ARIEL UNIVERSITY

GRADUATION PROJECT REPORT

A Deep Learning Approach to Classification of Encrypted Malicious & Ransomware Files

Author:
Lital SHEMESH
Elyasaf ZIDAR

Supervisor:
Dr. Amit DVIR

November 12, 2020

Contents

Chapter 1

Introduction

Classifiers are an invaluable tool for many tasks today, such as medical or genomics predictions, spam detection, face recognition, and finance. Many of these applications handle sensitive data, so it is important that the data and the classifier remain private. Consider the typical setup of supervised learning, which its algorithms consist of two phases: (i) the training phase during which the algorithm learns a model from a data set of labeled examples, and (ii) the classification phase that runs a classifier over a previously unseen feature vector, using the model to output a prediction.

In applications that handle sensitive data, it is important that the feature vectors and the model would remain secret to one or some of the parties involved. Consider the example of a medical study or a hospital having a model built out of the private medical profiles of some patients; the model is sensitive because it can leak information about the patients, and its usage has to be compliant. A client wants to use the model to make a prediction about her health (e.g., if she is likely to contract a certain disease, or if she would be treated successfully at the hospital), but does not want to reveal her sensitive medical profile. Ideally, the hospital and the client run a protocol at the end of which the client learns one bit ("yes/no"), and neither party learns anything else about the other party's input. A similar setting arises for a financial institution (e.g., an insurance company) holding a sensitive model, and a customer wanting to estimate rates or quality of service based on her personal information.

In this work, is constructed a classifier over both encrypted and non-encrypted dataset, in order to detect a Ransomware attack. In the cuber-security world, a **Ransomware** attack is a type of malware from crypto-virology that threatens to publish the victim's data or perpetually block access to it unless a ransom is paid. While some simple ransomware may lock the system in a way which is not difficult for a knowledgeable person to reverse, more advanced malware uses a technique called crypto-viral extortion, in which it encrypts the victim's files, making them inaccessible, and demands a ransom payment to decrypt them. Since Ransomware attacks learn and overcome defense tools, the need for refinement and defense tools is still a critical issue.

Chapter 2

Classical Approaches

- **Machine learning** - involves learning the patterns in data to create a model. However, the difficulty in using ML is in finding the correct algorithm to match with the type of data and the needed outcome. Thus, we use Deep-Learning model instead in order to overcome this issue. Among the classical machine learning algorithms, one can find the following (as well as the Neural Networks for Deep Learning which are presented too):

  - **K-Nearest-Neighbors** (KNN) - uses one nearest training data point to identity the closet neighbors and use it for prediction. This model classifies arbitrary k neighbors and designates classes with more neighbors as a label based on the number of neighbors belonging to each class. Therefore, the decision boundary is divided into the area designated by each class, then the data are classified based on the boundary. The important parameters in this model are the number of neighbors and how the distance between data points is estimated, and the performance of the model depends on these parameters.
  - **Decision Tree** -The decision tree model is widely used for classification and regression problems, and continues learning yes or no questions to reach a decision. This model divides the data into a list of yes or no questions, and the data are repeatedly divided until each divided area (leaf of the decision tree) has one target value (one class or one regression analysis result). A leaf node consisting of only one target is called a pure node. The disadvantage of this model is that if the model splits the data until all of the leaf nodes become pure nodes, the model becomes very complex and overfitting. There are two ways to control overfit: pre-pruning and post-pruning. Pre-pruning is a strategy that discontinues the tree generation early, while post-pruning is a strategy that deletes or merges with fewer data points following tree generation.
  - **Support Vector Machine** (SVM) - This model is an extension of the input data to generate a more complex model that cannot be defined as a simple hyperplane. For this reason, straight lines and hyperplanes are not flexible, and linear models are very limited for low dimensional data sets. In order to solve this problem, a new feature is added by multiplying features or exponentiating features. However, there is a disadvantage in that the computation cost is increased as the features increase. In order to reduce the computational cost, there is a kernel trick that can learn the classifier at a high dimension without generating many new features. This model does not actually extend the data, but instead calculates the distance (specifically, the scalar product) between the data points to the extended features. Based on the results, the SVM classifies each training

data point as the decision boundary between the two classes, and the data points located at the boundary are called support vectors. Thus, in order to predict for a new data point, this model measures the distance to each support vector, which is computed by the Gaussian kernel.

