Project Flow

Import Libraries and Dataset

At the project beginning, we import all the needed modules for training our model. We can easily import the dataset and start working on that because the Keras library already contains many datasets and MNIST is one of them. We call mnist.load_data() function to get training data with its labels and also the testing data with its labels.

Data preprocessing

Model cannot take the image data directly so we need to perform some basic operations and process the data to make it ready for our neural network. The dimension of the training data is (60000*28*28). One more dimension is needed for the CNN model so we reshape the matrix to shape (60000*28*28*1). Analyzing and reshaping the data and applying one hot encoding to the data is done in data preprocessing.

Create model

Its time for the creation of the CNN model for this Python-based artificial intelligence project. A convolutional layer and pooling layers are the two wheels of a CNN (Convolutional Neural Network) model. The reason behind the success of CNN for image classification problems is its feasibility with grid structured data. We will use the Adadelta optimizer for the model compilation.

Train the model

To start the training of the model we can simply call the model.fit() function of Keras. It takes the training data, validation data, epochs, and batch size as the parameter. The training of model takes some time. After successful model training, we can save the weights and model definition in the 'digit_classifier.h5' file.

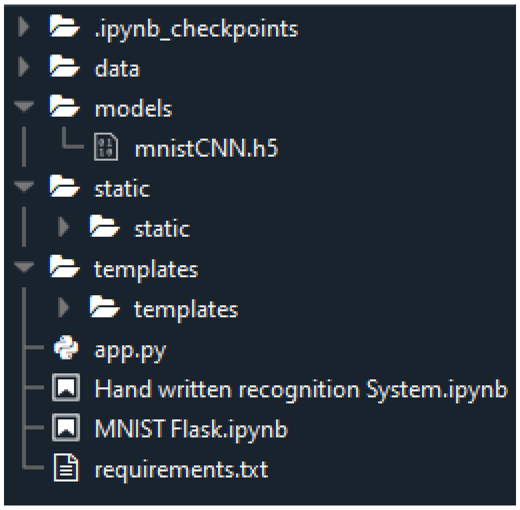
Evaluate the model

To evaluate how accurate our model works, we have around 10,000 images in our dataset. In the training of the data model, we do not include the testing data that's why it is new data for our model. Around 99% accuracy is achieved with this well-balanced MNIST dataset.

Develop web app using Flask

Although it doesn't require a specific architecture, there are some good practice to follow :

- **app.py :** Is the main code that will run our Flask application. It will contain the different routes for our application, respond to HTTP requests and decide what to display in the templates. In our case, it will also call our CNN classifier, operate pre-processing steps for our input data and make prediction.
- **Templates folder :** A template is an HTML file which can receive Python objects and is linked to the Flask application. Hence, our html pages will be stored in this folder.
- **Static folder :** Style sheets, scripts, images and other elements that will never be generated dynamically must be stored in this folder. We will place our Javascript and CSS files in it.



```
.ipynb_checkpoints
data
models
└ mnistCNN.h5
static
  static
templates
  templates
app.py
Hand written recognition System.ipynb
MNIST Flask.ipynb
requirements.txt
```