Midterm Project



**AI and CyberSecurity DSCI6015**

**Cloud-based PE Malware Detection API**

**Sayesh Kumar Chittoor**

**University of New Haven**

**Dr. Vahid Behzadan**

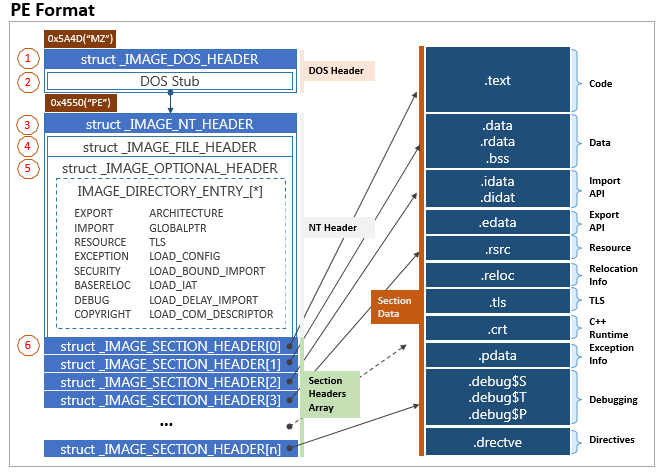
# Summary

The construction of a cloud-based PE (Portable Executable) malware detection API is successfully documented in this study. The Portable Executable (PE) file classification system used by the API is based on a deep neural network architecture called MalConv, which was trained using the EMBER-2018 v2 dataset. Streamlit was used to create an intuitive client application, Amazon SageMaker was utilized for model deployment, and Google Colab was utilized for model construction and training. The project was conducted using the Python programming language, and the Pytorch library was employed to create the model.

# Introduction

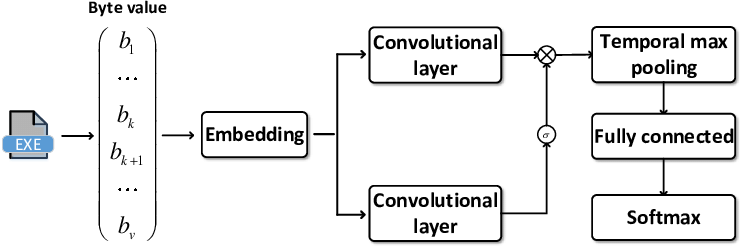
## PE Files Format

Operating systems for Windows employ a file format called Portable Executable (PE) to hold executable code and related data. Machine instructions, resources, imported libraries, and metadata are among the crucial details found in these files that are needed for the program to function. For drivers, programs, and dynamic link libraries (DLLs), PE files are frequently utilized. Their arrangement is methodical, featuring headers that offer details on the attributes of the file, like its structure, entry point, and group arrangement. Due to its ability to allow for the inspection and manipulation of executable material, the PE file format is essential for tasks like software analysis, reverse engineering, and virus detection.



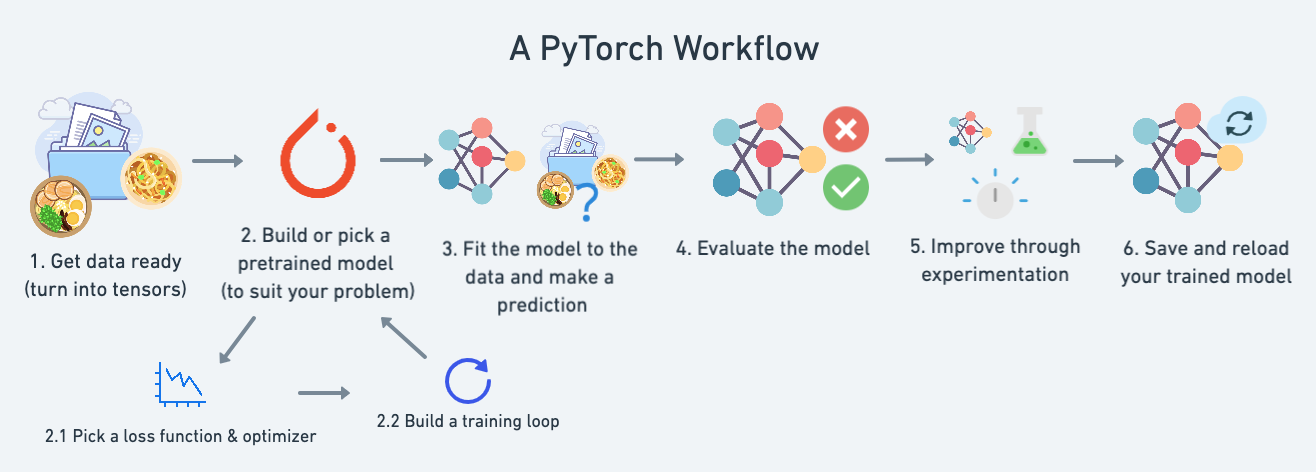
## Malconv Functionality

A deep learning model called MalConv was created to identify Windows Portable Executable (PE) files that are malicious. In order to extract relevant features and patterns that may point to harmful activity from the raw, byte-level content of PE files, it uses convolutional neural networks (CNNs).   
 MalConv seeks to overcome the shortcomings of conventional signature-based malware detection techniques, which frequently find it difficult to stay up to date with the constantly changing threat landscape of malware. MalConv can identify intricate patterns and correlations in PE files by utilizing deep learning, which frees it from the need for pre-established signatures or heuristics and allows for successful malware detection. This method provides a more reliable and flexible way to detect malware variants that are more complex and have never been seen before.



