Import biblioteki **TensorFlow** (https://www.tensorflow.org/) z której będziemy korzystali w **uczeniu maszynowym**:

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np


import keras
from keras.models import Sequential
from keras.layers import Dense
```

Numbers recognition - dataset **MNIST**

Download dataset

```
(train_data, train_labels), (test_data, test_labels) = tf.keras.datasets.mnist.load_data()
```

```
(train_data, train_labels), (test_data, test_labels) = tf.keras.datasets.mnist.load_data()
```

```
data = np.concatenate([train_data, test_data])
```

```
data.shape
```

```
(70000, 28, 28)
```

```
label = np.concatenate([train_labels,test_labels])
```

```
label.shape
```

```
(70000,)
```

Informations about dataset

```
train_data.shape,train_labels.shape
```

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

```
test_data.shape,test_labels.shape
```

```
((10000, 28, 28), (10000,))
```

```
train_data[0]
```

```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   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],
```

```
train_labels[0]
```

```
5
```

One-hot encoding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

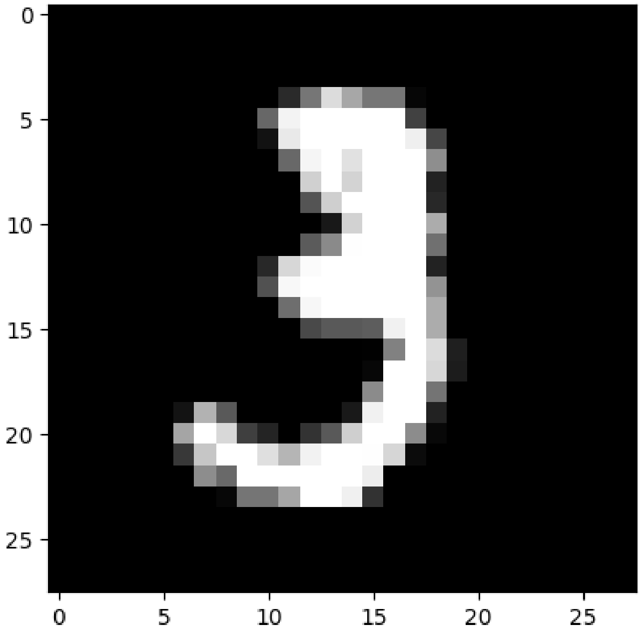
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
```




```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

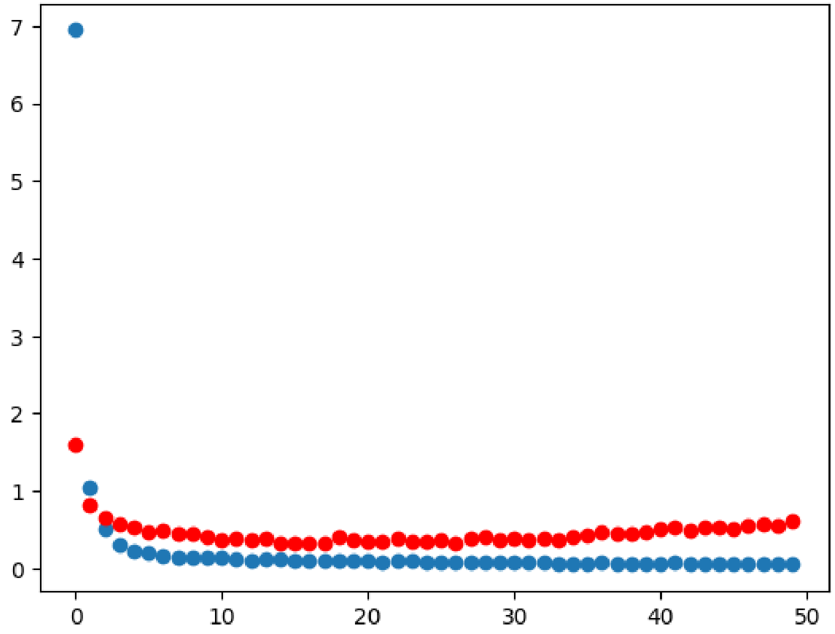
```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 128)               100480

 dense_13 (Dense)            (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

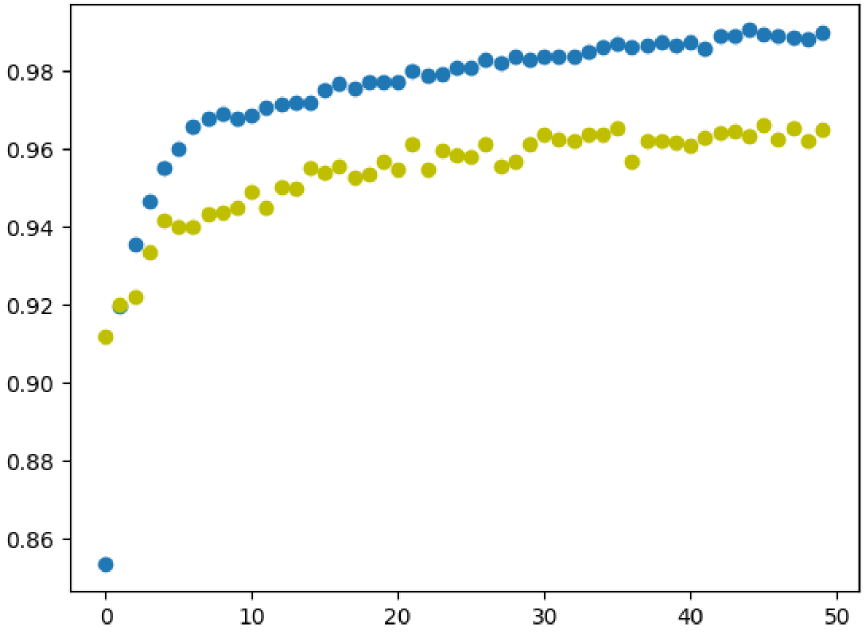
```
batch_size = 128
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
375/375 [==============================] - 1s 3ms/step - loss: 0.0754 - accuracy: 0.9859 - val_loss: 0.4677 - val_accuracy: 0.9566
Epoch 38/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0635 - accuracy: 0.9862 - val_loss: 0.4423 - val_accuracy: 0.9621
Epoch 39/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0612 - accuracy: 0.9871 - val_loss: 0.4498 - val_accuracy: 0.9621
Epoch 40/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0666 - accuracy: 0.9864 - val_loss: 0.4637 - val_accuracy: 0.9614
Epoch 41/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0615 - accuracy: 0.9873 - val_loss: 0.5013 - val_accuracy: 0.9607
Epoch 42/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0807 - accuracy: 0.9858 - val_loss: 0.5323 - val_accuracy: 0.9628
Epoch 43/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0568 - accuracy: 0.9890 - val_loss: 0.4959 - val_accuracy: 0.9638
Epoch 44/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0574 - accuracy: 0.9891 - val_loss: 0.5289 - val_accuracy: 0.9644
Epoch 45/50
375/375 [==============================] - 1s 4ms/step - loss: 0.0499 - accuracy: 0.9903 - val_loss: 0.5323 - val_accuracy: 0.9633
Epoch 46/50
375/375 [==============================] - 2s 4ms/step - loss: 0.0497 - accuracy: 0.9895 - val_loss: 0.5072 - val_accuracy: 0.9660
Epoch 47/50
375/375 [==============================] - 1s 4ms/step - loss: 0.0575 - accuracy: 0.9890 - val_loss: 0.5552 - val_accuracy: 0.9624
Epoch 48/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0625 - accuracy: 0.9885 - val_loss: 0.5624 - val_accuracy: 0.9653
Epoch 49/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0645 - accuracy: 0.9881 - val_loss: 0.5592 - val_accuracy: 0.9619
Epoch 50/50
375/375 [==============================] - 1s 3ms/step - loss: 0.0527 - accuracy: 0.9898 - val_loss: 0.6130 - val_accuracy: 0.9650
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

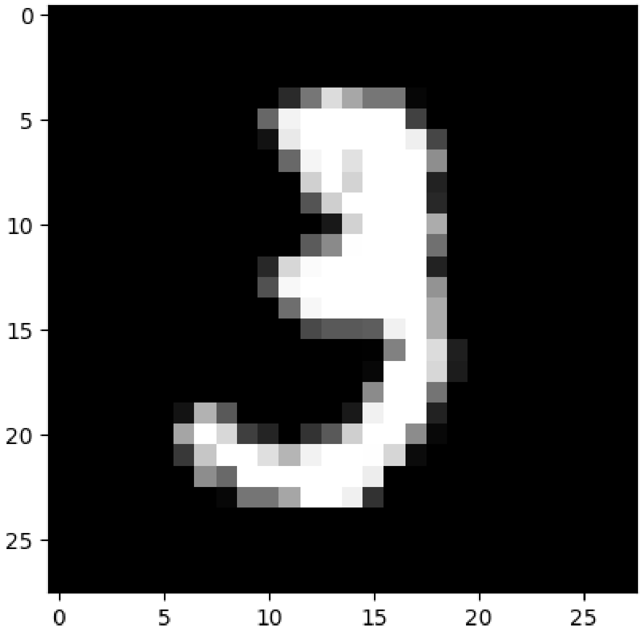



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 0.6223078370094299
Test accuracy: 0.9627000093460083
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7adb9caa28c0> triggered tf.funct
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 47ms/step
array([[0.0000000e+00, 1.6704997e-37, 0.0000000e+00, 1.0000000e+00,
        0.0000000e+00, 1.3962363e-19, 0.0000000e+00, 3.5204495e-34,
        1.7884261e-32, 4.6899234e-18]], dtype=float32)
```



Validation split 0.8

```
test_data = data[:56000]
train_data = data[56000:]
test_labels = label[:56000]
train_labels = label[56000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(56000, 28, 28) (14000, 28, 28) (56000,) (14000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

    ((14000, 28, 28), (14000, 10))

```
test_data.shape,test_labels.shape
```

    ((56000, 28, 28), (56000, 10))

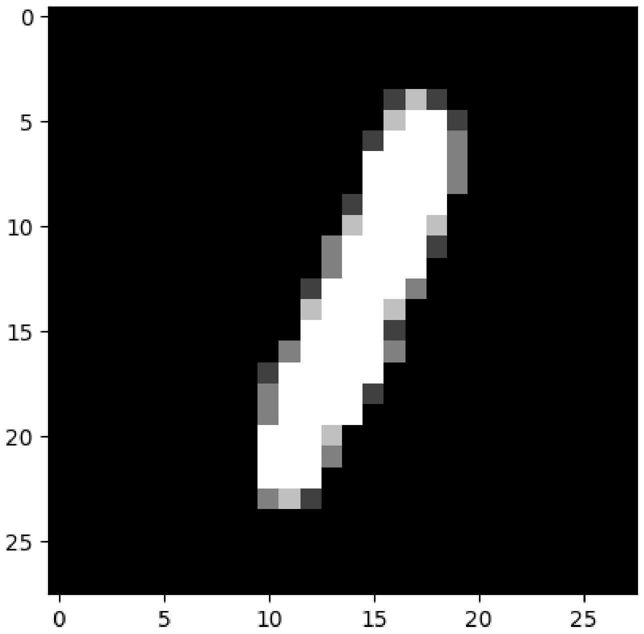
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

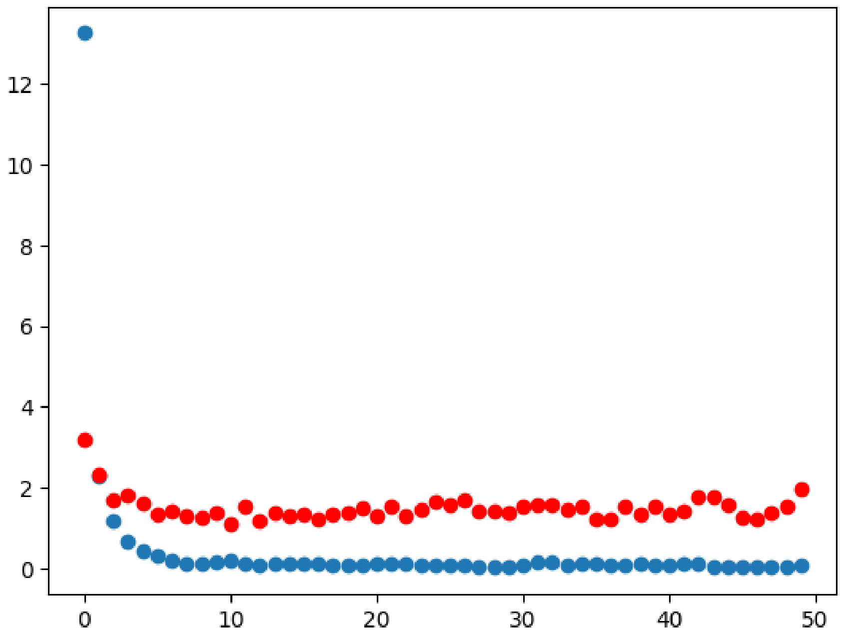
```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_14 (Dense)            (None, 128)               100480

 dense_15 (Dense)            (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
```

