SOLUTION ARCHITECTURE

Importing the libraries and loading the dataset

All the modules that we will require for training our model will first be imported. There are already certain datasets in the Keras library, and MNIST is one of them. Therefore, importing the dataset and using it are both simple processes. The training data, its labels, as well as the testing data and its labels, are returned to us by the mnist.load data() method.

Preprocessing the data

We must execute some operations and process the data in order to prepare it for our neural network because the image data cannot be supplied straight into the model. The training data's dimension is. We must rearrange the matrix to fit the additional dimension that the CNN model will need.

Creating the model

Our CNN model will now be developed in a Python data science project. Convolutional and pooling layers are the most common components of CNN models. Because it performs better for data that are represented as grid structures, CNN is a good choice for challenges involving picture categorization. When training, the dropout layer minimizes offer fitting of the model by deactivating part of the neurons. The Adadelta optimizer will then be used to compile the model.

Training the model

The training of the model will begin with the help of Keras' model.fit() function. It requires the batch size, epochs, training data, and validation data. The model's training process takes some time. We store the weights and model definition in the "mnist.h5" file after training.

Evaluating the model

Our dataset, which contains 10,000 photos, will be used to gauge how well our model performs. Since the testing data was not used to train the model, it is new data for our model. Since the MNIST dataset is nicely balanced, we can get an accuracy of about 99%.

Creating GUI to predict digits

In a new file that we've made for the GUI, we've built an interactive window that allows us to draw numbers on a canvas and then identify those numbers with a button. The Python standard library includes the Tkinter library. Predict digit(), a function we developed, uses the trained model to predict the digit from the image as input.After that, we develop the App class, which is in charge of developing our app's GUI. By capturing the mouse event, we build a canvas on which we may draw, and by pressing a button, we call the predict digit() function and display the outcomes.

IBM-Project-16649-16596