Cloud-based Deep Learning PE Malware Detection

Sharara Hossain

Problem Statement

This report outlines the process to train a deep convolutional neural network based on the MalConv architecture to classify PE files as malware or benign. The EMBER dataset is used for model training. After training, the model is deployed to the AWS cloud using AWS SageMaker to create an endpoint for local PE file classification. The report covers the steps involved in training, evaluating and packaging the model, uploading it to Amazon S3, creating a SageMaker model object, deploying the model as an endpoint, testing the endpoint, and discusses the implications of the deployment.

Introduction

Malware detection is a critical task in cybersecurity, and machine learning models have shown promise in automating this process. This report documents the deployment of a malware detection model based on the MalConv architecture using Amazon SageMaker, a cloud-based machine learning platform.

PE Files

PE (Portable Executable) format is a common format for executable, object code, DLLs, FON font files, and core dumps in Windows operating systems. PE files contain various data structures, including DOS Header, PE File Header, Image Optional Header, Section Table, Data Dictionaries, and Sections. These files are vulnerable to malicious code injection, making them a prime target for malware attacks.

The Ember datset contains features extracted from PE files. The MalConv architecture was employed for its effectiveness in capturing the sequential nature of byte-level data. The model was implemented using Python 3.x and the PyTorch framework.

MalConv

MalConv is a deep learning architecture designed specifically for the classification of PE files as malware or benign. It addresses the challenge of presenting raw bytes sequentially with linear complexity dependence, enabling scalability with sequence length. By employing convolutional layers, MalConv captures both local and global context, overcoming the positional variance present in executable files.

Dataset

EMBER (Elastic Malware Benchmark for Empowering Researchers) is a benchmark dataset containing features from PE files. The EMBER2017 dataset comprises features from 1.1 million PE files and facilitates feature extraction based on the LIEF project. The files are classed into three categories- malware, benign and unlabeled.

Methodology

1. **Data Processing:** Features extracted using the LIEF feature extractor are normalized using Standard Scaler. This makes sure the GPU is not overloaded. Unlabeled data is discarded. The input arrays are then reshaped and transformed into byte sequences. The training set consists of 800k data points and for testing 200k data points are separated.
2. **Training the MalConv Model:** The MalConv model is constructed using PyTorch framework. The model consists of two Conv layers and a gated layer along with a maxpooling layer. A Sigmoid activation function is employed at the classification layer, and the model is trained with binary cross-entropy loss and a learning rate of 0.001. It is trained for 13 epochs.

```
MalConv(
  (embed): Embedding(256, 8)
  (filter): Conv1d(8, 128, kernel_size=(512,), stride=(512,))
  (attention): Conv1d(8, 128, kernel_size=(512,), stride=(512,))
  (fc1): Linear(in_features=128, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=1, bias=True)
  (relu): ReLU()
  (maxpool): AdaptiveMaxPool1d(output_size=1)
  (sigmoid): Sigmoid()
)
```

3. **Evaluating the Model Performance:** The model is serialized to a pickle file after every epoch. Binary cross entropy loss is calculated after each epoch. Finally the model is evaluated over the test dataset for accuracy, precision and recall.

```
Epoch 3, Training Loss: 0.1767809932060089
Validation Loss: 0.13538787926559137
Model checkpoint saved to vMalConv/model_epoch_3.pt
Epoch 4, Training Loss: 0.11798364543332684
Validation Loss: 0.12117938681835125
Model checkpoint saved to vMalConv/model_epoch_4.pt
Epoch 5, Training Loss: 0.09826269757171954
Validation Loss: 0.12156597245816714
Model checkpoint saved to vMalConv/model_epoch_5.pt
Epoch 6, Training Loss: 0.08822460069890646
Validation Loss: 0.1301822431996254
Model checkpoint saved to vMalConv/model_epoch_6.pt
Epoch 7, Training Loss: 0.08110414630870325
Validation Loss: 0.1314195684100628
Model checkpoint saved to vMalConv/model_epoch_7.pt
Epoch 8, Training Loss: 0.07405904789271234
Validation Loss: 0.1327621054663003
Model checkpoint saved to vMalConv/model_epoch_8.pt
Epoch 9, Training Loss: 0.0707812625763994
Validation Loss: 0.16424127275045428
Model checkpoint saved to vMalConv/model_epoch_9.pt
Epoch 10, Training Loss: 0.06669016188833633
Validation Loss: 0.14050454745732077
Model checkpoint saved to vMalConv/model_epoch_10.pt
Epoch 11, Training Loss: 0.06326528408428231
Validation Loss: 0.17956580665959623
Model checkpoint saved to vMalConv/model_epoch_11.pt
Epoch 12, Training Loss: 0.06304482300018412
Validation Loss: 0.13598500096777388
Model checkpoint saved to vMalConv/model_epoch_12.pt
Epoch 13, Training Loss: 0.062024606894945114
Validation Loss: 0.16019812103914685
Model checkpoint saved to vMalConv/model_epoch_13.pt
```

Epochs	Threshold	Accuracy	Precision	Recall
13	predicted.round()	0.5	0.4706	0.0002
13	predicted > 0.5	0.4990	0.4995	0.9961
8	predicted.round()	0.5005	0.5002	0.9996
8	predicted > 0.5	0.5073	0.5044	0.8351

4. **Deploying the Model as AWS SageMaker Endpoint:** The saved model is deployed in AWS SageMaker, and an endpoint is created.

Results and Discussion

The trained model achieves an accuracy of 50% on the testing dataset. Challenges faced include issues calling the SageMaker endpoint, possibly due to computation limitations.

For testing the model performance I experimented with two thresholding methods and compared two model checkpoints. The model seems to score about as good as a uniform classifier would, which is a severe underperformance for a deep learning algorithm. The model precision is not very good either but it seems to be doing well in the recall metric, meaning most actual positive samples were identified correctly.

Going back to model training, validation loss is notably higher than the training loss, indicating that the model may be overfitting to the training data. Regularization may increase model performance.

An important note here is that the model was trained on malware/benign classes and the test data may contain unlabeled class data as well which could have affected the model performance. At the end of this experiment the 3rd model is chosen to proceed to the next steps.

The model deployment was not successful because of resource limitations.

Conclusion

The project successfully trains a MalConv model using the EMBER dataset and deploys it on AWS SageMaker. Deploying the model in the cloud enables end-users to identify malware in PE files efficiently. Further optimization of hyperparameters and addressing endpoint call issues can enhance model performance. The code implementation is available on GitHub.

References

1. EMBER dataset: https://github.com/elastic/ember
2. LIEF project: https://lief-project.github.io/
3. MalConv paper: https://arxiv.org/abs/1710.09435
4. Malware Classification paper: https://arxiv.org/abs/1804.04637