1. Import the library tensorflow,keras,ggplot2,readr.

2. Load the training and test dataset in the working directory the rows and columns of an image are set to (28,28)

3. Data i.e. the 784 pixel columns are transformed into a 28X28 matrix form which is further converted into image form and these pixels are separated from labels.And the same is performed for the training as well as test data.
4. Unflattening of the x_train and y_train.
5. Since the class names are not included in the dataset we will store them in a vector to use them later with the images.
6. Rotate the matrix.
7. Now we will take a function to plot the image formed in step 3.
8. Further,we will plot 16 random images from the training set.

   Model building:-

9. Now set the hyperparameter (batch_size =128, num_class = 10, epoch =6) to determine the rate at which the samples are fed to the model for training.
10. The dataset of 60,000 examples is divided into batches of 128, which means it takes 469 iterations to complete 1 epoch.
11. T**he labels for the classes are one hot encoded. One hot encoding is the process in which categorical variables are converted into forms (000,010,001) which can be fed easily to ML algorithms to do a better job in prediction.**
12. Then we move onto the layering process, in which we add layers one by one.
13. Initially we add the first convolution layer through layer_conv_2d() method which takes number of filters to be 32, filter size to be 5x5 and the activation function is relu, which activates neurons in our CNN and enables learning. The input_shape is also provided.
14. Now the activation map is of size 24x24 (from N-M+1)
15. Then we add another convolution layer with number of filters 64, filter size 3x3 and activation function being relu, there is no input shape in this case as it will acquire/learn this attribute from the previous step.
16. Now the activation map is of size 22x22.
17. In the next step we add the max pooling layer which assists in dimensionality reduction along with noise suppression, the pool size/filter size is 2x2.
18. With the help of the formula (N-F)/S + 1, where N is the current dimensions, F is the filter size, and S is stride which by default is one. Which in this case will become 21x21
19. After this we add the dropout layer, which prevents overfitting by dropping certain neurons.
20. Then, the flattening layer is added which assists in reducing the dimensionality by converting it to 1D.
21. Now we add one of the most important layers being the dense layer, which takes 128 units as an argument depicting 128 neurons. Dense layers feeds all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer. The activation in this case was relu.

22. Now, the dropout layer is yet again added with a modified rate of 50% as compared to 25% previously.
23. Now finally the dense layer is added again as a part of the fully connected layer, with this time the activation being soft max, which gives the prediction or output probabilities for each class/label. Each node contains a score that indicates the probability that the current image belongs to one of the 10 digit classes.
24. After this model is compiled with the help of calibration of optimizer, metric and loss.
25. Now we've approached the fitting part of the CNN where the model is fit to the train data. The hyper parameters used here are the batch size, epoch and the validation data as mentioned before.
26. We pass the validation of test data to the fit function so that keras knows what data to test the metric against when the evaluate function is run on the model.
27. Finally we move on to the visual representation part of our prediction to better analyse the fitting of the model.