```
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

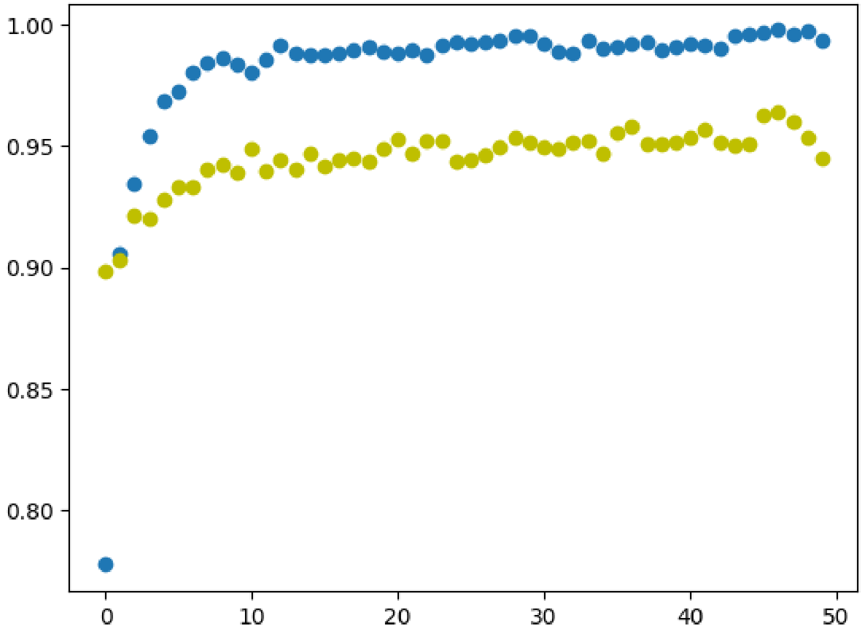
```
batch_size = 128
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
88/88 [==============================] - 0s 4ms/step - loss: 0.1001 - accuracy: 0.9915 - val_loss: 1.4050 - val_accuracy: 0.9568
Epoch 43/50
88/88 [==============================] - 0s 3ms/step - loss: 0.1233 - accuracy: 0.9902 - val_loss: 1.7838 - val_accuracy: 0.9518
Epoch 44/50
88/88 [==============================] - 0s 4ms/step - loss: 0.0422 - accuracy: 0.9956 - val_loss: 1.7605 - val_accuracy: 0.9504
Epoch 45/50
88/88 [==============================] - 0s 4ms/step - loss: 0.0420 - accuracy: 0.9962 - val_loss: 1.5653 - val_accuracy: 0.9511
Epoch 46/50
88/88 [==============================] - 0s 3ms/step - loss: 0.0344 - accuracy: 0.9971 - val_loss: 1.2519 - val_accuracy: 0.9629
Epoch 47/50
88/88 [==============================] - 0s 4ms/step - loss: 0.0186 - accuracy: 0.9979 - val_loss: 1.2166 - val_accuracy: 0.9643
Epoch 48/50
88/88 [==============================] - 0s 4ms/step - loss: 0.0347 - accuracy: 0.9965 - val_loss: 1.3804 - val_accuracy: 0.9604
Epoch 49/50
88/88 [==============================] - 0s 4ms/step - loss: 0.0324 - accuracy: 0.9977 - val_loss: 1.5495 - val_accuracy: 0.9536
Epoch 50/50
88/88 [==============================] - 0s 3ms/step - loss: 0.0748 - accuracy: 0.9938 - val_loss: 1.9561 - val_accuracy: 0.9454
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

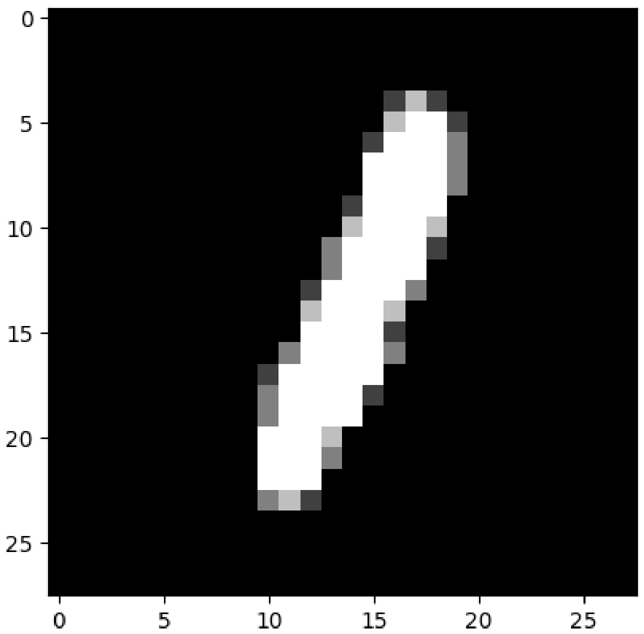



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
    Test loss: 2.763029098510742
    Test accuracy: 0.9341607093811035
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7adb9c8fb010> triggered tf.funct
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 43ms/step
array([[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        3.7452694e-34, 0.0000000e+00]], dtype=float32)
```



Validation split 0.7

```
test_data = data[:49000]
train_data = data[49000:]
test_labels = label[:49000]
train_labels = label[49000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(49000, 28, 28) (21000, 28, 28) (49000,) (21000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((21000, 28, 28), (21000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((49000, 28, 28), (49000, 10))
```

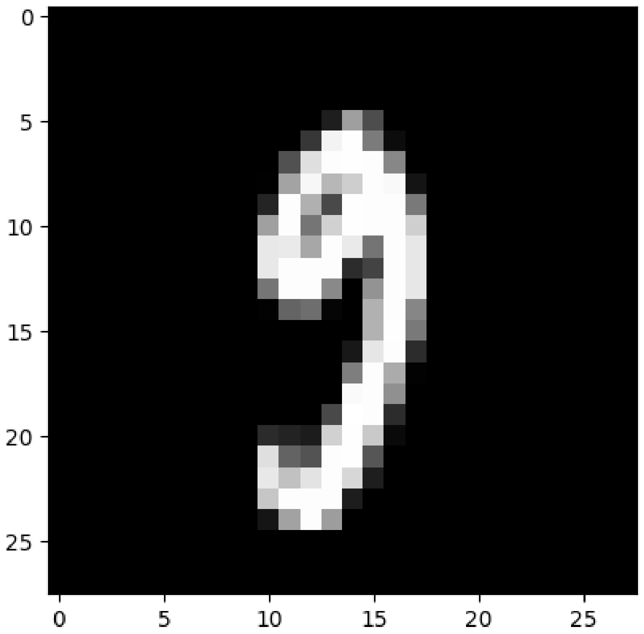
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```




```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_16 (Dense)            (None, 128)               100480

 dense_17 (Dense)            (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
```

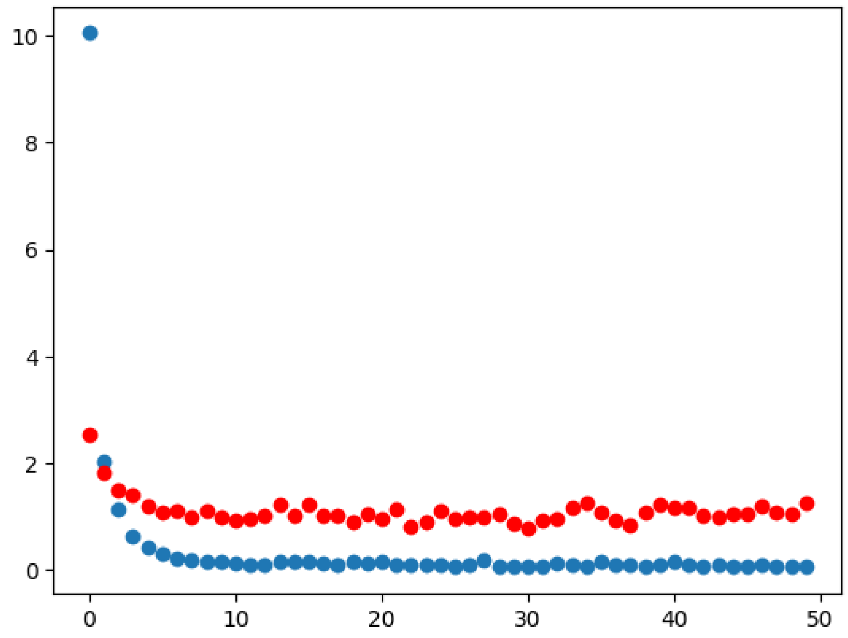
```
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
batch_size = 128
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

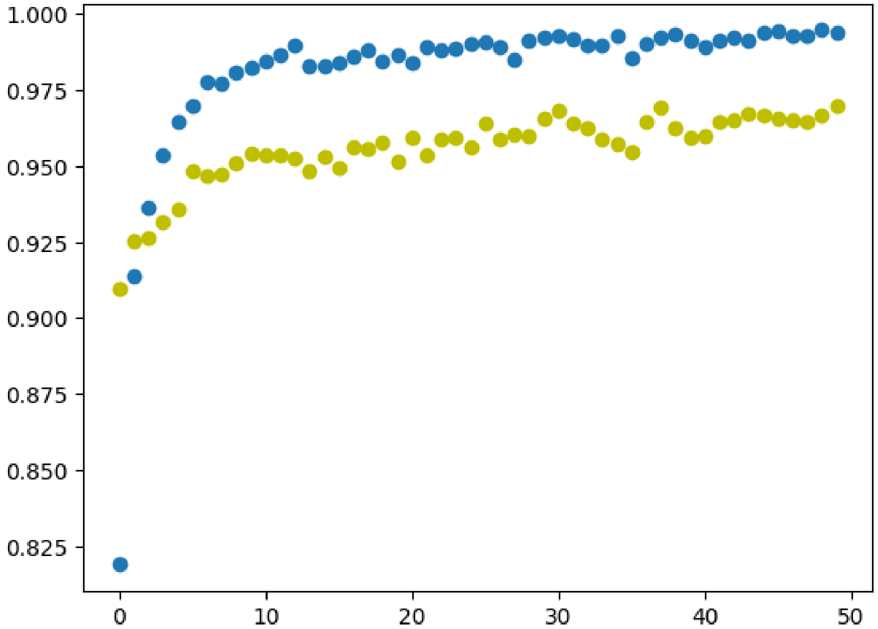
```
Epoch 1/50
132/132 [==============================] - 1s 5ms/step - loss: 10.0412 - accuracy: 0.8189 - val_loss: 2.5454 - val_accuracy: 0.9095
Epoch 2/50
132/132 [==============================] - 0s 3ms/step - loss: 2.0317 - accuracy: 0.9140 - val_loss: 1.8289 - val_accuracy: 0.9252
Epoch 3/50
132/132 [==============================] - 0s 4ms/step - loss: 1.1431 - accuracy: 0.9362 - val_loss: 1.4875 - val_accuracy: 0.9262
Epoch 4/50
132/132 [==============================] - 0s 3ms/step - loss: 0.6417 - accuracy: 0.9538 - val_loss: 1.4206 - val_accuracy: 0.9317
Epoch 5/50
132/132 [==============================] - 1s 4ms/step - loss: 0.4407 - accuracy: 0.9646 - val_loss: 1.1908 - val_accuracy: 0.9360
Epoch 6/50
132/132 [==============================] - 0s 4ms/step - loss: 0.3100 - accuracy: 0.9698 - val_loss: 1.0802 - val_accuracy: 0.9486
Epoch 7/50
132/132 [==============================] - 0s 3ms/step - loss: 0.2164 - accuracy: 0.9777 - val_loss: 1.1206 - val_accuracy: 0.9469
Epoch 8/50
132/132 [==============================] - 1s 4ms/step - loss: 0.1991 - accuracy: 0.9774 - val_loss: 1.0083 - val_accuracy: 0.9476
Epoch 9/50
132/132 [==============================] - 1s 5ms/step - loss: 0.1679 - accuracy: 0.9808 - val_loss: 1.1188 - val_accuracy: 0.9512
Epoch 10/50
132/132 [==============================] - 1s 5ms/step - loss: 0.1486 - accuracy: 0.9825 - val_loss: 0.9944 - val_accuracy: 0.9540
Epoch 11/50
132/132 [==============================] - 1s 5ms/step - loss: 0.1434 - accuracy: 0.9845 - val_loss: 0.9420 - val_accuracy: 0.9536
Epoch 12/50
132/132 [==============================] - 1s 5ms/step - loss: 0.1144 - accuracy: 0.9867 - val_loss: 0.9780 - val_accuracy: 0.9538
Epoch 13/50
132/132 [==============================] - 1s 4ms/step - loss: 0.0928 - accuracy: 0.9895 - val_loss: 1.0311 - val_accuracy: 0.9526
Epoch 14/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1635 - accuracy: 0.9830 - val_loss: 1.2272 - val_accuracy: 0.9483
Epoch 15/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1709 - accuracy: 0.9828 - val_loss: 1.0227 - val_accuracy: 0.9531
Epoch 16/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1674 - accuracy: 0.9842 - val_loss: 1.2257 - val_accuracy: 0.9495
Epoch 17/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1371 - accuracy: 0.9859 - val_loss: 1.0267 - val_accuracy: 0.9564
Epoch 18/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1115 - accuracy: 0.9885 - val_loss: 1.0099 - val_accuracy: 0.9557
Epoch 19/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1716 - accuracy: 0.9846 - val_loss: 0.9056 - val_accuracy: 0.9576
Epoch 20/50
132/132 [==============================] - 1s 4ms/step - loss: 0.1263 - accuracy: 0.9867 - val_loss: 1.0672 - val_accuracy: 0.9517
Epoch 21/50
```

```
132/132 [==============================] - 0s 3ms/step - loss: 0.1731 - accuracy: 0.9838 - val_loss: 0.9573 - val_accuracy: 0.9593
Epoch 22/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1011 - accuracy: 0.9892 - val_loss: 1.1302 - val_accuracy: 0.9538
Epoch 23/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1163 - accuracy: 0.9885 - val_loss: 0.8128 - val_accuracy: 0.9590
Epoch 24/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1120 - accuracy: 0.9887 - val_loss: 0.9017 - val_accuracy: 0.9593
Epoch 25/50
132/132 [==============================] - 0s 3ms/step - loss: 0.0929 - accuracy: 0.9904 - val_loss: 1.1108 - val_accuracy: 0.9564
Epoch 26/50
132/132 [==============================] - 0s 3ms/step - loss: 0.0824 - accuracy: 0.9908 - val_loss: 0.9721 - val_accuracy: 0.9643
Epoch 27/50
132/132 [==============================] - 0s 3ms/step - loss: 0.1005 - accuracy: 0.9895 - val_loss: 0.9801 - val_accuracy: 0.9588
Epoch 28/50
132/132 [==============================] - 1s 4ms/step - loss: 0.1847 - accuracy: 0.9852 - val_loss: 1.0015 - val_accuracy: 0.9602
Epoch 29/50
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

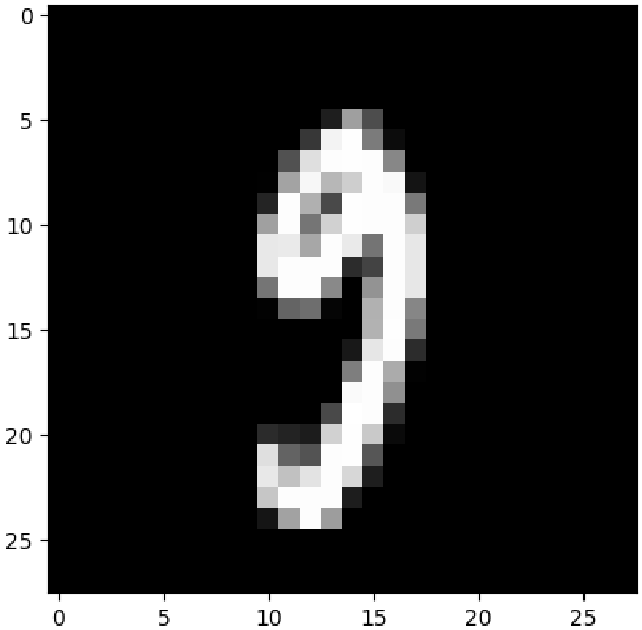


```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 2.086263656616211
Test accuracy: 0.9490816593170166
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
1/1 [==============================] - 0s 41ms/step
array([[1.6403934e-11, 1.1675589e-22, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 2.5161540e-26, 0.0000000e+00, 1.0482298e-17,
        0.0000000e+00, 1.0000000e+00]], dtype=float32)
```



