Transfer Learning MNIST

- Train a simple convnet on the MNIST dataset the first 5 digits [0-4].
- Freeze convolutional layers and fine-tune dense layers for the classification of digits [5-9].

MNIST Dataset

The MNIST database contains 60,000 training images and 10,000 testing images taken from Ame
American high school students. The MNIST dataset is one of the most common datasets used fo
many different sources. In fact, even Tensorflow and Keras allow us to import and download the M

Let's import keras and load MNIST dataset

```
%tensorflow_version 2.x
import tensorflow as tf
```

```
tensorflow.__version__
```

```
'2.1.0-rc1'
```

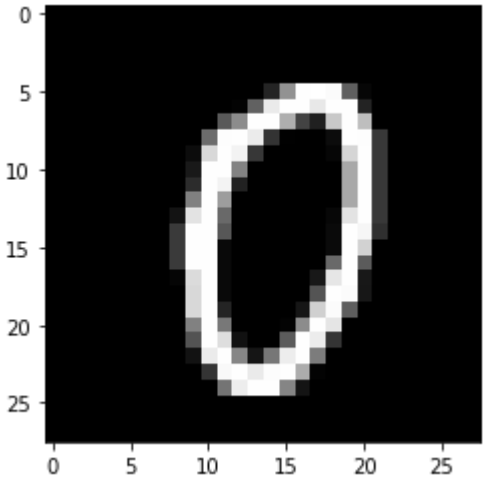
```
# Initialize the random number generator
import random
random.seed(0)

import warnings
warnings.filterwarnings("ignore")

from tensorflow.keras.backend import backend
from tensorflow.keras.datasets import mnist

# the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

X_train and X_test contain greyscale RGB codes (from 0 to 255) while y_train and y_test contains l
number they actually are.

Let's visualize some numbers using matplotlib

```
import matplotlib.pyplot as plt
%matplotlib inline
print("Label: {}".format(y_train[1000]))
plt.imshow(X_train[1000], cmap='gray')
```

