**Resist Group Campus A**: **Image Classification Platform Using Convolutional Neural Networks**

# 1. Introduction

* NGIEKA ARIANE WHITNEY
* NDI WALTER KONI
* GLORY KINKANG BONGBUNRI

### 1.1 Objective

This project aims to design and develop an end-to-end image classification platform using a trained Convolutional Neural Network (CNN) model. The main components of the project include:

* Software architecture design
* AI model development
* Web application integration with Flask

## 2. Software Architecture Design

### 2.1 System Overview

The platform follows a modular architecture divided into three main layers:

* **Data Layer**: Handles image dataset preprocessing and storage.
* **Model Layer**: Implements the neural network for image classification.
* **Application Layer**: Integrates the Flask web application with APIs and a user interface.

### 2.2 Architecture Diagram

***Fig 01: Architectural Diagram***

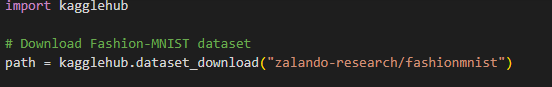
### 2.3 Design Decisions

* Selection of CNN model for accurate image classification
* Flask for lightweight web application development
* Modularity: Separate layers to allow independent updates.
* Scalability: Flask for lightweight web integration; Keras for model efficiency.

## 3. AI Model Development

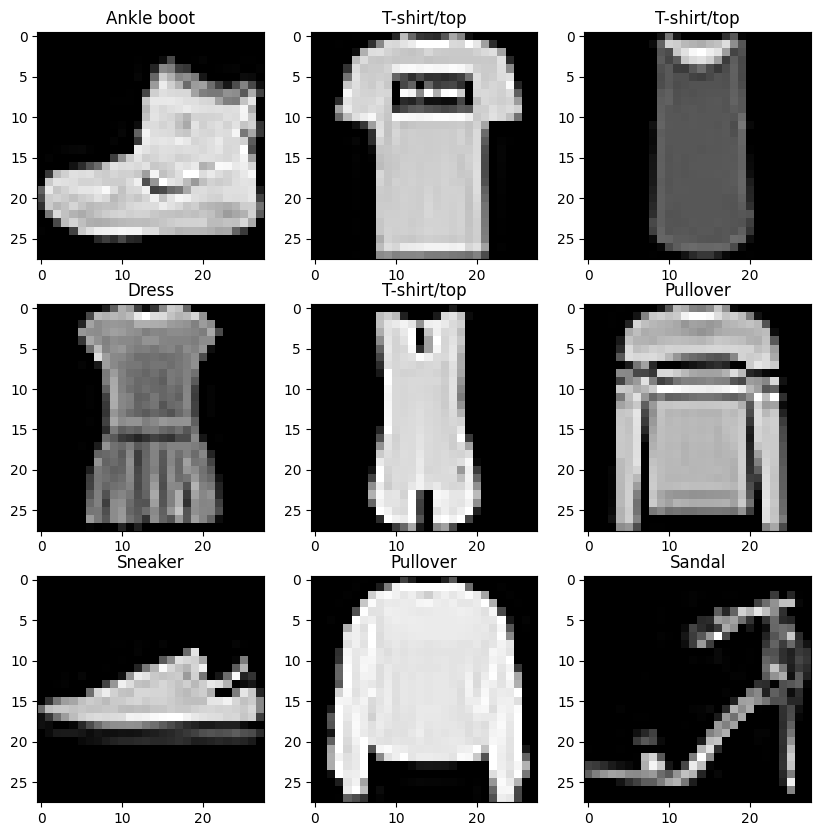
### 3.1 Dataset Selection and Preprocessing

