

[illegible]

[illegible]


```

0,      [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,      0,  0],
0,      [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,      0,  0],
0,      [ 0,  0,  0,  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=uint8)

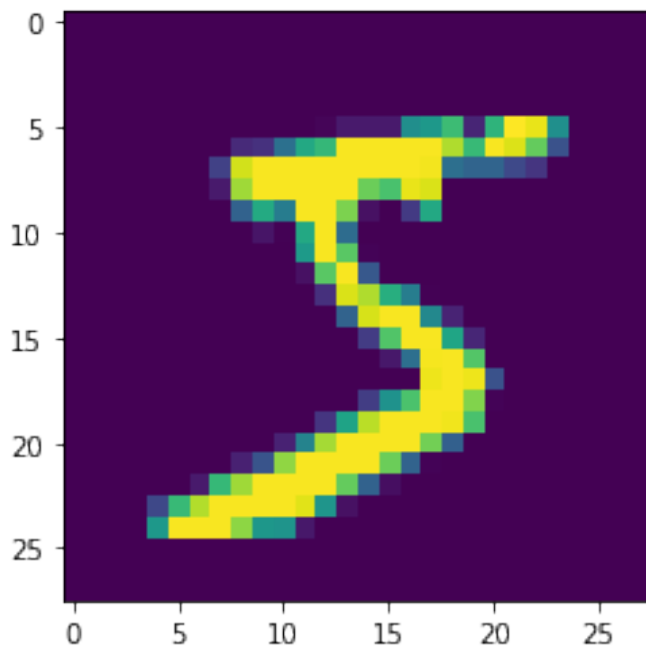
```

```
y_train[0]
```

```
5
```

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

```
<matplotlib.image.AxesImage at 0x7fc4986466d0>
```



```
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
```

```
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

```
number_of_classes = 10
```

```
Y_train = np_utils.to_categorical(y_train, number_of_classes)
```

```
Y_test = np_utils.to_categorical(y_test, number_of_classes)
```

```

Y_train[0]

array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)

model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1),
activation="relu"))
model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(Flatten())
model.add(Dense(number_of_classes, activation="softmax"))

model.compile(loss='categorical_crossentropy', optimizer="Adam",
metrics=["accuracy"])

model.fit(X_train, Y_train, batch_size=32, epochs=5,
validation_data=(X_test,Y_test))

Epoch 1/5
1875/1875 [=====] - 192s 102ms/step - loss:
0.2245 - accuracy: 0.9518 - val_loss: 0.1058 - val_accuracy: 0.9701
Epoch 2/5
1875/1875 [=====] - 197s 105ms/step - loss:
0.0685 - accuracy: 0.9788 - val_loss: 0.0962 - val_accuracy: 0.9752
Epoch 3/5
1875/1875 [=====] - 190s 101ms/step - loss:
0.0468 - accuracy: 0.9854 - val_loss: 0.0900 - val_accuracy: 0.9749
Epoch 4/5
1875/1875 [=====] - 190s 102ms/step - loss:
0.0351 - accuracy: 0.9891 - val_loss: 0.0993 - val_accuracy: 0.9748
Epoch 5/5
1875/1875 [=====] - 191s 102ms/step - loss:
0.0270 - accuracy: 0.9917 - val_loss: 0.1005 - val_accuracy: 0.9764

<keras.callbacks.History at 0x7fc493e218d0>

metrics = model.evaluate(X_test, Y_test, verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)

Metrics (Test Loss & Test Accuracy):
[0.10052110999822617, 0.9764000177383423]

prediction = model.predict(X_test[:4])
print(prediction)

1/1 [=====] - 0s 92ms/step
[[1.5678695e-09 1.6640128e-14 2.0494097e-12 1.5698962e-08 5.4015579e-
15
 3.6338055e-13 2.2240399e-20 1.0000000e+00 2.9577885e-08 1.9005494e-
08]
 [5.8188578e-09 1.2512093e-10 9.9999821e-01 7.4831279e-09 1.0770124e-
10

```

```

2.9252167e-18 1.6483800e-06 1.5410843e-14 1.2811967e-07 3.3103555e-
12]
[1.2689595e-09 9.9028254e-01 3.9091717e-08 1.3732340e-10 9.6216686e-
03
2.9094124e-07 1.9340013e-10 4.5208512e-07 9.5003670e-05 2.4108826e-
10]
[1.0000000e+00 7.3556976e-16 3.5439882e-12 4.7910155e-14 3.2022885e-
12
1.5000925e-12 1.5939531e-11 4.1566353e-14 7.7353792e-12 1.2456662e-
09]]

```

```

print(numpy.argmax(prediction, axis=1))
print(Y_test[:4])

```

```

[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

```

```

model.save("model.h5")

```

```

model=load_model("model.h5")

```

```

from keras.datasets import mnist
from matplotlib import pyplot
(X_train,y_train),(X_test,y_test)=mnist.load_data()
print('X_train:' +str(X_train.shape))
print('y_train:' +str(y_train.shape))
print('X_test:' +str(X_test.shape))
print('y_test:' +str(y_test.shape))
from matplotlib import pyplot
for i in range(9):
    pyplot.subplot(330+1+i)
    pyplot.imshow(X_train[i],cmap=pyplot.get_cmap('gray'))
    pyplot.show()

```

```

X_train:(60000, 28, 28)
y_train:(60000,)
X_test:(10000, 28, 28)
y_test:(10000,)

```