```
Label: 0
<matplotlib.image.AxesImage at 0x7f00544c22b0>
```



Question 1

```
# This is formatted as code
```

Create two datasets

- First having digits from 0 to 4
- Second having digits from 5 to 9

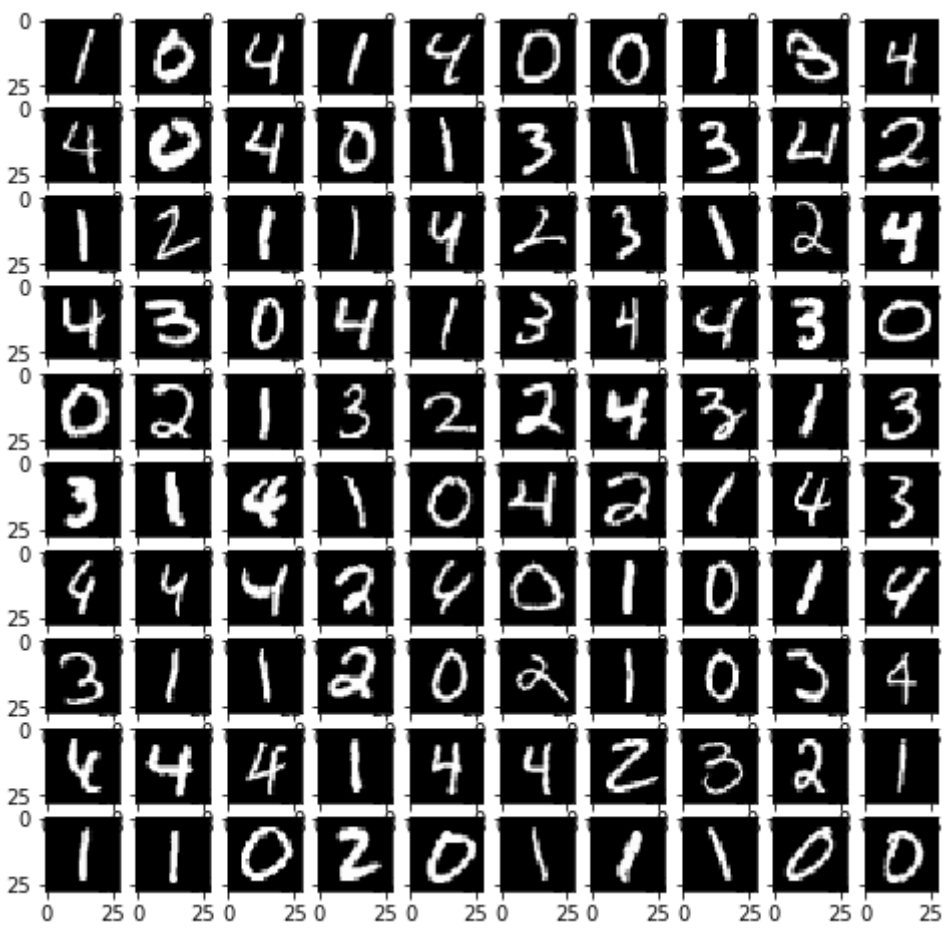
Hint: use labels to separate data

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

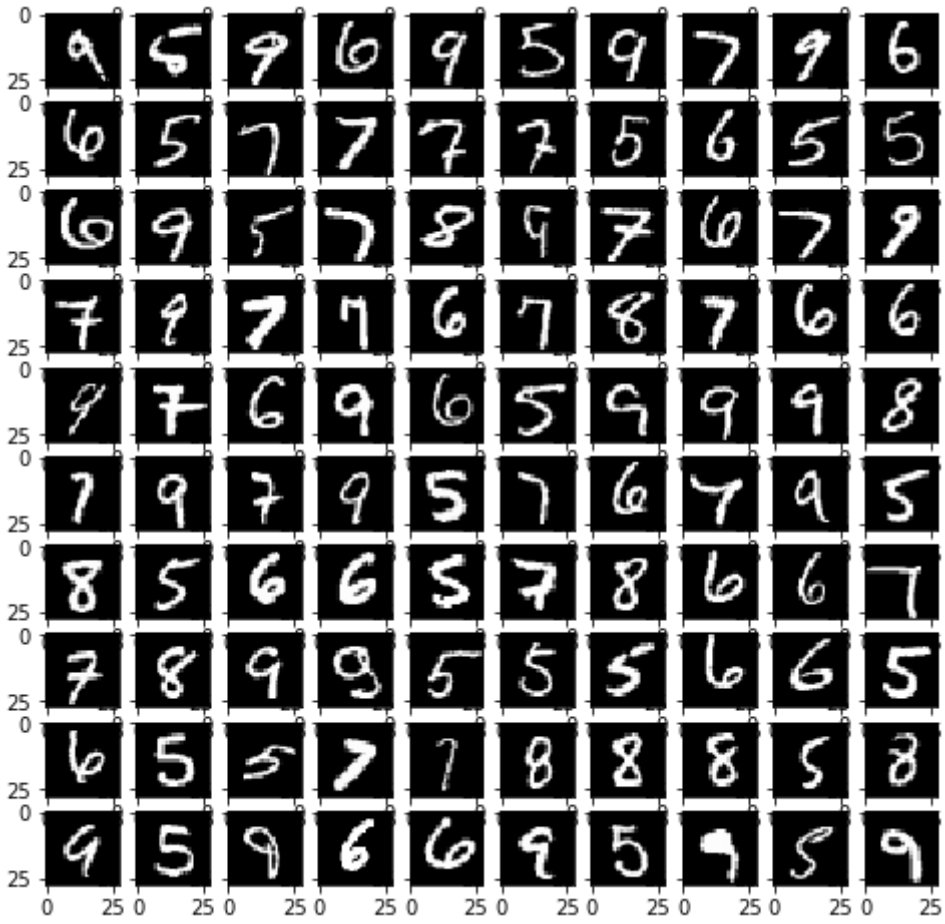
```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

```
X_train_lt5 = X_train[y_train < 5]
y_train_lt5 = y_train[y_train < 5]
X_test_lt5 = X_test[y_test < 5]
y_test_lt5 = y_test[y_test < 5]

X_train_gte5 = X_train[y_train >= 5]
y_train_gte5 = y_train[y_train >= 5] - 5
X_test_gte5 = X_test[y_test >= 5]
y_test_gte5 = y_test[y_test >= 5] - 5
```

```
w=10
h=10
fig=plt.figure(figsize=(8, 8))
columns = 10
rows = 10
for i in range(1, columns*rows +1):
    img = X_test_lt5[i]
    fig.add_subplot(rows, columns, i)
    plt.imshow(img, cmap='gray')
plt.show()
```



```
w=10
h=10
fig=plt.figure(figsize=(8, 8))
columns = 10
rows = 10
for i in range(1, columns*rows +1):
    img = X_test_gte5[i]
    fig.add_subplot(rows, columns, i)
    plt.imshow(img, cmap='gray')
plt.show()
```