* Using Fashion-MNIST



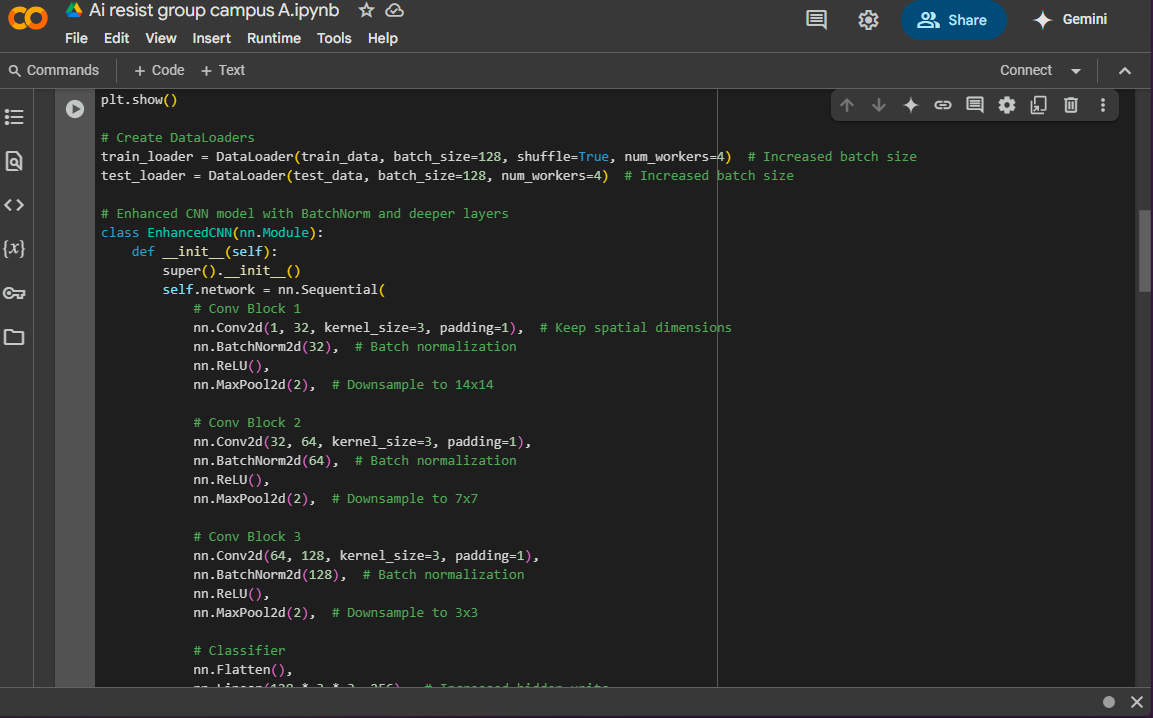
**Fig 03**

* Data augmentation and normalization for improved training

*****Fig 04: data processing Sample Data***

### 3.2 CNN Model Implementation

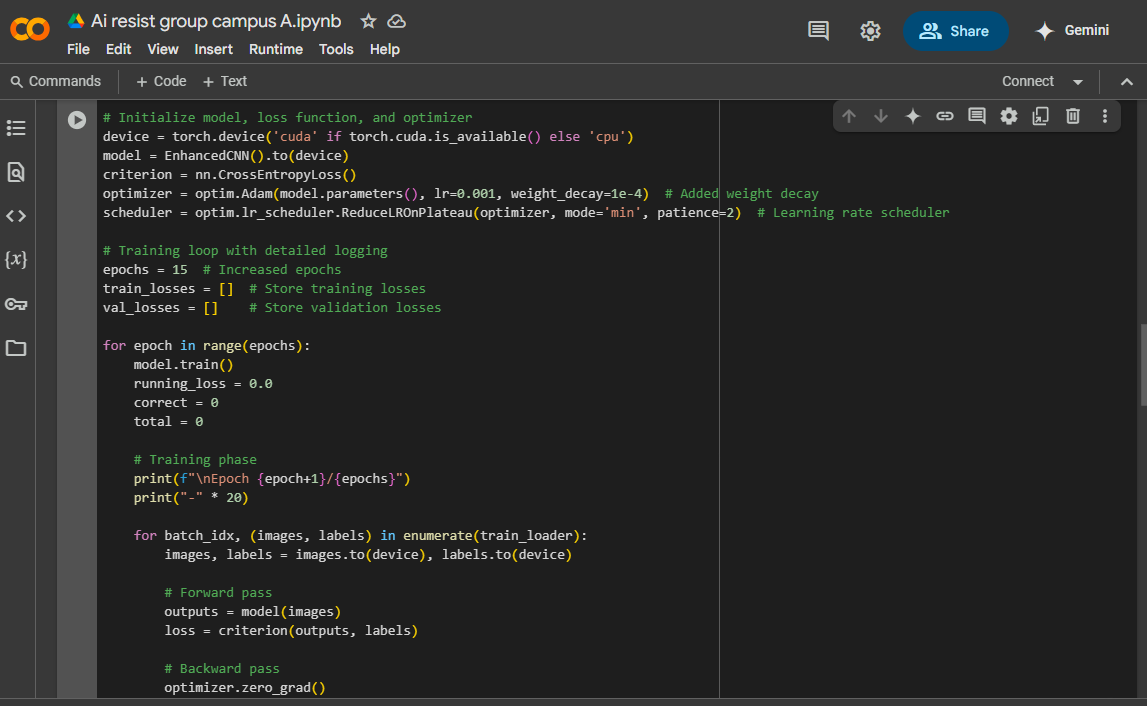
* Model architecture includes convolutional, pooling, and dense layers
* Activation functions such as ReLU and Softmax
* Compilation using an optimizer like Adam