Validation split 0.6

```
test_data = data[:42000]
train_data = data[42000:]
test_labels = label[:42000]
train_labels = label[42000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(42000, 28, 28) (28000, 28, 28) (42000,) (28000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((28000, 28, 28), (28000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((42000, 28, 28), (42000, 10))
```

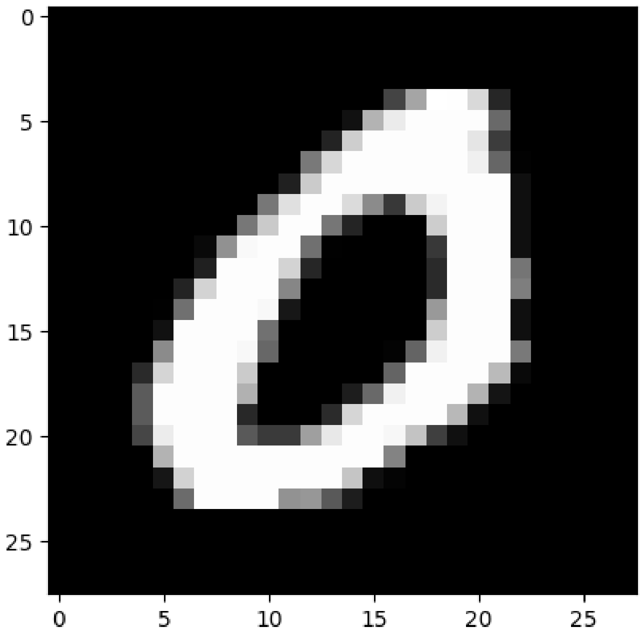
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

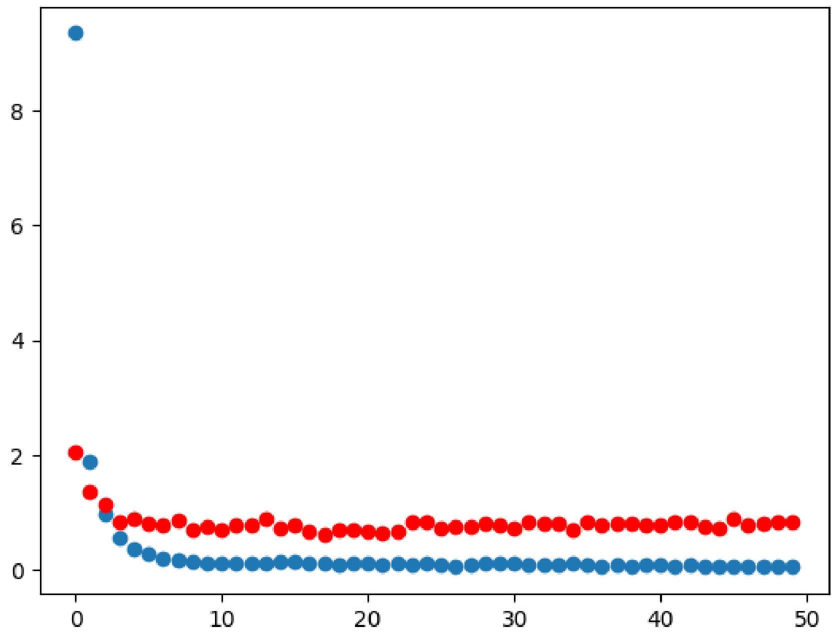
```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_18 (Dense)            (None, 128)               100480

 dense_19 (Dense)            (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
```

```
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

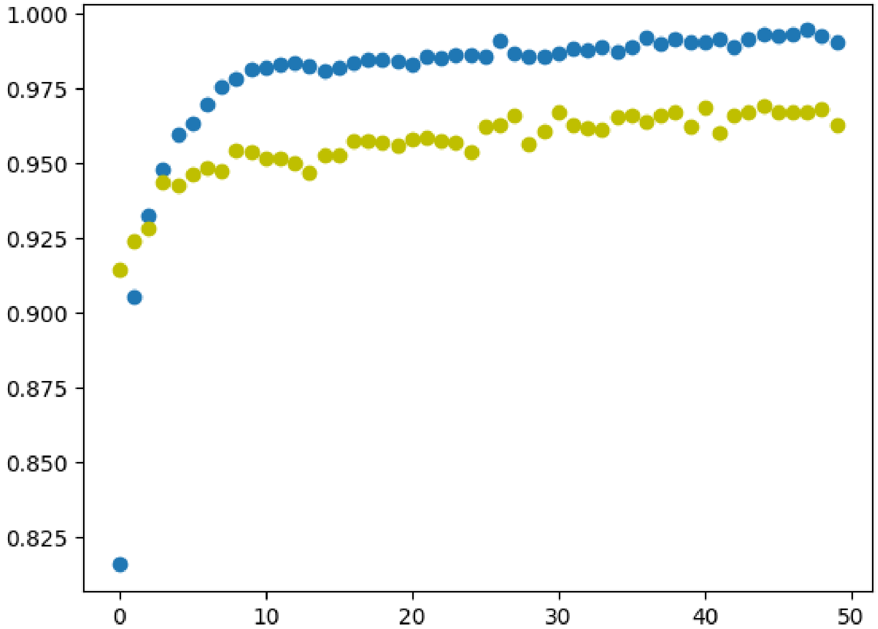
```
batch_size = 128
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
175/175 [==============================] - 1s 4ms/step - loss: 0.0681 - accuracy: 0.9915 - val_loss: 0.8282 - val_accuracy: 0.9600
Epoch 43/50
175/175 [==============================] - 1s 3ms/step - loss: 0.1006 - accuracy: 0.9889 - val_loss: 0.8409 - val_accuracy: 0.9659
Epoch 44/50
175/175 [==============================] - 1s 3ms/step - loss: 0.0745 - accuracy: 0.9914 - val_loss: 0.7500 - val_accuracy: 0.9670
Epoch 45/50
175/175 [==============================] - 1s 3ms/step - loss: 0.0502 - accuracy: 0.9930 - val_loss: 0.7226 - val_accuracy: 0.9689
Epoch 46/50
175/175 [==============================] - 1s 3ms/step - loss: 0.0634 - accuracy: 0.9927 - val_loss: 0.8891 - val_accuracy: 0.9668
Epoch 47/50
175/175 [==============================] - 1s 3ms/step - loss: 0.0687 - accuracy: 0.9930 - val_loss: 0.7870 - val_accuracy: 0.9671
Epoch 48/50
175/175 [==============================] - 1s 3ms/step - loss: 0.0533 - accuracy: 0.9943 - val_loss: 0.8233 - val_accuracy: 0.9670
Epoch 49/50
175/175 [==============================] - 1s 4ms/step - loss: 0.0745 - accuracy: 0.9925 - val_loss: 0.8407 - val_accuracy: 0.9679
Epoch 50/50
175/175 [==============================] - 1s 3ms/step - loss: 0.0759 - accuracy: 0.9905 - val_loss: 0.8435 - val_accuracy: 0.9627
```

```python
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```python
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

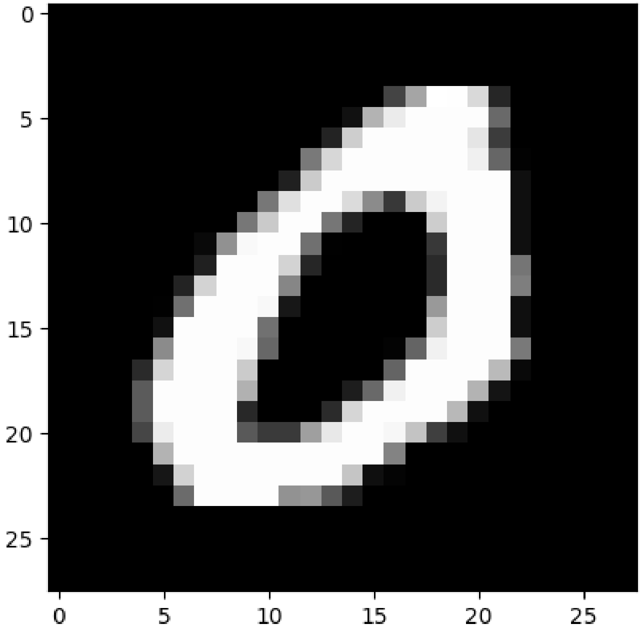



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 1.2243731021881104
Test accuracy: 0.9535238146781921
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 66ms/step
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```



Model no 1.

Validation split 0.85714285714 (default)

```
test_data = data[:60000]
train_data = data[60000:]
test_labels = label[:60000]
train_labels = label[60000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(60000, 28, 28) (10000, 28, 28) (60000,) (10000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

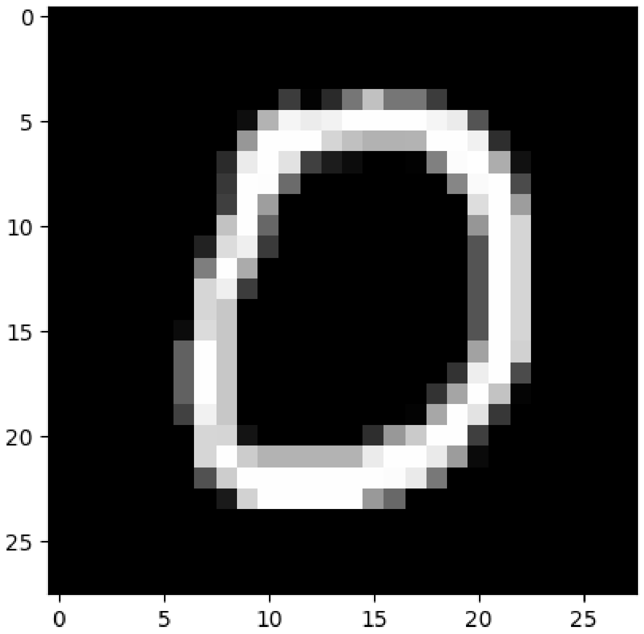
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 64, use_bias=True, activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_20 (Dense)            (None, 128)               100480    
                                                                 
 dense_21 (Dense)            (None, 64)                8256      
                                                                 
 dense_22 (Dense)            (None, 10)                650       
```

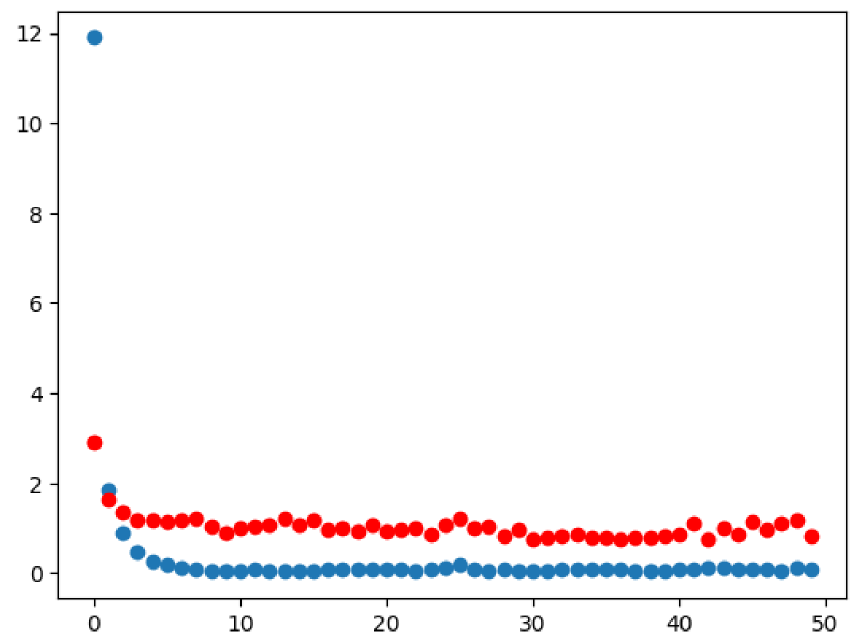
```
=================================================================
Total params: 109386 (427.29 KB)
Trainable params: 109386 (427.29 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
batch_size = 128
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

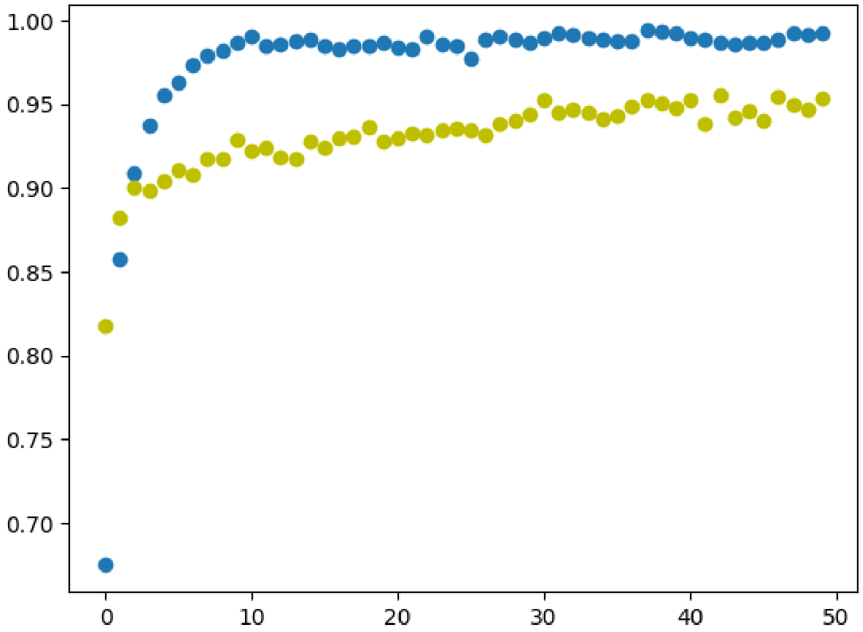
```
Epoch 1/50
63/63 [==============================] - 1s 6ms/step - loss: 11.8953 - accuracy: 0.6747 - val_loss: 2.9170 - val_accuracy: 0.8175
Epoch 2/50
63/63 [==============================] - 0s 4ms/step - loss: 1.8569 - accuracy: 0.8579 - val_loss: 1.6379 - val_accuracy: 0.8820
Epoch 3/50
63/63 [==============================] - 0s 4ms/step - loss: 0.9040 - accuracy: 0.9084 - val_loss: 1.3717 - val_accuracy: 0.9000
Epoch 4/50
63/63 [==============================] - 0s 4ms/step - loss: 0.4866 - accuracy: 0.9374 - val_loss: 1.1851 - val_accuracy: 0.8980
Epoch 5/50
63/63 [==============================] - 0s 4ms/step - loss: 0.2760 - accuracy: 0.9554 - val_loss: 1.1739 - val_accuracy: 0.9045
Epoch 6/50
63/63 [==============================] - 0s 4ms/step - loss: 0.2033 - accuracy: 0.9628 - val_loss: 1.1375 - val_accuracy: 0.9110
Epoch 7/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1263 - accuracy: 0.9737 - val_loss: 1.1748 - val_accuracy: 0.9075
Epoch 8/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0924 - accuracy: 0.9795 - val_loss: 1.2117 - val_accuracy: 0.9175
Epoch 9/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0671 - accuracy: 0.9816 - val_loss: 1.0324 - val_accuracy: 0.9175
Epoch 10/50
63/63 [==============================] - 0s 7ms/step - loss: 0.0548 - accuracy: 0.9865 - val_loss: 0.9073 - val_accuracy: 0.9290
Epoch 11/50
63/63 [==============================] - 1s 11ms/step - loss: 0.0353 - accuracy: 0.9906 - val_loss: 1.0076 - val_accuracy: 0.9220
Epoch 12/50
63/63 [==============================] - 1s 10ms/step - loss: 0.0790 - accuracy: 0.9846 - val_loss: 1.0269 - val_accuracy: 0.9240
Epoch 13/50
63/63 [==============================] - 0s 7ms/step - loss: 0.0646 - accuracy: 0.9854 - val_loss: 1.0565 - val_accuracy: 0.9185
Epoch 14/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0606 - accuracy: 0.9874 - val_loss: 1.2062 - val_accuracy: 0.9175
Epoch 15/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0451 - accuracy: 0.9883 - val_loss: 1.0591 - val_accuracy: 0.9275
Epoch 16/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0630 - accuracy: 0.9850 - val_loss: 1.1727 - val_accuracy: 0.9245
Epoch 17/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0941 - accuracy: 0.9826 - val_loss: 0.9767 - val_accuracy: 0.9295
Epoch 18/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0817 - accuracy: 0.9852 - val_loss: 0.9905 - val_accuracy: 0.9305
Epoch 19/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0803 - accuracy: 0.9847 - val_loss: 0.9236 - val_accuracy: 0.9365
```

