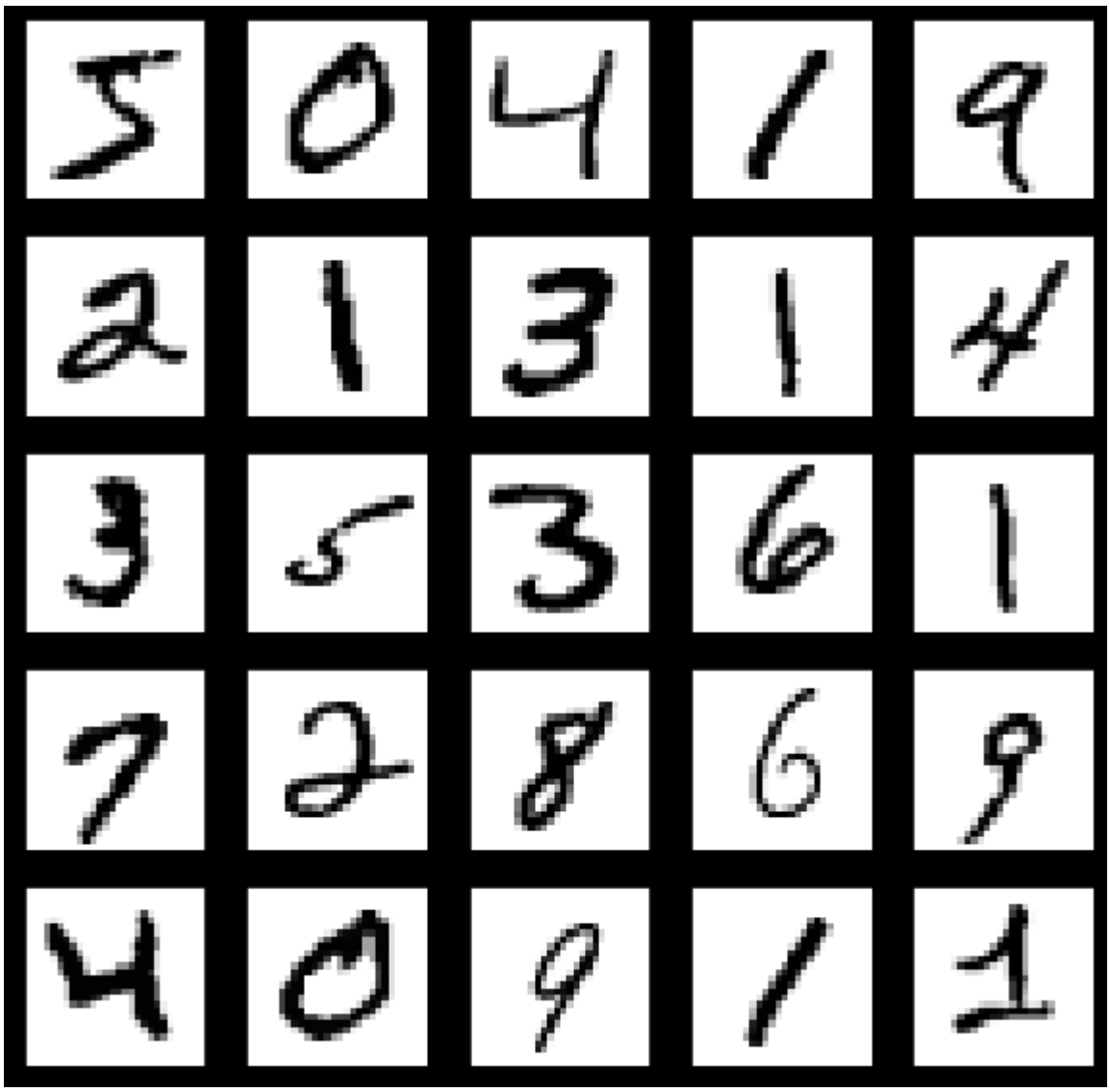
Mid-Course Test

edureka!

Scenario – 1

Description

The dataset is MNIST dataset includes images of o to 9 handwritten digits. The objective is to classify images into specific classes using Single Layer Perceptron.