  – **Neural Networks** - In our project, we utilize Multi-Layer Perceptrons (MLP), a neural network model that is relatively easy to use for classification and regression. This model repeats the process of generating a weighted sum, which is a process from the linear regression model, to the output many times, as opposed to just once. This model is a model that computes the hidden unit which composes the intermediate stage and calculates the weight sum again. Such a hidden unit constitutes a hidden layer, and the deep running is performed due to the hidden layer. The most important parameter in this model is the number of units in the hidden layer, and the disadvantage is that the computation cost increases with an increasing number of hidden units.

- **Honeypot** - is a computer security mechanism designed to detect, deflect, and in some cases neutralize attempts to unauthorized use of information systems (Figure 2.1). In general, a honey trap contains data that appears to be part of the database or system data, and which allegedly contains valuable information to attackers, but is in fact isolated, monitored and blocked by attackers. Honeypots can be classified based on their deployment (use/action) and based on their level of involvement. Based on deployment, honeypots may be classified as production honeypots, or research honeypots;

  – Production honeypots are easy to use, capture only limited information, and are used primarily by corporations. Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security. Normally, production honeypots are low-interaction honeypots, which are easier to deploy. They give less information about the attacks or attackers than research honeypots.

  – Research honeypots are run to gather information about the motives and tactics of the black hat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better protect against those threats. Research honeypots are complex to deploy and maintain, capture extensive information, and are used primarily by research, military, or government organizations.

  Unfortunately, Honeypots have several disadvantages. It is because of these disadvantages that honeypots do not replace any security mechanisms; they only work with and enhance your overall security architecture;

  1. **Narrow Field Of View** - The greatest disadvantage of honeypots is they have a narrow field of view: They only see what activity is directed against them. If an attacker breaks into your network and attacks a variety of systems, your honeypot will be blissfully unaware of the activity unless it is attacked directly. If the attacker has identified your honeypot for what it is, she can now avoid that system and infiltrate your organization, with the honeypot never knowing she got in. As noted earlier, honeypots have

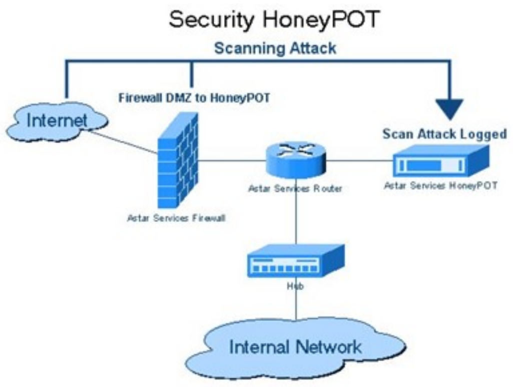

FIGURE 2.1: An illustration of the Honeypot trap. The image was taken from Wikipedia Honeypot Page

a microscope effect on the value of the data you collect, enabling you to focus closely on data of known value. However, like a microscope, the honeypot's very limited field of view can exclude events happening all around it.

2. **Finger Printing** - Another disadvantage of honeypots, especially many commercial versions, is fin-gerprinting. Fingerprinting is when an attacker can identify the true identity of a honeypot because it has certain expected characteristics or behaviors. For example, a honeypot may emulate a Web server. Whenever an attacker connects to this specific type of honeypot, the Web server responds by sending a common error message using standard HTML. This is the exact response we would expect for any Web server. However, the honeypot has a mistake in it and misspells one of the HTML commands, such as spelling the word length as legnht. This misspelling now becomes a fingerprint for the honeypot, since any attacker can quickly identify it because of this error in the Web server emulation. An incorrectly implemented honeypot can also identify itself. For example, a honeypot may be designed to emulate an NT IIS Web server, but the honeypot also has certain characteristics that identify it as a Unix Solaris server. These contradictory identities can act as a signature for a honeypot. There are a variety of other methods to fingerprint a honeypot that we discuss later in the book.

