A Report on Deep k-Nearest Neighbors

Buhua Liu
Department of Computer Science
Hong Kong Bapatist University
Hong Kong SAR, China
csbhliu@comp.hkbu.edu.hk

I. Introduction

Deep neural networks (DNNs) have demonstrated impressive performance on various machine learning tasks, such as object recognition, object detection, image generation, machine translation, etc. However, the reliability of deep learning is often called into question due to the black-box impression. Specifically, deep learning generally cannot rationalize its predictions, in other words, it does not answer the question of why. What is worse is that the existence of adversarial examples, i.e., carefully crafted inputs intended to fool the DNNs, reveals the lack of robustness of deep learning models in adversarial settings. Papernot and McDaniel [1] introduce the *Deep k-Nearest Neighbors* (DkNN), exploiting the structure of deep learning models to enable new learning-based inference and decision strategies. Combining the k-nearest neighbors algorithm with representations learned by each layer of the DNNs, DkNN as a hybrid classifier can achieve desirable properties such as robustness and interpretability. For a test input, compare it to its neighboring training points according to the distance in the representation space of each layer. The distribution of labels of these neighboring points can be a better confidence estimate than conventional softmax probabilities, especially for inputs outside the model's training manifold, including malicious inputs like adversarial examples, thus providing protection against inputs that are beyond the model's understanding. The authors argues that this is because the nearest neighbors can be used to estimate the nonconformity of, i.e., the lack of support for, a prediction in training data. In addition, due to the explainable nature of k-nearest neighbors algorithm, these neighbors also make the predictions of DkNN human-interpretable.

II. Problem Definition

Despite the breakthroughs they have enabled, the adoption of deep neural networks (DNNs) in security and safety critical applications remains limited in part because they are often considered as *black-box* models whose performance is not entirely understood and are controlled by a large set of parameters-modern DNN architectures are often parameterized with over a million values. An essential part of the design philosophy of DNNs is to learn a modular model whose components (layers of neurons) are simple in isolation yet powerful and expressive in combination due to their orchestration as a composition of non-linear functions. There are three well-identified criticisms directly relevant to the security of deep learning: the lack of reliable *confidence* estimates, model *interpretability* and *robustness*.

III. Methodology

Deep neural networks are designed to learn a hierarchical set of representations of the input domain. Harnessing this intrinsic modularity of deep learning, Papernot and McDaniel [1] introduce the *Deep k-Nearest Neighbors* (DkNN) classification algorithm, which enforces conformity of the predictions made by a DNN on test inputs with respect to the model's training data. Specifically, for each layer in the DNN, the DkNN performs a nearest neighbor search to find training points for which the layer's output is closest to the layer's output on the test input of interest. The labels of these neighboring training points can be analysed to ensure that the intermediate computations performed by each layer remain conformal with the final model's prediction. In this way, confidence can be viewed as estimating the distance between the test input and the model's training points, interpretability is achieved by finding points on the training manifold supporting the prediction; robustness is achieved when the prediction's support is consistent across the layers of the DNN, i.e., prediction with high confidence. In order to maintain the integrity of the model, the authors create a novel characterization of confidence, called *credibility*, that spans the hierarchy of representations within a DNN: any credible classification must be supported by evidence from the training data. In contrast, a lack of credibility indicates that the sample must be ambiguous or adversarial. The pseudo-code for the Deep k-Nearest Neighbors (DkNN) procedure is presented in Algorithm 1

In its simplest form, Algorithm 1 can be understood as creating a nearest neighbors classifier in the space defined by each DNN layer. Specifically, given the trained neural network f and a test input x:

1) We run input x through the DNN f to obtain the l representations output by its layers: $\{f_\lambda(x) | \lambda \in 1..l\}$.
2) For each of these representations $f_\lambda(x)$, we use a nearest neighbors classifier based on *locality-sensitive hashing* (LSH) to find the k training points whose representations at layer λ are closest to the one of the test input (i.e.,$f_\lambda(x)$).
3) For each layer λ, we collect the multi-set Ω_λ of labels assigned in the training dataset to the k nearest representations found at the previous step.

