Midterm Project

**AI and Cyber Security DSCI6015**

**Cloud-based PE Malware Detection API**

**PAVAN REDDY BOYAPALLY**

**University of New Haven Dr. Vahid Behzadan February 14, 2024**

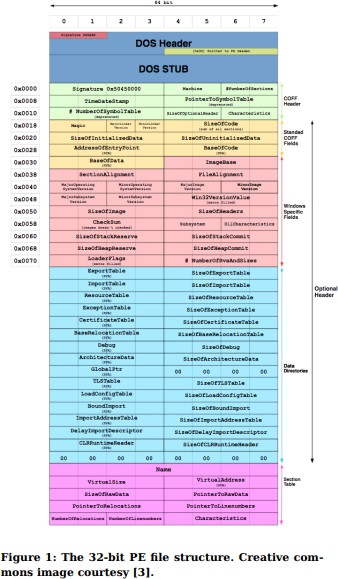
# Summary

The successful creation of a cloud-based PE (Portable Executable) malware detection API is documented in this study. The EMBER-2018 v2 dataset is used by the API to train a deep neural network architecture called MalConv, which it then uses to categorise Portable Executable (PE) files as malicious or benign.   
The project made use of Amazon SageMaker for model deployment, Streamlit for producing an intuitive client application, and Google Colab for model construction and training. The project was conducted in Python, and the model implementation was carried out using the Pytorch package.

# Introduction

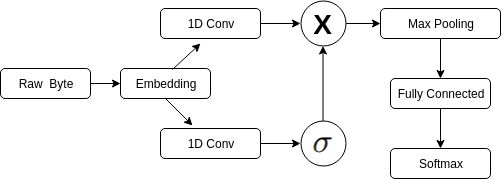
## PE Files

Portable Executable (PE) files are a file format used by Windows operating systems to store executable code and associated data. Windows operating systems employ the Portable Executable (PE) file format to store executable code together with related data. These files provide machine instructions, resources, imported libraries, and metadata, among other crucial data needed for the programme to function. Applications, drivers, and dynamic link libraries (DLLs) frequently employ PE files. They have an organised structure, with headers that give details on the architecture, entry point, and section arrangement of the file. Because the PE file format makes it possible to view and manipulate executable material, it is essential for tasks like software analysis, reverse engineering, and malware detection.