```
Epoch 20/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0790 - accuracy: 0.9865 - val_loss: 1.0741 - val_accuracy: 0.9275
Epoch 21/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0878 - accuracy: 0.9841 - val_loss: 0.9493 - val_accuracy: 0.9300
Epoch 22/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0935 - accuracy: 0.9829 - val_loss: 0.9672 - val_accuracy: 0.9330
Epoch 23/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0538 - accuracy: 0.9902 - val_loss: 0.9885 - val_accuracy: 0.9320
Epoch 24/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0742 - accuracy: 0.9856 - val_loss: 0.8687 - val_accuracy: 0.9345
Epoch 25/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1153 - accuracy: 0.9846 - val_loss: 1.0897 - val_accuracy: 0.9350
Epoch 26/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1814 - accuracy: 0.9774 - val_loss: 1.2034 - val_accuracy: 0.9345
Epoch 27/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0681 - accuracy: 0.9885 - val_loss: 0.9909 - val_accuracy: 0.9320
Epoch 28/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0602 - accuracy: 0.9906 - val_loss: 1.0242 - val_accuracy: 0.9385
Epoch 29/50
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```



```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

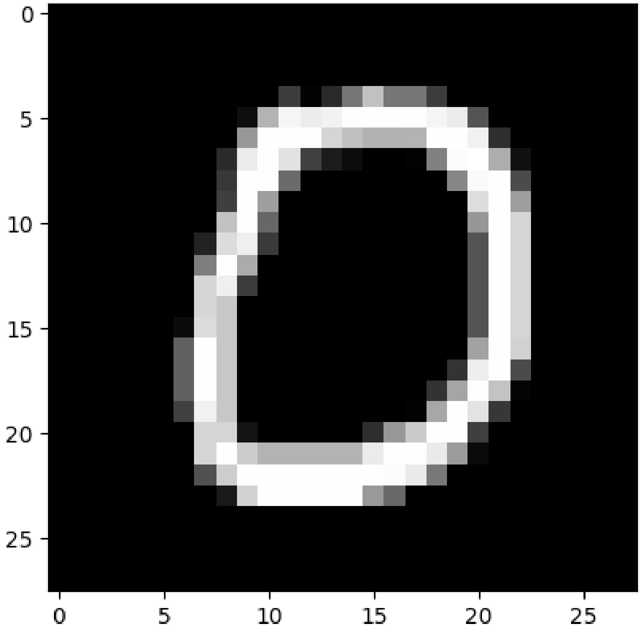



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 1.5609794855117798
Test accuracy: 0.9293166399002075
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 48ms/step
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```



Model no 2.

Validation split 0.85714285714 (default)

```
test_data = data[:60000]
train_data = data[60000:]
test_labels = label[:60000]
train_labels = label[60000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(60000, 28, 28) (10000, 28, 28) (60000,) (10000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

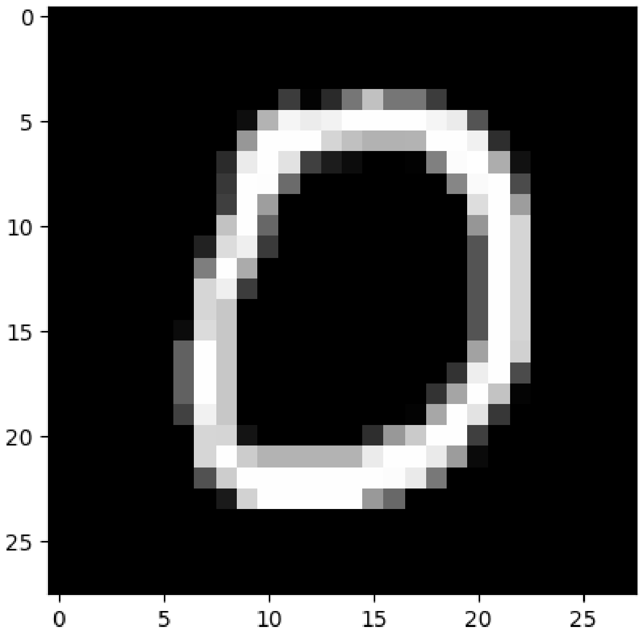
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 256, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 128, use_bias=True, activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_23 (Dense)            (None, 256)               200960

 dense_24 (Dense)            (None, 128)               32896

 dense_25 (Dense)            (None, 10)                1290
```

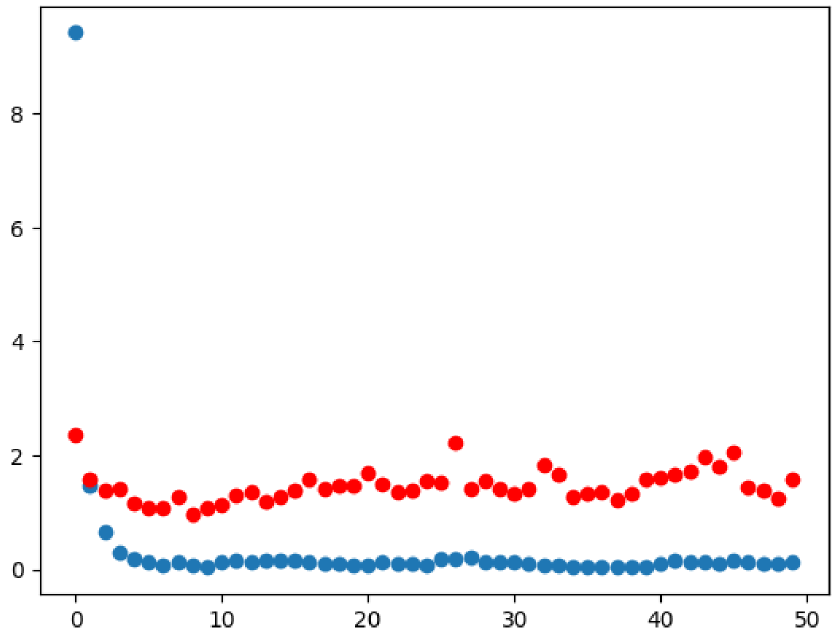
```
=================================================================
Total params: 235146 (918.54 KB)
Trainable params: 235146 (918.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
batch_size = 128
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

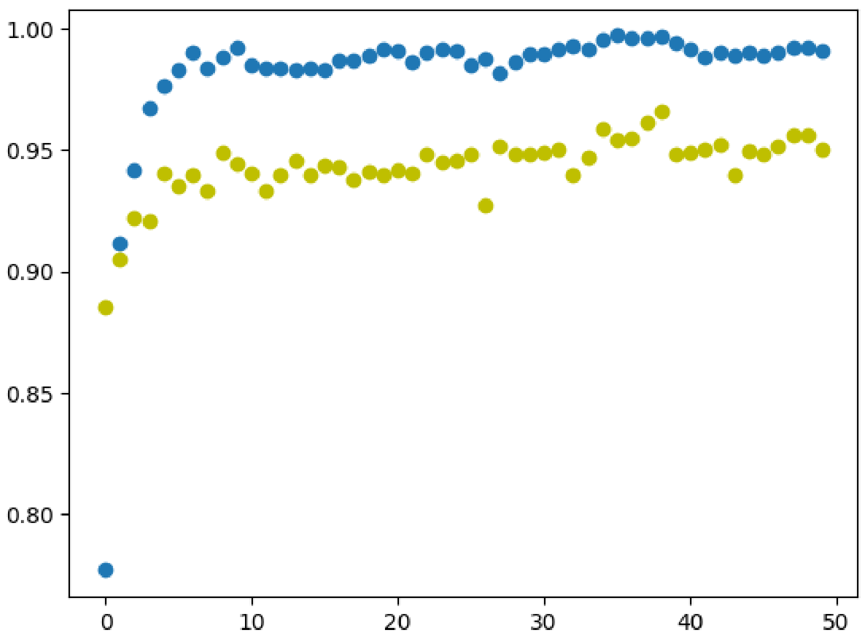
```
Epoch 1/50
63/63 [==============================] - 1s 7ms/step - loss: 9.4166 - accuracy: 0.7769 - val_loss: 2.3691 - val_accuracy: 0.8855
Epoch 2/50
63/63 [==============================] - 0s 4ms/step - loss: 1.4590 - accuracy: 0.9118 - val_loss: 1.5762 - val_accuracy: 0.9050
Epoch 3/50
63/63 [==============================] - 0s 4ms/step - loss: 0.6502 - accuracy: 0.9416 - val_loss: 1.3939 - val_accuracy: 0.9220
Epoch 4/50
63/63 [==============================] - 0s 4ms/step - loss: 0.2945 - accuracy: 0.9672 - val_loss: 1.4233 - val_accuracy: 0.9205
Epoch 5/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1869 - accuracy: 0.9766 - val_loss: 1.1635 - val_accuracy: 0.9405
Epoch 6/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1148 - accuracy: 0.9833 - val_loss: 1.0740 - val_accuracy: 0.9350
Epoch 7/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0660 - accuracy: 0.9900 - val_loss: 1.0671 - val_accuracy: 0.9400
Epoch 8/50
63/63 [==============================] - 0s 5ms/step - loss: 0.1323 - accuracy: 0.9840 - val_loss: 1.2620 - val_accuracy: 0.9335
Epoch 9/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0774 - accuracy: 0.9880 - val_loss: 0.9732 - val_accuracy: 0.9490
Epoch 10/50
63/63 [==============================] - 0s 7ms/step - loss: 0.0499 - accuracy: 0.9925 - val_loss: 1.0671 - val_accuracy: 0.9445
Epoch 11/50
63/63 [==============================] - 1s 12ms/step - loss: 0.1139 - accuracy: 0.9849 - val_loss: 1.1242 - val_accuracy: 0.9405
Epoch 12/50
63/63 [==============================] - 0s 5ms/step - loss: 0.1580 - accuracy: 0.9836 - val_loss: 1.3119 - val_accuracy: 0.9335
Epoch 13/50
63/63 [==============================] - 1s 17ms/step - loss: 0.1336 - accuracy: 0.9837 - val_loss: 1.3693 - val_accuracy: 0.9395
Epoch 14/50
63/63 [==============================] - 1s 12ms/step - loss: 0.1523 - accuracy: 0.9831 - val_loss: 1.1795 - val_accuracy: 0.9455
Epoch 15/50
63/63 [==============================] - 1s 10ms/step - loss: 0.1428 - accuracy: 0.9836 - val_loss: 1.2778 - val_accuracy: 0.9400
Epoch 16/50
63/63 [==============================] - 1s 8ms/step - loss: 0.1470 - accuracy: 0.9830 - val_loss: 1.3756 - val_accuracy: 0.9435
Epoch 17/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1167 - accuracy: 0.9868 - val_loss: 1.5726 - val_accuracy: 0.9430
Epoch 18/50
63/63 [==============================] - 0s 7ms/step - loss: 0.1048 - accuracy: 0.9868 - val_loss: 1.4242 - val_accuracy: 0.9375
Epoch 19/50
63/63 [==============================] - 0s 8ms/step - loss: 0.1114 - accuracy: 0.9886 - val_loss: 1.4597 - val_accuracy: 0.9410
```

```
Epoch 20/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0605 - accuracy: 0.9916 - val_loss: 1.4744 - val_accuracy: 0.9400
Epoch 21/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0616 - accuracy: 0.9909 - val_loss: 1.7014 - val_accuracy: 0.9420
Epoch 22/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1158 - accuracy: 0.9865 - val_loss: 1.4936 - val_accuracy: 0.9405
Epoch 23/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1072 - accuracy: 0.9900 - val_loss: 1.3613 - val_accuracy: 0.9485
Epoch 24/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0954 - accuracy: 0.9912 - val_loss: 1.3705 - val_accuracy: 0.9450
Epoch 25/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0740 - accuracy: 0.9906 - val_loss: 1.5630 - val_accuracy: 0.9455
Epoch 26/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1761 - accuracy: 0.9849 - val_loss: 1.5273 - val_accuracy: 0.9480
Epoch 27/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1914 - accuracy: 0.9874 - val_loss: 2.2212 - val_accuracy: 0.9275
Epoch 28/50
63/63 [==============================] - 0s 4ms/step - loss: 0.2209 - accuracy: 0.9815 - val_loss: 1.4095 - val_accuracy: 0.9515
Epoch 29/50
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```



```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

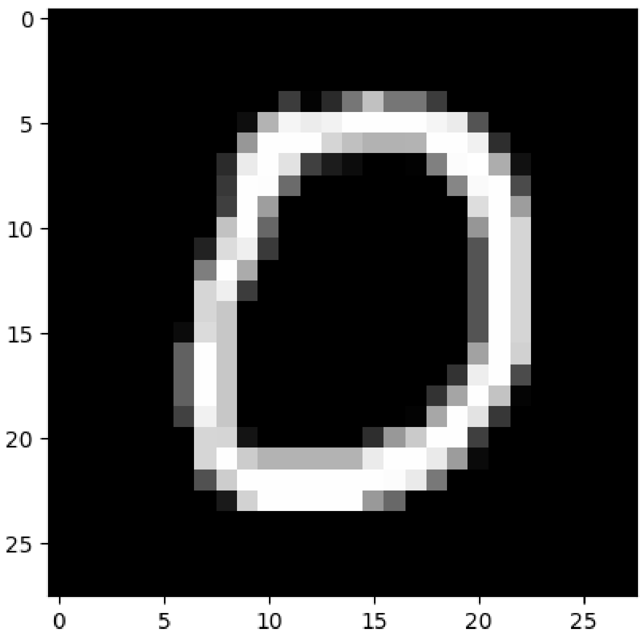



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 2.420621871948242
Test accuracy: 0.9312333464622498
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 51ms/step
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```



Model no 3.

