Handwritten Number Recognition using Tensor flow with Java GUI

Anvitha Hiriadka

MS in Information Systems

College of Engineering, Northeastern University

April 2024

Abstract

This project presents a handwritten number recognition system implemented in JavaFX for graphical user interface (GUI) development, integrated with a TensorFlow model. The user interacts with the system through an intuitive GUI created using JavaFX, providing a platform for drawing handwritten digits. The TensorFlow model is trained using MNIST dataset to recognize digits, which accurately predicts the digit's identity. TensorFlow's flexible and efficient deep learning framework enables the development of a robust model capable of accurately classifying handwritten digits. Later, after model training, it could recognize the digit present in the new image data. The integration of JavaFX and TensorFlow facilitates seamless user interaction and efficient digit recognition, demonstrating the synergy between GUI development and machine learning for practical applications.

Approach:

1) Build the Java FX - GUI application in Java for the handwritten digit input
2) Mnist Dataset is loaded from the TensorFlow.Mnist Dataset
3) Handwritten number recognizer using TensorFlow is built, leveraging convolutional neural networks (CNNs) to accurately classify digits from the MNIST dataset.
4) Training the tensor flow model
5) Evaluated the model’s accuracy and loss.

6) Inter process communication to trigger python code that recognizes and performs prediction of handwritten number.

JavaFX GUI

The Graphical User Interface is built using the JavaFX library.

The border pane and canvas pane are used for GUI Layout. Two buttons are present in the UI. Clear Button for clearing the canvas performing erase function and “PREDICT” Button for performing the prediction operation of handwritten number on canvas.

The GUI is as follows:

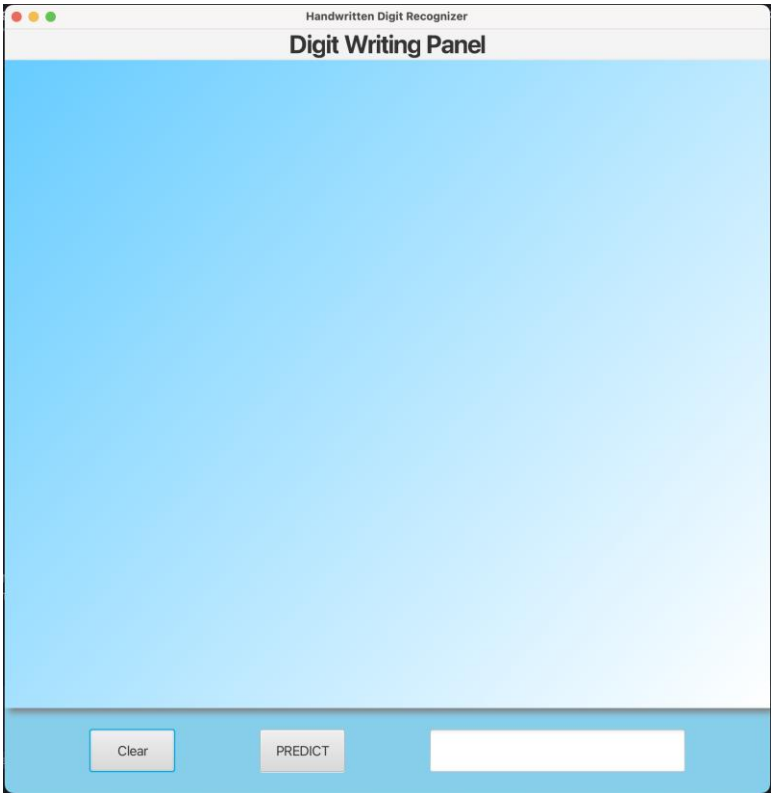

Figure 1. GUI for Handwritten Number Recognition

Event Handler for the GUI

When the user interacts with the UI, by drawing the numbers between (0-9) , an event is said to have occurred. To handle this events, event handler classic defined.

This class will have all the function to handle the events such as Mouse Pressed, Mouse Dragged, Predict and Clear button actions, trigger of the model to predict the number.

Image processing

The drawing made on the canvas[UI] by the user, is captured as a snapshot and later resized to 28 X 28 pixels and converted to greyscale before it is fed to the model for prediction.