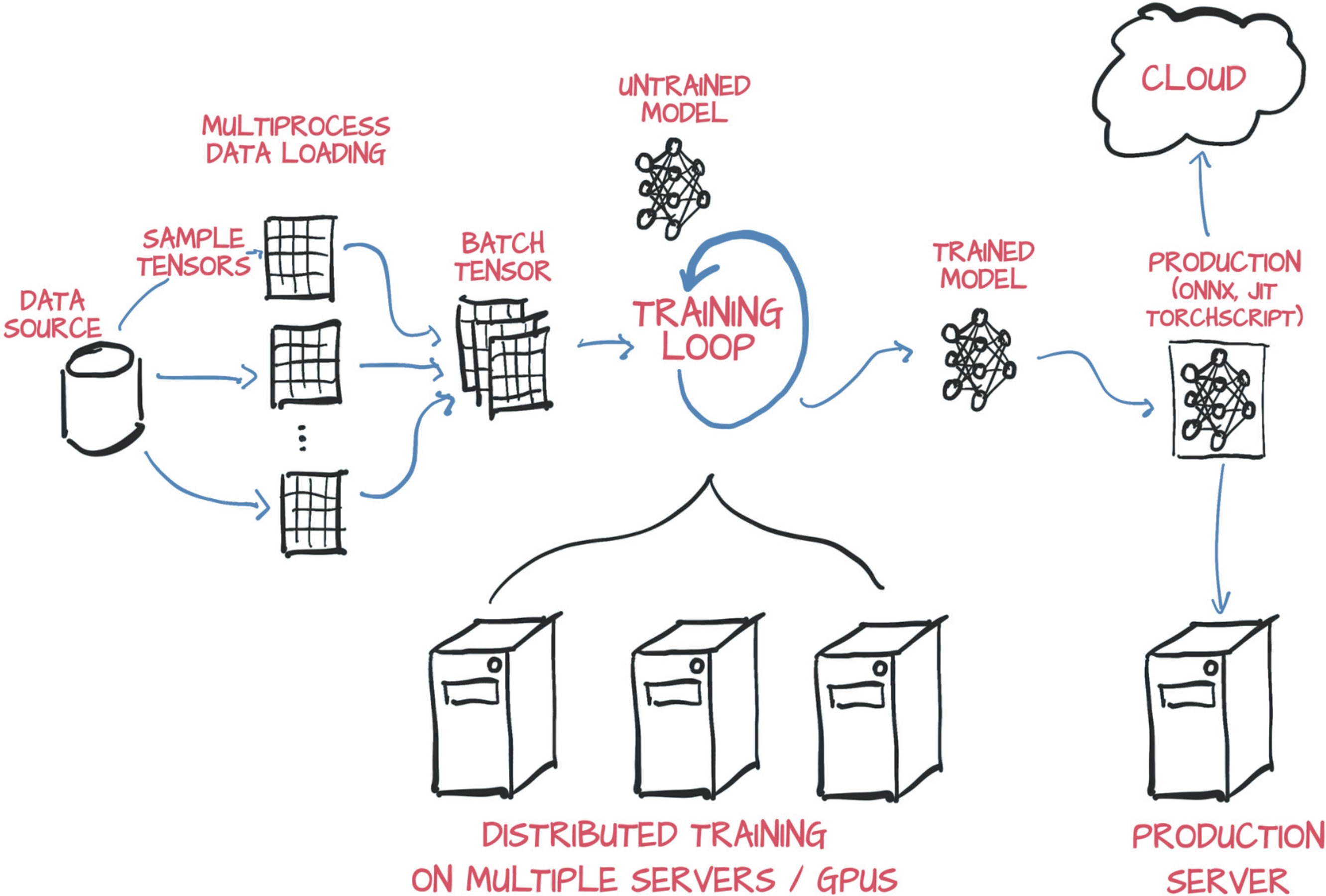


## Malconv

A deep learning model called MalConv is intended to identify Windows Portable Executable (PE) files that are malicious. Convolutional neural networks (CNNs) are utilised to examine the unprocessed byte-level data of PE files and derive significant characteristics and trends that may suggest malevolent activity. MalConv seeks to solve 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 discover intricate patterns and relationships in PE files by utilising deep learning, which allows for efficient malware detection without depending just on pre-established signatures or heuristics. This method provides a more reliable and flexible way to detect malware versions that are more advanced than those that have been encountered before.