Validation split 0.85714285714 (default)

```
test_data = data[:60000]
train_data = data[60000:]
test_labels = label[:60000]
train_labels = label[60000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(60000, 28, 28) (10000, 28, 28) (60000,) (10000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

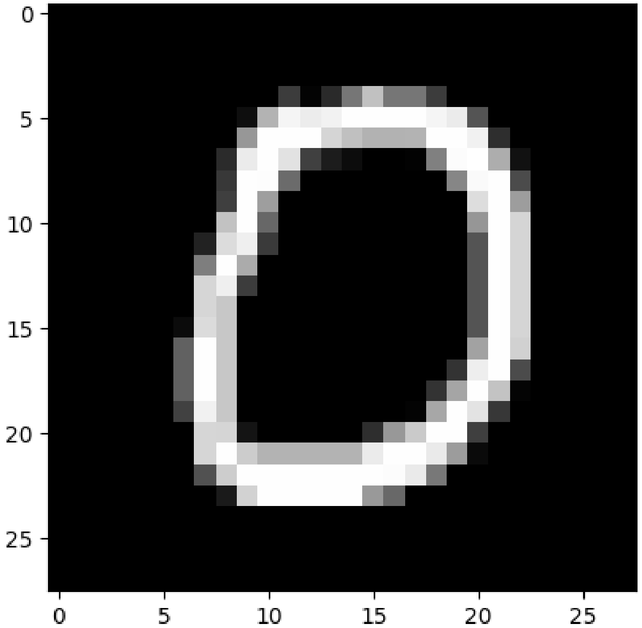
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```




```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 64, use_bias=True, activation = "relu"))
model.add(Dense(units = 64, use_bias=True, activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_26 (Dense)            (None, 128)               100480

 dense_27 (Dense)            (None, 64)                8256
```

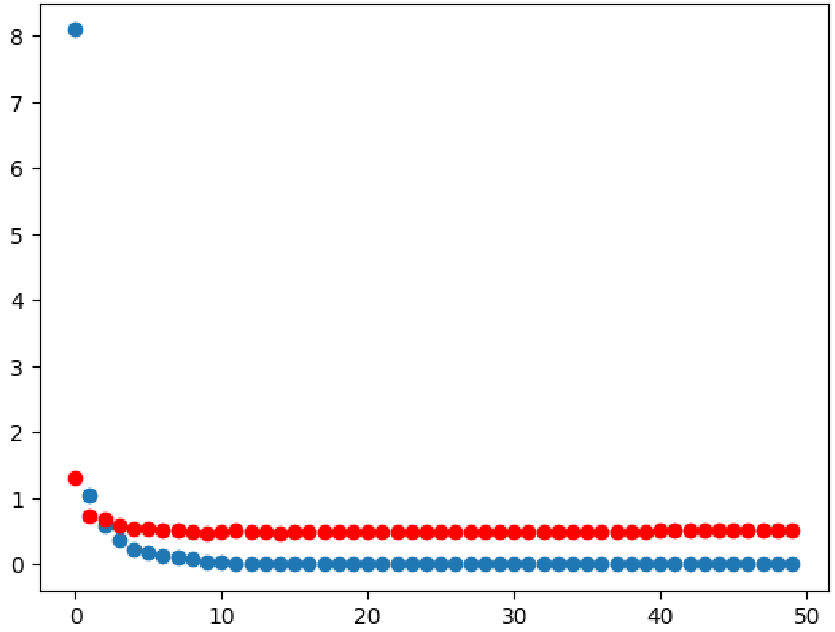
```
 dense_28 (Dense)            (None, 64)                4160

 dense_29 (Dense)            (None, 10)                650

=================================================================
Total params: 113546 (443.54 KB)
Trainable params: 113546 (443.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
batch_size = 128
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

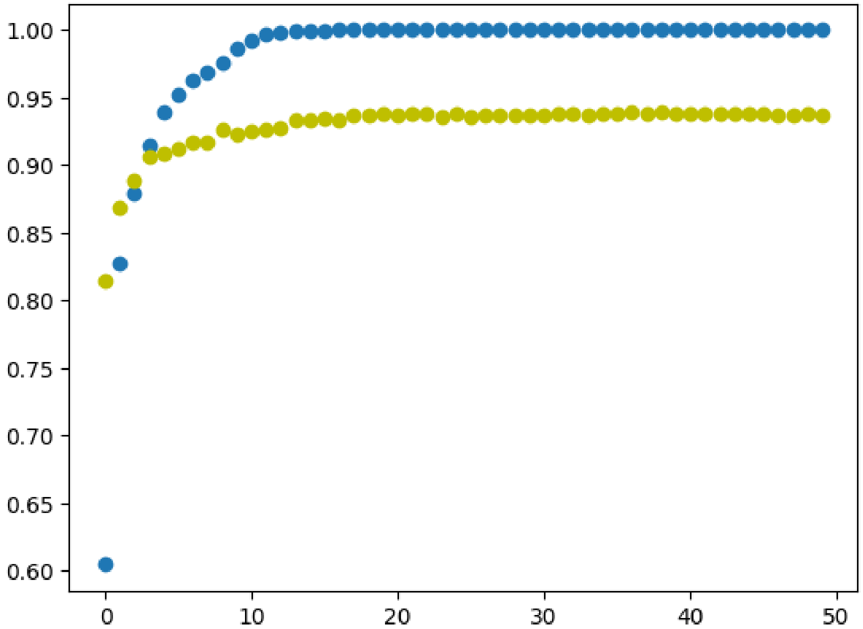
```
Epoch 1/50
63/63 [==============================] - 1s 6ms/step - loss: 8.0931 - accuracy: 0.6046 - val_loss: 1.3226 - val_accuracy: 0.8145
Epoch 2/50
63/63 [==============================] - 0s 4ms/step - loss: 1.0409 - accuracy: 0.8278 - val_loss: 0.7454 - val_accuracy: 0.8690
Epoch 3/50
63/63 [==============================] - 0s 4ms/step - loss: 0.5915 - accuracy: 0.8789 - val_loss: 0.6775 - val_accuracy: 0.8890
Epoch 4/50
63/63 [==============================] - 0s 4ms/step - loss: 0.3824 - accuracy: 0.9153 - val_loss: 0.5871 - val_accuracy: 0.9065
Epoch 5/50
63/63 [==============================] - 0s 4ms/step - loss: 0.2347 - accuracy: 0.9394 - val_loss: 0.5421 - val_accuracy: 0.9085
Epoch 6/50
63/63 [==============================] - 0s 4ms/step - loss: 0.1717 - accuracy: 0.9529 - val_loss: 0.5447 - val_accuracy: 0.9120
Epoch 7/50
63/63 [==============================] - 0s 5ms/step - loss: 0.1296 - accuracy: 0.9624 - val_loss: 0.5195 - val_accuracy: 0.9170
Epoch 8/50
63/63 [==============================] - 0s 7ms/step - loss: 0.0972 - accuracy: 0.9691 - val_loss: 0.5188 - val_accuracy: 0.9165
Epoch 9/50
63/63 [==============================] - 0s 7ms/step - loss: 0.0769 - accuracy: 0.9762 - val_loss: 0.4844 - val_accuracy: 0.9270
Epoch 10/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0441 - accuracy: 0.9865 - val_loss: 0.4749 - val_accuracy: 0.9230
Epoch 11/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0311 - accuracy: 0.9920 - val_loss: 0.4920 - val_accuracy: 0.9250
Epoch 12/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0171 - accuracy: 0.9969 - val_loss: 0.5203 - val_accuracy: 0.9260
Epoch 13/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0129 - accuracy: 0.9979 - val_loss: 0.4887 - val_accuracy: 0.9275
Epoch 14/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0087 - accuracy: 0.9989 - val_loss: 0.4954 - val_accuracy: 0.9330
Epoch 15/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0049 - accuracy: 0.9998 - val_loss: 0.4814 - val_accuracy: 0.9335
Epoch 16/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0039 - accuracy: 0.9998 - val_loss: 0.4900 - val_accuracy: 0.9345
Epoch 17/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0028 - accuracy: 1.0000 - val_loss: 0.4940 - val_accuracy: 0.9340
```

```
Epoch 18/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.4885 - val_accuracy: 0.9375
Epoch 19/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.4908 - val_accuracy: 0.9365
Epoch 20/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.4900 - val_accuracy: 0.9385
Epoch 21/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.4904 - val_accuracy: 0.9375
Epoch 22/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.4911 - val_accuracy: 0.9385
Epoch 23/50
63/63 [==============================] - 0s 7ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.4918 - val_accuracy: 0.9380
Epoch 24/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.4940 - val_accuracy: 0.9360
Epoch 25/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.4972 - val_accuracy: 0.9380
Epoch 26/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.4993 - val_accuracy: 0.9360
Epoch 27/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.5015 - val_accuracy: 0.9370
Epoch 28/50
63/63 [==============================] - 0s 5ms/step - loss: 9.8011e-04 - accuracy: 1.0000 - val_loss: 0.5007 - val_accuracy: 0.9375
Epoch 29/50
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

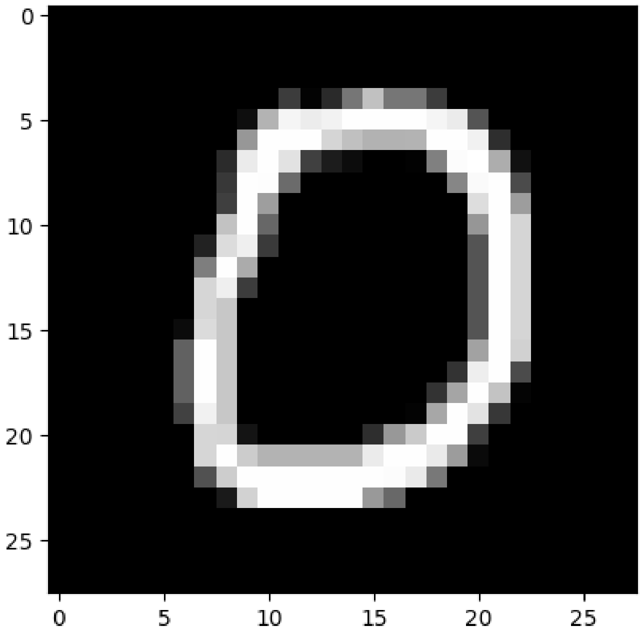



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 0.7605254054069519
Test accuracy: 0.9075000286102295
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 94ms/step
array([[1.0000000e+00, 3.4098838e-35, 2.2874737e-26, 5.6333921e-27,
        5.0748078e-26, 1.8851508e-20, 4.0799978e-32, 1.7487098e-33,
        1.2505732e-23, 3.8149031e-33]], dtype=float32)
```



Model no 4.

Validation split 0.85714285714 (default)

```
test_data = data[:60000]
train_data = data[60000:]
test_labels = label[:60000]
train_labels = label[60000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(60000, 28, 28) (10000, 28, 28) (60000,) (10000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

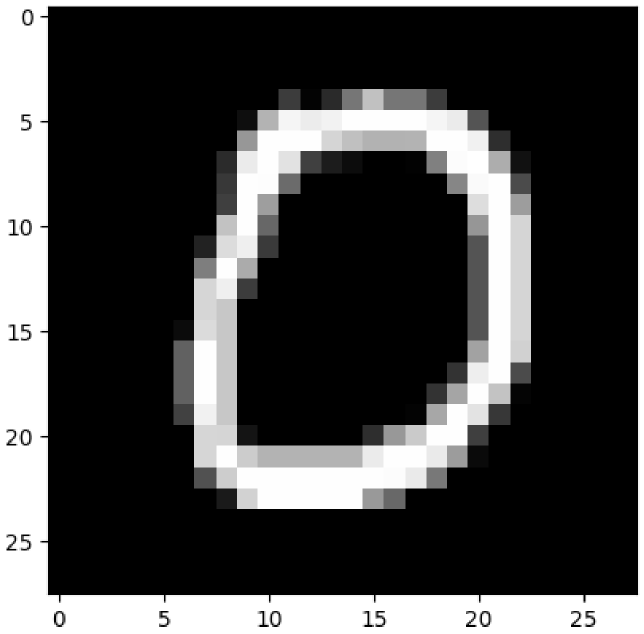
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 256, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 128, use_bias=True, activation = "relu"))
model.add(Dense(units = 64, use_bias=True, activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

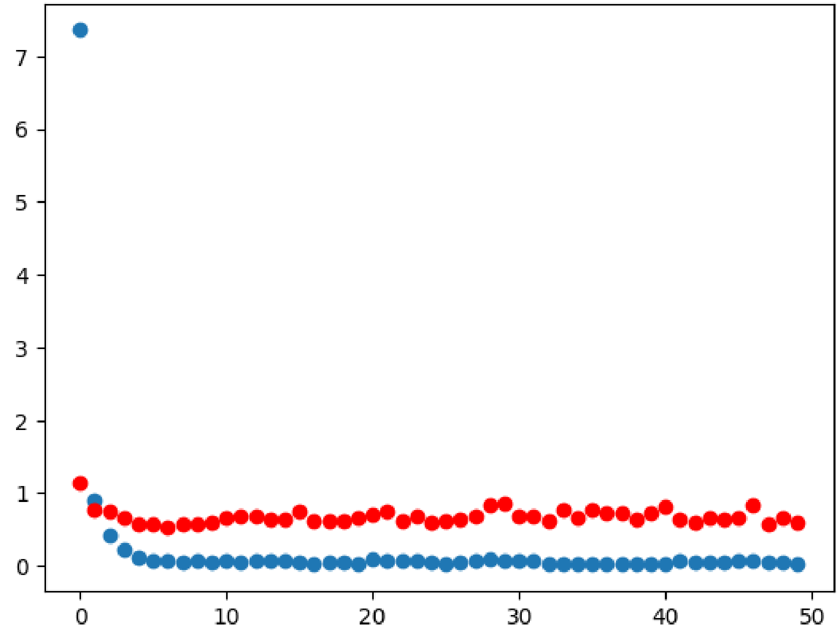
```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_30 (Dense)            (None, 256)               200960

 dense_31 (Dense)            (None, 128)               32896
```

```
 dense_32 (Dense)            (None, 64)                8256

 dense_33 (Dense)            (None, 10)                650

=================================================================
Total params: 242762 (948.29 KB)
Trainable params: 242762 (948.29 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

