Develop a Neural Network that can Read Handwriting

The objective of this research is to create a neural network that is very accurate at reading handwritten characters. The neural network will be able to identify the characters in fresh photographs after being trained on a collection of handwritten character images.

```
import tensorflow as tf
```

```
m = tf.keras.datasets.mnist
```

```
(x_train, y_train),(x_test, y_test) = m.load_data()
```

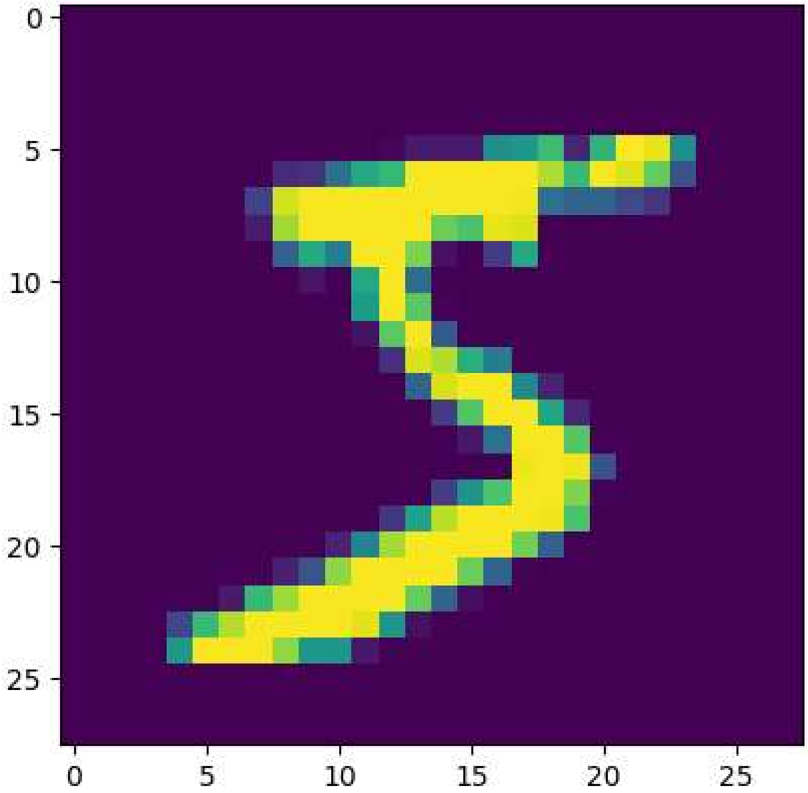
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist
11490434/11490434 [==============================] - 0s 0us/step
```

```
x_train.shape
```

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

```
import matplotlib.pyplot as plt
```

```
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
plt.show()
plt.imshow(x_train[0], cmap = plt.cm.binary)
```



```
<matplotlib.image.AxesImage at 0x7f4cbf377e80>
```

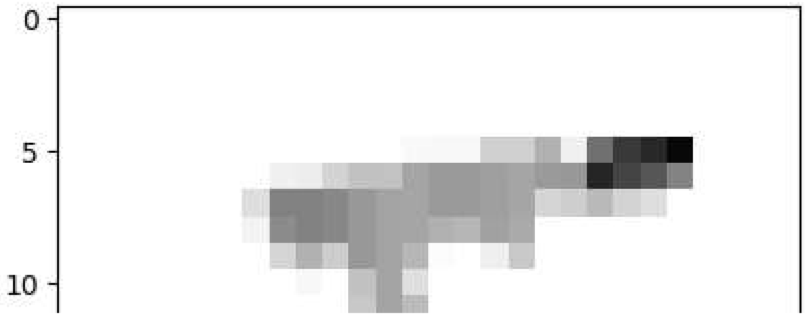
```
print (x_train[0])
```

```
[[  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
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
```

```
x_train = tf.keras.utils.normalize (x_train, axis = 1)
x_test = tf.keras.utils.normalize(x_test, axis=1)
plt.imshow(x_train[0], cmap = plt.cm.binary)
```

```
<matplotlib.image.AxesImage at 0x7f4cbd24cc10>
```



```
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.          ]]
```

```
print (y_train[0])
```

```
     5
```

```
import numpy as np
```

```
IMG_SIZE = 28
x_trainr = np.array(x_train).reshape(-1, IMG_SIZE, IMG_SIZE,1)
x_testr = np.array(x_test).reshape(-1, IMG_SIZE, IMG_SIZE,1)
```