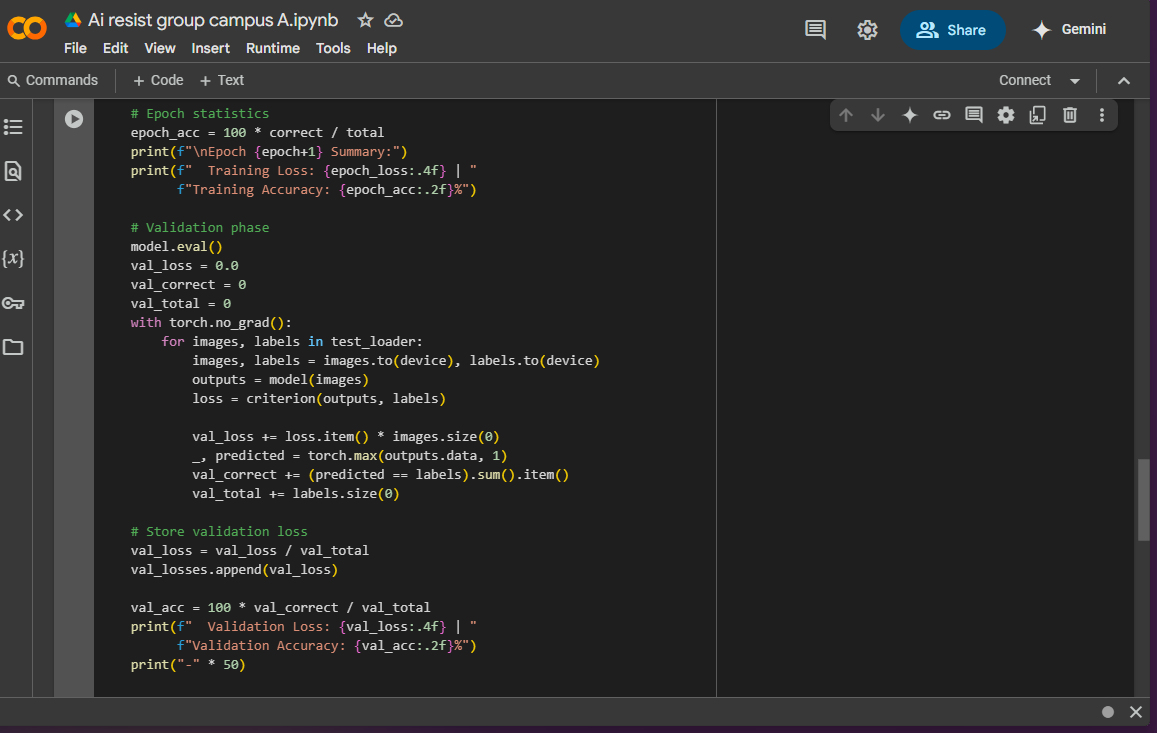
***Fig 05: CNN Model Summary and Training Code***