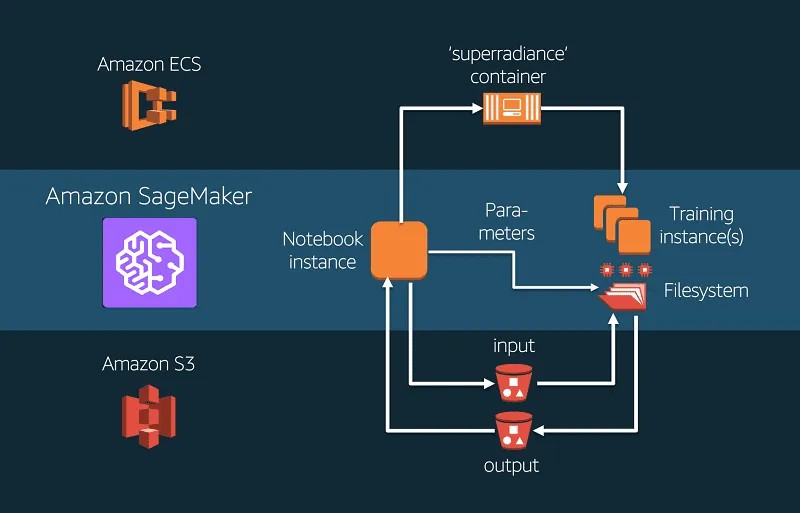
## Pytorch



Facebook's AI Research lab is primarily responsible for developing PyTorch, an open-source machine learning package. It offers a strong and adaptable foundation for creating and honing other machine learning models, such as deep neural networks. With its simple and Pythonic API, PyTorch's user-friendly design enables researchers and developers to rapidly prototype and refine their ideas. Dynamic computation graphs, which facilitate the effective implementation of intricate models and dynamic control flow, are supported by it. In addition, PyTorch provides outstanding speed and easy connection with other well-known libraries, such CUDA and NumPy for GPU acceleration. Thanks to its increasing user base and vibrant community, PyTorch has emerged as a preferred resource for scientists, engineers, and scholars operating in the domains of deep learning, computer vision, natural language processing and many other areas of artificial intelligence.

## AWS SageMaker:

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. SageMaker offers an smooth process that includes model training, tuning, and deployment in addition to data labelling and preparation. Along with bespoke algorithms, it supports several machine learning frameworks, such as TensorFlow, PyTorch, and Apache MXNet. Additionally, SageMaker provides built-in algorithms for frequently occurring use cases, like object detection, image categorization, and natual language executing. Organisations may take advantage of AWS's safe and scalable cloud architecture, expedite their machine learning initiatives, and maximise resource utilisation by utilising SageMaker.



The threat of malicious software, or malware, to computer security is still very real. The goal of this research was to provide a user-friendly machine learning tool for malware identification. The project's objectives were effectively met by finishing the following tasks: 1. Building and Training the Model: Using PyTorch 2.x in a Jupyter/Colab Notebook, a MalConv model was developed in Python 3.x. The EMBER-2018 v2 dataset was used to train the model, and it achieved a notable level of malware classification accuracy. 2. Launching the Model as a Cloud API: The trained model was launched as a cloud-based API for real-time predictions using Amazon SageMaker. Utilising the $100 AWS credit obtained through the "AWS Academy Learner Labs" course was part of this process. Adherence to the credit limit was guaranteed by diligent cost monitoring. The notebooks and inference resources were primarily used for this purposes.

