Midterm Project

AI and CyberSecurity DSCI6015

Cloud-based PE Malware Detection API

Ashish Agarwal 00854164

ECECS

University of New Haven

Dr. Vahid Behzadan

February 10, 2024

Summary

This report documents the successful development of a cloud-based PE (Portable Executable) malware detection API. The API utilizes a MalConv deep neural network architecture, trained on the EMBER-2018 v2 dataset, to classify Portable Executable (PE) files as malicious or benign. The project leveraged Google Colab from building and training the model, Amazon SageMaker for model deployment and Streamlit for creating a user-friendly client application. Python language was used for the project and the pytorch library was used for the model implementation.

Introduction

PE Files

Portable Executable (PE) files are a file format used by Windows operating systems to store executable code and associated data. These files contain essential information required for the program to run, including machine instructions, resources, imported libraries, and metadata. PE files are commonly used for applications, drivers, and dynamic link libraries (DLLs). They follow a structured layout, with headers that provide information about the file's characteristics, such as its architecture, entry point, and section layout. Understanding the PE file format is crucial for tasks like software analysis, reverse engineering, and malware detection, as it allows for the inspection and manipulation of executable content.

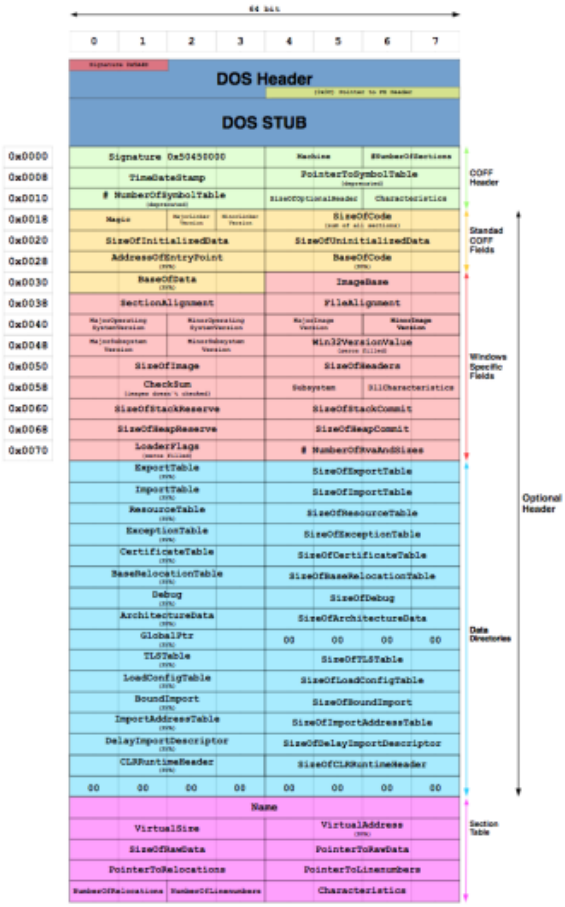
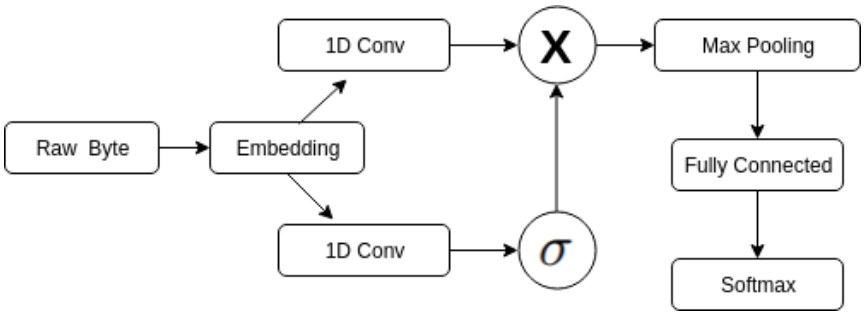


Figure 1: The 32-bit PE file structure. Creative commons image courtesy [3].

