CS5260
Neural Networks and Deep Learning II

Defending Adversarial Attacks against CNN, By Tey Shiwei (A0112101M)

Problem Definition

Convolutional neural networks (CNN) have been widely used for image classification and significant results have been achieved in various tasks. With all the well-built open source libraries around the web, it is relatively convenient to build an image classification model. However, CNN model can be easily fooled by an adversarial image created from the original image with some small perturbation. The robustness of CNN model is questionable and this implies that training a good CNN model is more than just fitting it to the training data.

In this project, a CNN model that classifies images into 4 different groups (artifacts, cancer, normal and other) was given. However certain types of images were not correctly classified by the CNN model. Clean images refer to the set of images that are classified correctly by the model and adversarial images refer to the set of images that are mis-classified by the model. The objective of this project is to classify both clean and adversarial images correctly as much as possible.

Various defence methods are available online, such as pre-processing the images before inputting to the model, re-training the CNN model with adversarial images and post-processing the output distribution of the CNN model to improve the accuracy of the classification model. While some of the adversarial images will be classified correctly after deploying the defences, the trade-off for using defences will be a decline in the accuracy of the output of clean images. Hence, the harmonic mean between accuracy of clean and adversarial images output will be the metric to evaluate the effectiveness of the defence method used for this project. In other words, the aim is to recover as much adversarial images accuracy while maintaining good accuracy on the clean images results.

Since the CNN model is given and fixed, the defence method used in this project will be focusing on pre-processing the images before feeding into the CNN and post-process the softmax output of the CNN model.

Figure 1 CNN Model with Defence

Brief Description of the Process Flow

1. Images are loaded without normalization (Pixel values between 0 to 1)
2. Pre-process the non-normalized images
3. CNN Model to predict pre-processed normalized images
4. Post-process the softmax output from CNN to generate final classification results

Data Sampling for Hyperparameters Tuning

To determine the best set of hyperparameters used for the pre and post processors, a simple brute force search will be carried out. However, running the whole image set (around 30k images) on a machine learning model its rather time consuming. To get a rough optimal value of the set of hyperparameters, a sampled of the data (500 to 5k) will be used to estimate the optimal values of the hyperparameters. Once the estimated optimal values of the parameters were found, the exact optimal values for the set of hyperparameters will be fine tuned using the entire image set.

An example to tune the learning rate,

- Initial value of optimal learning rate is unknown
- Data are sampled to obtain the estimated optimal learning rate by searching from [0.01 to 0.99] with 0.01 increment
- Assume estimated optimal learning rate = 0.15, the exact learning rate will be tuned based on the entire dataset from [0.1 to 0.2] with 0.01 increment

This searching method is applied to all hyperparameters of all the models that were tested.

Exploration Phase

The initial result of the CNN model without any defence method is of course 100% on clean images and 0% on adversarial images. As a start of this project, I have tried every single defence method from ***Adversarial Robustness Toolbox (ART)*** and analyse each and every of the defence method based on the accuracy, complexity of hyperparameters and computational time. The best defence processor will be the one with high accuracy and short computational time.

Here are the lists of *pre-processors* used: (Accuracy of clean and adversarial images shown are tuned with the estimated optimal hyperparameters)

- Total Variance Minimization: Good accuracy but hard to tune and time consuming
- Spatial Smoothing: Very high accuracy (Clean: 97%, Adv 50%), easy to tune and very fast
- JPEG Compression: Very high accuracy (Clean: 93%, Adv 60%), easy to tune and very fast
- Feature Squeezing: Good accuracy on clean images (Clean: 100%, Adv 30%), easy to tune and very fast

The two promising pre-processors are Spatial Smoothing and JPEG Compression, which are inexpensive to implement yet effective. JPEG compression method removes additive perturbation from the image by eliminating high frequency signal components while spatial smoothing reduces image noise by smoothing out each pixel value by its nearby pixels.

For *post-processors*:

- Gaussian Noise: Not very reliable as the result highly depends on the generated Gaussian distribution. It could be very good but also very bad sometimes based on the generated noise distribution.
- High Confidences: Currently the best post-processor among ***ART*** that suits this project but the additional improvement from pre-processor is only 1%-5%
- Tried Reverse Sigmoid, Rounded, Class Labels methods: Results are slightly lesser than High confidence method but not very effective
- Light GBM method: High accuracy (additional 10% from pre-processor) and short computational time

Light GBM is chosen as the post-processor model because of its high performance and short computational time. A well-tuned light GBM model is able to further increase the classification accuracy by 10%.