Chapter 3

Dataset

The BBC articles fulltext and category in Kaggle

Chapter 4

Solution Approach

Experimental Framework Design

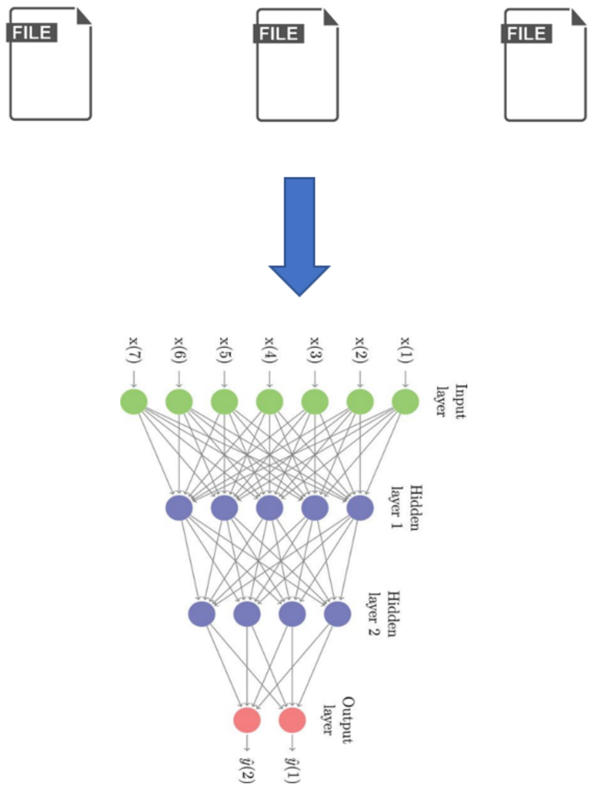

FIGURE 4.1: Blabla

4.1 Feature Engineering

4.2 Neural Network Architecture

Our feature engineering process resulted in 426 test instances, which were converted into feature-vector based learning examples. These examples were divided into training and test sets, using the well-known Cross-Validation method, such that 80% of the learning examples were used for the training set, and the remaining 20% for test. Due to the tabular structure of the transformed dataset, the model chosen for the classification problem was a Deep Neural Network with a Fully Connected (DNN-FC) architecture. A fully connected neural network consists of a series of fully connected layers. A fully connected layer is a function from $\mathbb{R}^m$ to $\mathbb{R}^n$ - where each output dimension depends on each input dimension.

The Neural Network has three hidden layers, and a dropout layer between any two layers, as well as between the last hidden layer and the output layer, with p = 0.5 for all the dropout layers (see Table 4.1 for a specific input size and Neural Network layer dimension). Since our problem is an instance of a supervised classification, we used the Cross-Entropy (CE) loss function. The Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

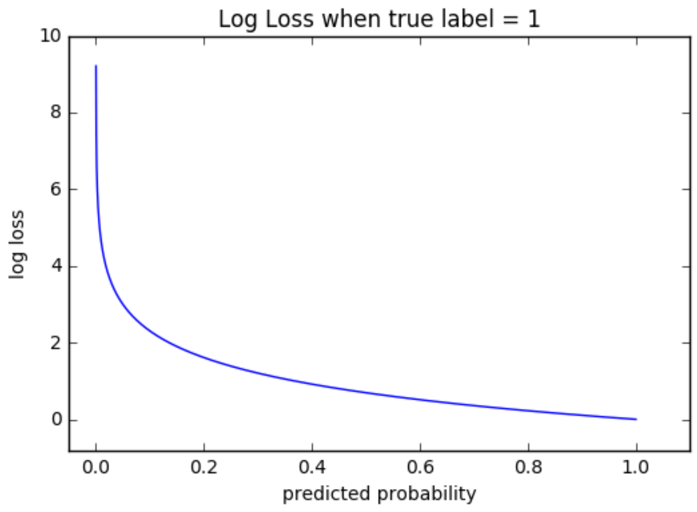

FIGURE 4.2: An illustration of the Cross-Entropy loss function (the image was taken from ML Glossary CE Loss Function).