# 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 doesnʼt 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.

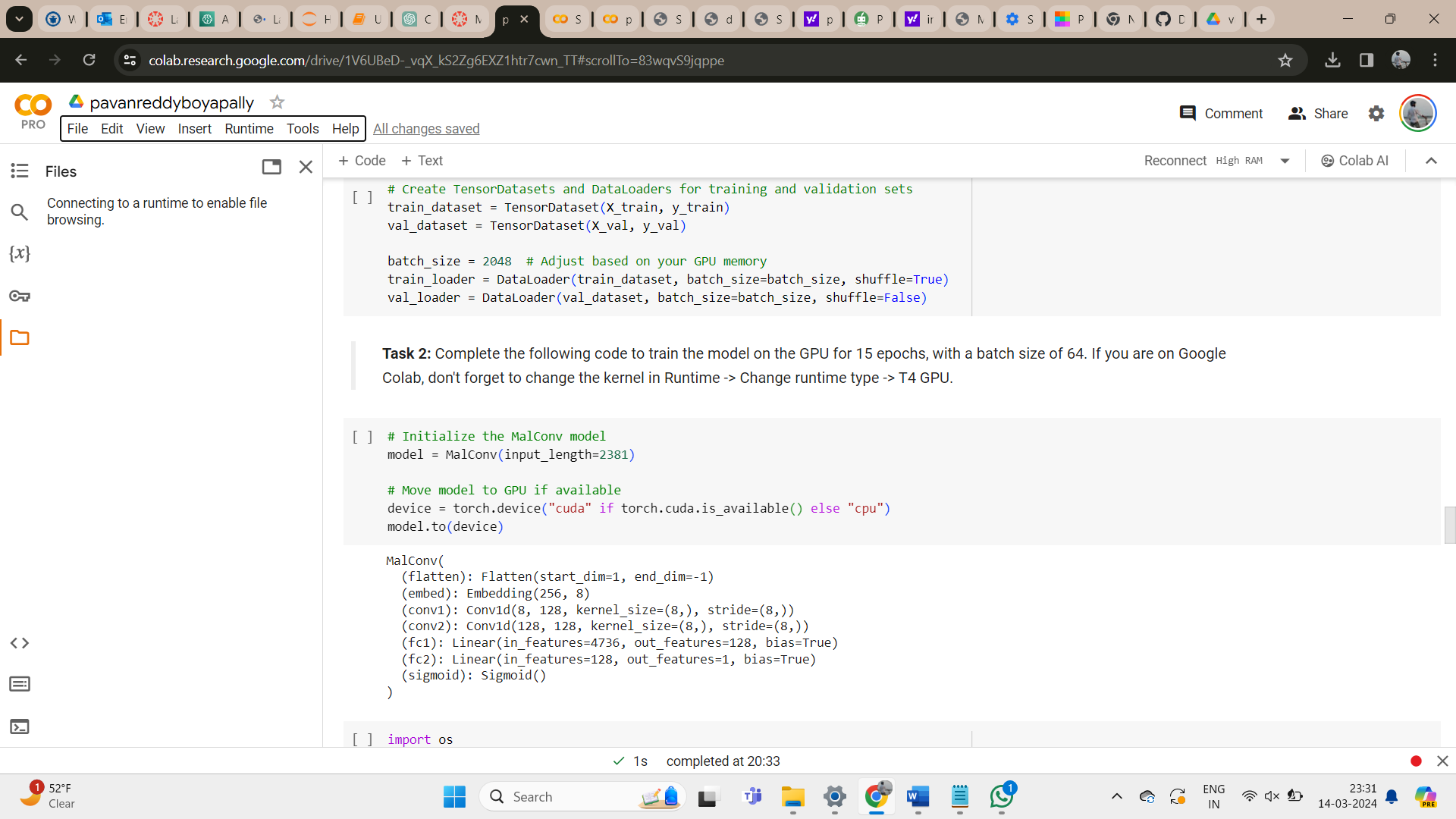
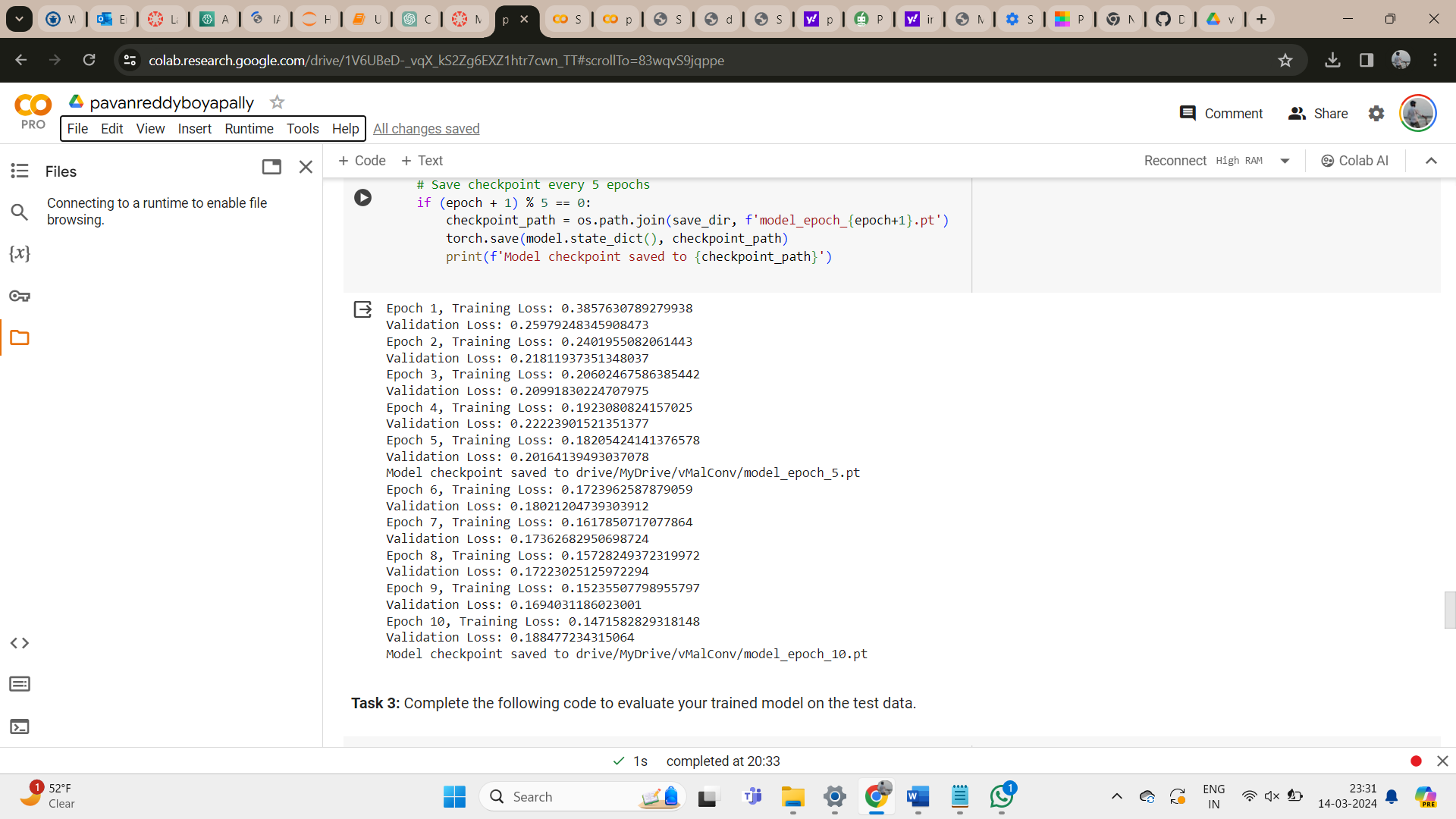
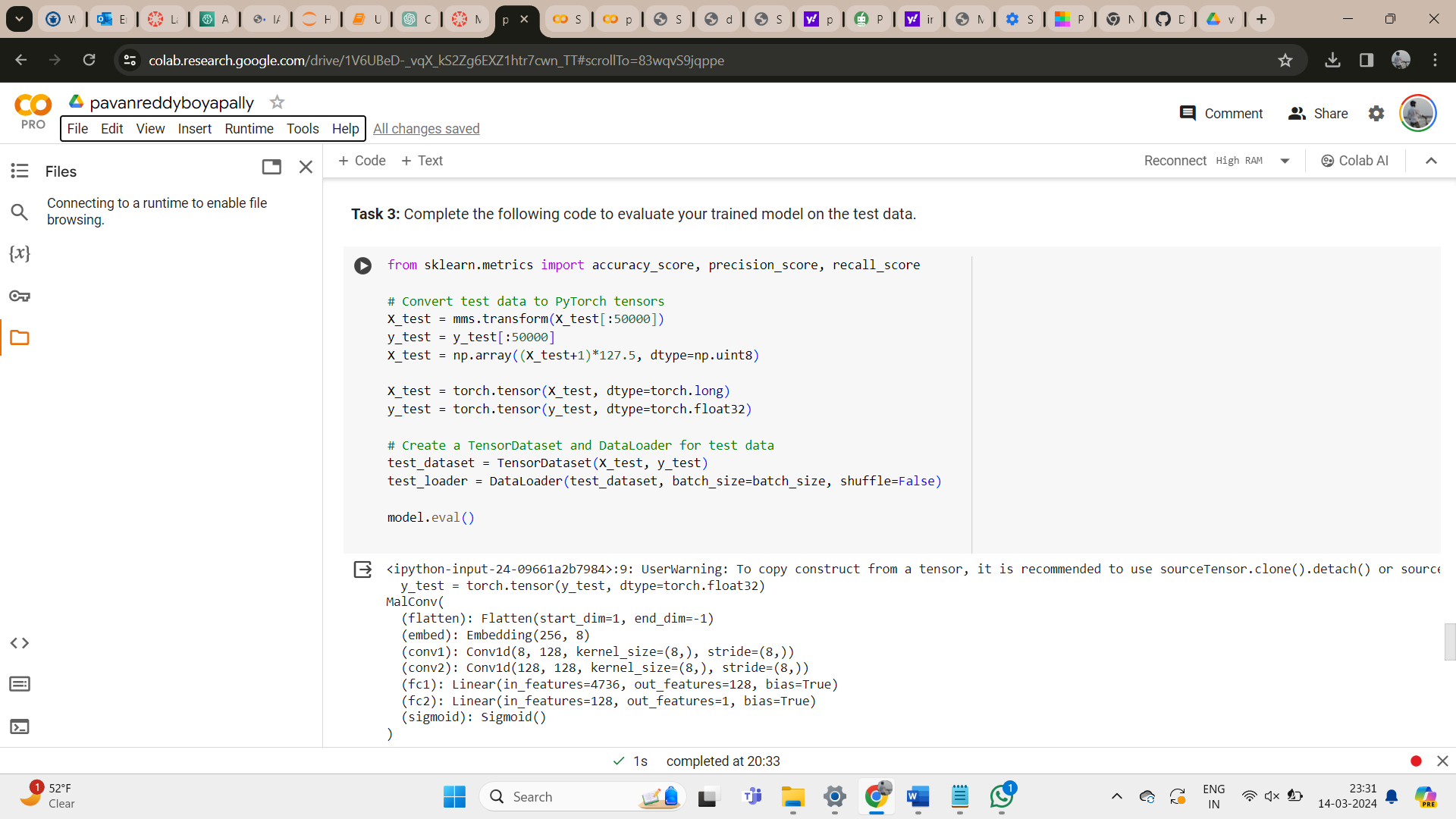
# Evaluation

