Lab 3 - Cloud Computing

TASK 7

Printing a basic Hello World

```
In [12]: print("Hello World")
```

```
Hello World
```

TASK 8

Training a model

```
In [3]: import tensorflow as tf
        from tensorflow import keras
        import numpy as np
        import matplotlib.pyplot as plt
        print(tf.__version__)
```

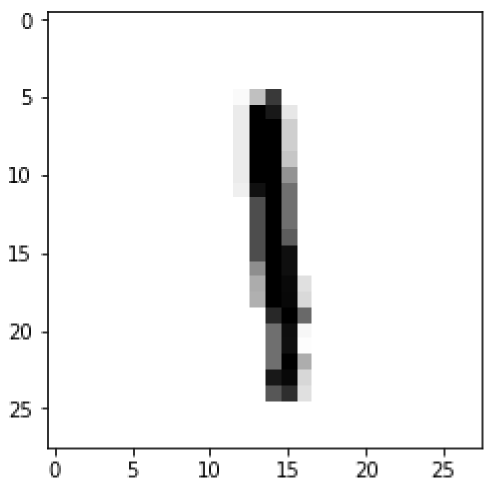
```
1.4.1
```

```
In [4]: mnist = tf.keras.datasets.mnist
        (x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11476992/11490434 [============================>.] - ETA: 0s
```

```
In [5]: import matplotlib.pyplot as plt
        plt.imshow(x_train[8], cmap=plt.cm.binary)
```

```
Out[5]: <matplotlib.image.AxesImage at 0x7fc7e496af60>
```



In [6]:
```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
```

In [7]:
```
from keras.utils import to_categorical
```

```
Using TensorFlow backend.
```

In [8]:
```
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)
```

In [9]:
```
from keras import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(10, activation='sigmoid', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
=================================================================
Total params: 7,960
Trainable params: 7,960
Non-trainable params: 0
_________________________________________________________________
```

In [10]:
```
model.compile(loss="categorical_crossentropy",
 optimizer="sgd",
 metrics = ['accuracy'])
model.fit(x_train, y_train, epochs=5)
```

```
Epoch 1/5
60000/60000 [==============================] - 4s 68us/step - loss: 1.9443
- acc: 0.5252
Epoch 2/5
60000/60000 [==============================] - 4s 61us/step - loss: 1.3377
- acc: 0.7145
Epoch 3/5
60000/60000 [==============================] - 4s 64us/step - loss: 0.9780
- acc: 0.7907
Epoch 4/5
60000/60000 [==============================] - 4s 62us/step - loss: 0.7864
- acc: 0.8269
Epoch 5/5
60000/60000 [==============================] - 4s 64us/step - loss: 0.6734
- acc: 0.8438
```

Out[10]:
```
<keras.callbacks.History at 0x7fc7e7eae5f8>
```

In [11]:

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

```
10000/10000 [==============================] - 0s 40us/step
Test accuracy: 0.8595
```