The graph in Figure 4.2 shows the range of possible loss values given a true observation. As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly. Log loss penalizes both types of errors, but especially those predictions that are confident and wrong!

As for the learning rate of the Neural Network, we used the Adam Optimizer, an adaptive learning rate method that computes individual learning rates for different parameters, with an initial learning rate value of $\alpha = 1 \cdot 10^{-4}$. Consequently, we chose a Softmax [1, 2] activation layer to represent the network's output, in the form of a probability vector of size 2 for the Softmax layer which predicts the probability of an instance being labeled as Enc and Non_Enc, with one probability for each class.

Layer	**In Features**	**Out Features**
Input	4	8
1^{st} Hidden Layer	8	16
2^{nd} Hidden Layer	16	32
3^{rd} Hidden Layer	32	2
Output	2	Softmax Decision Function

TABLE 4.1: Neural Network Architecture

Chapter 5

Experimental Evaluation

In order to verify that the suggested

In a classification task, the precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been).

Thus, in a classification task (Figure 5.1), a precision score of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but says nothing about the number of items from class C that were not labeled correctly) whereas a recall of 1.0 means that every item from class C was labeled as belonging to class C (but says nothing about how many items from other classes were incorrectly also labeled as belonging to class C).

Yet,

In statistical analysis of binary classification, the F1 score [1] is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of correctly identified positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of correctly identified positive results divided by the number of all samples that should have been identified as positive. The F1 score is the harmonic mean of the precision and recall. The highest possible value of F1 is 1, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero.

Model	**Precision**	**Recall**	**F1-Score**
NN	85.503%	81.679%	83.547%

TABLE 5.1: Blabla.

[1]Calculated as: $2\frac{Precision*Recall}{Precision+Recall}$

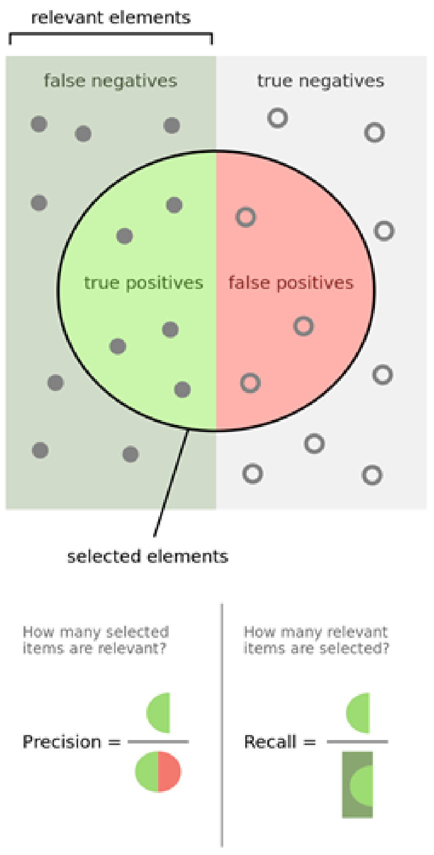

FIGURE 5.1: An illustration of the Cross-Entropy loss function (the image was taken from Wikipedia Precision and Recall Page).

Chapter 6

Discussion

While our preliminary results are promising, both from the textual and hybrid aspects, there is a notable drawback in this approaches, and is the semi-dependency in one language (we term this dependency as as semi one, since the Hebrew language is just a specific use-case, yet the models are trained over one language only).

Bibliography

[1] Zhenyue Qin and Dongwoo Kim. “Rethinking Softmax with Cross-Entropy: Neural Network Classifier as Mutual Information Estimator”. In: *arXiv preprint arXiv:1911.10688* (2019).

[2] Zhenyue Qin and Dongwoo Kim. “Softmax Is Not an Artificial Trick: An Information-Theoretic View of Softmax in Neural Networks”. In: *arXiv preprint arXiv:1910.02629* (2019).