Artificial Intelligence Project Progress Report

By:
Anurag Datta Roy
Rajat Tripathi
Suyash Shetty

Outline:

The topic of our project is *Malware Detection using Artificial Neural Networks.* Malware is any software used to disrupt computer operations, gather sensitive information, gain access to private computer systems, or display unwanted advertising. Our aim is, given any malicious file, to determine what class that particular malware belongs to. To do this, we create a neural network model which trains over a hand-curated dataset of existing malware binaries. The model trains over multiple iterations of forward-backward propagation over the network, at the end of which it converges to a local minima, thereby giving the optimum results.

The intuition behind the application of neural networks for this particular task is simple. Neural Networks with at least one hidden layer can be considered universal approximators. They have sufficient representational power to be able to approximate any continuous function, which makes them optimal for tasks such as pattern recognition.

Dataset:

The bulk of the data in our dataset was retrieved from *theZoo* [1], which is a project that was initiated to make the possibility of malware analysis open and available to the public. The malicious data is provided in two formats: binary and source code. The binary files are live, malware executable files whereas the source codes are C/C++ files which produce a malware object on compilation. We decided to ignore the malware source codes and chose to only work with malware binaries. [1] also provides a configuration file, which is essentially an SQL database dump which was provided to find malwares indexed in the directory. The database has many attributes out of which we extracted only those attributes which we felt were necessary. Attributes such as *Location*, *Name* and *Type* were extracted for each entry in the database and then filtered such that only information about the binaries were retrieved.

Using Python, some preliminary data preprocessing was performed. The binaries were first loaded in Python using the NumPy library. The binary files are stored as a NumPy array of integers in the Python runtime environment. However, since each malware file is different in content and size, the length of the arrays for each malware differs. To combat this, we set a threshold h, which we truncate/pad the arrays to. If the array is longer than h, then we truncate it and if it is shorter than h, then we pad it so it is of length h. Then we reshape the array into a square matrix and normalize the entries so that they lie in the interval [0,255]. This produces a grayscale representation of a malware binary. This grayscale representation

can be used to analyze patterns in the malware files which form the basis for our neural network.

Architecture:
Our rudimentary model is a simple 2-layer neural network with a single hidden layer. It contains one input layer, one hidden layer and one output layer which is a softmax classifier. We pass our grayscale malware binaries to the input layer, whose outputs are sent to the hidden layer, whose outputs are then sent to the output layer.

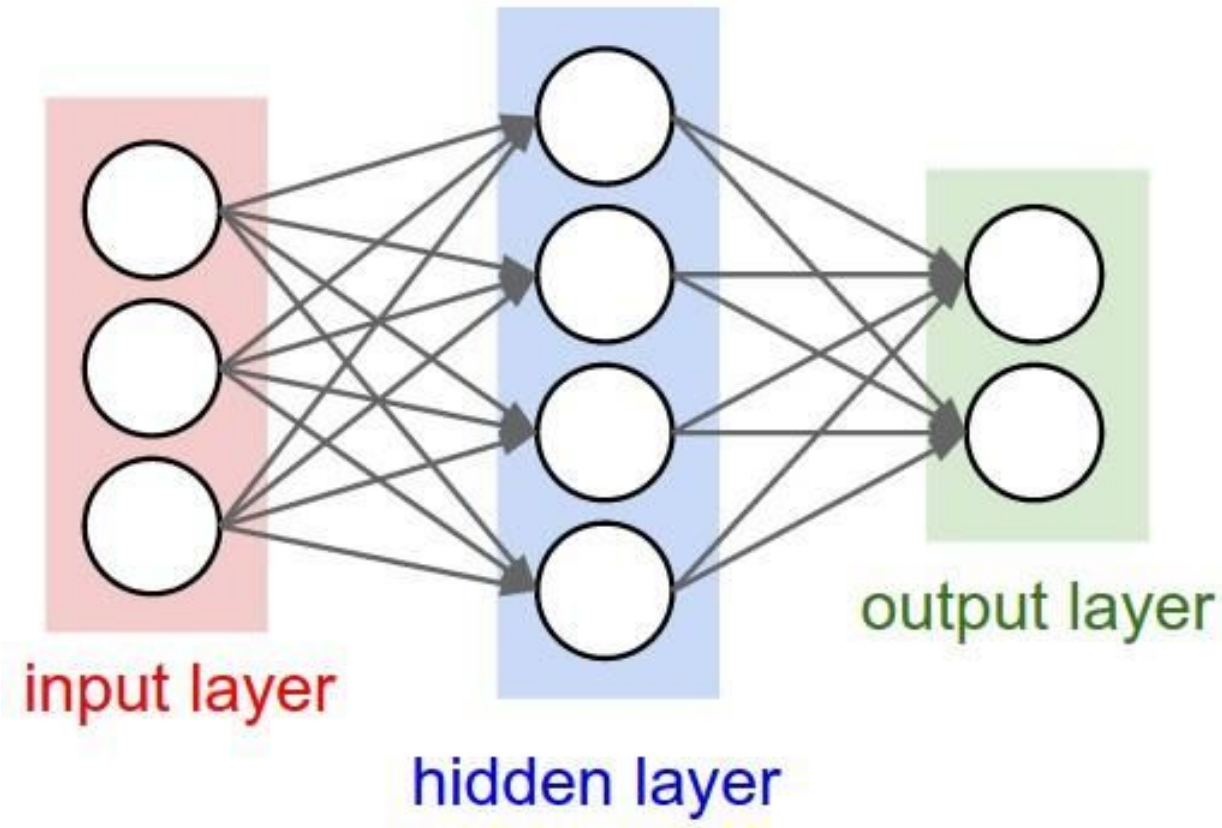

Figure 1: A schematic of 2-layer Neural Network

The input to the network are the grayscale malware matrices which were created in the data preprocessing step. The output of the first layer is, $h = RELU(W_1 * X + b_1)$, where W and B are called weights and biases of the architecture and *relu* [2] is the activation function. The output of the hidden layer is $y = W_2 * h + b_2$. The final output y is then fed into a softmax classifying function which gives us the class that the input belongs to.

This architecture is the first model of our project and most probably not the final model that we settle upon. By intuition and past experiences in neural networks, we understand that even though neural networks are considered to be universal approximators, in practice a network with only a single hidden layer does not provide sufficient representational power.

Proposed Algorithm:

Step 1: Read binaries into 8 bit for each pixel in grayscale image. (Done in preprocessing step.)

Step 2: Resize grayscale image so all images are of same dimensions

Step 3: Feed the resized images into an artificial neural network and perform classification.

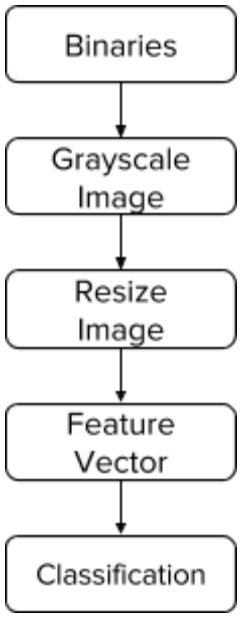

Figure 2: Flow of the proposed algorithm

References:

[1] "theZoo aka Malware DB by ytisf." [Online]. Available: http://thezoo.morirt.com/.

[2] G. E. Hinton, *Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair*.