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

4) We use all multi-sets Ω_λ to compute the prediction of the DkNN according to the framework of *conformal prediction*.

For more details about the motivation and implementation details, please refer to the original paper [1].

IV. Experiments

A. Experimental Setup

We implement the DkNN classifier described in Algorithm 1[1]. We experiment with the MNIST dataset and the DNN architecture used is detailed in Table I. The model was trained with Adam at a learning rate of 10^{-3}. We set the number of neighbors $k = 75$ and the size of the calibration set $|A| = 750$, which is obtained by holding out a subset of the test set not used for evaluation.

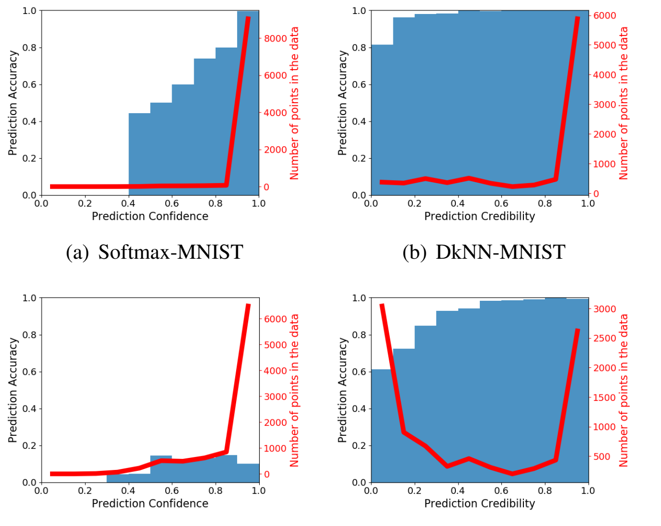
TABLE I
DNN ARCHITECTURES FOR EVALUATION

Layer	Layer Parameters (filters, kernel shape, strides, padding)
Conv1	64, (8*8), (2*2), same
Conv2	128, (6*6), (2*2), valid
Conv3	128, (5*5), (1*1), valid
Linear	10units

B. Identifying Adversarial Examples with the DkNN Algorithm

We craft adversarial examples using the Fast Gradient Sign Method (FGSM) [2]. In Figure 1, we plot reliability diagrams comparing the DkNN credibility with the softmax probabilities output by the DNN on MNIST original clean and adversarial examples. The number of points in each bin is reflected by the red line. Hence, credibility on adversarial examples is low for the DkNN, when compared to clean test inputs–unless the DkNN's prediction is correct. However, the softmax outputs high confidence on both clean and adversarial examples. The blue bars indicate the prediction accuracies. The left column shows that undefended DNN makes significant errors on most adversarial examples while the right column shows the good robustness of DkNN against adversarial examples (see [1] for more experiments and interpretations).

[1]https://github.com/Buhua-Liu/Adversarial-examples--Attack-and-Defense



(a) Softmax-MNIST (b) DkNN-MNIST

(c) Softmax-FGSM (d) DkNN-FGSM

Fig. 1. **Reliability diagrams of DNN softmax confidence (left) and DkNN credibility (right) on original (top) and adversarial (bottom) MNIST data**–blue bars (left axis) indicate the mean accuracy of predictions binned by credibility; the red line (right axis) illustrates data density across bins. The softmax outputs high confidence on most of the data while DkNN credibility spreads across the value range. On the other hand, the DkNN's credibility is better calibrated (i.e., it assigns low confidence to adversarial examples) than probabilities output by the softmax of an undefended DNN.

V. Conclusion

In this project, we reproduce the DkNN algorithm and validate its effectiveness on MNIST dataset with both clean and adversarial examples. The DkNN algorithm benefits from its intrinsic design, which can provide desirable properties–confidence, robustness and interpretability. However, the weakness is the fine-tuning of the hyper-parameters–the number of nearest neighbors and the size of the calibration set besides those of the DNN itself. Furthermore, the training time will scale up with the increase of the model depth and width since the computational cost of kNN algorithm performed at each layer will become larger. Finally, in practice, the fact that better-calibrated prediction requires more calibration data also limits the application of DkNN.

References

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

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