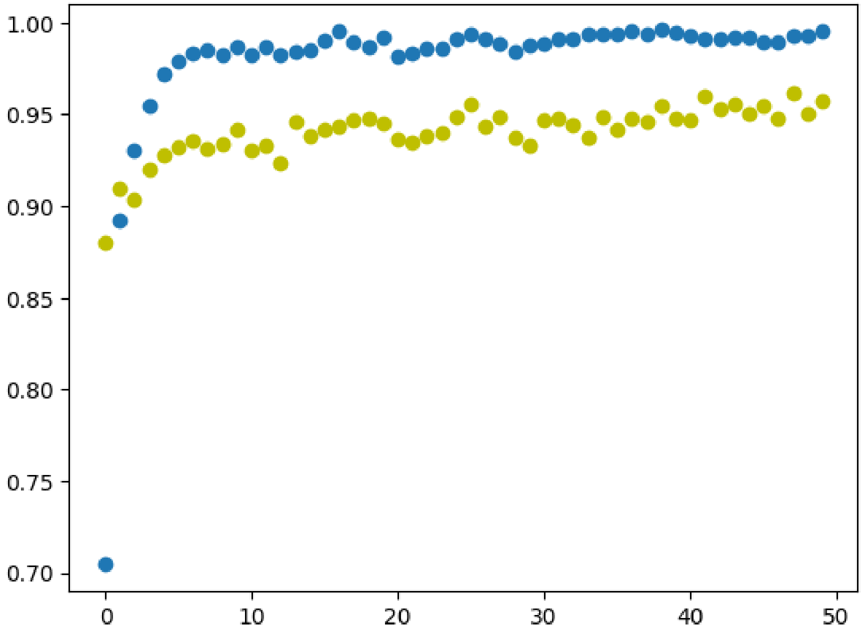
```
batch_size = 128
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
Epoch 39/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0220 - accuracy: 0.9959 - val_loss: 0.6469 - val_accuracy: 0.9545
Epoch 40/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0298 - accuracy: 0.9944 - val_loss: 0.7367 - val_accuracy: 0.9475
Epoch 41/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0351 - accuracy: 0.9933 - val_loss: 0.8223 - val_accuracy: 0.9465
Epoch 42/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0688 - accuracy: 0.9908 - val_loss: 0.6381 - val_accuracy: 0.9595
Epoch 43/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0510 - accuracy: 0.9915 - val_loss: 0.5978 - val_accuracy: 0.9530
Epoch 44/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0580 - accuracy: 0.9916 - val_loss: 0.6545 - val_accuracy: 0.9555
Epoch 45/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0481 - accuracy: 0.9920 - val_loss: 0.6436 - val_accuracy: 0.9500
Epoch 46/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0711 - accuracy: 0.9894 - val_loss: 0.6655 - val_accuracy: 0.9550
Epoch 47/50
63/63 [==============================] - 0s 4ms/step - loss: 0.0724 - accuracy: 0.9898 - val_loss: 0.8346 - val_accuracy: 0.9475
Epoch 48/50
63/63 [==============================] - 0s 5ms/step - loss: 0.0393 - accuracy: 0.9933 - val_loss: 0.5842 - val_accuracy: 0.9620
Epoch 49/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0422 - accuracy: 0.9927 - val_loss: 0.6548 - val_accuracy: 0.9505
Epoch 50/50
63/63 [==============================] - 0s 6ms/step - loss: 0.0203 - accuracy: 0.9958 - val_loss: 0.5909 - val_accuracy: 0.9575
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

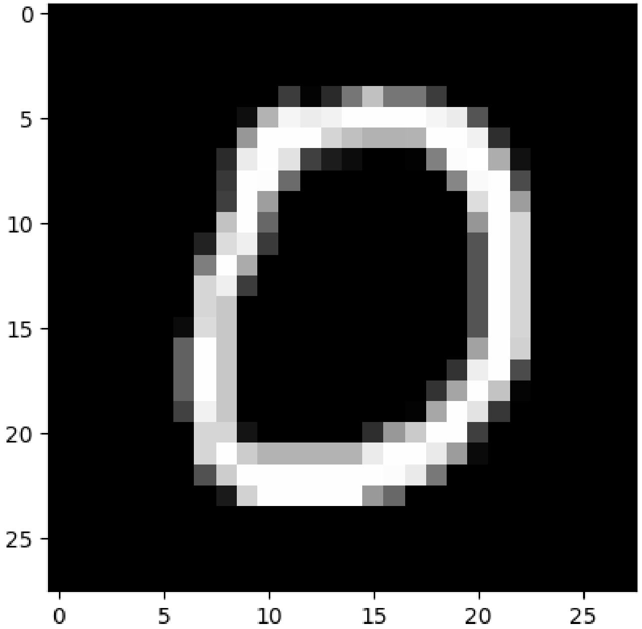



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 0.8955525755882263
Test accuracy: 0.9413833618164062
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 59ms/step
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```



Epochs 75

Validation split 0.85714285714 (default)

```
test_data = data[:60000]
train_data = data[60000:]
test_labels = label[:60000]
train_labels = label[60000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(60000, 28, 28) (10000, 28, 28) (60000,) (10000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

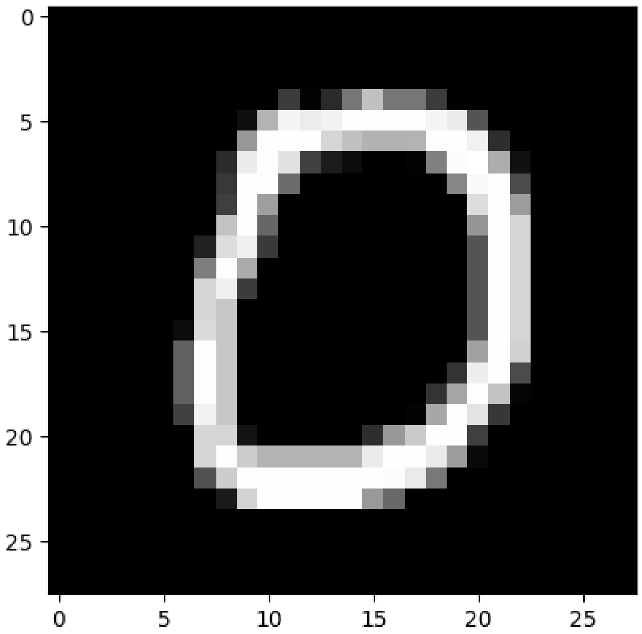
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

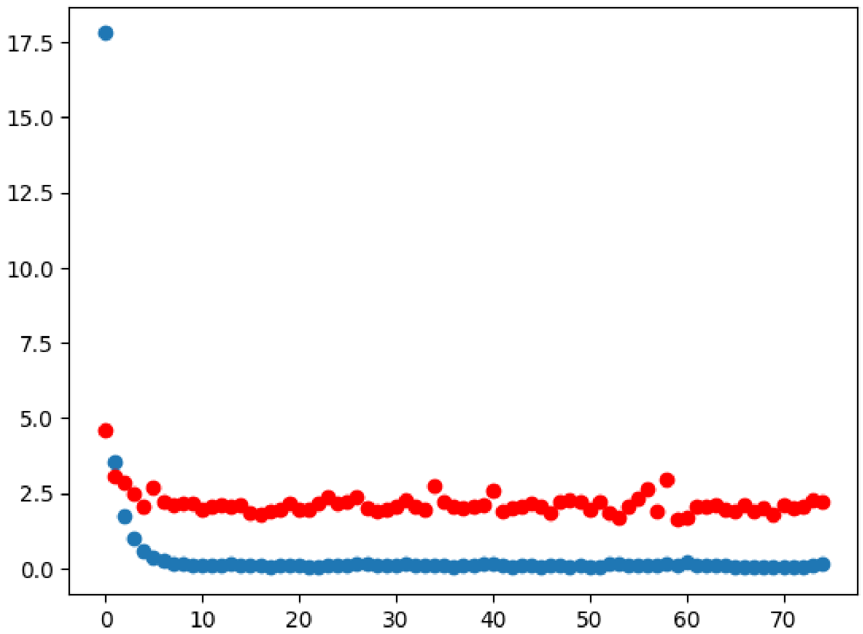
```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_34 (Dense)            (None, 128)               100480

 dense_35 (Dense)            (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
```

```
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

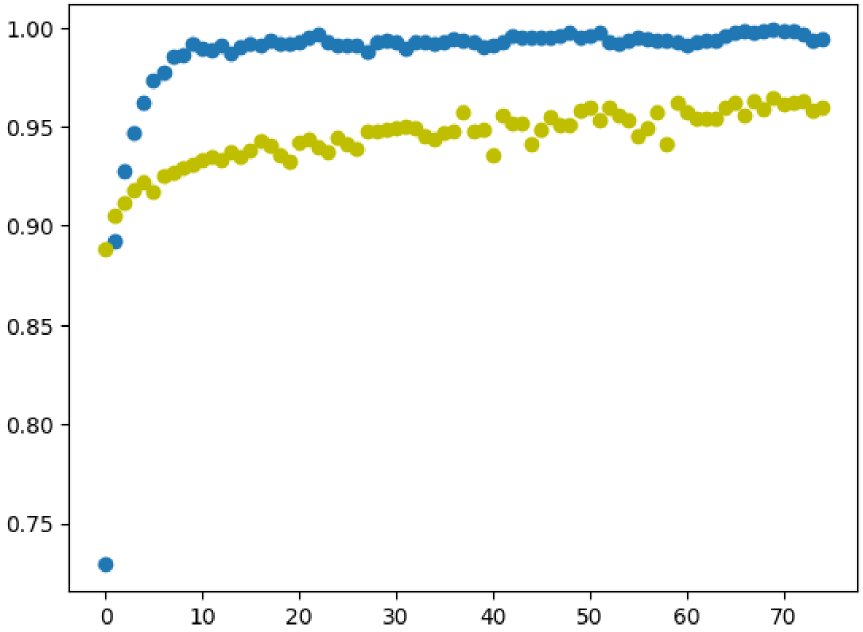
```
batch_size = 128
epochs = 75

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
63/63 [==============================] - 0s 4ms/step - loss: 0.0147 - accuracy: 0.9981 - val_loss: 2.0917 - val_accuracy: 0.9560
Epoch 68/75
63/63 [==============================] - 0s 4ms/step - loss: 0.0338 - accuracy: 0.9973 - val_loss: 1.9060 - val_accuracy: 0.9630
Epoch 69/75
63/63 [==============================] - 0s 4ms/step - loss: 0.0228 - accuracy: 0.9985 - val_loss: 1.9935 - val_accuracy: 0.9590
Epoch 70/75
63/63 [==============================] - 0s 5ms/step - loss: 0.0197 - accuracy: 0.9986 - val_loss: 1.8097 - val_accuracy: 0.9645
Epoch 71/75
63/63 [==============================] - 0s 4ms/step - loss: 0.0206 - accuracy: 0.9983 - val_loss: 2.1103 - val_accuracy: 0.9615
Epoch 72/75
63/63 [==============================] - 0s 4ms/step - loss: 0.0228 - accuracy: 0.9983 - val_loss: 1.9877 - val_accuracy: 0.9620
Epoch 73/75
63/63 [==============================] - 0s 4ms/step - loss: 0.0493 - accuracy: 0.9969 - val_loss: 2.0495 - val_accuracy: 0.9630
Epoch 74/75
63/63 [==============================] - 0s 4ms/step - loss: 0.1163 - accuracy: 0.9934 - val_loss: 2.2720 - val_accuracy: 0.9585
Epoch 75/75
63/63 [==============================] - 0s 4ms/step - loss: 0.1277 - accuracy: 0.9942 - val_loss: 2.2304 - val_accuracy: 0.9600
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

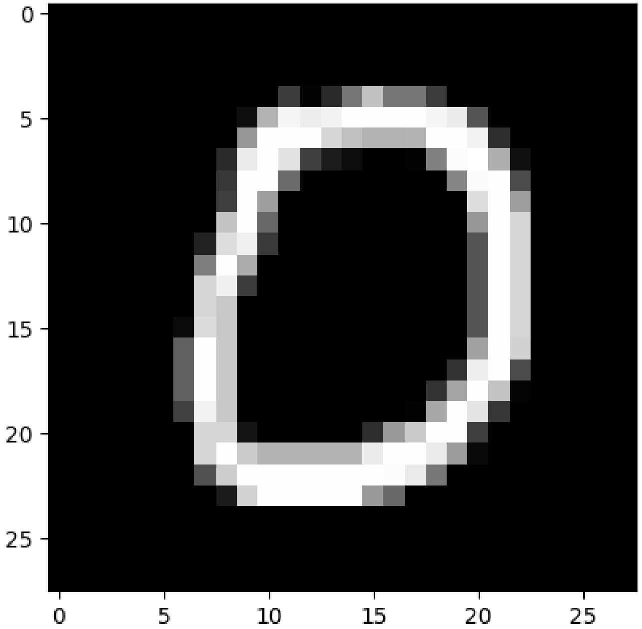



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 3.6567094326019287
Test accuracy: 0.9387500286102295
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 42ms/step
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```



