Assignment - 6 (CS5830)

Build a FastAPI for MNIST digit prediction

-Salem Aslam (BE20B027)

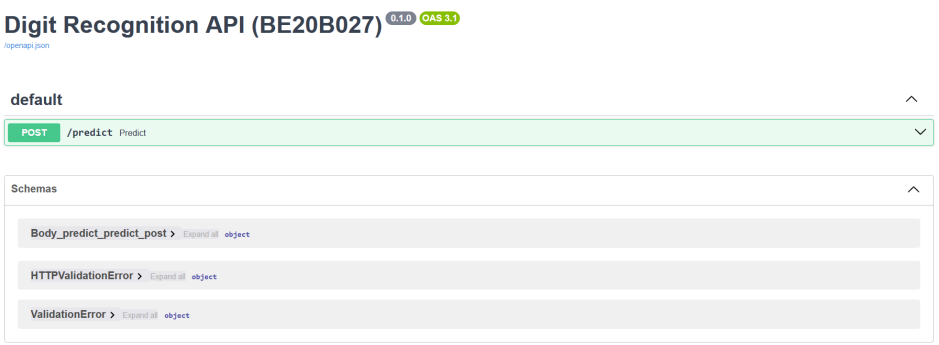
1. Introduction

This assignment implements a FastAPI application designed to function as a digit recognition API. Users can interact with the API by uploading an image containing a handwritten digit, and the API will predict the digit using a pre-trained MNIST model.

2. About the assignment

The project utilizes Python libraries like FastAPI, Pillow (PIL Fork), NumPy, and TensorFlow (through Keras) to achieve its functionality. The core functionalities involve:

- **API Endpoint (`/predict`):** This endpoint accepts image uploads via POST requests.
    - It validates the image format (ensuring JPEG, JPG, or PNG).
    - It preprocesses the image by converting it to grayscale and resizing it to the model's expected dimensions (28x28).
    - The image data is flattened and normalized for compatibility with the model.
    - Prediction is performed using a pre-trained MNIST model, returning the predicted digit label in the API response.
- **Helper Functions:**
    - `load_model`: Loads the pre-trained MNIST model from a specified file path.
    - `predict_image`: Takes a model and a preprocessed image, performs prediction, and returns the predicted digit class as a string.
    - `format_image` (commented out in this version): Resizes the input image to the dimensions required by the model.



3. Model Architecture

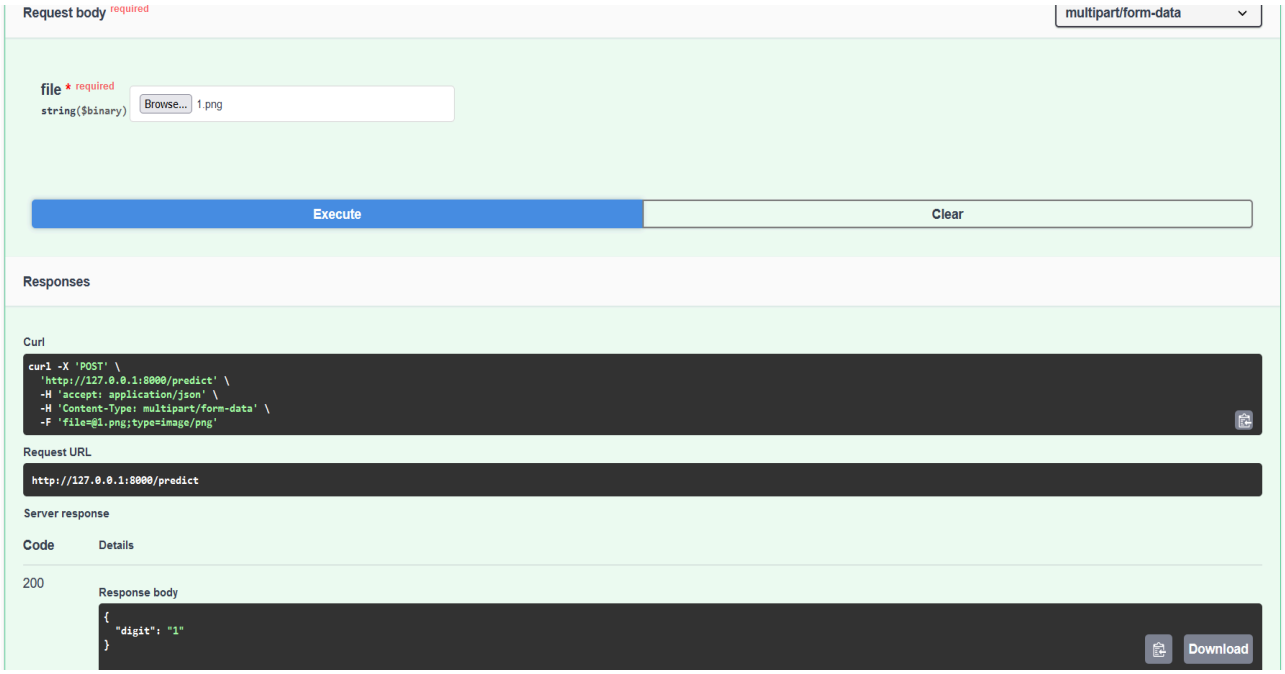
The architecture below defines a neural network for digit recognition with:

- Input layer for flattened images (784 pixels).
- Two hidden layers with 256 & 128 neurons (sigmoid activation).
- Output layer with 10 neurons (softmax activation for digit probabilities)

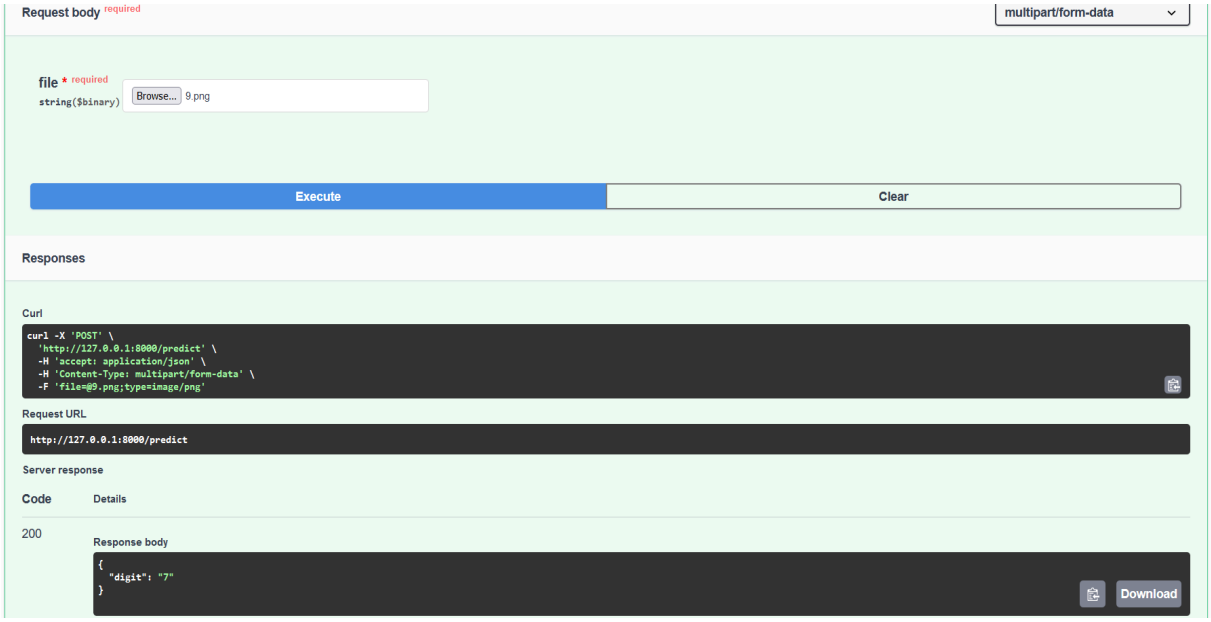
```
model = Sequential([
    Input(shape=(784,)),
    Dense(units=256, activation='sigmoid'),
    Dense(units=128, activation='sigmoid'),
    Dense(units=10, activation='softmax')
])
return model
```

4. Results

The model achieved good prediction on the original MNIST test data, correctly identifying all 10 digits correctly.



However, when presented with custom handwritten digits in ms paint, the model's performance was very poor, incorrectly predicting all of them except the digit 1.



One reason for this behavior is the difference between the training data and the custom inputs. The MNIST dataset likely contains a wider variety of stroke thicknesses and writing styles compared to the simpler, potentially thinner digits drawn in MS Paint. The model, trained solely on the MNIST data, may not have learned the features necessary to accurately classify these thinner handwritten digits.

This suggests the model may benefit from data augmentation techniques to improve its generalization ability to unseen variations in handwritten digits.

5. Conclusion

While the model performs well on the original MNIST data, its reliance on unaugmented training data hinders its robustness to slight variations in handwritten digit appearances. Implementing data augmentation techniques during training could significantly improve the model's ability to handle diverse digit styles.