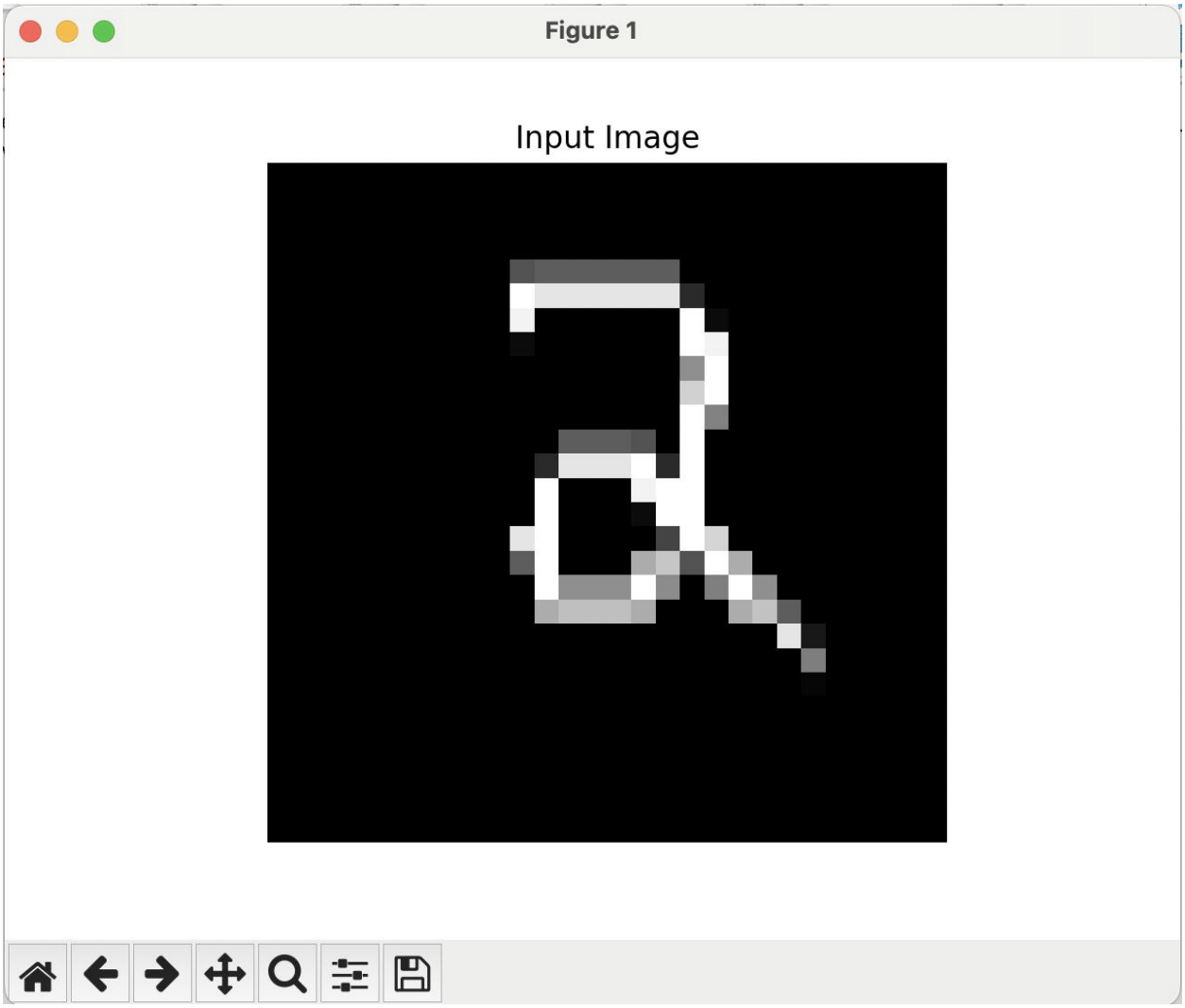

Figure 2. Image after Pre-processing

Number Recognition Model

Data Set: MNIST Dataset

- The MNIST database is a large database of handwritten digits commonly used for training image processing systems.
- It consists of 70,000 grey scale images of handwritten digits (0 to 9) each measuring 28 x 28 pixels.
- Using this dataset because of its simplicity, accessibility, and well-defined structure suitable for training the Number recognizer model.