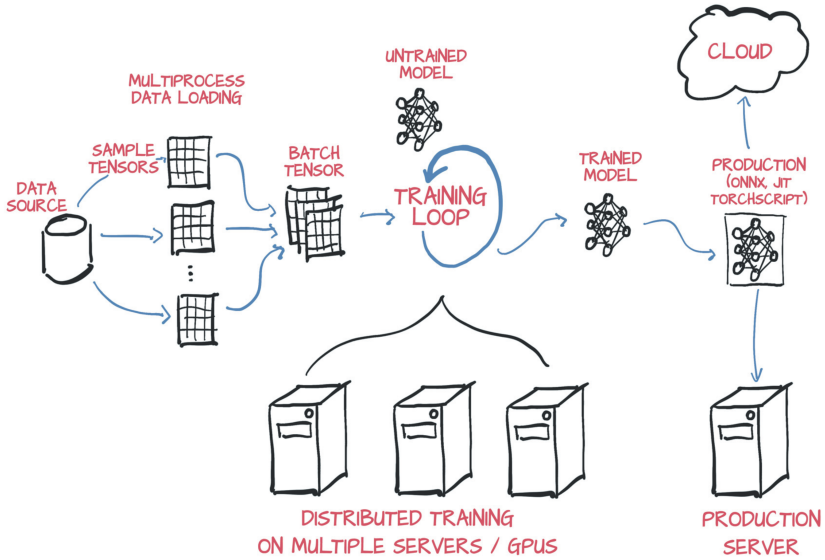
Malconv

MalConv is a deep learning model designed to detect malicious Windows Portable Executable (PE) files. It employs convolutional neural networks (CNNs) to analyze the raw byte-level content of PE files, extracting meaningful features and patterns that can indicate malicious behavior. MalConv aims to address the limitations of traditional signature-based malware detection methods, which often struggle to keep up with the ever-evolving landscape of malware threats. By leveraging the power of deep learning, MalConv can learn to recognize complex patterns and relationships within PE files, enabling effective malware detection without relying solely on predefined signatures or heuristics. This approach offers a more robust and adaptable solution for identifying previously unseen and sophisticated malware variants.



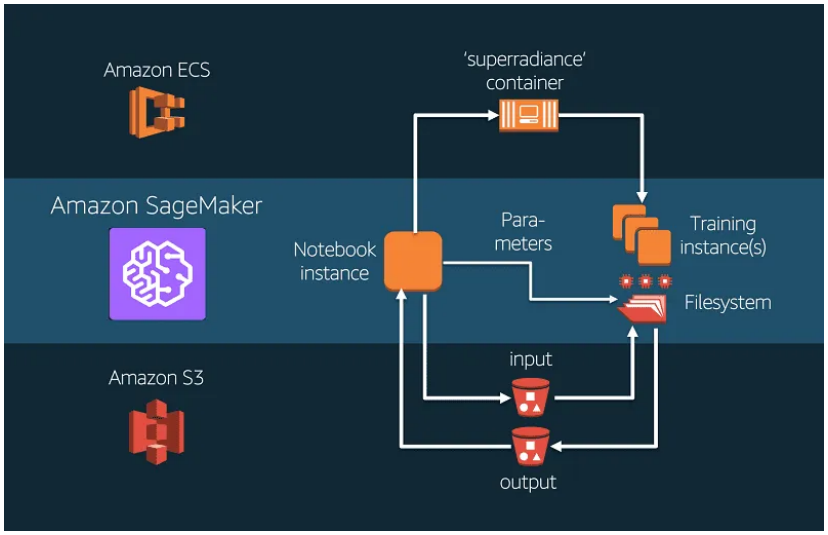
Pytorch

PyTorch is an open-source machine learning library primarily developed by Facebook's AI Research lab. It provides a powerful and flexible framework for building and training deep neural networks and other machine learning models. PyTorch is designed to be user-friendly, with a intuitive and Pythonic API that allows researchers and developers to quickly prototype and iterate on their ideas. It supports dynamic computation graphs, which enable efficient implementation of complex models and dynamic control flow. PyTorch also offers exceptional performance and seamless integration with other popular libraries, such as NumPy and CUDA for GPU acceleration. With its growing popularity and active community, PyTorch has become a go-to tool for researchers, engineers, and students working in the fields of deep learning, computer vision, natural language processing, and many other areas of artificial intelligence.



