Backdoor Detection using Pruning Defence in BadNets

Name : Rithviik Srinivasan (rs8385)

1. Introduction

Neural networks' widespread adoption in various fields has led to increased concerns regarding their vulnerability to adversarial attacks, notably backdoor attacks. This report focuses on developing a backdoor detection system for BadNets trained on the YouTube Face dataset. Using the pruning defence technique, the project aims to rectify compromised BadNets by effectively detecting and mitigating backdoor attacks. Evaluation involves testing on both clean and backdoored validation and test datasets to showcase the defence's effectiveness. Pruning, a key methodology employed, systematically removes individual channels from a pooling layer. Channels are eliminated based on the descending order of average activation levels observed across the entire validation set. This iterative process assesses the model's accuracy after each channel removal, ceasing pruning when accuracy declines by a specified threshold (X%) compared to the original accuracy. This approach ensures retention of channels significantly contributing to the model's accuracy. Emphasizing validation using a clean dataset, the pruning process validates model accuracy and behaviour on test data. This validation assesses accuracy and attack success rate, offering insights into the model's performance when channels are removed. Pruning's goal is to uphold optimal accuracy while fortifying the model against potential attacks. Backdoor attacks involve inserting imperceptible cues triggering misclassification in neural networks. BadNets, trained on tainted datasets with inserted backdoors, are susceptible to classifying both clean and manipulated inputs, posing a substantial threat.

1.1 Objectives and Methodology

The primary goal of this project is to devise a defence mechanism against backdoor attacks targeting BadNets trained on the YouTube Face dataset. The objectives are twofold: first, to repair the compromised BadNet using the pruning defence strategy, and second, to evaluate the repaired network's efficacy in detecting and distinguishing clean inputs from backdoored ones. The proposed approach to address backdoor attacks on BadNets involves pruning the last pooling layer of the BadNet, i.e., removing channels one at a time from this layer. The selection of channels for removal is based on their average activation values computed over the entire validation set. Pruning continues until the validation accuracy drops by at least X% below the original accuracy, thereby producing a new network termed B'. The repaired network G, referred to as the "GoodNet," is constructed by utilizing both B and B'. For each test input, the model runs the input through both networks. If the classification outputs from B and B' are identical (i.e., class i), G outputs class i. However, if the outputs differ, G identifies the input as backdoored and assigns it to class N+1.

2. Implementation Details

2.1 Dataset and Repair Process

The project utilized the YouTube Face dataset and loaded the BadNet, B, along with its weights for repair and evaluation. Additionally, the validation and test datasets (clean and backdoored) were loaded for model evaluation. The repair process involved pruning the last pooling layer of the BadNet, sequentially removing channels based on decreasing average activation values. At each pruning step, the validation accuracy was computed, and pruning ceased when the accuracy dropped by X% or more.

2.2 GoodNet Creation

Once the repaired BadNet, B', was obtained, the GoodNet, G, was created. For every test input, G processed the input through both B and B' networks. The decision logic to output a specific class or the backdoor class (N+1) was applied based on the classification outputs of B and B'.

3. Results

The performance of the defence mechanism was evaluated using key metrics: **Clean Classification Accuracy**: Accuracy of correctly classifying clean inputs. **Attack Success Rate:** Rate of correctly identifying backdoored inputs.Three repaired networks were obtained corresponding to X values of 2%, 4%, and 10%. The showed varying performance metrics for each repaired network.

3.3 Graphs and Tables

Graph 1:

Clean Classification Accuracy vs. Pruning index

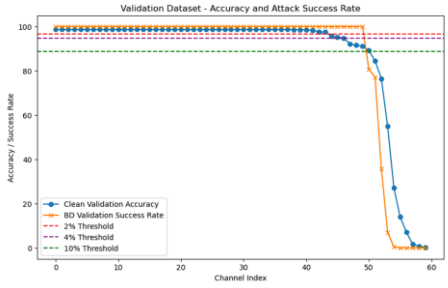

Table 1: Summary of Repair Efficacy

Table summarizing the clean classification accuracy and attack success rate for each repaired network at different pruning percentages.

model	Accuracy	Attack_Rate
repaired_2%	95.744349	100.000000
repaired_4%	92.127825	99.984412
repaired_10%	84.333593	77.201871

5. Analysis of Results

The obtained results depict the efficacy of the defence mechanism. These results demonstrate a trade-off between clean accuracy and the ability to detect backdoored inputs as the pruning percentage increases.

- **Repaired (2%):** This model shows a high clean accuracy of 95.74%. It correctly classifies clean inputs with a high degree of accuracy. However, the backdoor attack rate is 100%, indicating that it fails to detect backdoored inputs effectively. This suggests that although the model is proficient at identifying clean inputs, it is vulnerable to the backdoor attack.
- **Repaired (4%):** With a pruning percentage of 4%, the clean accuracy remains relatively high at 92.13%. It maintains a good level of accuracy in classifying clean inputs. However, the backdoor attack rate is still substantial at 99.98%. Despite slightly lower clean accuracy than the 2% repaired model, it performs similarly in detecting backdoored inputs.
- **Repaired (10%):** This model exhibits a significant decline in clean accuracy, dropping to 84.33%. The model's effectiveness in correctly identifying clean inputs decreases notably as more channels are pruned. Additionally, the backdoor attack rate decreases to 77.20%. While the attack rate decreases, it's at the expense of a considerable loss in clean accuracy, indicating a more robust defence against

6. Conclusion

Overall, the results suggest that as the pruning percentage increases, the model's ability to correctly classify clean inputs diminishes while its capacity to detect backdoors improves. There's a clear trade-off between accurately classifying clean data and identifying backdoored inputs, highlighting the need to balance these aspects based on the desired application and security requirements.

GitHub Repository: https://github.com/unrealbatman/MLCybersecurityLab4.git