Analyze the use of Intermediate layer representations in the attack and defence of Neural Networks

May 24, 2021

Abstract

Adversarial examples, malicious inputs crafted to fool a Deep Neural Network(DNN) pose a security risk in their commercial deployment. Malicious inputs can be predicted to belong to a wrong class with higher confidence than legitimate inputs are to the correct class. In this project, we tried to analyze the robustness of using internal representations or the intermediate layer representation of the input data of Neural Network for detecting adversarial examples using Deep-KNN(DkNN) [2] in the context of (a) poisoned model - a model trained on a poisoned data set and(b) against adversarial examples fine-tuned for increasing perturbation on a targeted layer using Intermediate Level Attack (ILA) [3].

1 Introduction to Adversarial attacks

A DNN builds hierarchical representations of the input data that grows increasingly abstract where the final layer output can be good enough to be classified by a linear classifier.

- Deep Neural Networks(DNN) used in Vision tasks are susceptible to attacks from malicious inputs that can cause the network to misclassify it's input with a high degree of confidence where a Human observer would not be easily defeated. An example from [1] below illustrates one such case
- Misclassifcation can happen when the final representation of the image evolved through successive layers moves to the wrong region in a models decision space.

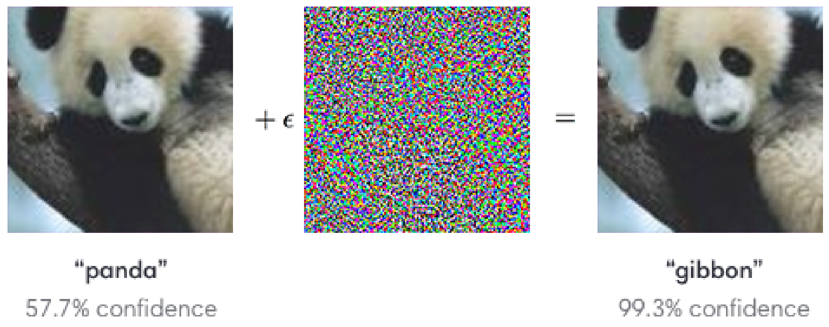


Figure 1: Adversarial attack

In the project we seek to identify two types of attacks, Poisoned data attacks and Intermediate layer attacks using Deep kNN from [2] which is detailed in further sections.

1.1 Poisoned data attacks

These attacks are carried out during the training phase by corrupting the training data. There are two categories

- Collision attacks: Adversary aims to introduce infected samples to training set to degrade the testing performance of a trained model.
- Backdoor attacks: Adversary introduces a trigger in training set to fool the trained model. A trigger can be a patch or a seemingly innocuous detail that is deliberately added so that the model may identify it with the label rather than associating the label with other details in the Image.

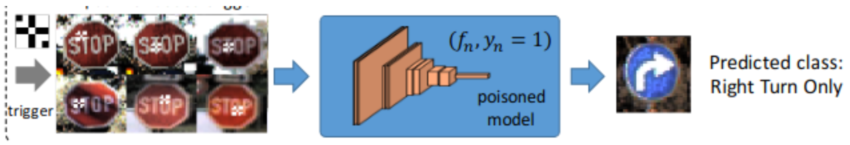


Figure 2: poisoned data

In the project we have focused on backdoor attacks for creation of poisoned data samples for MNIST.

1.2 Intermediate layer attack(ILA)

This type of attack from [3] aims at increasing the transferability of adversarial example by fine-tuning an existing adversary crafted in a white-box setting for a black-box attack.

- In the image space there is a strong correlation between perceptibility of the image and the norm of the perturbation . However, it is said in [3] that in the feature space the norm of the perturbation is not intrinsically tied to the perceptibility in the image space. Thus, we can increase the norm of the perturbation without significantly changing the perception in the Image space
- ILA makes use of this idea to increase the perturbation of the image in the feature space to attack a model. This is illustrated in the picture taken from WIML talk