Question 2

Print shape of the data

- print shape of all variables of both the datasets you created

```
print(X_train_lt5.shape)
print(y_train_lt5.shape)
print(X_test_lt5.shape)
print(y_test_lt5.shape)
```

```
(30596, 28, 28)
(30596,)
(5139, 28, 28)
(5139,)
```

```
print(X_train_gte5.shape)
print(y_train_gte5.shape)
print(X_test_gte5.shape)
print(y_test_gte5.shape)
```

```
(29404, 28, 28)
```

Question 3

Reshape data

- reshape first dataset
- To be able to use the dataset in Keras, we need 4-dims numpy arrays.
- reshape features to pass it to a Conv2D layer
- channel = 1
- reshape features of first dataset only
- do not reshape labels

```
X_train = X_train_lt5.reshape(X_train_lt5.shape[0], 28, 28, 1)
X_test = X_test_lt5.reshape(X_test_lt5.shape[0], 28, 28, 1)
```

Question 4

Normalize data

- normalize first dataset
- we must normalize our data as it is always required in neural network models
- we can achieve this by dividing the RGB codes to 255 (which is the maximum RGB code min
- normalize X_train and X_test
- make sure that the values are float so that we can get decimal points after division

```
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255
```

```
X_train[0]
```

```
array([[[0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
```

```
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
```

```
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.2       ],
       [0.62352943],
       [0.99215686],
       [0.62352943],
       [0.19607843],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.1882353 ],
       [0.93333334],
       [0.9882353 ],
       [0.9882353 ],
       [0.9882353 ],
       [0.92941177],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
```

```
       [0.        ],
       [0.        ],
       [0.        ],
       [0.21176471],
       [0.8901961 ],
       [0.99215686],
       [0.9882353 ],
       [0.9372549 ],
       [0.9137255 ],
       [0.9882353 ],
       [0.22352941],
       [0.02352941],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.03921569],
       [0.23529412],
       [0.8784314 ],
       [0.9882353 ],
       [0.99215686],
       [0.9882353 ],
       [0.7921569 ],
       [0.32941177],
       [0.9882353 ],
       [0.99215686],
       [0.47843137],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.6392157 ],
       [0.9882353 ],
       [0.9882353 ],
```

```
        [0.9882353 ],
        [0.99215686],
        [0.9882353 ],
        [0.9882353 ],
        [0.3764706 ],
        [0.7411765 ],
        [0.99215686],
        [0.654902  ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.2       ],
        [0.93333334],
        [0.99215686],
        [0.99215686],
        [0.74509805],
        [0.44705883],
        [0.99215686],
        [0.89411765],
        [0.18431373],
        [0.30980393],
        [1.        ],
        [0.65882355],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.1882353 ],
        [0.93333334],
        [0.9882353 ],
        [0.9882353 ],
        [0.7019608 ],
        [0.04705882],
        [0.29411766],
        [0.4745098 ],
```

```
       [0.08235294],
       [0.        ],
       [0.        ],
       [0.99215686],
       [0.9529412 ],
       [0.19607843],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.14901961],
       [0.64705884],
       [0.99215686],
       [0.9137255 ],
       [0.8156863 ],
       [0.32941177],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.99215686],
       [0.9882353 ],
       [0.64705884],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.02745098],
       [0.69803923],
       [0.9882353 ],
       [0.9411765 ],
       [0.2784314 ],
       [0.07450981],
       [0.10980392],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.99215686],
```

```
       [0.9882353 ],
       [0.7647059 ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.22352941],
       [0.9882353 ],
       [0.9882353 ],
       [0.24705882],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.99215686],
       [0.9882353 ],
       [0.7647059 ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.7764706 ],
       [0.99215686],
       [0.74509805],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [1.        ],
       [0.99215686],
       [0.76862746],
       [0.        ],
```

```
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.29803923],
        [0.9647059 ],
        [0.9882353 ],
        [0.4392157 ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.99215686],
        [0.9882353 ],
        [0.5803922 ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.33333334],
        [0.9882353 ],
        [0.9019608 ],
        [0.09803922],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.02745098],
        [0.5294118 ],
        [0.99215686],
        [0.7294118 ],
        [0.04705882],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]],
```

```
       [[0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.33333334],
        [0.9882353 ],
        [0.8745098 ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.02745098],
        [0.5137255 ],
        [0.9882353 ],
        [0.88235295],
        [0.2784314 ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.33333334],
        [0.9882353 ],
        [0.5686275 ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.1882353 ],
        [0.64705884],
        [0.9882353 ],
        [0.6784314 ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
```

```
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.3372549 ],
       [0.99215686],
       [0.88235295],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.44705883],
       [0.93333334],
       [0.99215686],
       [0.63529414],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.33333334],
       [0.9882353 ],
       [0.9764706 ],
       [0.57254905],
       [0.1882353 ],
       [0.11372549],
       [0.33333334],
       [0.69803923],
       [0.88235295],
       [0.99215686],
       [0.8745098 ],
       [0.654902  ],
       [0.21960784],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
```

```
       [0.33333334],
       [0.9882353 ],
       [0.9882353 ],
       [0.9882353 ],
       [0.8980392 ],
       [0.84313726],
       [0.9882353 ],
       [0.9882353 ],
       [0.9882353 ],
       [0.76862746],
       [0.50980395],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.10980392],
       [0.78039217],
       [0.9882353 ],
       [0.9882353 ],
       [0.99215686],
       [0.9882353 ],
       [0.9882353 ],
       [0.9137255 ],
       [0.5686275 ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.09803922],
       [0.5019608 ],
       [0.9882353 ],
```

```
       [0.9882353 ],
       [0.99215686],
       [0.9882353 ],
       [0.5529412 ],
       [0.14509805],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
```

```
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ]],

        [[0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ]],

        [[0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ],
         [0.        ]
```

```
[0.        ],
[0.        ],
[0.        ],
[0.        ],
[0.        ],
[0.        ],
[0.        ],
[0.        ],
[0.        ],
[0.        ],
[0.        ],
[0.        ]]], dtype=float32)
```

Print shape of data and number of images

- for first dataset
- print shape of X_train
- print number of images in X_train
- print number of images in X_test

Question 5

One-hot encode the class vector

- encode labels of first dataset
- convert class vectors (integers) to binary class matrix
- convert y_train and y_test
- number of classes: 5
- we are doing this to use categorical_crossentropy as loss

Hint: you can use keras.utils.to_categorical

```
y_train = tensorflow.keras.utils.to_categorical(y_train_lt5,5)
y_test = tensorflow.keras.utils.to_categorical(y_test_lt5,5)
```

Question 6

We will build our model by using high level Keras.

