Deep Learning for Malicious Flow Detection

Yun-Chun Chen

Department of Electrical Engineering, National Taiwan University

Introduction

Cyber security has become an important issue that cannot be overlooked. However, traditional methods to defend these threats are signature oriented meaning that malware will not be detected by anti-virus software if they change their behavior.

Recently, many literature proposed different methods to detect malware. Even though these methods can achieve a high accuracy, we discover that the precision, which is also an important evaluation index, of these methods is rather low. When dealing with real world issue, imbalanced data distribution between classes often occurs. We propose an end-to-end trainable Tree-Shaped Deep Neural Network along with Quantity Dependent Backpropagation to tackle with imbalanced data issue.

Contributions

- We propose an ***end-to-end trainable Tree-Shaped Deep Neural Network*** which classifies the data in a layer-wise manner.
- We propose a ***Quantity Dependent Backpropagation*** which incorporates the knowledge of disparity between classes and overcome the difficulty of imbalanced learning.
- We further show the feasibility of ***real-time detection*** by executing a partial flow experiment and we also conduct a zero-shot learning experiment which demonstrates the generalization performance of deep learning in cyber security.

Algorithm

Backpropagation:

$$\theta_i^{l^+} = \theta_i^l - \eta \times \frac{\partial Loss}{\partial \theta_i^l} \tag{1}$$

We introduce a vector $\boldsymbol{F}$ into backpropagation and propose a ***Quantity Dependent Backpropagation*** which takes the disparity between classes into consideration and shows different sensitivities toward different classes. The mathematical formula is given by:

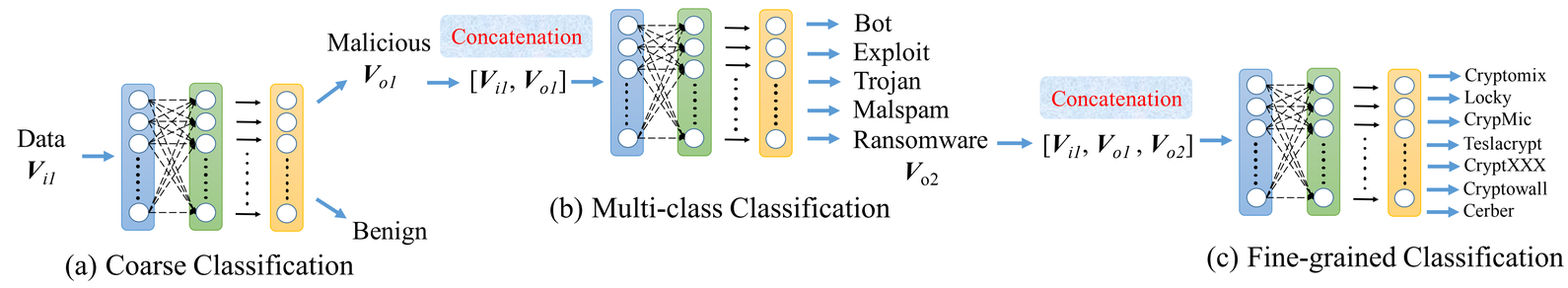
$$\theta_i^{l^+} = \theta_i^l - \eta \cdot \boldsymbol{F} \cdot \nabla Loss \tag{2}$$

$$\boldsymbol{F} = \left[\frac{c_1}{n_1}, \frac{c_2}{n_2}, ..., \frac{c_N}{n_N}\right] \tag{3}$$

$$\nabla Loss = \left[\frac{\partial Loss_1}{\partial \theta_i}, ..., \frac{\partial Loss_N}{\partial \theta_i}\right]^T \tag{4}$$

where $\boldsymbol{F}$ is a row vector, N represents the cardinality of the training data set, n_i represents the cardinality of the class where the i^{th} data belongs to, and c_i is the pre-selected coefficient for the class where the i^{th} data belongs to.

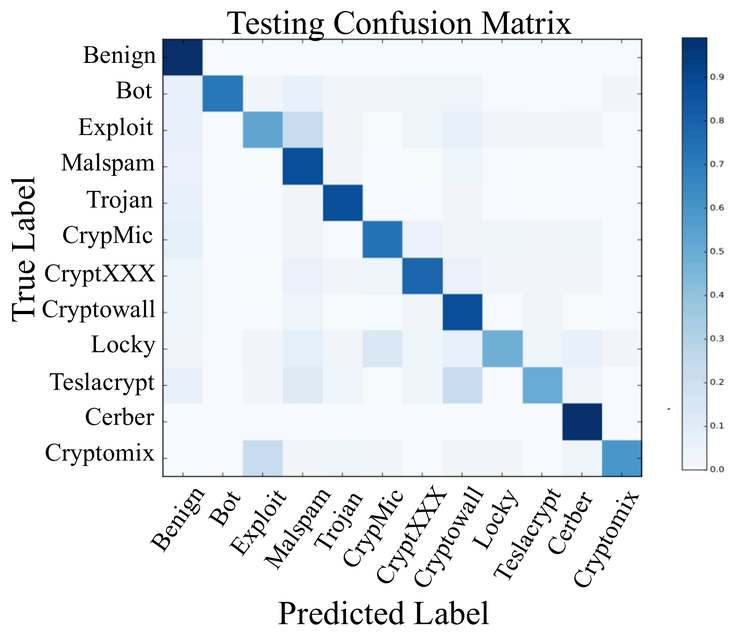
Tree-Shaped Deep Neural Network



(a) Coarse Classification (b) Multi-class Classification (c) Fine-grained Classification

This model performs a 12-class classification in three stages. Firstly, the model performs a coarse classification which preliminarily classifies the data into the class of benign flows and malicious flows. Secondly, the model conducts a multi-class classification on the malicious flows, which classifies them into 5 different categories according to the attack behavior. Lastly, the model executes a fine-grained classification on Ransomware family.