```
print ("Training Samples dimension",x_trainr.shape)
print ("Testing Sapmles dimension",x_testr.shape)
```

```
     Training Samples dimension (60000, 28, 28, 1)
     Testing Sapmles dimension (10000, 28, 28, 1)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
```

```
mod = Sequential()
mod.add(Conv2D(64, (3,3), input_shape = x_trainr.shape[1:]))
mod.add(Activation("relu"))
mod.add(MaxPooling2D(pool_size=(2,2)))
mod.add(Conv2D(64, (3,3)))
mod.add(Activation("relu"))
mod.add(MaxPooling2D(pool_size=(2,2)))
mod.add(Conv2D(64, (3,3)))
mod.add(Activation("relu"))
mod.add(MaxPooling2D(pool_size=(2,2)))
mod.add (Flatten())
mod.add (Dense(64))
mod.add(Activation("relu"))
mod.add (Dense(32))
mod.add(Activation("relu"))
mod.add(Dense(10))
mod.add(Activation('softmax'))


mod.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 activation (Activation)     (None, 26, 26, 64)        0

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        36928

 activation_1 (Activation)   (None, 11, 11, 64)        0

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 activation_2 (Activation)   (None, 3, 3, 64)          0

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 64)         0
 2D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 activation_3 (Activation)   (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 activation_4 (Activation)   (None, 32)                0

 dense_2 (Dense)             (None, 10)                330

 activation_5 (Activation)   (None, 10)                0

=================================================================
Total params: 81,066
Trainable params: 81,066
Non-trainable params: 0
_________________________________________________________________
```

```
print("Total Training Samples = ",len(x_trainr))
```

```
Total Training Samples =  60000
```

```
mod.compile(loss ="sparse_categorical_crossentropy", optimizer ="adam", metrics=['accuracy'])
```

```
mod.fit (x_trainr,y_train,epochs=5, validation_split = 0.3)
```

```
Epoch 1/5
1313/1313 [==============================] - 92s 68ms/step - loss: 0.3282 - accuracy: 0
Epoch 2/5
1313/1313 [==============================] - 82s 63ms/step - loss: 0.1026 - accuracy: 0
Epoch 3/5
1313/1313 [==============================] - 81s 62ms/step - loss: 0.0743 - accuracy: 0
Epoch 4/5
1313/1313 [==============================] - 82s 62ms/step - loss: 0.0567 - accuracy: 0
Epoch 5/5
1313/1313 [==============================] - 82s 63ms/step - loss: 0.0480 - accuracy: 0
<keras.callbacks.History at 0x7f4cc025b700>
```

```
test_loss, test_acc = mod.evaluate(x_testr, y_test)
print ("Test Loss on 10,000 test samples",test_loss)
print ("Validation Accuracy on 10,000 test samples",test_acc)
```

```
313/313 [==============================] - 6s 20ms/step - loss: 0.0683 - accuracy: 0.978
Test Loss on 10,000 test samples 0.0683475211262703
Validation Accuracy on 10,000 test samples 0.9786999821662903
```