Link : MNIST - dataset

Neural Network Model

Tensor flow for developing a neural network model for the handwritten number recognition.

Steps involved in building neural network model

Normalizing the Data:

The original pixel values range from 0-255, where 0 is black and 255 is white. Therefore, the pixel values of the images are normalized to a range between 0 and 1. This step helps in improving the training process by making the data more uniform.

Training the model:

The model is trained using the data set .

Performance Evaluation:

The accuracy and performance of the model is evaluated.

The accuracy results of 98.94% was achieved.

After the above steps, the model is saved and used later for predicting the number using new image data.

Dependencies

- This project requires Java 8 and JavaFX for GUI.
- For number recognition model - tensorflow, Python 3.11.5.

Files Created for Developing the Application

- GUI_Number_Recognizer.java
    - Purpose: Entry point of the JavaFX application.
    - Contents: Contains the main method to launch the application.
- Event_handler_canvas.java
    - Purpose: Handles the events on the Canvas
    - Contents: Contains various functions to handle the events happening the UI(Mouse pressed, Mouse Dragged, Button actions) and the Model trigger
- Canvas_EventHandler.java
    - Purpose: Interface for handling mouse on a canvas.
    - Contents: contains the methods for Mouse pressed and Mouse Dragged
- Image_processing.java
    - Purpose: Class for proceesing the image
    - Contents: contains the methods resizing the image and save the image as .png file
- number_recognizer_model2.py
    - Purpose: This file has the Tensor flow based model training, accuracy, loss tests and model is saved and loaded here for prediction using new images of number
    - Contents: contains Model training and digit recognition.
- handwritten_digit_model2.h5 : this is the trained model, which is loaded to predict new number images.

Class Diagram of the Application

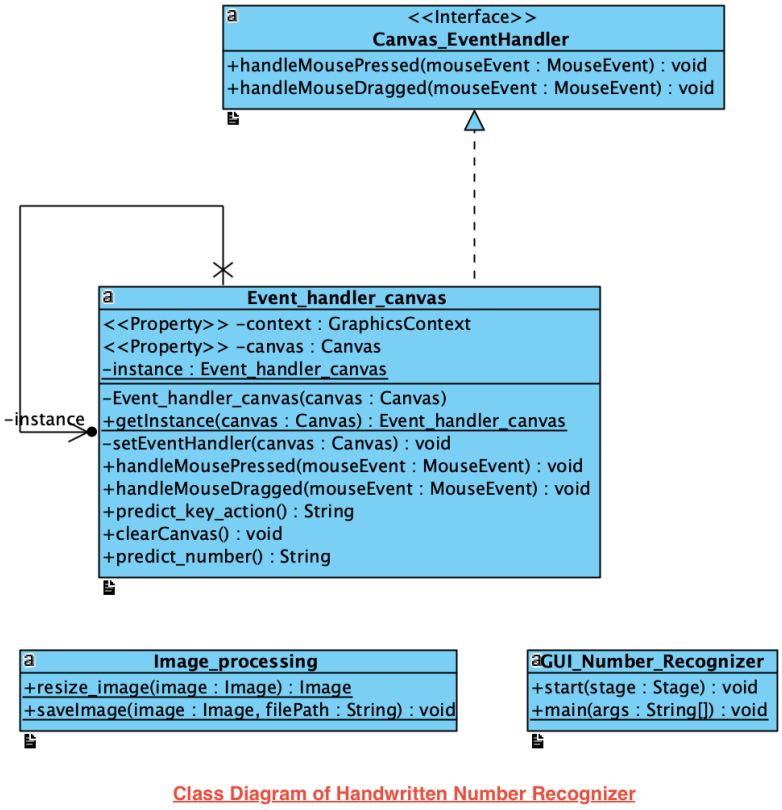

Figure 3. Class Diagram of the Handwritten number recognizer

Usage

- Launch the application.
- Draw a number on the canvas using the touch screen or mouse (anything between 0-9).
- Click the "PREDICT" button to predict the drawn number.
- The predicted number will be displayed on the output field of GUI.

