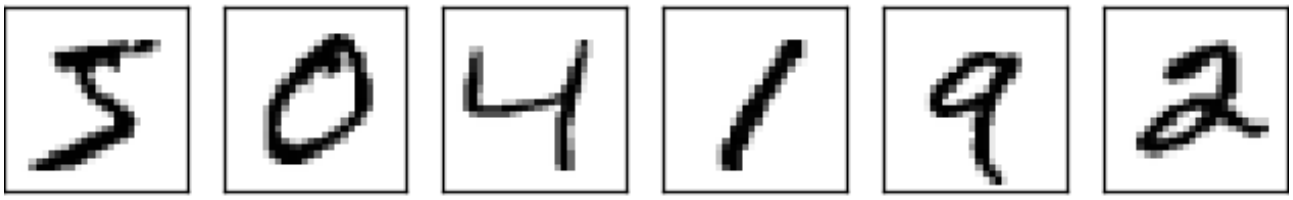
Classification of Handwritten Digits using ANN

Input data

Images of handwritten digits

Image size : 28 x 28 pixels = 784 pixels

Each pixel has a value between 0 and 255.
Zero represents black, 255 represents white,
and a number in between represents a shade of grey.



Data loading and normalisation

Input data of 60,000 training images and 10,000 testing images is in the form of 28x28 array.

```
from keras.datasets import mnist
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()
```

Reshape the images into a single vector of 784 elements.

```
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
```

Normalise the input data values so that they are in the range 0-1, instead of 0-255.

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

In the MNSIT dataset, the output is a digit in the range [0,9]. For our Machine Learning algorithm, we need to convert this to one-hot encoding. So instead of a single digit, the output is a vector of length 10. The values of the vector are all zeros, except for the index representing the given digit, which has an entry of one. So, the digit 5 is represented by [0,0,0,0,0,1,0,0,0,0] and 8 is represented by [0,0,0,0,0,0,0,0,1,0].

```
Y_train = np_utils.to_categorical(Y_train, 10)
Y_test = np_utils.to_categorical(Y_test, 10)
```

ANN structure

(write this part on your own and run the code)

```
    from keras.models import Sequential
    from keras.layers.core import Dense, Dropout, Activation
    from keras.utils import np_utils
```

Input layer of 784 nodes. Each node corresponding to each pixel value of the image.

Two hidden layers of 512 nodes each, using ReLU activation function.
At each iteration of training, randomly drop 20% of the neural network connections. This helps in preventing over-fitting. This can simply be done by passing the following argument inside the Sequential function:
Dropout=0.2

Output layer of 10 nodes, corresponding to the 10 digits of decimal number system (0-9).
Each node represents the probability of the input image representing a particular digit.
Use softmax activation function for this output layer.

Some parameters to be used for compilation and fitting of the ANN:
loss='categorical_crossentropy'
optimizer='adam'
batch_size=128
epochs=20