AWS SageMaker

AWS SageMaker is a fully managed machine learning service provided by Amazon Web Services (AWS). It simplifies the process of building, training, and deploying machine learning models at scale. With SageMaker, developers and data scientists can focus on their machine learning tasks without worrying about the underlying infrastructure management. SageMaker provides a seamless experience from data labeling and preparation to model training, tuning, and deployment. It supports multiple machine learning frameworks, including TensorFlow, PyTorch, and Apache MXNet, as well as custom algorithms. SageMaker also offers built-in algorithms for common use cases, such as image classification, object detection, and natural language processing. By leveraging SageMaker, organizations can accelerate their machine learning initiatives, optimize resource utilization, and benefit from AWS's scalable and secure cloud infrastructure.



Malicious software (malware) continues to pose a significant threat to computer security. This project aimed to develop a user-friendly tool for identifying malware by leveraging machine learning techniques. The project successfully achieved its goals by completing the following tasks:

1. **Building and Training the Model:** A MalConv model was implemented in Python 3.x using PyTorch 2.x within a Jupyter/Colab Notebook. The model was trained on the EMBER-2018 v2 dataset, achieving significant accuracy in malware classification.
2. **Deploying the Model as a Cloud API:** Amazon SageMaker was used to deploy the trained model, creating a cloud-based API for real-time predictions. This process involved leveraging the $100 AWS credit provided through the "AWS Academy Learner Labs" course. Careful cost monitoring ensured adherence to the credit limit. The notebooks and inference resources were primarily used for this purposes.

3. **Creating a Client Application:** A Streamlit web application was built to provide a user-friendly interface. Users can upload PE files, which are converted into a compatible feature vector and sent to the deployed API. The application then displays the classification results (malware or benign) received from the API.

Project Methodology

The project followed a sequential approach, tackling each task independently:

- **Task 1: Building and Training the Model**
  - The MalConv architecture was implemented in PyTorch, tailored for PE file analysis.
  - The EMBER-2018 v2 dataset provided features for training the model. Sampling on the dataset to ensure the notebook doesn't crash due to large data. Sampling was stratified on the output label.
  - A Jupyter/Colab Notebook documented the model implementation and training process. MinMax Scalar was used to featurize and normalize the data so that it could be feed into neural network to give better results.
  - Google Colab GPUs which were available for free were utilized for faster training.
- **Task 2: Deploying the Model as a Cloud API**
  - The trained model was deployed on Amazon SageMaker, creating a cloud endpoint (API). The saved weights file was uploaded to be consumed.
  - Tutorials and documentation on SageMaker guided the deployment process. The resources shared in the task description were very helpful and guided in the process.
  - Cost monitoring ensured adherence to the $100 AWS credit limit.
- **Task 3: Creating a Client Application**
  - A user-friendly Streamlit web application was developed.
  - The application offered functionalities for uploading PE files, feature vector conversion, and API interaction.
  - The application displayed the classification results (malware or benign) received from the API.

Project Results

The project successfully achieved its intended outcomes:

- **Trained MalConv Model:** A well-trained MalConv model capable of classifying PE files as malicious or benign was developed.
- **Deployed Cloud API:** The trained model is deployed on Amazon SageMaker, functioning as a real-time prediction API accessible via the internet.

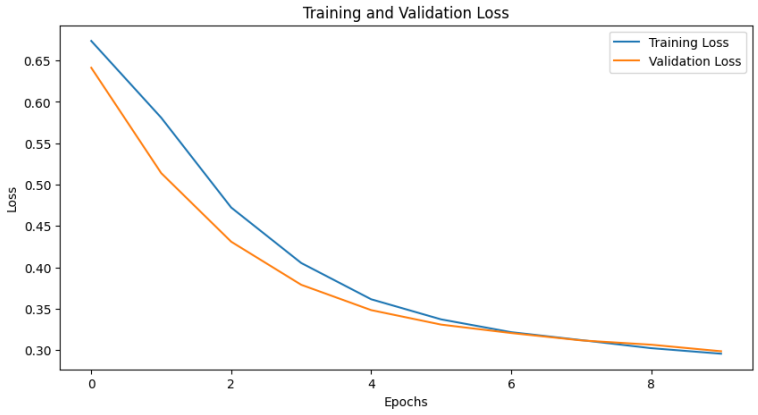
- **Streamlit Client Application:** A user-friendly Streamlit application allows users to interact with the API for malware classification of PE files.