Test Cases Covered

Test Case 1: Mouse Input

- Description: Drawing numbers using a mouse.
- Steps: Run the application, draw numbers using the mouse, click "PREDICT"

- Expected Result: The drawn number is correctly recognized, predicted and displayed.

Test Case 2: Touch Screen Input

- Description: Drawing numbers using a touch screen(laptop).
- Steps: Run the application, draw numbers using touch screens, click "PREDICT"
- Expected Result: The drawn number is correctly recognized, predicted and displayed.

Test Case 3: Tested number recognition from digits 0-9

- Description: Drawing numbers using a touch screen(laptop)/Mouse.
- Steps: Run the application, draw numbers using touch screens, click "PREDICT"
- Expected Result: The drawn number is correctly recognized, predicted and displayed

Results

The number (0-9) are predicted and results are displayed as shown in the images below:

Figure 4. Result of Handwritten Number Prediction 0

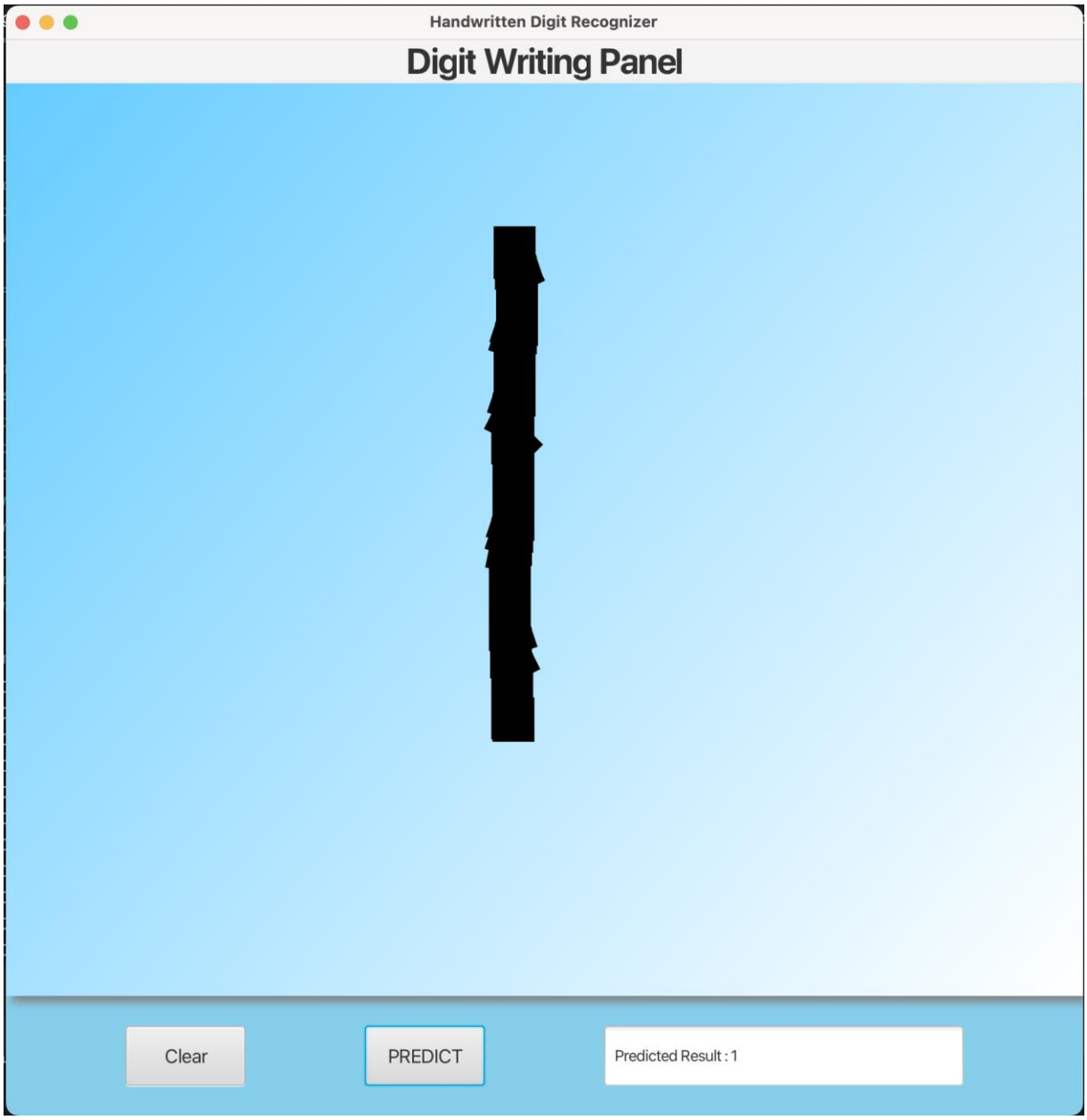

Figure 5. Result of Handwritten Number Prediction 1

Figure 6. Result of Handwritten Number Prediction 2

Figure 7. Result of Handwritten Number Prediction 3

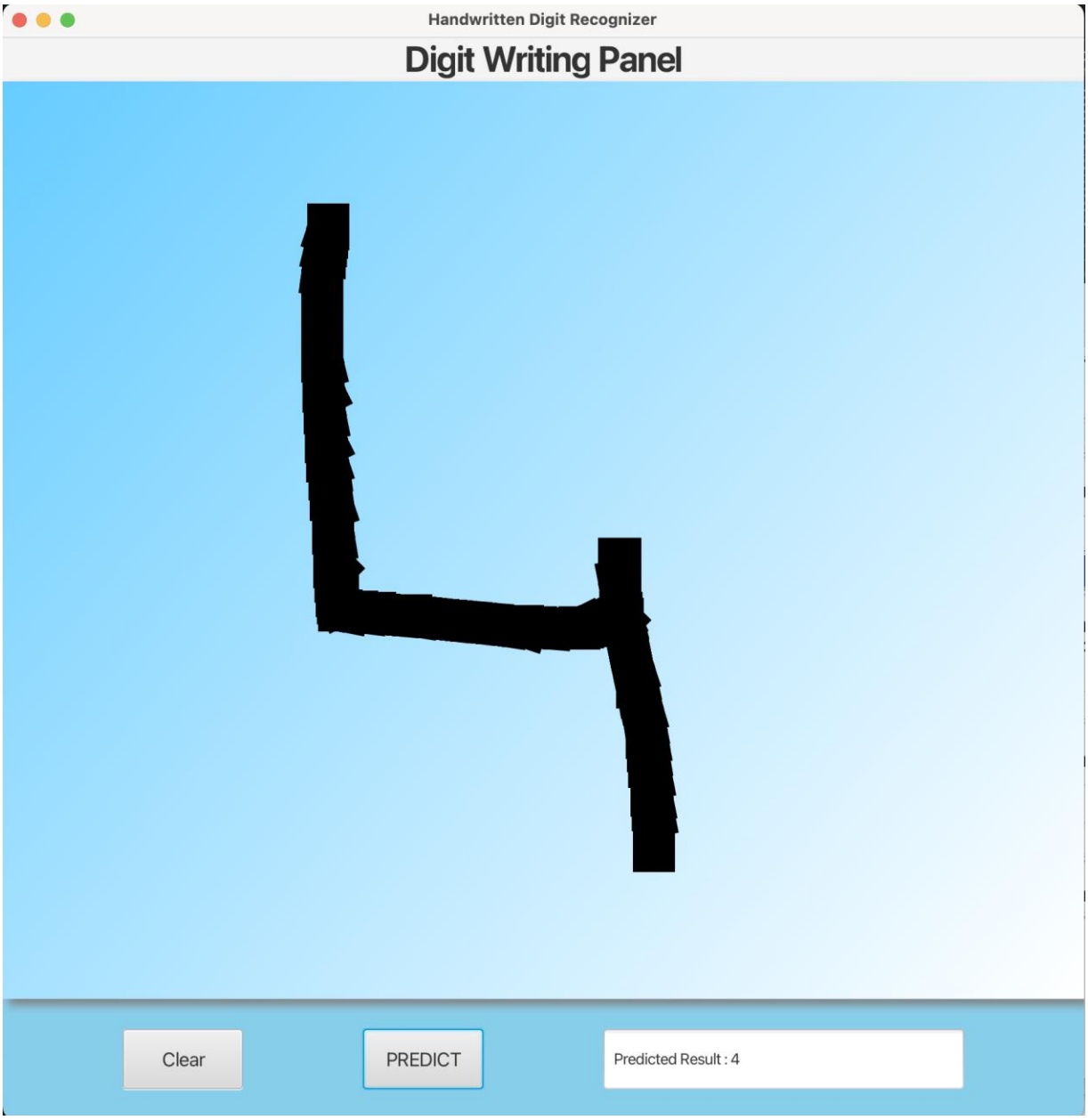

Figure 8. Result of Handwritten Number Prediction 4

Figure 9. Result of Handwritten Number Prediction 5

Figure 10. Result of Handwritten Number Prediction 6

Figure 11. Result of Handwritten Number Prediction 7

Figure 12. Result of Handwritten Number Prediction 8