### 3.3 Training the Model

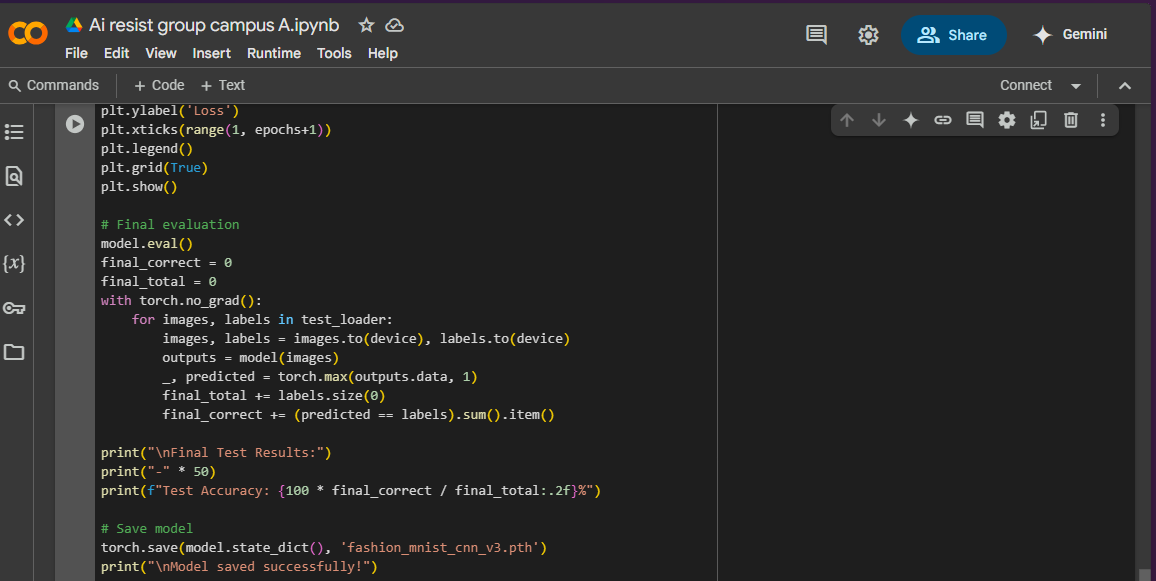
* Training the model on the dataset
* Evaluation of model accuracy and loss