Initialize a sequential model

- define a sequential model
- add 2 convolutional layers
    - no of filters: 32
    - kernel size: 3x3
    - activation: "relu"
    - input shape: (28, 28, 1) for first layer
- add a max pooling layer of size 2x2

- add a dropout layer
    - dropout layers fight with the overfitting by disregarding some of the neurons while trair
    - use dropout rate 0.2

```
from __future__ import absolute_import, division, print_function
import numpy as np
import tensorflow.keras as keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
import pickle
from matplotlib import pyplot as plt
import seaborn as sns
plt.rcParams['figure.figsize'] = (15, 8)


tf.keras.backend.clear_session()



batch_size = 128
num_classes = 5
epochs = 10


#Initialize the model
model = Sequential()

#Add a Convolutional Layer with 32 filters of size 3X3 and activation function as 'ReLU'
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(28,28,1),name='conv_1'))

#Add a Convolutional Layer with 64 filters of size 3X3 and activation function as 'ReLU'
model.add(Conv2D(32, (3, 3), activation='relu',name='conv_2'))

#Add a MaxPooling Layer of size 2X2
model.add(MaxPooling2D(pool_size=(2, 2),name='max_1'))

#Apply Dropout with 0.25 probability
model.add(Dropout(0.25,name='drop_1'))
```

Question 7

Add classification layers

- do this after doing question 6
- flatten the data
    - add Flatten later
    - flatten layers flatten 2D arrays to 1D array before building the fully connected layers

- add 2 dense layers
    - number of neurons in first layer: 128
    - number of neurons in last layer: number of classes
    - activation function in first layer: relu
    - activation function in last layer: softmax
    - we may experiment with any number of neurons for the first Dense layer; however, the l the number of output classes
- you can add a dropout layer in between, if necessary

```
#Flatten the layer
model.add(Flatten())

#Add Fully Connected Layer with 128 units and activation function as 'ReLU'
model.add(Dense(128, activation='relu',name='dense_1'))

#Apply Dropout with 0.2 probability
model.add(Dropout(0.2,name='drop_2'))

#Add Fully Connected Layer with 10 units and activation function as 'softmax'
model.add(Dense(num_classes, activation='softmax',name='dense_2'))
```

Question 8

Compile and fit the model

- compile your model
    - loss: "categorical_crossentropy"
    - metrics: "accuracy"
    - optimizer: "sgd"
- fit your model
    - give train data - features and labels
    - batch size: 128
    - epochs: 10
    - give validation data - features and labels

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import categorical_crossentropy

#To use adam optimizer for learning weights with learning rate = 0.001
optimizer = Adam(lr=0.001)
#Set the loss function and optimizer for the model training
model.compile(loss=categorical_crossentropy,
              optimizer=optimizer,
              metrics=['accuracy'])
```

```
#Training on the dataset
model.fit(X_train, y_train,
          batch_size=batch_size,
          epochs=10,
          verbose=1,
          validation_data=(X_test, y_test))
```

```
Train on 30596 samples, validate on 5139 samples
Epoch 1/10
30596/30596 [==============================] - 2s 58us/sample - loss: 0.1163 - accura
Epoch 2/10
30596/30596 [==============================] - 1s 44us/sample - loss: 0.0281 - accura
Epoch 3/10
30596/30596 [==============================] - 1s 44us/sample - loss: 0.0169 - accura
Epoch 4/10
30596/30596 [==============================] - 1s 44us/sample - loss: 0.0115 - accura
Epoch 5/10
30596/30596 [==============================] - 1s 44us/sample - loss: 0.0100 - accura
Epoch 6/10
30596/30596 [==============================] - 1s 43us/sample - loss: 0.0093 - accura
Epoch 7/10
30596/30596 [==============================] - 1s 44us/sample - loss: 0.0052 - accura
Epoch 8/10
30596/30596 [==============================] - 1s 44us/sample - loss: 0.0048 - accura
Epoch 9/10
30596/30596 [==============================] - 1s 44us/sample - loss: 0.0041 - accura
Epoch 10/10
30596/30596 [==============================] - 1s 44us/sample - loss: 0.0034 - accura
<tensorflow.python.keras.callbacks.History at 0x7f00c58bf4a8>
```

Question 9

Evaluate model

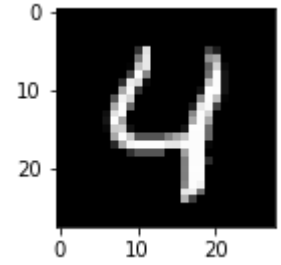
- evaluate your model and get accuracy
- use test features and labels