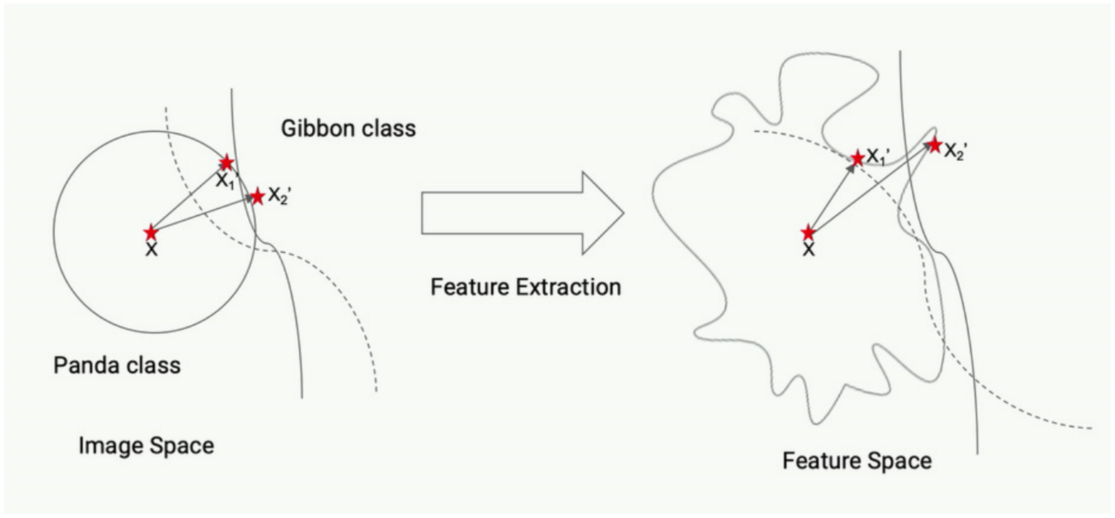


Figure 3: Perturbation in the feature space

2 Introduction to Deep kNN

DeepkNN from [2] uses the intermediate layer information of a DNN to identify adversarial examples

- DeepkNN fits a kNN classifier on the internal representation of the training points at each layer.
- During inference, across the layers, identify the training points whose internal representations are close to the internal representation of the input.
- Gauge the homogeneity of the labels of the nearest training points across layers and compare it to the label assigned by DNN.
- Calibrate the DeepkNN with a holdout data, not used in training but whose correct labels are known to get a baseline estimate for homogenity.
- Based on the above factors, DeepkNN gives a credibility for the current prediction along with confidence measure. Credibility says how relevant is the training set for the current predictions.

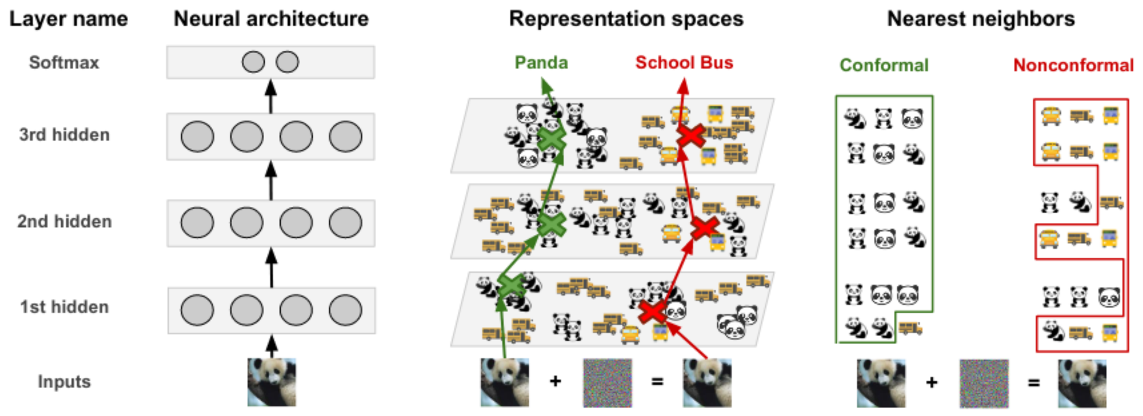


Figure 4: DeepkNN intuition

3 Application of DeepkNN and Results

In this section we detail our plan for applying DeepkNN on poison attacks and ILA. We have results only for poison data attack.
Code for the experiment is adapated from Poison_data and DeepkNN.

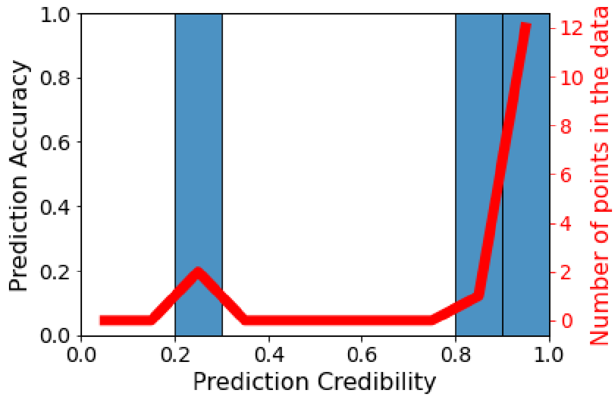
3.1 Poisoned data

- Construct a poisoned MNIST data using **secml** library. Details of the parameters used is listed in B.
- Use the poisoned dataset during training of DkNN and also separately without using in training and only as test dataset to check how good is a trained DkNN to detect such poisoned sample based on credibility estimate returned by the network.
- We check how the poisoned dataset influences the DNN decision for an input by identifying what percentage of the nearest neighbours at each layer are poisoned data points.
- We have used the same CNN architecture as in [2] listed in A