Epochs 100

Validation split 0.85714285714 (default)

```
test_data = data[:60000]
train_data = data[60000:]
test_labels = label[:60000]
train_labels = label[60000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(60000, 28, 28) (10000, 28, 28) (60000,) (10000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

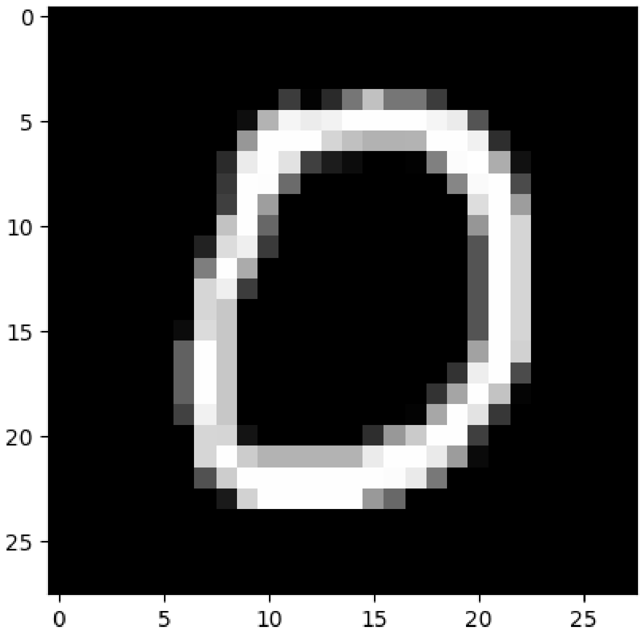
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

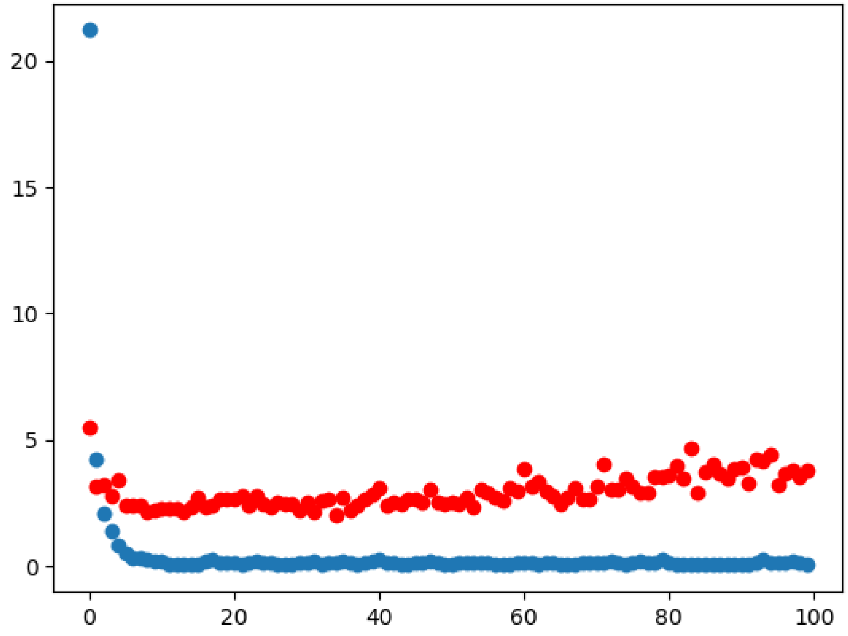
```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_36 (Dense)            (None, 128)               100480

 dense_37 (Dense)            (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
```

```
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

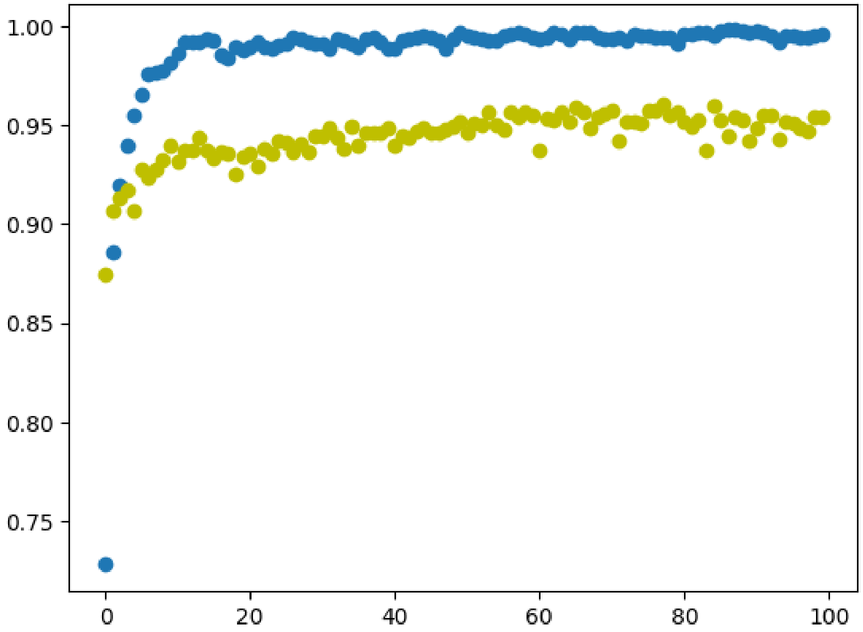
```
batch_size = 128
epochs = 100

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
63/63 [==============================] - 0s 4ms/step - loss: 0.0806 - accuracy: 0.9969 - val_loss: 3.2400 - val_accuracy: 0.9550
Epoch 93/100
63/63 [==============================] - 0s 4ms/step - loss: 0.1044 - accuracy: 0.9956 - val_loss: 4.2229 - val_accuracy: 0.9550
Epoch 94/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2306 - accuracy: 0.9921 - val_loss: 4.1469 - val_accuracy: 0.9430
Epoch 95/100
63/63 [==============================] - 0s 6ms/step - loss: 0.1412 - accuracy: 0.9950 - val_loss: 4.3874 - val_accuracy: 0.9520
Epoch 96/100
63/63 [==============================] - 0s 6ms/step - loss: 0.1199 - accuracy: 0.9955 - val_loss: 3.1825 - val_accuracy: 0.9510
Epoch 97/100
63/63 [==============================] - 0s 6ms/step - loss: 0.1339 - accuracy: 0.9946 - val_loss: 3.6308 - val_accuracy: 0.9485
Epoch 98/100
63/63 [==============================] - 0s 6ms/step - loss: 0.1523 - accuracy: 0.9946 - val_loss: 3.7904 - val_accuracy: 0.9470
Epoch 99/100
63/63 [==============================] - 0s 6ms/step - loss: 0.0930 - accuracy: 0.9954 - val_loss: 3.5162 - val_accuracy: 0.9545
Epoch 100/100
63/63 [==============================] - 0s 6ms/step - loss: 0.0773 - accuracy: 0.9964 - val_loss: 3.7648 - val_accuracy: 0.9545
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

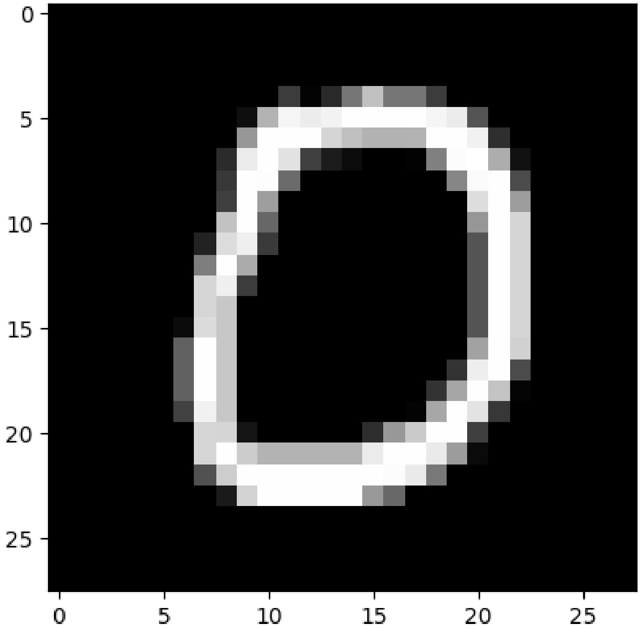



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 5.24637508392334
Test accuracy: 0.941266655921936
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 48ms/step
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```



Batch_size 256

Validation split 0.85714285714 (default)

```
test_data = data[:60000]
train_data = data[60000:]
test_labels = label[:60000]
train_labels = label[60000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(60000, 28, 28) (10000, 28, 28) (60000,) (10000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

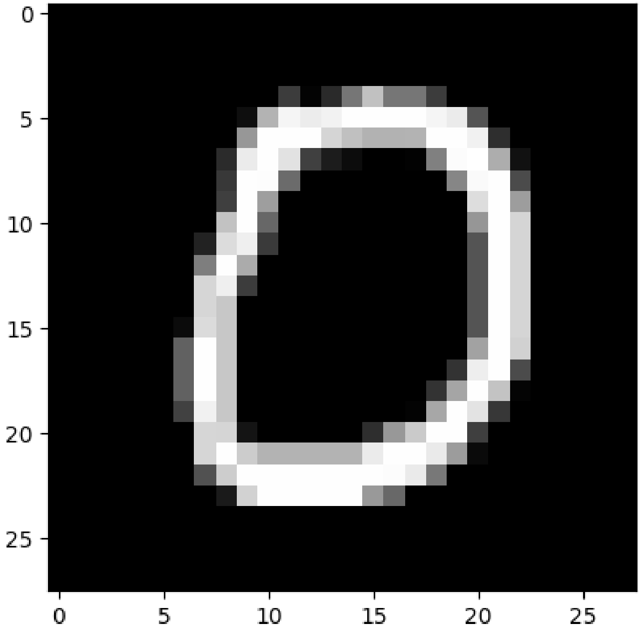
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```




```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

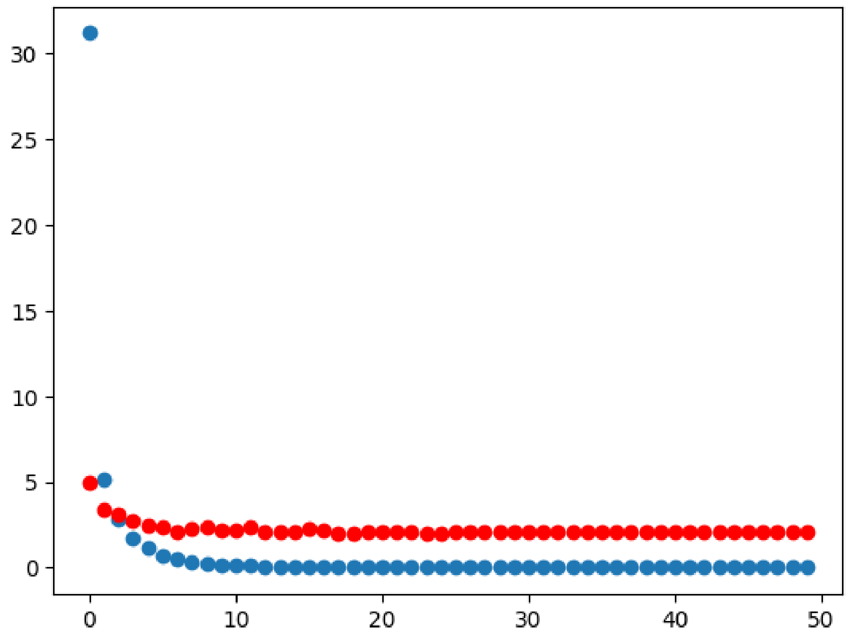
```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_38 (Dense)            (None, 128)               100480

 dense_39 (Dense)            (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
```

```
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

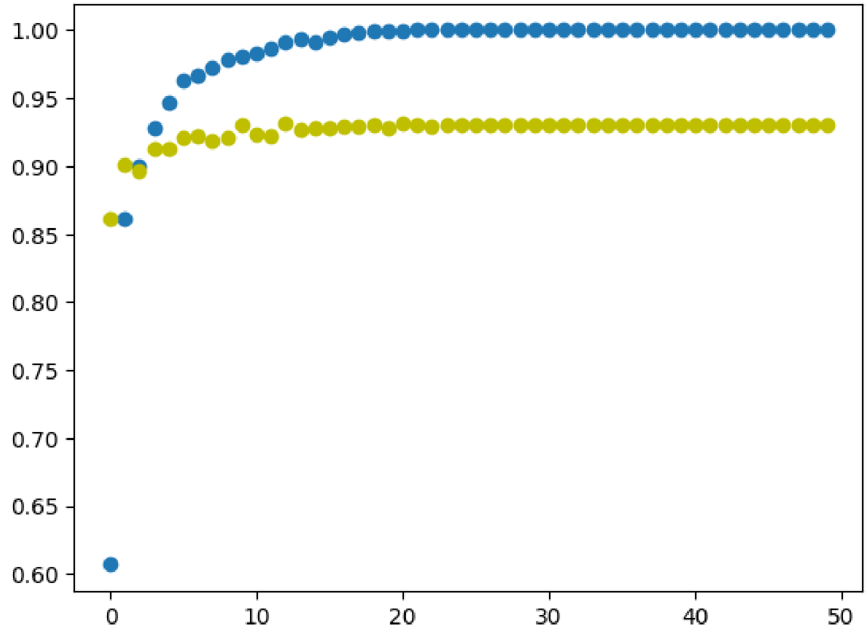
```
batch_size = 256
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
32/32 [==============================] - 0s 4ms/step - loss: 3.5822e-06 - accuracy: 1.0000 - val_loss: 2.0481 - val_accuracy: 0.9305
Epoch 43/50
32/32 [==============================] - 0s 4ms/step - loss: 3.4313e-06 - accuracy: 1.0000 - val_loss: 2.0483 - val_accuracy: 0.9305
Epoch 44/50
32/32 [==============================] - 0s 4ms/step - loss: 3.3147e-06 - accuracy: 1.0000 - val_loss: 2.0485 - val_accuracy: 0.9305
Epoch 45/50
32/32 [==============================] - 0s 5ms/step - loss: 3.2087e-06 - accuracy: 1.0000 - val_loss: 2.0486 - val_accuracy: 0.9305
Epoch 46/50
32/32 [==============================] - 0s 4ms/step - loss: 3.1195e-06 - accuracy: 1.0000 - val_loss: 2.0487 - val_accuracy: 0.9305
Epoch 47/50
32/32 [==============================] - 0s 4ms/step - loss: 3.0197e-06 - accuracy: 1.0000 - val_loss: 2.0489 - val_accuracy: 0.9305
Epoch 48/50
32/32 [==============================] - 0s 5ms/step - loss: 2.9416e-06 - accuracy: 1.0000 - val_loss: 2.0491 - val_accuracy: 0.9305
Epoch 49/50
32/32 [==============================] - 0s 4ms/step - loss: 2.8533e-06 - accuracy: 1.0000 - val_loss: 2.0493 - val_accuracy: 0.9305
Epoch 50/50
32/32 [==============================] - 0s 4ms/step - loss: 2.7804e-06 - accuracy: 1.0000 - val_loss: 2.0495 - val_accuracy: 0.9305
```

```python
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```python
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

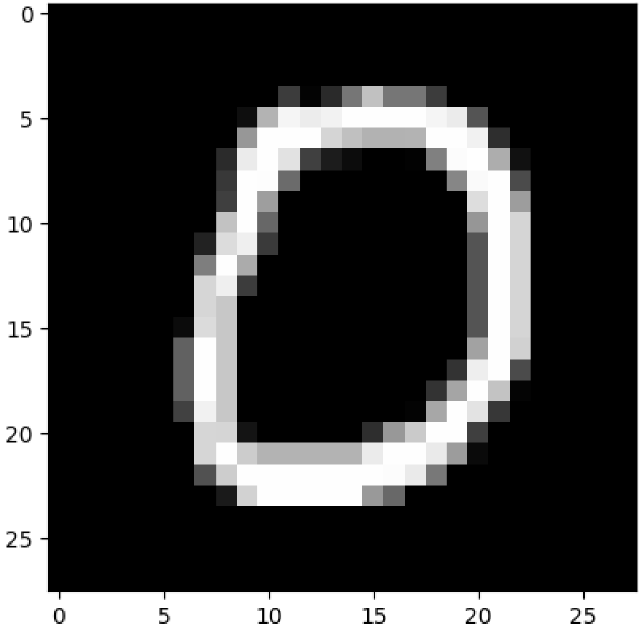



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 3.025714397430420
Test accuracy: 0.909500002861023
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 48ms/step
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```



Number of epochs - 64

Validation split 0.85714285714 (default)

