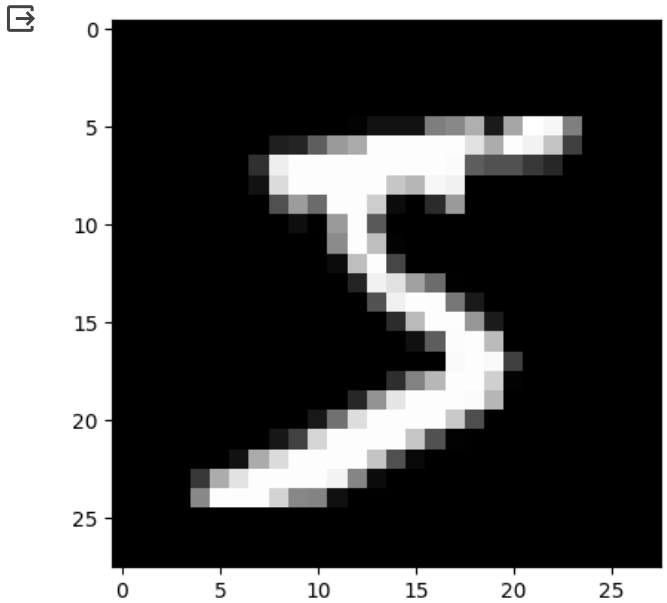
```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
from sklearn import metrics
```

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

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

```
plt.imshow(x_train[0], cmap='gray')
plt.show()
```




```
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   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]]
```

```
print("x_train shape: ",x_train.shape)
print("y_train shape: ",y_train.shape)
print("x_test shape: ",x_test.shape)
print("y_test shape: ",y_test.shape)
```

```
x_train shape:  (60000, 28, 28)
y_train shape:  (60000,)
x_test shape:  (10000, 28, 28)
y_test shape:  (10000,)
```

```
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train = x_train/255 # Each image has Intensity from 0 to 255
x_test = x_test/255
```

```
num_classes = 10
y_train = np.eye(num_classes)[y_train]
y_test = np.eye(num_classes)[y_test]
```

```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=RMSprop(), metrics=['accuracy'])
```

```
batch_size = 128
epochs = 20
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Epoch 1/20
469/469 [==============================] - 11s 22ms/step - loss: 0.2561 - accuracy: 0.9220 - val_loss: 0.1122 - val_accuracy: 0.9631
Epoch 2/20
469/469 [==============================] - 8s 16ms/step - loss: 0.1031 - accuracy: 0.9679 - val_loss: 0.0865 - val_accuracy: 0.9730
Epoch 3/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0739 - accuracy: 0.9767 - val_loss: 0.0946 - val_accuracy: 0.9718
Epoch 4/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0585 - accuracy: 0.9820 - val_loss: 0.0708 - val_accuracy: 0.9785
Epoch 5/20
469/469 [==============================] - 8s 17ms/step - loss: 0.0478 - accuracy: 0.9849 - val_loss: 0.0792 - val_accuracy: 0.9780
Epoch 6/20
469/469 [==============================] - 9s 18ms/step - loss: 0.0399 - accuracy: 0.9878 - val_loss: 0.0705 - val_accuracy: 0.9805
Epoch 7/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0345 - accuracy: 0.9889 - val_loss: 0.0653 - val_accuracy: 0.9825
Epoch 8/20
469/469 [==============================] - 8s 17ms/step - loss: 0.0297 - accuracy: 0.9905 - val_loss: 0.0657 - val_accuracy: 0.9835
Epoch 9/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0267 - accuracy: 0.9918 - val_loss: 0.0687 - val_accuracy: 0.9843
Epoch 10/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0234 - accuracy: 0.9923 - val_loss: 0.0675 - val_accuracy: 0.9835
Epoch 11/20
469/469 [==============================] - 8s 16ms/step - loss: 0.0214 - accuracy: 0.9931 - val_loss: 0.0631 - val_accuracy: 0.9855
```

```
Epoch 12/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0176 - accuracy: 0.9943 - val_loss: 0.0730 - val_accuracy: 0.9834
Epoch 13/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0176 - accuracy: 0.9942 - val_loss: 0.0710 - val_accuracy: 0.9848
Epoch 14/20
469/469 [==============================] - 8s 17ms/step - loss: 0.0143 - accuracy: 0.9955 - val_loss: 0.0738 - val_accuracy: 0.9850
Epoch 15/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0141 - accuracy: 0.9955 - val_loss: 0.0784 - val_accuracy: 0.9846
Epoch 16/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0129 - accuracy: 0.9958 - val_loss: 0.0792 - val_accuracy: 0.9834
Epoch 17/20
469/469 [==============================] - 8s 17ms/step - loss: 0.0117 - accuracy: 0.9961 - val_loss: 0.0799 - val_accuracy: 0.9835
Epoch 18/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0108 - accuracy: 0.9966 - val_loss: 0.0776 - val_accuracy: 0.9850
Epoch 19/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0095 - accuracy: 0.9968 - val_loss: 0.0805 - val_accuracy: 0.9843
Epoch 20/20
469/469 [==============================] - 8s 16ms/step - loss: 0.0106 - accuracy: 0.9966 - val_loss: 0.0739 - val_accuracy: 0.9853
```

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.07389672100543976
Test accuracy: 0.9853000044822693
```