attack: JSMA[10] with two variants- one, where the target class is randomly chosen, and the second, where the attacker uses predictions made by the kNN classifier. Based on work on generating adversarial examples for kNNs[9], we hypothesize that instead picking the target which is the nearest neighbour of the input in the embedding space of the DNN (but has a different label), will a generate stronger adversary. To maintain high credibility for the predictions, we pick the 2nd most likely labels predicted by the DkNN on the test set (which has an accuracy of 12.6%) as the target classes. Additionally, we also created an attack where the targets were picked as the farthest neighbour (i.e., least likely label from the DkNN). We expect this JSMA variant to misclassify, but with lower credibility since the non-conformity values will be high for these targets.

Result

Contrary to our hypothesis, both the DNN and DkNN perform significantly better on the targeted adversaries(Table 3). Misclassification rate of the DNN was lower by 10% when the second prediction from the DkNN was used as the target as compared to the last prediction. Analysis of adversarial images showed fewer perturbations in both variants compared to standard JSMA. While we expected adversaries that are neighbours in the embedding space to confuse the model, the similar activations actually helped the DkNN to recover the true class of the input. These results, along with those reported by the authors on adversarial examples generated using hidden layer activations (methodology based on [7]), suggest that the underlying kNN classifiers are robust to any attacks that use information from the DNN embeddings to construct the adversarial images.

5. Conclusion and Future Work

Our experiments confirm the human interpretability of the 3 metrics from the DkNN algorithm. But the interpretability is accurate only when the data fed into the DNN is either from the same distribution as the training data or when the adversarial attacks are designed only against the DNN. In both the black-box and white-box settings, it is possible to break both the DNN and the kNN classifiers and bring down the performance of the DkNN algorithm. It is important to note that, the DkNN algorithm does not provide good results against heavily perturbed images. But we believe that the human interpretability of the DkNN algorithm is important and should be improved to ensure good performance in a white box setting.

References

[1] A. Andoni, P. Indyk, T. Laarhoven, I. P. Razenshteyn, and L. Schmidt. Practical and optimal LSH for angular distance. *CoRR*, abs/1509.02897, 2015.

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

[3] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016.

[4] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[5] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[6] N. Papernot and P. McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

[7] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015.

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

[9] Y. Wang, S. Jha, and K. Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. *arXiv preprint arXiv:1706.03922*, 2017.

[10] R. Wiyatno and A. Xu. Maximal jacobian-based saliency map attack. *CoRR*, abs/1808.07945, 2018.

[11] C. Xiao, J. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially transformed adversarial examples. *CoRR*, abs/1801.02612, 2018.