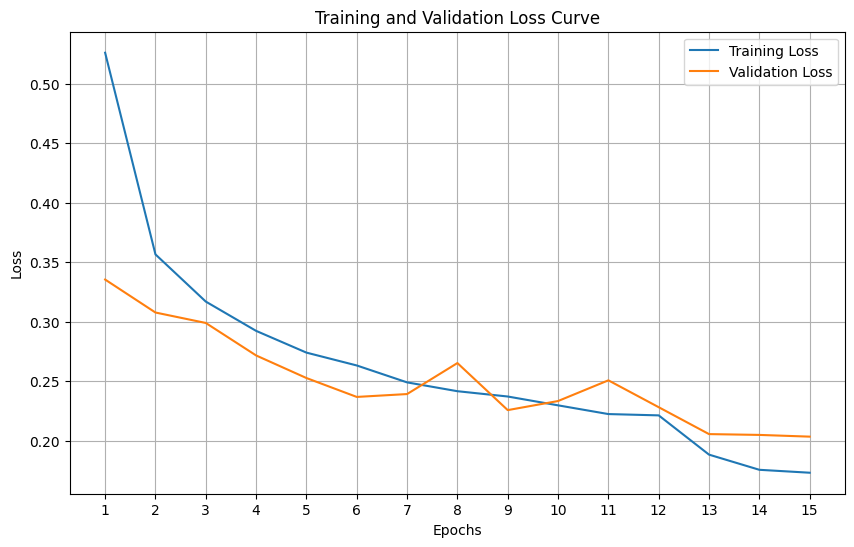
**Fig 06**



**Fig 07**

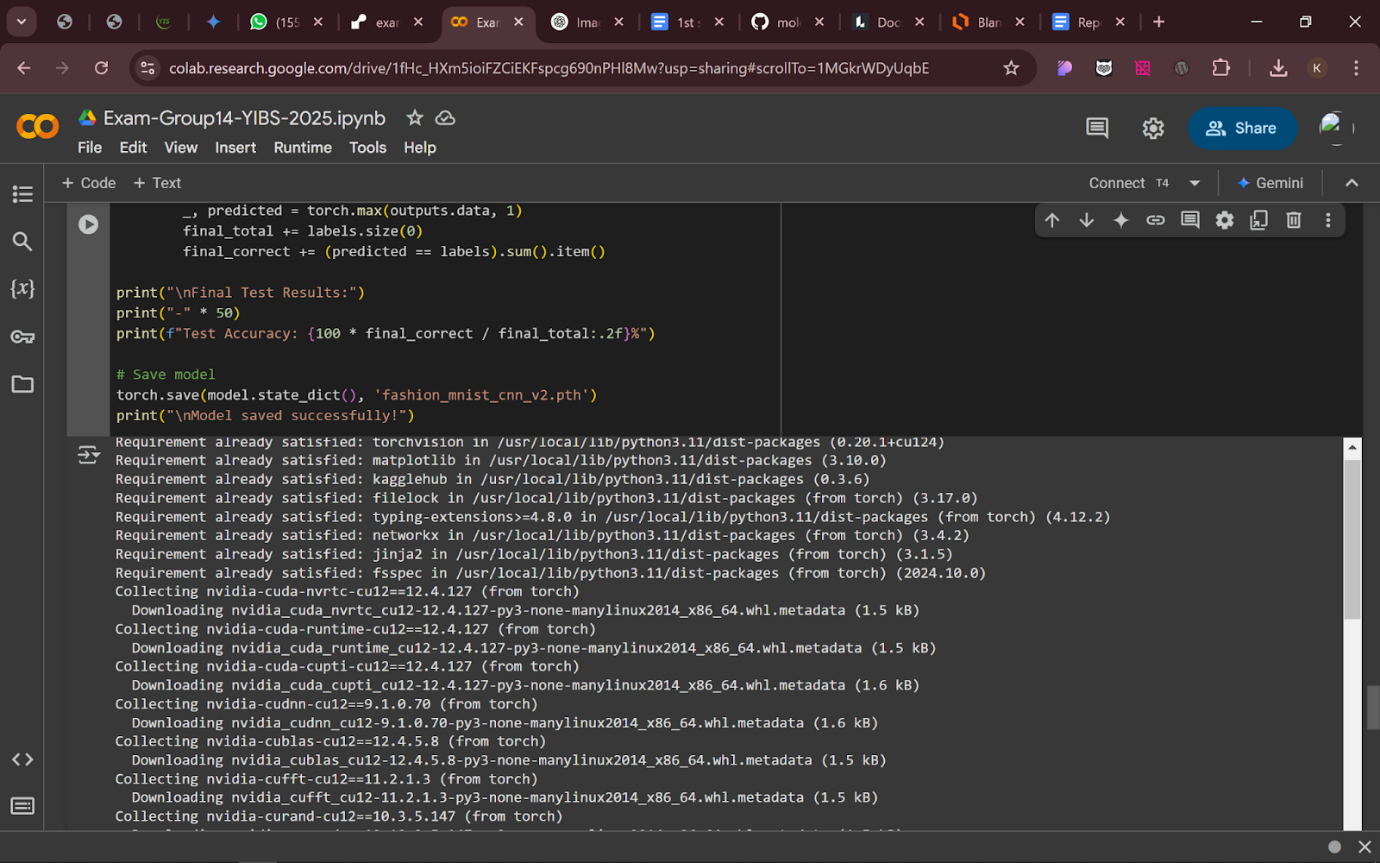


**Fig 08**

*****Fig 09: Training Process and Accuracy/Loss Graph***

### 3.4 Saving and Loading the Model

* Model saved in .pth format for deployment

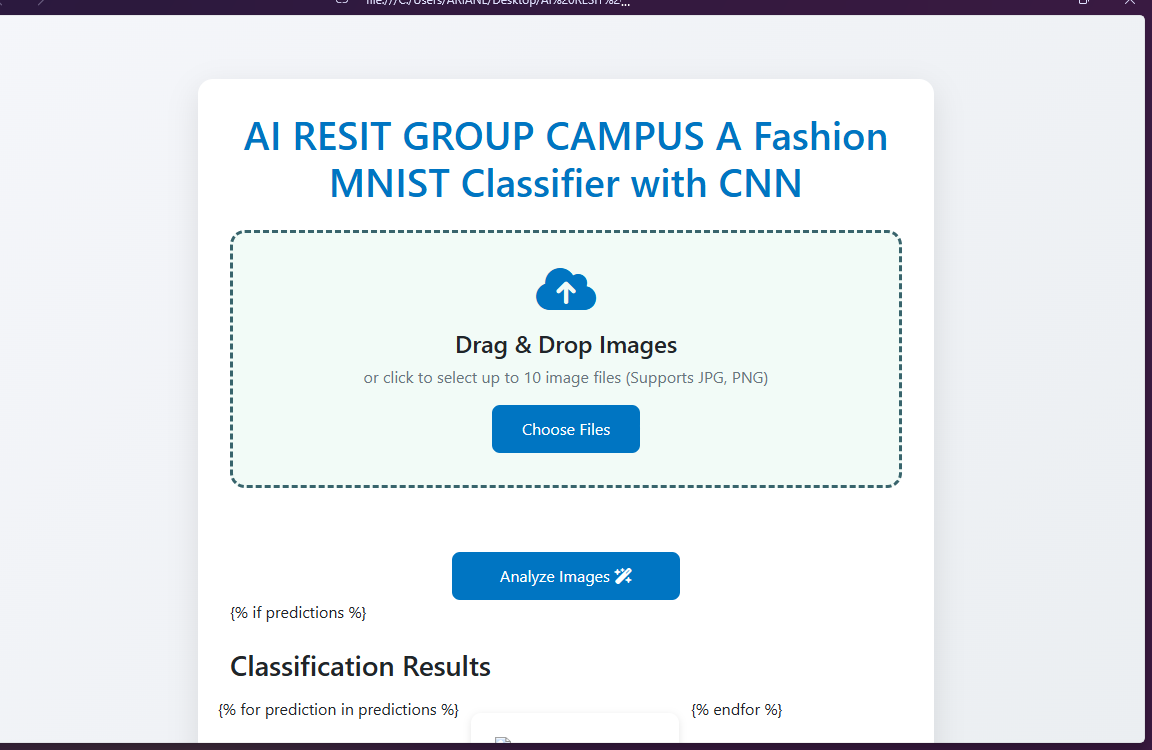


***Fig 10: Insert Screenshot of Saved Model File Location***

## 4. Application Development with Flask

### 4.1 Web Application Features

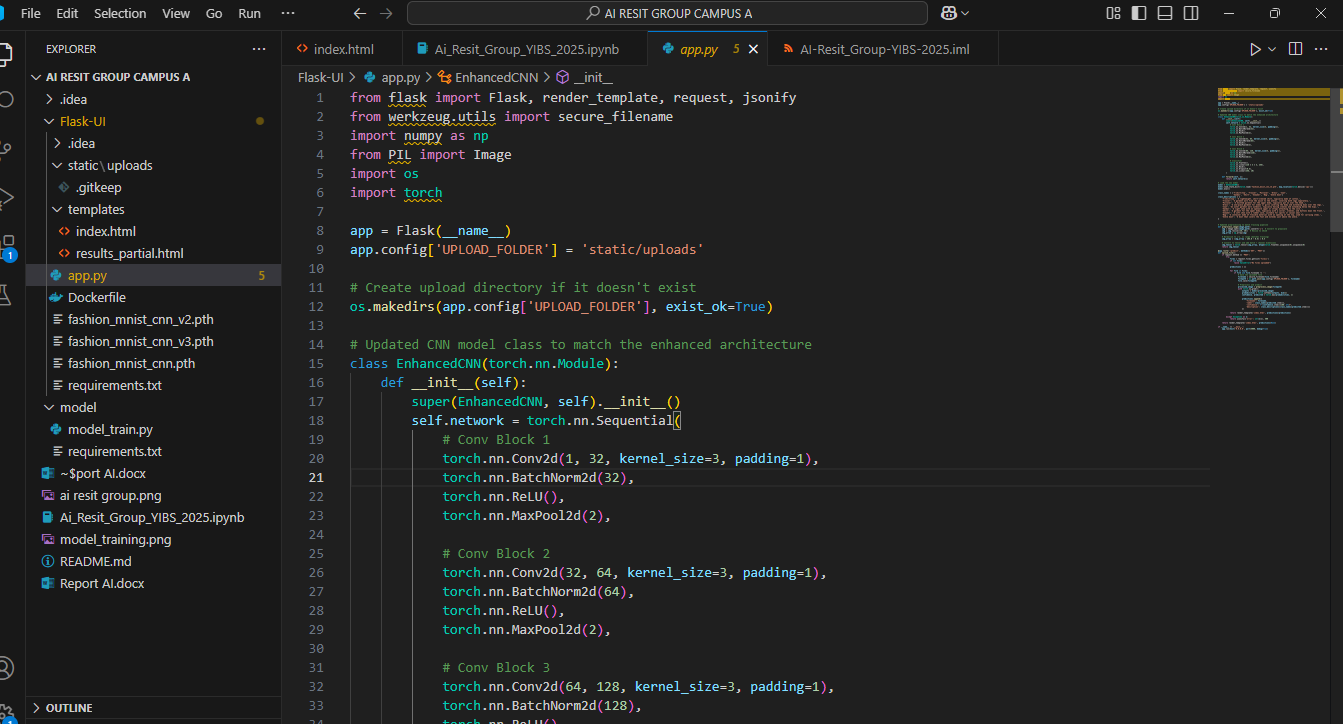
* **Upload Images**: A file upload form for users to submit images.
* **Classify Images**: API endpoint to classify uploaded images.