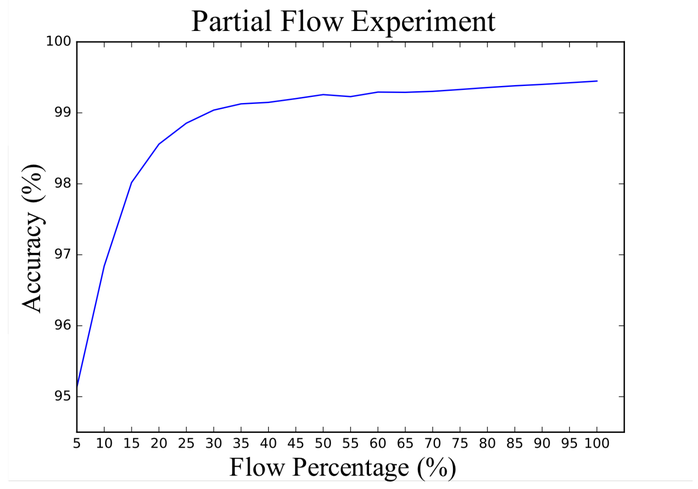
Experimental Analysis



Accuracy and precision of different approaches

Method	Accuracy	Precision
DNN + Backpropagation	59.08%	8.33%
DNN + Oversampling (10000 samples/class)	85.18%	65.9%
DNN + Undersampling (45 samples/class)	68.89%	49.45%
DNN + Incremental Learning	78.84%	71.23%
DNN + QDBP	84.56%	62.3%
SVM (RBF)	83.87%	38.8%
Random Forest	98.9%	68.25%
TSDNN + QDBP	**99.63%**	**85.4%**

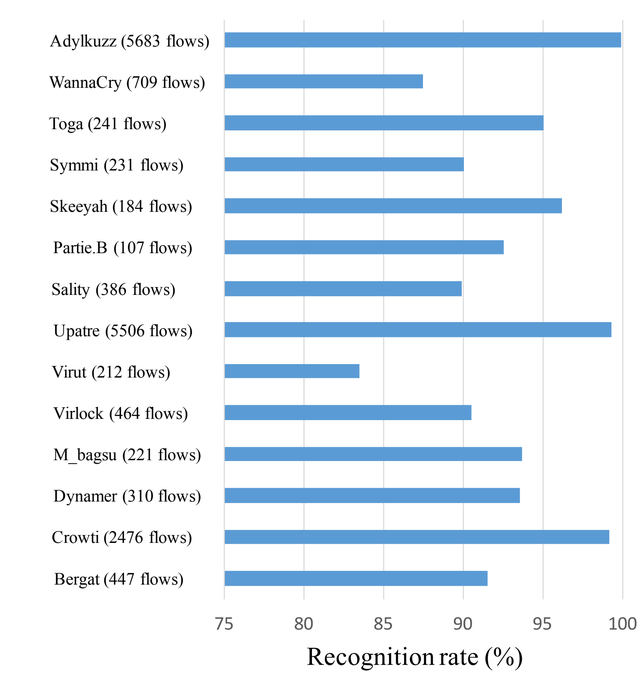
Partial Flow Detection



For the sake of real-time detection, we devise an experiment by dividing each attack into fractions and only consider a portion of the data to test the potentiality of being a malware.

From the experimental result, we can conclude that our model is able to distinguish the malicious flow by only considering the first 5 percent of the entire flow without analyzing the entire flow which shows the possibility of a real-time detection.

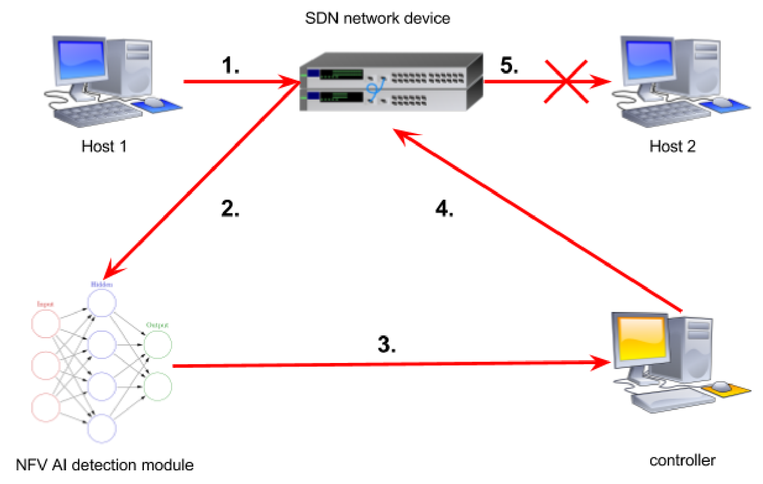
Zero-Shot Detection



To evaluate the generalization performance of the proposed model, we would like to examine the ability of TSDNN to identify some malware that has never been trained by our model.

We collect 14 different kinds of malware to evaluate the ability of our model to perceive potential threats and the experimental result is illustrated in the figure above.

SDN



The entire framework works as follows:

- A network flow comes out from host1 to host2 passing SDN.
- SDN mirrors the network flow to TSDNN model (NFV).
- If a malicious flow is detected, TSDNN will notify the controller.
- Once the controller receives the notification, it will add one new flow entry to block the malicious flow.
- Malicious flow is blocked.