ECE-GY 9163
ML for Cyber Security
Fall 2023

Designing a Backdoor Detector for BadNets Trained on the YouTube Face Dataset Using the Pruning Defense

Nishal Sundarraman (ns5429)

12-04-2023

1 Introduction

The increasing reliance on neural networks in various applications has raised significant concerns regarding their security, particularly from backdoor attacks. These attacks, which subtly manipulate network behavior, pose a critical threat to the integrity of neural network applications. This report details the development of a novel backdoor detector designed to safeguard BadNets, a type of compromised neural network, using a dataset derived from YouTube Faces. The primary defense mechanism employed is network pruning, a technique aimed at enhancing model robustness against such insidious threats.

2 Dataset

In this experiment, the YouTube Face dataset is partitioned into two distinct subsets: clean and poisoned. The clean subsets, specifically the validation and test sets named "valid.h5" and "test.h5," are employed for fine-tuning and evaluating the model's performance. To simulate backdoor attacks, poisoned datasets featuring a sunglasses trigger - labeled as "bd_valid.h5" and "bd_test.h5" - are utilized to test the efficacy of the defense mechanism.

3 Methodology

In this project, a pruning defense was applied to a neural network, strategically eliminating channels according to their activation intensities. This pruning process continued until the model's accuracy declined by specific predefined thresholds, set at 2%, 4%, and 10%. Utilizing TensorFlow and Keras for implementation, the performance of the pruned models was assessed against both the unaltered and the poisoned datasets. The evaluation focused on the models' ability to distinguish between clean and backdoored inputs, relying on the level of agreement between the model predictions. The primary goal of this strategy was to find an optimal balance between maintaining accuracy and efficiently detecting backdoor attacks.

1. **Environment Setup and Essential Libraries:** The project commenced with the setup of a Python environment and the importation of key libraries. TensorFlow and Keras were utilized for neural network modeling, NumPy for numerical computations, Matplotlib for visualizations, and OpenCV for image processing.

2. **Data Integration and Loading:** The project leverages Google Colab for its computational needs. A custom 'data_loader' function was created to handle the YouTube Face dataset, ensuring efficient data management and preprocessing. This function is vital for preparing the dataset for subsequent training and analysis.

3. **Data Loader Function Implementation:** The 'data_loader' function is instrumental in reading the dataset, modifying its structure to suit the requirements of the neural network, and ensuring the data is in a format that is ready for model input.

4 Results

Model	Repaired Clean Accuracy	Attack Rate
2% Repaired	95.7443	100
4% Repaired	92.1278	99.9844
10% Repaired	84.3336	77.2097

Table 1: Accuracy and Attack Rate of Repaired Models

The experiment focused on implementing a pruning defense in a neural network to counteract backdoor attacks. This process involved selectively removing channels based on their activation levels and monitoring the consequent impact on accuracy. The pruning was executed until the model's accuracy experienced predefined thresholds of reduction at 2%, 4%, and 10%. The evaluation of the pruned models was conducted using TensorFlow and Keras, assessing their performance against both original (clean) and poisoned datasets.

Pruning Results:

1. **Initial Accuracy Thresholds:** The pruning process was structured to halt upon reaching specific accuracy drop thresholds. The experiment successfully saved different versions of the model corresponding to these accuracy decreases.
2. **Model Performance:** The model demonstrated a distinct drop in accuracy at each predefined threshold. The results indicate that the model was saved after each significant accuracy drop, ensuring a comprehensive evaluation of the pruning defense at various stages.
3. **Output Summary:**
   During the pruning process, the following outcomes were observed and logged:
   (a) **2% Accuracy Drop:** The model was saved after reaching at least a 2% drop in accuracy.
   (b) **4% Accuracy Drop:** Subsequently, the model experienced a further accuracy decline of at least 4%, prompting another save.
   (c) **10% Accuracy Drop:** Finally, the model was saved after a substantial accuracy reduction of at least 10

The results show that the pruning defense was effective in manipulating the model's accuracy, providing insights into the balance between model accuracy and backdoor attack resilience. The saved models at different accuracy levels facilitate a nuanced analysis of the trade-offs involved in implementing pruning defenses in neural networks.