* **Display Results**: Shows predicted category and confidence scores.



***Fig 11: Flask Application UI***

### 4.2 Flask Code Implementation

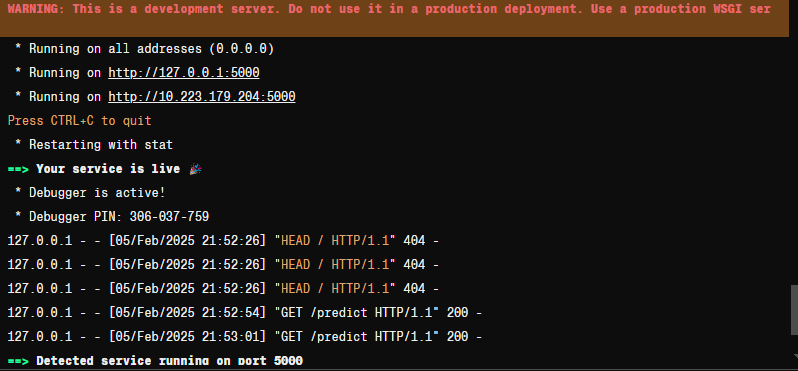
* Routes for image upload and model inference



**Fig 12: Integration with the CNN model**

### 4.3 Running the Web Application

* Local execution of Flask app



**Fig 14**

# 5. Testing and Deployment

### 5.1 Local Testing

* Testing for correct image classification and error handling

**Fig 15**

* Results for testing with a t-shirt

**Fig 16**

### 5.2 Optional Deployment

* Containerizing the application for deployment
* Hosting on platforms used RENDER

## 6. Conclusion

The project successfully implements an image classification platform with a well-structured architecture, a trained CNN model, and an interactive Flask web application. Future enhancements could include batch processing, real-time updates, and additional deployment optimizations.

Important links :

* **Collab**

<https://colab.research.google.com/drive/1X8aetzvWSKyPKYyDhw4pK-7DXXa5lOZj?usp=sharing>

* **Github repository:**

<https://github.com/molo-biloa/Exam-Group14-YIBS-2025.git/>