```
#Testing the model on test set
score = model.evaluate(X_test, y_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

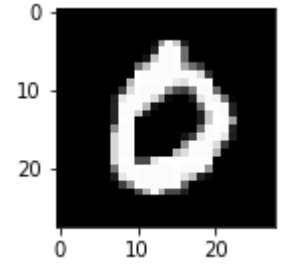
```
5139/5139 [==============================] - 0s 74us/sample - loss: 0.0075 - accuracy
Test loss: 0.007510769576315465
Test accuracy: 0.9976649
```

```
import numpy as np

plt.figure(figsize=(2,2))
plt.imshow(X_test[3].reshape(28,28),cmap="gray")
```

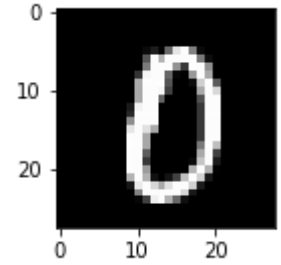
```
plt.show()
print(np.argmax(model.predict(X_test[3].reshape(1,28,28,1))))


plt.figure(figsize=(2,2))
plt.imshow(X_test[2].reshape(28,28),cmap="gray")
plt.show()
print(np.argmax(model.predict(X_test[2].reshape(1,28,28,1))))


plt.figure(figsize=(2,2))
plt.imshow(X_test[33].reshape(28,28),cmap="gray")
plt.show()
print(np.argmax(model.predict(X_test[33].reshape(1,28,28,1))))


plt.figure(figsize=(2,2))
plt.imshow(X_test[59].reshape(28,28),cmap="gray")
plt.show()
print(np.argmax(model.predict(X_test[59].reshape(1,28,28,1))))


plt.figure(figsize=(2,2))
plt.imshow(X_test[36].reshape(28,28),cmap="gray")
plt.show()
print(np.argmax(model.predict(X_test[36].reshape(1,28,28,1))))
```



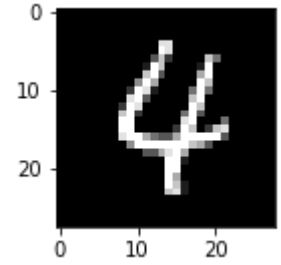
4



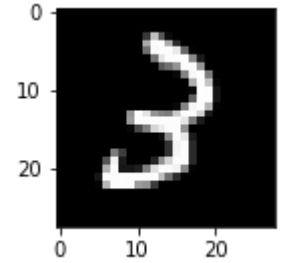
0



0



4



3

```
#Set the path where you want to store the model and weights.
model.save('./cnn_r8_lab.h5')
```

```
model.save_weights('./cnn_r8_lab.h5')
```

Question 10

Transfer learning

Now we will apply this model on second dataset (5-9 digits)

- fix the first convolution layers so that the weights in the convolution layers dont get updated
- get the second dataset
- train the last 2 dense layers
- predict the accuracy and loss

Make only dense layers trainable

- set trainalble = False for all layers other than Dense layers

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
import h5py

tf.keras.backend.clear_session()



#Freezing layers in the model which don't have 'dense' in their name # we have bulit archi
for layer in model.layers:
  if('dense' not in layer.name): #prefix detection to freeze layers which does not have de
    #Freezing a layer
    layer.trainable = False

#Module to print colourful statements
from termcolor import colored

#Check which layers have been frozen
for layer in model.layers:
  print (colored(layer.name, 'blue'))
  print (colored(layer.trainable, 'red'))
```