```
test_data = data[:60000]
train_data = data[60000:]
test_labels = label[:60000]
train_labels = label[60000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(60000, 28, 28) (10000, 28, 28) (60000,) (10000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

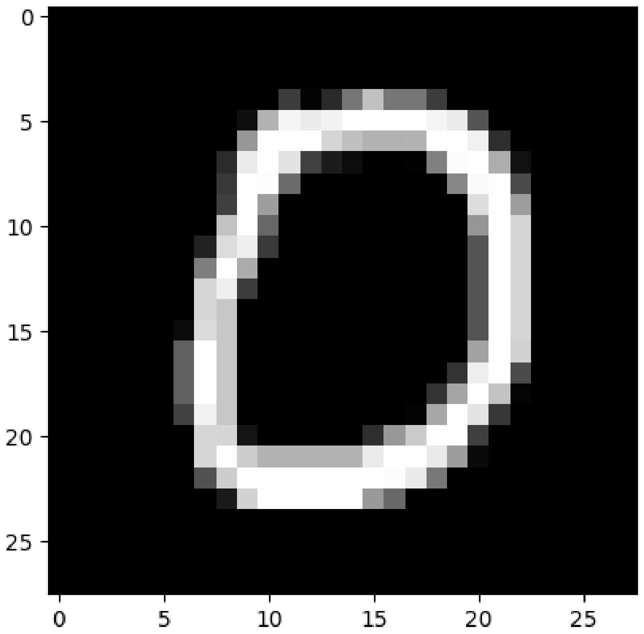
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))
```

```
model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

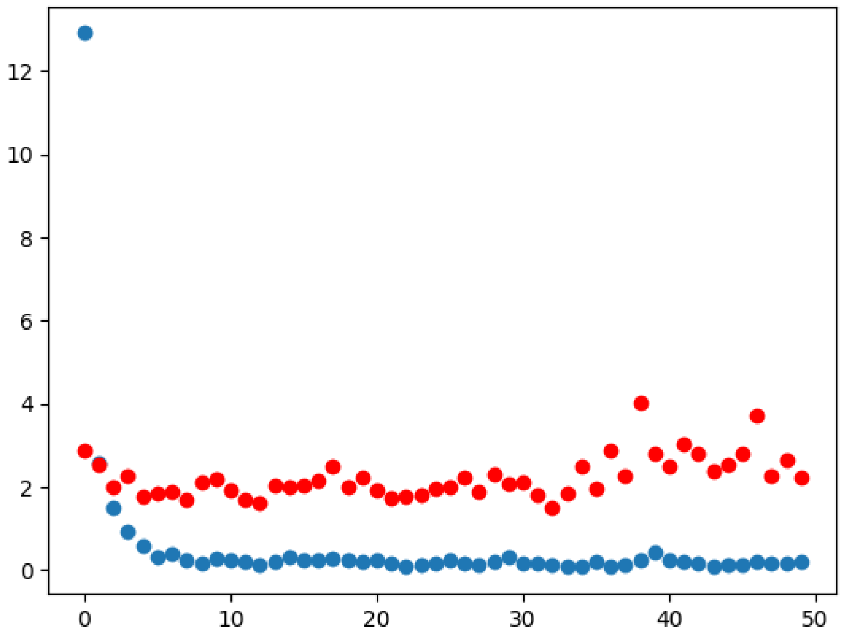
```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_40 (Dense)            (None, 128)               100480    
                                                                 
 dense_41 (Dense)            (None, 10)                1290      
                                                                 
=================================================================
Total params: 101770 (397.54 KB)
```

```
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

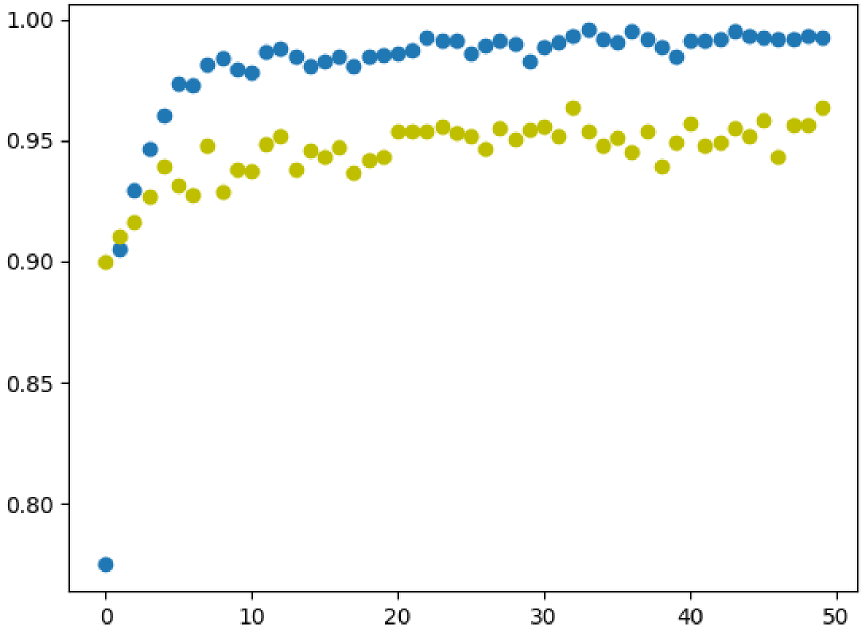
```
batch_size = 64
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
125/125 [==============================] - 0s 4ms/step - loss: 0.2081 - accuracy: 0.9911 - val_loss: 3.0274 - val_accuracy: 0.9480
Epoch 43/50
125/125 [==============================] - 1s 4ms/step - loss: 0.1456 - accuracy: 0.9921 - val_loss: 2.8146 - val_accuracy: 0.9490
Epoch 44/50
125/125 [==============================] - 1s 4ms/step - loss: 0.0942 - accuracy: 0.9952 - val_loss: 2.3573 - val_accuracy: 0.9550
Epoch 45/50
125/125 [==============================] - 0s 3ms/step - loss: 0.1132 - accuracy: 0.9930 - val_loss: 2.5478 - val_accuracy: 0.9515
Epoch 46/50
125/125 [==============================] - 0s 4ms/step - loss: 0.1270 - accuracy: 0.9925 - val_loss: 2.7792 - val_accuracy: 0.9580
Epoch 47/50
125/125 [==============================] - 0s 4ms/step - loss: 0.2129 - accuracy: 0.9918 - val_loss: 3.6977 - val_accuracy: 0.9435
Epoch 48/50
125/125 [==============================] - 0s 4ms/step - loss: 0.1684 - accuracy: 0.9921 - val_loss: 2.2614 - val_accuracy: 0.9565
Epoch 49/50
125/125 [==============================] - 0s 4ms/step - loss: 0.1477 - accuracy: 0.9933 - val_loss: 2.6413 - val_accuracy: 0.9565
Epoch 50/50
125/125 [==============================] - 0s 4ms/step - loss: 0.1825 - accuracy: 0.9925 - val_loss: 2.2346 - val_accuracy: 0.9635
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```




```
plt.scatter(np.arange(epochs),h.history['accuracy'])
plt.scatter(np.arange(epochs),h.history['val_accuracy'],c='y')
plt.show()
```

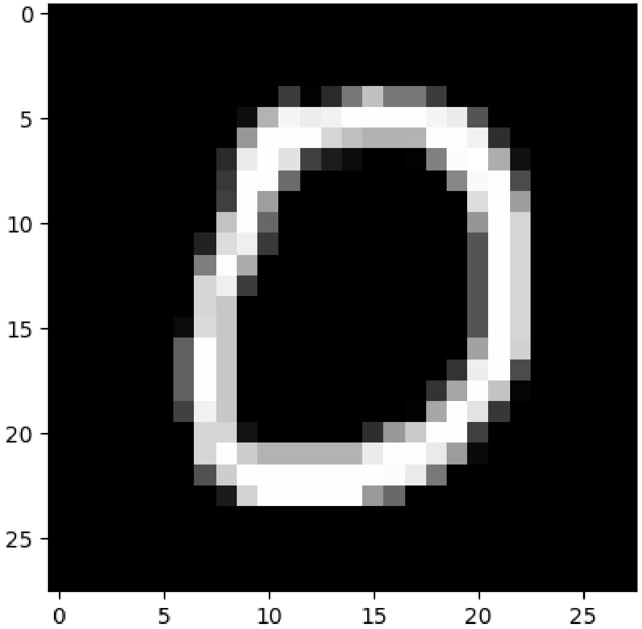



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 3.8052914142608643
Test accuracy: 0.9400500059127808
```

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)

picture = train_data[img_index].reshape(-1,784)

model.predict(picture)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 69ms/step
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```



Learning rate 0.0007

Validation split 0.85714285714 (default)

```
test_data = data[:60000]
train_data = data[60000:]
test_labels = label[:60000]
train_labels = label[60000:]
```

```
print(test_data.shape,train_data.shape,test_labels.shape,train_labels.shape)
```

```
(60000, 28, 28) (10000, 28, 28) (60000,) (10000,)
```

One-hot coding

```
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

```
train_data.shape,train_labels.shape
```

```
((10000, 28, 28), (10000, 10))
```

```
test_data.shape,test_labels.shape
```

```
((60000, 28, 28), (60000, 10))
```

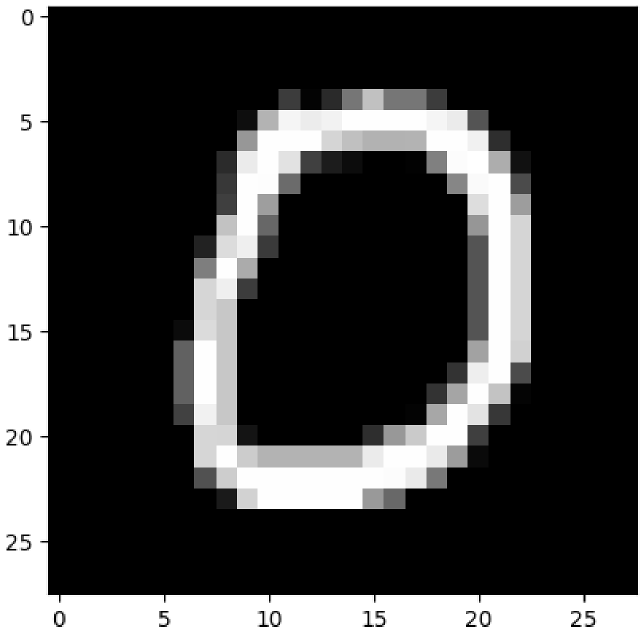
```
train_labels[0]
```

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

Visulization

```
def plot_image(img_index):
    label_index = train_labels[img_index]
    plt.imshow(train_data[img_index]/255, cmap = 'gray')
    print(label_index)

img_index = 10
plot_image(img_index)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```




```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))


model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 10, use_bias=True, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.007)
#opt = keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_42 (Dense)            (None, 128)               100480

 dense_43 (Dense)            (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
```

```
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
batch_size = 128
epochs = 50

h = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
Epoch 1/50
63/63 [==============================] - 1s 6ms/step - loss: 40.5557 - accuracy: 0.5930 - val_loss: 1.3223 - val_accuracy: 0.5495
Epoch 2/50
63/63 [==============================] - 0s 5ms/step - loss: 1.0807 - accuracy: 0.6879 - val_loss: 0.9276 - val_accuracy: 0.7285
Epoch 3/50
63/63 [==============================] - 0s 5ms/step - loss: 0.8409 - accuracy: 0.7660 - val_loss: 0.8779 - val_accuracy: 0.7935
Epoch 4/50
63/63 [==============================] - 0s 4ms/step - loss: 0.6871 - accuracy: 0.8076 - val_loss: 0.9245 - val_accuracy: 0.7890
Epoch 5/50
63/63 [==============================] - 0s 5ms/step - loss: 0.5934 - accuracy: 0.8294 - val_loss: 0.8093 - val_accuracy: 0.8310
Epoch 6/50
63/63 [==============================] - 0s 6ms/step - loss: 0.5231 - accuracy: 0.8506 - val_loss: 0.6695 - val_accuracy: 0.8595
Epoch 7/50
63/63 [==============================] - 0s 7ms/step - loss: 0.4556 - accuracy: 0.8636 - val_loss: 0.7397 - val_accuracy: 0.8545
Epoch 8/50
63/63 [==============================] - 0s 7ms/step - loss: 0.4378 - accuracy: 0.8763 - val_loss: 0.6983 - val_accuracy: 0.8790
Epoch 9/50
63/63 [==============================] - 0s 6ms/step - loss: 0.4128 - accuracy: 0.8873 - val_loss: 0.6743 - val_accuracy: 0.8755
Epoch 10/50
63/63 [==============================] - 1s 9ms/step - loss: 0.3583 - accuracy: 0.9021 - val_loss: 0.5586 - val_accuracy: 0.8725
Epoch 11/50
63/63 [==============================] - 1s 8ms/step - loss: 0.3321 - accuracy: 0.9005 - val_loss: 0.6276 - val_accuracy: 0.8710
Epoch 12/50
63/63 [==============================] - 0s 7ms/step - loss: 0.3102 - accuracy: 0.9141 - val_loss: 0.6188 - val_accuracy: 0.9000
Epoch 13/50
63/63 [==============================] - 0s 6ms/step - loss: 0.2806 - accuracy: 0.9201 - val_loss: 0.5288 - val_accuracy: 0.9095
Epoch 14/50
63/63 [==============================] - 0s 6ms/step - loss: 0.2804 - accuracy: 0.9252 - val_loss: 0.5888 - val_accuracy: 0.9120
Epoch 15/50
63/63 [==============================] - 0s 6ms/step - loss: 0.2757 - accuracy: 0.9281 - val_loss: 0.5783 - val_accuracy: 0.9100
Epoch 16/50
63/63 [==============================] - 0s 6ms/step - loss: 0.2922 - accuracy: 0.9273 - val_loss: 0.6166 - val_accuracy: 0.9035
Epoch 17/50
63/63 [==============================] - 0s 6ms/step - loss: 0.3362 - accuracy: 0.9144 - val_loss: 0.6996 - val_accuracy: 0.9175
Epoch 18/50
63/63 [==============================] - 0s 6ms/step - loss: 0.2749 - accuracy: 0.9324 - val_loss: 0.6785 - val_accuracy: 0.9055
Epoch 19/50
```