CodeAlpha Internship

Task Title: Hand-Written character recognition Using Python

Task Description: Create a handwritten character recognition system that can recognize various handwritten characters or alphabets. You can extend this to recognize entire words or sentences.

Hand-Written Character Recognition

Building a handwritten character recognition model involves using a Convolutional Neural Network (CNN) to classify images of handwritten characters, such as digits or letters. The process starts with selecting a suitable dataset, like MNIST or EMNIST, followed by preprocessing the data through normalization and reshaping. After training the model on labeled data, it is evaluated using metrics like accuracy and a confusion matrix. The goal is to achieve a model that accurately recognizes and classifies handwritten characters.

Problem Definition and Dataset Selection

Objective: The goal is to build a model that can recognize handwritten characters from an image. This typically involves digits (like in the MNIST dataset) or alphabetic characters (like in the EMNIST dataset).

Dataset: Select an appropriate dataset, such as:

- **MNIST**: Contains 70,000 images of handwritten digits (0-9).
- **EMNIST**: A more extensive dataset with handwritten letters and digits.

Data Preprocessing

Normalization: Scale pixel values to the range $[0, 1]$ to help the model converge faster during training.

Reshaping: Ensure that the input data is in the correct shape for the model. For example, MNIST images are 28x28 pixels, so each image should be reshaped to (28, 28, 1).

Label Encoding: Convert the labels into a one-hot encoded format, which is typically used in classification tasks.

Model Training

Training Process: Train the model on the training dataset, typically using a batch size and a number of epochs. The model learns by minimizing the loss function over time.

Validation: Use a validation set to monitor the model's performance and prevent overfitting.

Model Evaluation

Test the Model: Evaluate the model on a separate test dataset to measure its accuracy and generalization.

Confusion Matrix: Generate a confusion matrix to see how well the model performs on each class.

Accuracy and Loss Plots: Plot the training and validation accuracy/loss over epochs to visualize the model's learning process.