Dataset:

Total Images: - 70,000

Train Images: - 60,000

Test Images:- 10,000

Image Size:- 28 X 28

Different Classes:

- Classes: **{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}**

Note: Please use google colab to work on this project Also, make sure to select GPU backend while selecting a runtime.

https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d

Problem Statement

Question 1

A. Load MNIST data from Keras Library and Split the same into Train and Test.

Output :-

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
The shape of data for train (60000, 28, 28)
```

B. Scale the values of train and test between 0 & 1 by dividing train & test by 255

Output:

Original X_train

```
print(X_train[0])

[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
```

Scaled X_Train

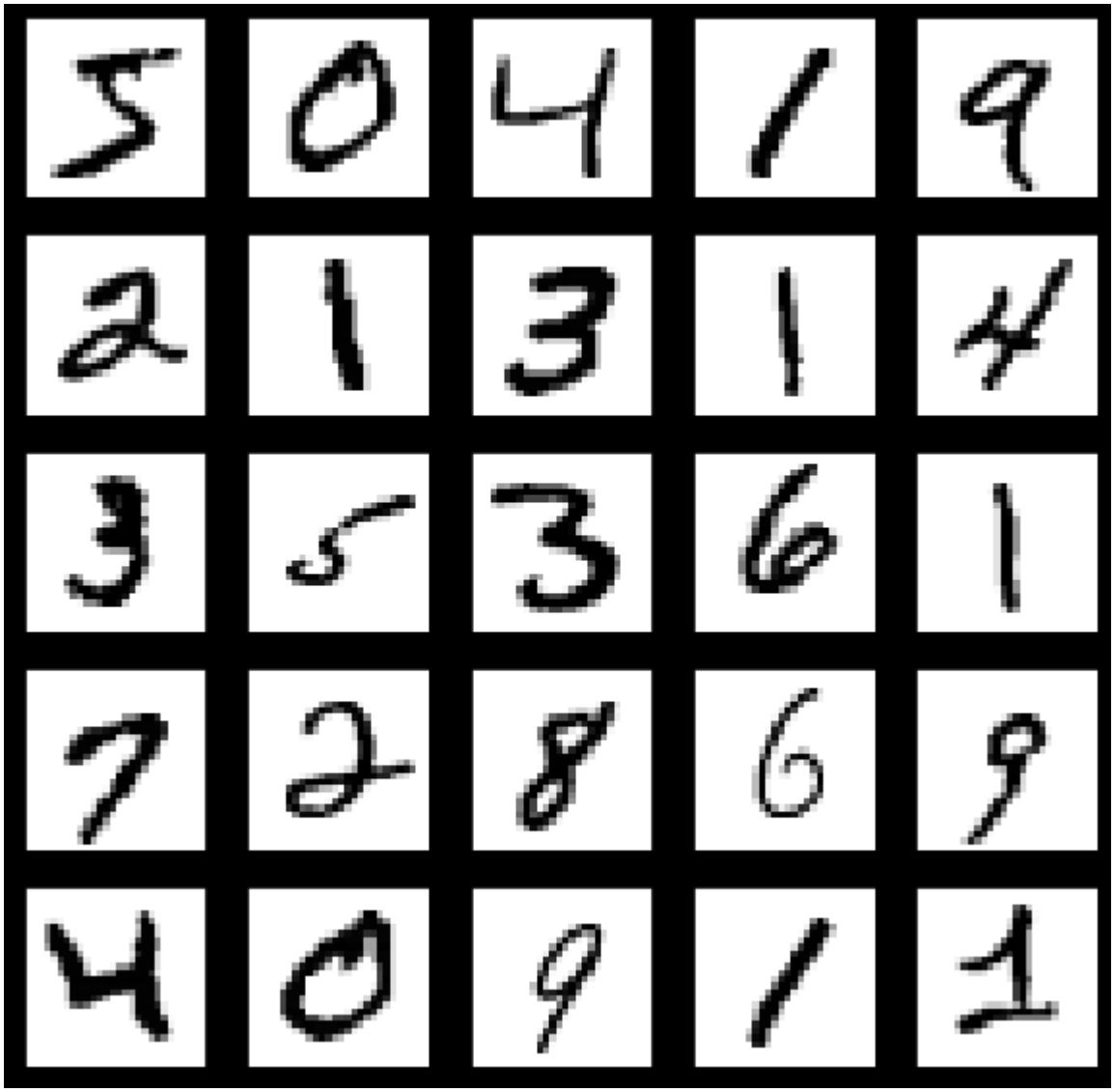
```
print(x_train[0])