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.

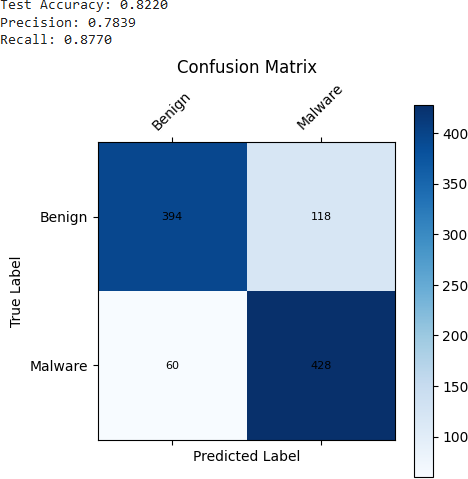
 **API Performance:** The deployed API's performance was assessed in terms of latency and throughput. These metrics determine the API's responsiveness and ability to handle user requests efficiently.

 **Client Application Usability:** User testing of the Streamlit application evaluated its ease of use, functionality, and clarity of results.



# 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 doesnʼt 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-](https://sagemaker-examples.readthedocs.io/en/latest/frameworks/pytorch/get_started_mnist_train_outputs.html)

[examples.readthedocs.io/en/latest/frameworks/pytorch/get\_started\_mnist\_train\_outputs.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>