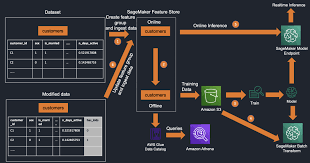
## Pytorch Overview

Facebook's AI Research Lab is the primary developer of PyTorch, an open-source machine learning package. For creating and honing deep neural networks and other machine learning models, it offers a strong and adaptable architecture. PyTorch is an easy-to-use platform that enables academics and developers to rapidly experiment and refine their ideas through an intuitive and Pythonic API. The effective implementation of complex models and dynamic control flow are made possible by its support for dynamic computation graphs. Together with its outstanding performance, PyTorch integrates easily with other well-known libraries, such CUDA and NumPy for GPU acceleration. PyTorch is now a required tool for researchers, developers, and students working in the domains of deep learning, computer vision, natural language processing, and other areas of AI.



## AWS SageMaker Functinality

AWS Amazon Web Services (AWS) offers a machine learning service called SageMaker that is fully managed. Building, training, and implementing machine learning models at scale is made easier by it. Developers and data scientists can concentrate on their machine learning jobs with SageMaker, as it takes care of the underlying infrastructure management for them. From data labeling and preparation to model training, tuning, and deployment, SageMaker offers a seamless experience. Along with bespoke algorithms, it supports several machine learning frameworks, such as TensorFlow, PyTorch, and Apache MXNet. Additionally, SageMaker comes with built-in algorithms for typical use cases like object detection, image categorization, and natural language processing. Organizations may take advantage of AWS's secure and scalable cloud architecture, expedite their machine learning activities, and maximize resource utilization by utilizing SageMaker.



Pernicious programming (malware) keeps on representing a critical danger to PC security. This venture intended to foster an easy to understand device for distinguishing malware by utilizing machine learning methods. The undertaking effectively accomplished its objectives by finishing the accompanying jobs:

**1. Building and Preparing the Model:** A MalConv model was carried out in Python 3.x utilizing PyTorch 2.x inside a Jupyter/Colab Scratch pad. The model was prepared on the Ash 2018 v2 dataset, accomplishing critical exactness in malware grouping.

**2. Conveying the Model as a Cloud Programming interface:** Amazon SageMaker was utilized to send the prepared model, making a cloud-based Programming interface for constant expectations. This interaction included utilizing the $100 AWS credit gave through the "AWS Institute Student Labs" course. Cautious expense checking guaranteed adherence to as far as possible. The scratch pad and deduction assets were fundamentally

1. **Creating a Client Application:** A Streamlit web application was attempted to give a client pleasant mark of communication. Clients can move PE records, which are changed over into a feasible component vector and sent off the conveyed Programming point of interaction. The application then shows the gathering results (malware or innocuous) got from the Programming point of interaction.

# Project Approach

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

### Task 1: Building and Training the Model

* With an emphasis on PE file analysis, the MalConv architecture was implemented in PyTorch.
* Features for model training were obtained from the EMBER-2018 v2 dataset. Samples are taken from the dataset to make sure there are no huge data crashes on the notebook. On the output label, sampling was categorized.
* The model's deployment and training were recorded in a Jupyter/Colab notebook. To enhance the quality of the output from the neural network, the data was first featurized and normalized using MinMax Scalar.
* Faster training was achieved by utilizing free Google Colab GPUs.

### Task 2: Deploying the Model as a Cloud API

* In order to create a cloud endpoint (API), the trained model was installed on Amazon SageMaker. It was uploaded and devoured, the saved weights file.
* Deployment was led by SageMaker tutorials and documentation. The information provided were very beneficial and provided guidance throughout the endeavor.
* The $100 AWS credit limit was adhered to with cost monitoring.

### Task 3: Creating a Client Application



* An easy-to-use online application called Streamlit was created.
* The program provided features for interacting with APIs, converting feature vectors, and uploading PE files.
* The program showed the malware or benign classification findings that it had obtained from the API.

# Project Outcomes

The project's desired results were effectively realized:

**MalConv Model:** A sophisticated model of MalConv was created that is able to identify if PE files are harmful or not.

**Cloud API:** The trained model is implemented as a real-time prediction API that can be accessed online through Amazon SageMaker, which is the cloud platform on which it is installed.   
  
**Client Application:** Users can interact with the API for malware categorization of PE files through the Streamlit Client Application, which is an intuitive application.

# Evaluation

Using the following metrics, the project's success may be assessed:

**Model Accuracy:** A hold-out test set was used to assess the correctness of the MalConv model in categorizing PE files. The model's efficacy in practical situations is guaranteed by this measure. The lack of overfitting and a good learning/training curve are demonstrated by the epoch history plot.   
**API Efficiency:** The latency and throughput of the implemented API were evaluated. The responsiveness and effectiveness of the API in managing user requests are assessed by these metrics.   
**Client Application Usage**: User testing was conducted to assess the usability, functionality, and clarity of findings of the Streamlit application for clients.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Results And Conclusion**

Our trained model achieved accuracy of 0.59 on the testing held out dataset with 0.56 precision and 0.813 recall. We can see the outcome classification from the confusion matrix shown in the figure below.

A screenshot of a computer

Description automatically generated

# Conclusion

This project goal of developing and successfully deploying a cloud-based malware detection API was achieved. The project demonstrates the effectiveness of machine earning in malware classification and the power of cloud platforms such as Amazon SageMaker and Google Colab to create scalable and usable applications.