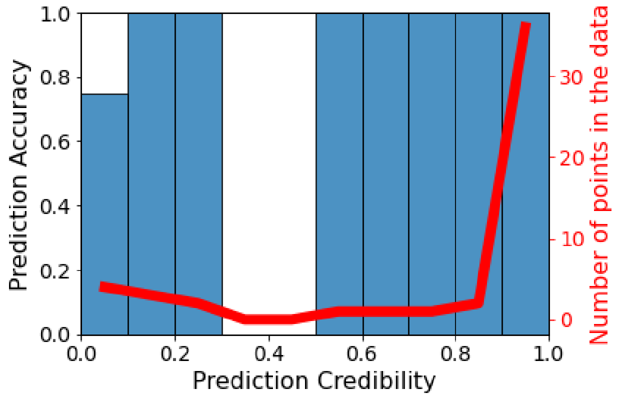
Experiment 1: Clean MNIST training set
After training a DkNN with a clean MNIST dataset we passed in a mixture of poisoned + non-poisoned samples to see

the credibility scores and accuracy. We observed that credibility scores were low for some samples even though accuracy may be high. These are suspect points and DNN labels should not be trusted. This is illustrated in the following graphs.

Result for 4/15 poisoning points(without separating poisoned and correctly classified samples):



Result for 25/50 poisoned points(without separating poisoned and correctly classified samples):



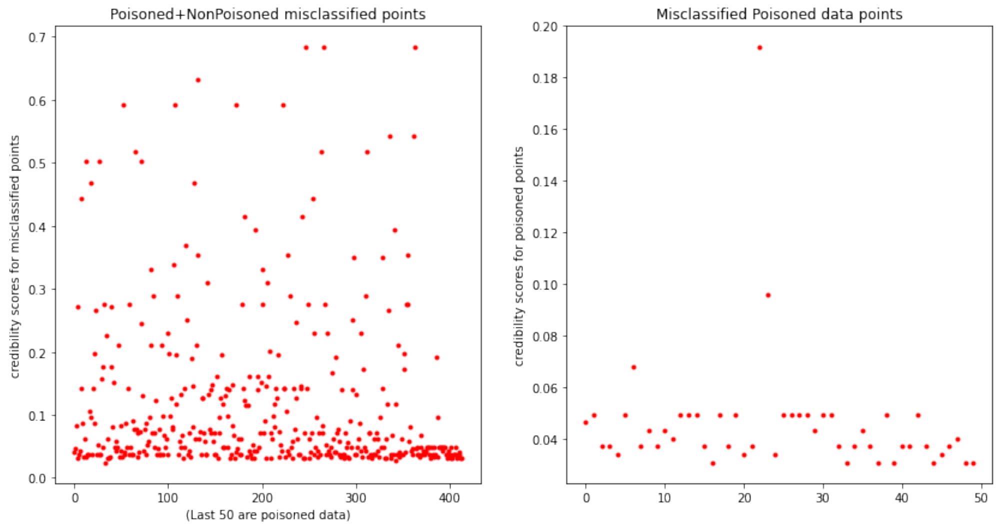
We can see that we have high credibility for most of the samples. This is because the poisoned dataset created from the secml library also included samples which were not actually poisoned and DNN itself was sufficient to correctly classify the samples, therefore high credibility. We conducted another experiment detailed below to asses the performance on only misclassified samples.

Experiment 2:

- Generate 50 poisoned MNIST data for 2 labels, '1' and '9' and combined with a clean MNIST data for two different sets of training data, having all 10 labels and having only 2 labels.
- Train a CNN with clean + poisoned dataset.

2.1 : Train with MNIST having all 10 labels + poisoned data with two labels

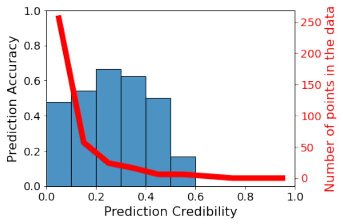
We had 414 misclassified points which included the poisoned data. Plots below show the credibility scores for misclassified and also only poisoned points.