[[0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.01176471 0.07058824 0.07058824 0.07058824 0.49411765 0.53333333
  0.68627451 0.10196078 0.65098039 1.         0.96862745 0.49803922
  0          0          0          0         ]
```

Scenario – 2

Problem Statement

Question 2: Display first 25 images from the training dataset and display the labels along with them

Output:-



QUESTION 3:

Load the data (again, important)

1. Reshape the data →

```
before reshape the data (60000, 28, 28)
After reshape the data (60000, 784)
```

2. Print the label for first image and after converting it to categori cal

```
class lebel of first image  : 5
After converting class lebel of first image  : [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

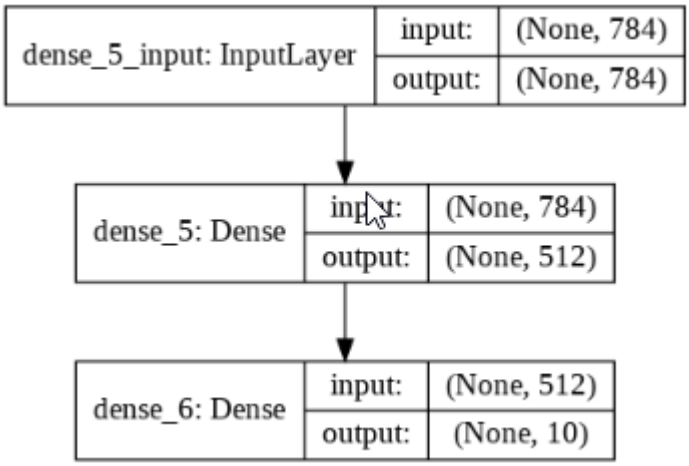
Question 4:

A. Build basic Single Layer Perceptron on the MNIST Data.

HINT:

1. Single Dense 1024 Neurons
2. Input Shape Being 784 Activation Function Relu.
3. Softmax Layer With 10 Neuron As Output
4. Loss- Categorical_Crossentropy
5. Optimizer - RMSPROP
6. Batch_Size = 5000, Epochs=50
7. Print The Accuracy And Loss Of SLP Model

Output (Architecture):-



Test Accuracy:-

```
Test loss: 0.06742082018023357
Test acc: 0.9789
```

Question 5:

B. Build basic Multilayer Perceptron on the MNIST Data.

HINT:-

1. Single dense layer 1024 neurons

2. input shape being 784

3. activation function relu.