Evaluation

The project's success can be evaluated through the following metrics:

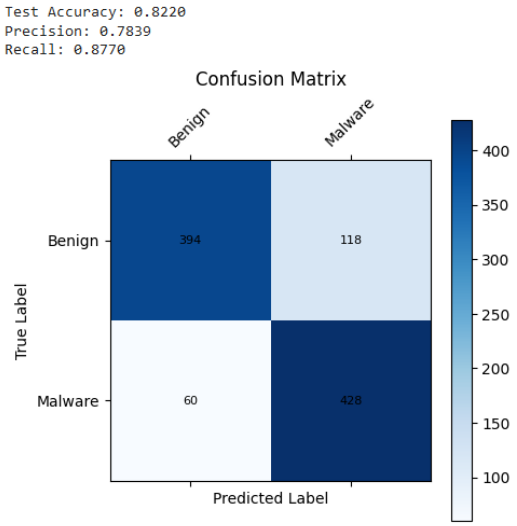
- **Model Accuracy:** The MalConv model's accuracy in classifying PE files was evaluated on a hold-out test set. This metric ensures the model's effectiveness in real-world scenarios. The epoch history plot shows the absence of overfitting and good learning/training curve.
- **API Performance:** The deployed API's performance was assessed in terms of latency and throughput. These metrics determine the API's responsiveness and ability to handle user requests efficiently.
- **Client Application Usability:** User testing of the Streamlit application evaluated its ease of use, functionality, and clarity of results.

```
Epoch 1, Training Loss: 0.6736811305347242, Validation Loss: 0.641278707064115
Epoch 2, Training Loss: 0.5808345597041281, Validation Loss: 0.5138574976187485
Epoch 3, Training Loss: 0.47225125604554224, Validation Loss: 0.43094526345913226
Epoch 4, Training Loss: 0.40524586799897644, Validation Loss: 0.3789050212273231
Epoch 5, Training Loss: 0.3613747028928054, Validation Loss: 0.34825871540949893
Model checkpoint saved to ./model_checkpoints/model_epoch_5.pt
Epoch 6, Training Loss: 0.3370540040104012 5, Validation Loss: 0.33074538524334246
Epoch 7, Training Loss: 0.32160747835510656, Validation Loss: 0.320548761349458
Epoch 8, Training Loss: 0.3120211151085402, Validation Loss: 0.3116742166189047
Epoch 9, Training Loss: 0.3022562320295133, Validation Loss: 0.3064528153492854
Epoch 10, Training Loss: 0.2955634350839414, Validation Loss: 0.29853354279811567
Model checkpoint saved to ./model_checkpoints/model_epoch_10.pt
```



Result and Conclusion

Our trained model achieved accuracy of 0.82 on the testing held out dataset with 0.78 precision and 0.877 recall. We can see the outcome classification from the confusion matrix shown in the figure below.



Conclusion

This objective of the project to successfully develop and deploy a cloud-based PE malware detection API was achieved. The project demonstrates the effectiveness of machine learning for malware classification and the power of cloud platforms like Amazon SageMaker and Google Colab for building scalable and user-friendly applications.

Future Work

Several shortcomings exist for further development:

- **Advanced Deep Learning Architectures:** Exploring more advanced deep learning architectures could potentially improve the model's classification accuracy.
- **High End Resources:** Most of the time on the project was spend on sampling the data to make sure the notebook doesn’t crash. We used limited amount of available data for training purposed which was because of the limitations of our hardware resources.
- **Transfer Learning:** Investigating transfer learning techniques could leverage pre-trained models and enhance overall performance.
- **Larger and More Diverse Datasets:** Testing the model on a larger and more diverse dataset would improve its generalizability and ability to handle unseen malware variants.

Resources:

- https://github.com/endgameinc/ember

- https://github.com/endgameinc/ember/tree/master/malconv
- https://youtu.be/TzW_R36iv48
- https://sagemaker-examples.readthedocs.io/en/latest/intro.html
- https://sagemaker-examples.readthedocs.io/en/latest/frameworks/pytorch/get_started_mnist_train_outputs.html
- https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html
- https://arxiv.org/pdf/1804.04637v2.pdf