Figure 13. Result of Handwritten Number Prediction 9

Installation and Configuration set up to run this JavaFx Application

To run this project, you need to configure JavaFX in IntelliJ IDEA. Follow these steps:

- Launch IntelliJ IDEA and open your JavaFX project.
- Navigate to Run/Debug Configurations:
- Click on the drop-down menu next to the green play button in the toolbar.
- Select "Edit Configurations..." from the list. This will open the Run/Debug Configurations window.
- Add a New Configuration:
- In the Run/Debug Configurations window, click on the "+" icon at the top left corner to add a new configuration.
- From the dropdown menu, select "Application".
- Configure the New Run/Debug Configuration:
- Give your new configuration a name, such as "JavaFX Application".
- Set the main class of your JavaFX application by clicking on the browse (...) button next to the "Main class" field and selecting the appropriate class that contains your main method(GUI_Number_Recognizer.java).
- Set the working directory if necessary.
- Specify JavaFX VM Options:
    - To specify JavaFX VM options, you typically need to add the following VM options:

      --module-path /path/to/javafx-sdk/lib --add-modules javafx.controls,javafx.fxml

    - Replace /path/to/javafx-sdk/lib with the actual path to the lib directory of your JavaFX SDK installation.
- Apply and Save Configuration:
- Click "Apply" to apply the changes.