4. Second Dense Layer 512 neurons and activation relu

5. Third Dense Layer 512 Neurons and activation relu

6. Softmax layer with 10 neurons as output

7. loss- categorical_crossentropy

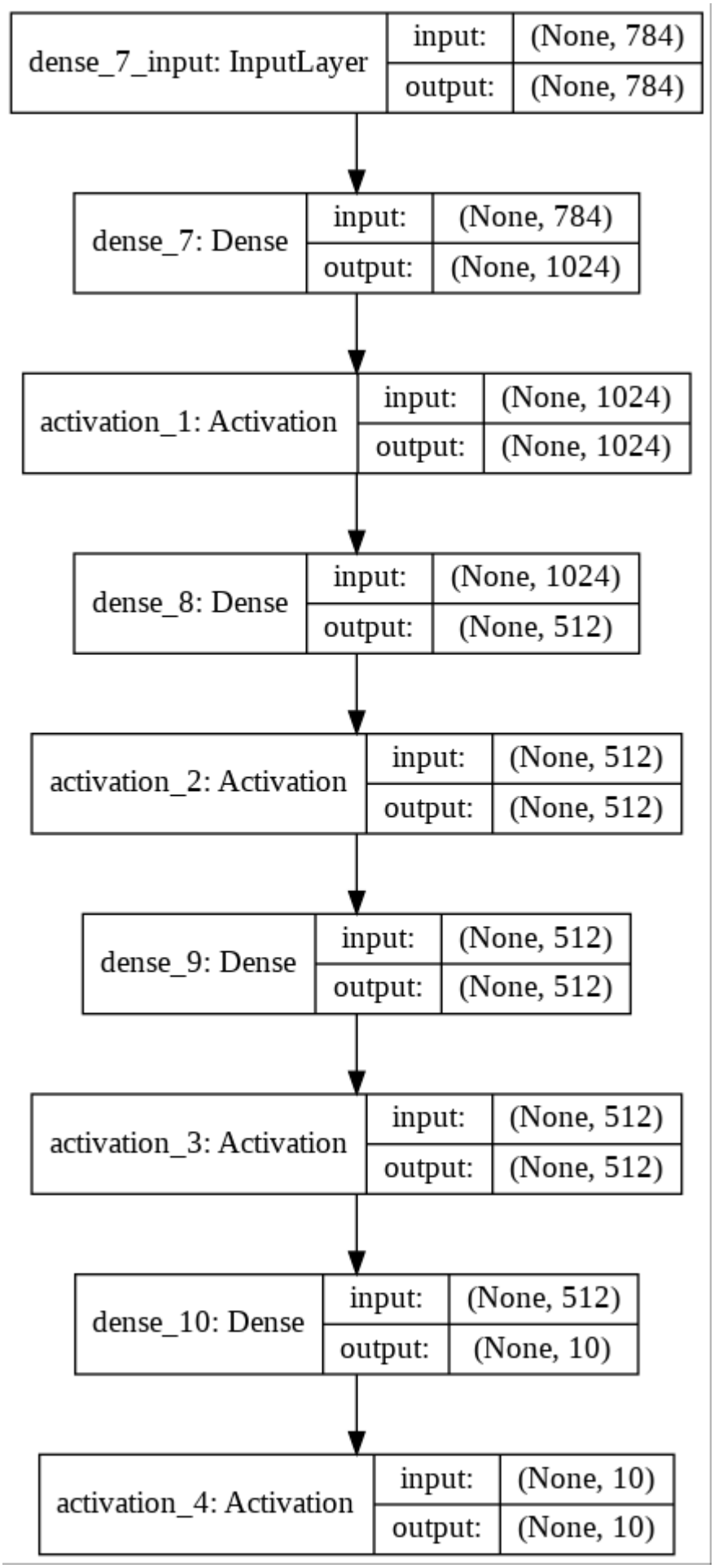
8. optimizer - adam

9. batch_size = 5000, epochs=50

10. Print the accuracy and loss of MLP model

Test Accuracy:-

```
Test loss: 0.07742713400616726
Test acc: 0.9823
```


dense_7_input: InputLayer
input: (None, 784)
output: (None, 784)
dense_7: Dense
input: (None, 784)
output: (None, 1024)
activation_1: Activation
input: (None, 1024)
output: (None, 1024)
dense_8: Dense
input: (None, 1024)
output: (None, 512)
activation_2: Activation
input: (None, 512)
output: (None, 512)
dense_9: Dense
input: (None, 512)
output: (None, 512)
activation_3: Activation
input: (None, 512)
output: (None, 512)
dense_10: Dense
input: (None, 512)
output: (None, 10)
activation_4: Activation
input: (None, 10)
output: (None, 10)

Question 6:

A. Build basic Convolution neural network on the MNIST Data.

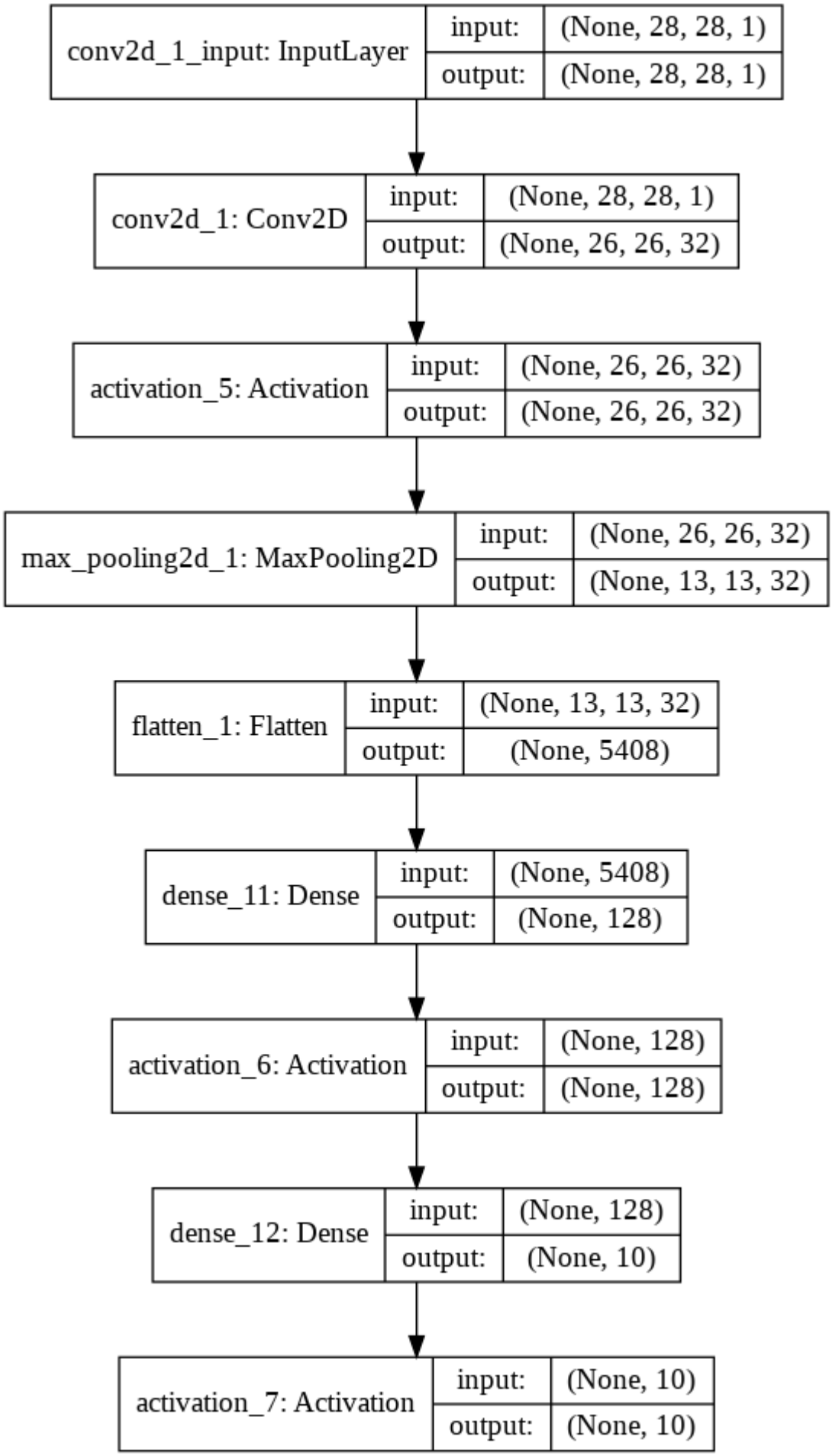
Reshape the Data:

```
before reshape the data (28, 28)
After reshape the data (60000, 28, 28, 1)
```

HINT:

1. Conv2D with 32 Neuron; Filter 3,3 ; Activation: Relu ; Stride (1,1)
2. MaxPool2D ; Pool Size (2,2)
3. Flatten the data again to send to desne layer
4. 128 Neuron single Dense Layer with relu
5. 10 neuron single dense layer with softmax as output layer

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
activation_5 (Activation)    (None, 26, 26, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 5408)              0
_________________________________________________________________
dense_11 (Dense)             (None, 128)               692352
_________________________________________________________________
activation_6 (Activation)    (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                1290
_________________________________________________________________
activation_7 (Activation)    (None, 10)                0
=================================================================
Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0
```


conv2d_1_input: InputLayer
input: (None, 28, 28, 1)
output: (None, 28, 28, 1)
conv2d_1: Conv2D
input: (None, 28, 28, 1)
output: (None, 26, 26, 32)
activation_5: Activation
input: (None, 26, 26, 32)
output: (None, 26, 26, 32)
max_pooling2d_1: MaxPooling2D
input: (None, 26, 26, 32)
output: (None, 13, 13, 32)
flatten_1: Flatten
input: (None, 13, 13, 32)
output: (None, 5408)
dense_11: Dense
input: (None, 5408)
output: (None, 128)
activation_6: Activation
input: (None, 128)
output: (None, 128)
dense_12: Dense
input: (None, 128)
output: (None, 10)
activation_7: Activation
input: (None, 10)
output: (None, 10)

B. Save the model into Json and .h5 file

model.json

model_try.h5

C. Load the model from disk

```
Loaded model from disk
Model: "sequential_16"
```