```
predictions = mod.predict([x_testr])
```

```
313/313 [==============================] - 7s 21ms/step
```

```
print(predictions)
```

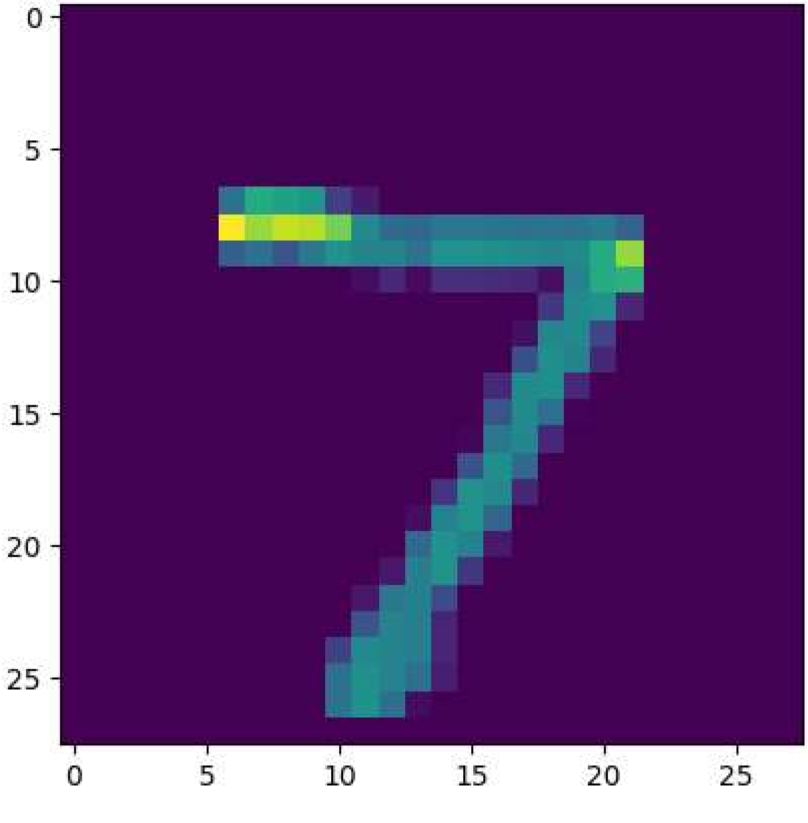
```
[[6.0464544e-10 2.1049615e-08 1.3939098e-06 ... 9.9999851e-01
  3.4068406e-10 2.7682974e-08]
 [1.4981175e-04 4.0685632e-06 9.9970090e-01 ... 3.7577622e-05
  8.7426806e-06 1.5823326e-07]
 [1.3877703e-06 9.9990636e-01 5.3089457e-06 ... 1.3208566e-07
  2.7656574e-06 5.1671771e-07]
 ...
 [5.9518911e-08 1.0387527e-06 1.1210544e-08 ... 9.3059434e-06
  5.5926097e-07 2.3022749e-05]
 [5.4276956e-08 2.4685649e-11 1.6341548e-10 ... 9.5684929e-16
  7.5553551e-08 2.2211875e-08]
 [4.0824875e-06 3.2617006e-08 4.3763222e-07 ... 3.3171453e-14
  2.0056157e-06 1.4004372e-07]]
```

```
print(np.argmax(predictions[1]))
```

```
    2
```

```
plt.imshow(x_test[0])
```

```
    <matplotlib.image.AxesImage at 0x7f4c79bf3f10>
```

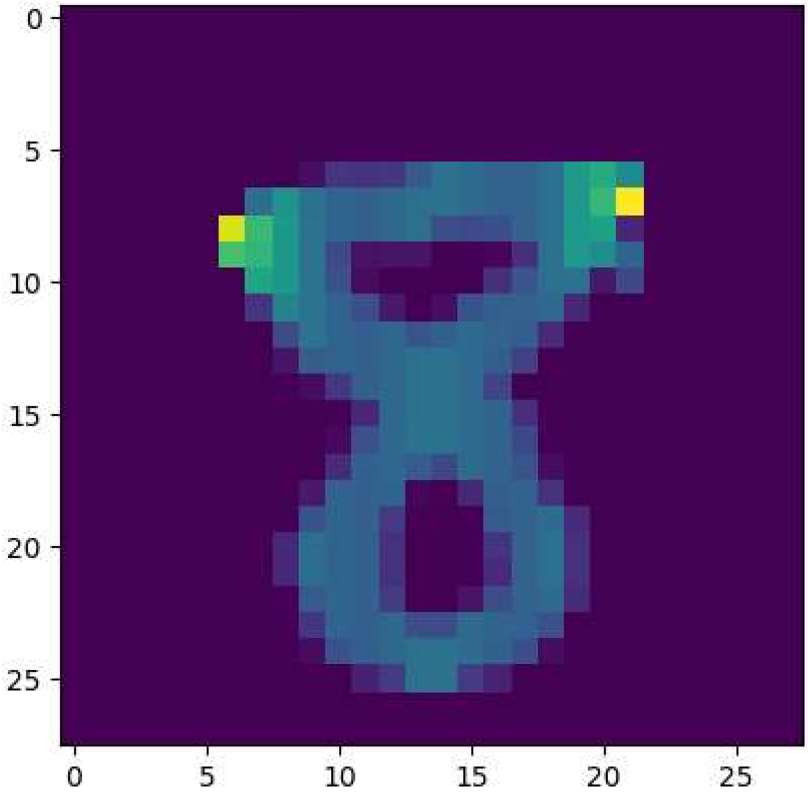


```
print(np.argmax(predictions[128]))
```

```
    8
```

```
plt.imshow(x_test[128])
```

```
<matplotlib.image.AxesImage at 0x7f4c79cd00d0>
```



0s completed at 6:58 PM