Observation: We see that the credibility score assigned to the poisoned points have very low variance and have very low overall max credibility score. We repeated the experiment with a new set of poisoning data and we observed the same behaviour of very low variance in credibility score and max value

Inference: This behaviour which requires more examination perhaps can give us an indication that poisoned points are always assigned very low credibility scores and we may narrow down the list suspected points by setting a threshold for credibility score.

Below plot shows prediction accuracy and credibility for misclassified data.



Among the misclassified points by DNN , DeepkNN was able to recover the correct labels for 213/364 non-poisoned points.

2.2:Train with a smaller MNIST data having only 2 labels + poisoned data and to find the influence of poisoned data in the final prediction.

- In this experiment we considered 1000 MNIST samples for labels '1' and '9' and added 50 poisoned points for the same set of labels.
- We estimated what percentage of the nearest neighbours for each of the misclassified points and for non-poisoned data points are made up of poisoned data.
- Through this measure we sought to identify if there is any correlation between poisoned points influencing mis-classification result and for non-poisoned data points.

Table 1: % of poisoned data in the nearest 75 neigbhours

	Layer 1(ReLU)	**Layer 2(ReLU)**	**Layer 3(ReLU)**	**Layer 4(FC)**
Non-poisoned	24.93	28.8	29.14	29.04
Misclassified	25.3	28.8	35.4	36.93

In our experiments influence of poisoned data points by way of being the nearest neighbour's monotonically increased as we moved towards the final layer.

3.2 ILA

We sought to apply DeepkNN to detect a finely tuned adversarial example for MNIST dataset on a CNN. However, even though we can generate a white box attack using FGSM(Fast gradient sign method) we were not able to use the ILA algorithm to transfer it for a black box attack. Hence, we were not able to obtain results.

Code and the datasets used is made available in the GitHub repo. All experiments were carried out on Google Colab.

4 Scope for further experiments

- Experiment with larger poisoned data-set with multiple labels and observe the influence of poisoned points in mis-classification. From this result we may remove the suspected training points and re-train.
- Experiment to validate if poisoned data points always have low variance in their credibility scores low maximum values as was observed.
- As observed, why does the influence of poisoned training points in influencing the DNN decision grows as we move towards the final layer.

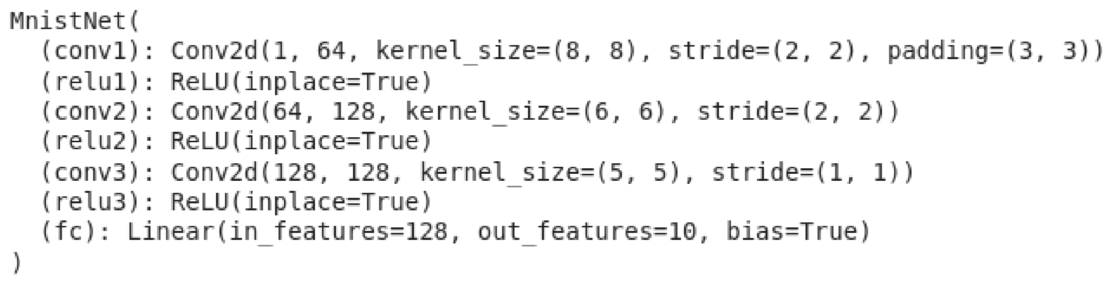
References

[1] Ian J. Goodfellow & JonathonShlens & Christian Szegedy *Explaining and Harnessing Adversarial Examples*.

[2] Nicolas Papernot & Patrick McDaniel *Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning*. *arXiv:1803.04765v1 [cs.LG] 13 Mar 2018*. Annalen der Physik, 322(10):891–921, 1905.

[3] Qian Huang & Isay Katsman & Horace He & Zeqi Gu *Enhancing Adversarial Example Transferability with an Intermediate Level Attack*.

A CNN Architecture



```
MnistNet(
  (conv1): Conv2d(1, 64, kernel_size=(8, 8), stride=(2, 2), padding=(3, 3))
  (relu1): ReLU(inplace=True)
  (conv2): Conv2d(64, 128, kernel_size=(6, 6), stride=(2, 2))
  (relu2): ReLU(inplace=True)
  (conv3): Conv2d(128, 128, kernel_size=(5, 5), stride=(1, 1))
  (relu3): ReLU(inplace=True)
  (fc): Linear(in_features=128, out_features=10, bias=True)
)
```

B Poison data generation parameters using secml library

Norm:L2
Radius:2.5eps
Feature space bound:[0,1]