Experimental Study

Pre-processor

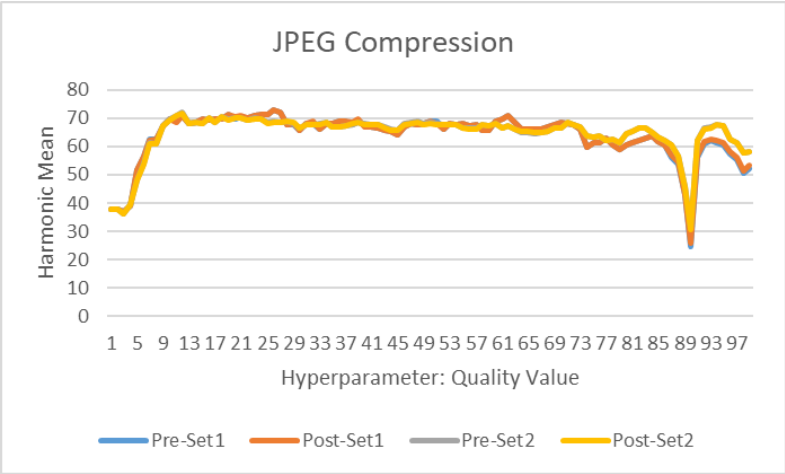

Figure 2 JPEG Compression Harmonic Mean

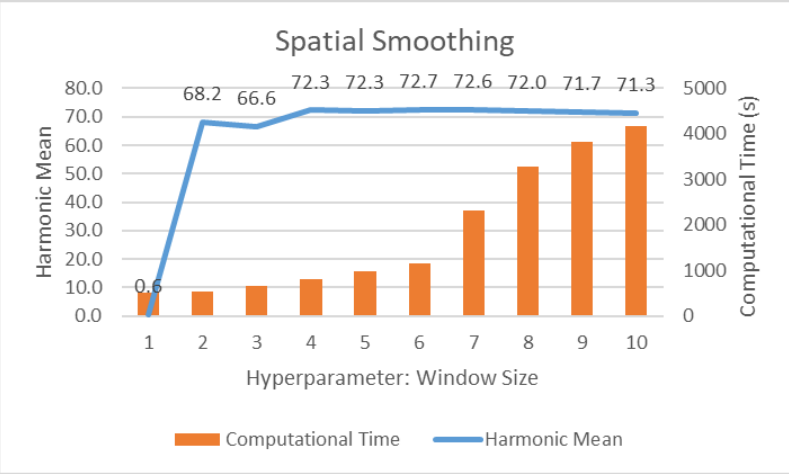

Figure 3 Spatial Smoothing Harmonic Mean

The two best pre-processor models' performance were analysed by changing its hyperparameter.

- Hyperparameter of JPEG Compression: Image quality value.
  - As shown in Figure 2, since the computational time is constant across all the quality values, the optimal quality value chosen is 22.
- Hyperparameter of Spatial Smoothing: Window Size.
  - The optimal window size chosen is 6 because of its relatively low computational time and high harmonic mean. The harmonic mean stabilizes when window size > 3.

Despite both pre-processors achieve similar performances, Spatial Smoothing method was chosen as it achieves a greater performance and compatibility with Light GBM as the post-processor for this project.

Post-processor

Light GBM is used as the post-processor to further improve the output accuracy based on the softmax output values of the CNN model. The function of Light GBM as the post-processor is to learn the generated softmax distribution of the CNN model (after pre-processing the input images) to predict the final class of the images. According to distribution classifier method [1], there is a high similarity between the softmax distribution of clean and adversarial images. Hence, the Light GBM model is shown to be able to learn the softmax distribution of adversarial images and classify it to the correct label based on the learned softmax distribution of the clean images.

To tune the Light GBM, a Grid Search is run to obtain the best parameter values and the range values of the hyperparameters are bounded according to the Light GBM documentation to prevent overfitting. The final hyperparameters values for Light GBM are *{Learning Rate: 0.1, Number of leaves: 81, Number of rounds: 50*} with a train-validation split ratio of 0.2.

A further study was done by tweaking the train-validation split ratio from 0.1 to 0.4 with fixed seeds to understand the error bound of the chosen training set data. The observed harmonic mean value tends to stabilize around 10% to 12% improvements which implies that the number of training data above 10k is enough to represent the entire dataset (30k).

In short, the original CNN model is combined with Spatial Smoothing as the pre-processor and Light GBM as the post-processor.

Figure 4 CNN Classifier with Spatial Smoothing and LightGBM as pre and post processor

Results

Pre-processor	Post-processor	Model accuracy	Pre-processed accuracy	Post-processed accuracy
Spatial Smoothing	High Confidence	44.05%	62.56%	64.11%
Spatial Smoothing	Light GBM	44.05%	62.56%	93.39%
JPEG Compression	High Confidence	44.05%	72.25%	73.19%
JPEG Compression	Light GBM	44.05%	72.25%	86.34%

The table above shows results of improved accuracy based on a sample set of data with 44% clean images and 56% adversarial images.

- The initial model accuracy was 44.05% without any defence methods.
- After pre-processor, JPEG Compression (72.25%) yields a higher accuracy than Spatial Smoothing (62.56%)
- However, Spatial Smoothing + Light GBM (93.39%) works better than JPEG Compression + Light GBM (86.34%)

Final Results

The following results are generated with all the given image data: 6929 clean images and 22228 adversarial images.

- Model accuracy (Total number of corrects/ Total number of images)
    - Initial: 23.76%
    - After Spatial Smoothing: 69.36%
    - After Light GBM: 81.41%
- Final evaluation score:
    - Clean: 6368 / 6929
    - Adv: 17369 / 22228
    - Harmonic mean: 0.8446

Code structure

1. Data directories
    - Cell to define folder directories
2. Define functions
    - All functions used for the defence algorithm are defined in this cell
3. Code to pre-train Light GBM Model
    - The Light GBM model has been pre-trained with the given dataset. This cell will be commented and not needed to run the main algorithm.
4. Defence Algorithm
    - Implementation of the defence algorithm using Spatial Smoothing as pre-processor and Light GBM as post-processor
5. Evaluate Results
    - Using the harmonic mean metric to evaluate the results. This cell will be commented as well.

File structure

- model
    - model.pt
- images
    - artifacts
    - cancer_regions
    - normal_regions
    - other
- results
    - A0112101M.txt
- A0112101M
    - A0112101M.ipynb
    - lgb_model.txt
    - load_model_36.pyc
    - load_model_37.pyc

References

[1] Kou C, Lee HK, Ng TK, Chang EC. Enhancing Transformation-based Defences using a Distribution Classifier. In ICLR 2020.

[2] Kou C, Lee HK, Ng TK. A compact network learning model for distribution regression. Neural Networks, 2018.