**ARTIFICIAL INTELLIGENCE AND CYBERSECURITY**

**MIDTERM PROJECT REPORT**

**On**

**Cloud-based PE Malware Detection API**

 By

                                        Yashaswini Gouru(00667970)

**Under the guidance of**

Vahid Behzadan



**Tagliatela College of Engineering**

**Department of Computer Science**

**(Data Science)**

**300 Boston Post Rd, West Haven, CT 06516**

The main idea of this project is to implement and deploy machine learning models for malware classification. The technical implementation of this project is comprised of three main tasks that need to be completed sequentially:

**Task 1 - Training:**

In this we will be creating and training a deep neural network based on the MalConv architecture to classify PE files as malware or benign. As for the dataset, you will be using the EMBER-2017 v2.

( https://github.com/endgameinc/ember ).

**Requirements:**

The model must be implemented in Python 3.x using TensorFlow (1.x or 2.x) and Keras and needs to be coded and documented in a Jupyter Notebook.

**Approach:**

The data has been preprocessed and vectorized to send it into neural network. The vectorization has been done using the ember function present in the link. This vectorization function preprocesses the raw data and return the vectorized data. The whole training process is done in jupyter notebook in the local machine. The vectorizing the data takes about 40min of time.

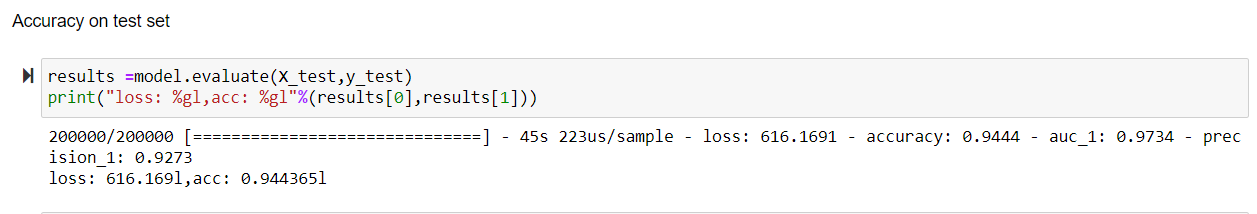
After the vectorization when the data is fed directly into neural network model. I got an accuracy of about 42%, this shows the either the model is not good, or it is due to vast data. First, I took relevant samples (samples not having labels as -1) out of 900000 samples 600000 samples are relevant in training set. Now a standard scalar imported from sklearn is used to scale the data (Preprocessing step).Scaled the data of train set and the test set, I have again sent the scaled data to my model and the accuracy has been increased to 92%. Calculated the accuracy, f1score and precision on test data.

**The Neural Network Model:**

TensorFlow 1x or 2x are used. I have used the 2x version of TensorFlow and the keras is imported. The Input and the dense layers from keras are added to create a neural model. Binary cross entropy as loss and the accuracy as metrics are used while compiling the model. When I try to fit the train data in my model a shape error has occurred due to which I have to reshape the input and the labels data. After reshaping and fitting the data successfully in to my model with a validation split of 0.2, I got an accuracy of about 94% for my training data.

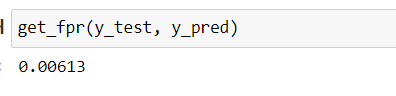
Scaling and reshaping the Test set and the accuracy, precision, Loss and the F1 score, False positive rate is calculated successfully and confusion matrix for benign and malicious samples is also found.

Accuracy:94.40%

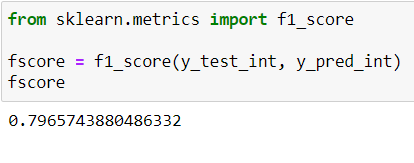


Loss:616

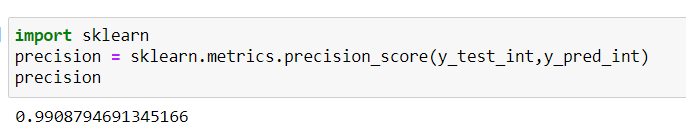
False positive rate: 6.13%



F1 score: 0.7965



Precision:99.08%



The model is saved as json file and the weights are saved as h5 file.

**Task2:** **Deploy your model on the cloud**

### To do this we need an Amazon Sagemaker. Amazon Sagemaker deploy the model on the cloud and create an endpoint (~ API) so that other applications can make use of the model. Amazon Sagemaker imports boto3 client. From keras models we upload the model and the weight files from local drive to keras\_model directory in Sagemaker. Keras model is used to export the loaded files to TensorFlow ProtoBuf format. From TensorFlow model we convert the files to amazon Sagemaker readable format. In this we zip the loaded files to tar.gz format and upload it to S3.

### Now we deploy the trained model in AWS Sagemaker. We take tar.gz file from S3 and create a instance type ml.m4.xlarge. This takes about 6min to complete, and after this the model is finally deployed and an endpoint is created

'sagemaker-tensorflow-2019-08-05-03-29-25-591'

### Testing the model deployment, this can be done either inside Sagemaker or outside Sagemaker. I have done outside Sagemaker and have given access key id and the secret access key to check the model on endpoint.

**Task3:** **Create a client**

Created a python script which already has most of the imports required. The python script has the client connection code which connects to the client and acquires the model in S3. The body of the code is already present in EMBER repository, this files helps to extract the features and the extracted data is sent through the model and the model predicts whether the file is benign or malicious and gives the output as either 0 or 1.The client python file has AWS ID and AWS secret key which has been removed in the code. This file checks whether the model created works on other malware files or not and can be accessed from cloud directly.

**Limitations:**

The main complication of this project is version control. The model gave errors when some versions are used on some packages.

I got errors while deploying when the file model was saved as h5 file instead of json file.

The model was first run on GPU runtime in Google Colab and the RAM collapsed. Next, I have run it on TPU Runtime I got an output I have run the model for 3 or 4 times and then could not connect to TPU. Finally, I run this project on my local Jupyter notebook.

**Conclusion:**

This project helps me in learning more about Neural Networks and how to reshape the data that fits properly into the model. Deploying the model on Sagemaker helps me to know more about builder function and exporting the model on cloud. Even though the project is time taking but interesting it helped me to learn more.

**Bibliography:**

# Endgame Malware Benchmark for Research Paper.

# EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models by [Hyrum S. Anderson](https://arxiv.org/search/cs?searchtype=author&query=Anderson%2C+H+S), [Phil Roth](https://arxiv.org/search/cs?searchtype=author&query=Roth%2C+P)

<https://towardsdatascience.com/malware-detection-using-deep-learning-6c95dd235432>