- Click "OK" to close the Run/Debug Configurations window.
- Run or Debug Your JavaFX Application:
    - Now, you can run or debug your JavaFX application using the newly created run/debug configuration.
    - Click on the green play button in the toolbar and select the configuration you just created from the dropdown list.

- After the installation and configuration of the JavaFX, You can run the Application GUI_Number_Recognizer.java

Github Link

Link for Handwritten Number Recognizer code - Final Project

Citation

Dataset:

MNSIT Database : https://en.wikipedia.org/wiki/MNIST_database

GUI:

1. Geron,A.(2017). Hands-On Machine Learning with Scikit-Learn & TensorFlow(1st Ed.).O'REILLY
2. JavaFX - Quick Guide. (n.d.). https://www.tutorialspoint.com/javafx/javafx_quick_guide.htm
3. JavaFX - Layout BorderPane. (n.d.). https://www.tutorialspoint.com/javafx/layout_borderpane.htm
4. JavaFX - Quick Guide. (n.d.-b). https://www.tutorialspoint.com/javafx/javafx_quick_guide.htm
5. BorderPane (JavaFX 8). (2015, February 10). https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/BorderPane.html
6. GeeksforGeeks. (2021, September 1). JavaFX BorderPane Class. GeeksforGeeks. https://www.geeksforgeeks.org/javafx-borderpane-class/
7. GeeksforGeeks. (2018, September 18). JavaFX Font Class. GeeksforGeeks. https://www.geeksforgeeks.org/javafx-font-class/
8. JavaFX CSS Reference Guide. (n.d.). https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html