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
```

Rozpoznawanie cyfr - zbiór **MNIST**

Pobranie danych

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step
```

Wybrane informacje o zbiorze

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

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

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

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

```
train_data[0]
```

```
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
        253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
        195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
         11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]], dtype=uint8)
```

```
train_labels[0]
```

```
5
```

Kodowanie one-hot

```
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))
```

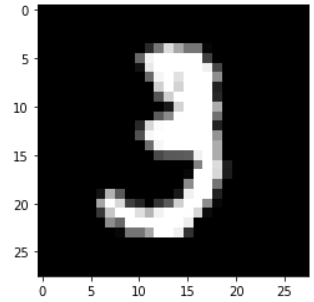
```
train_labels[0]
```

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

Wizualizacja

```
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.]
```



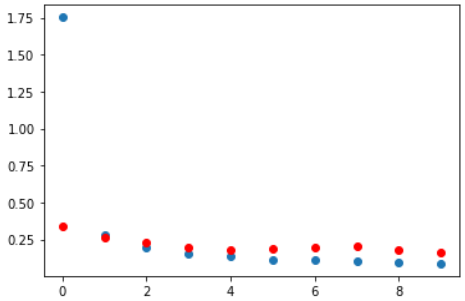
```
train_images = train_data.reshape((-1, 784))
test_images = test_data.reshape((-1, 784))


model = Sequential()
model.add(Dense(units = 128, use_bias=True, input_shape=(784,), activation = "relu"))
model.add(Dense(units = 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 = tf.keras.optimizers.Adam(learning_rate=0.002)
#opt = tf.keras.optimizers.SGD(learning_rate=0.001)

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

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

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 125,898
Trainable params: 125,898
Non-trainable params: 0
_________________________________________________________________
```

```
batch_size = 128
epochs = 10

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

```
Epoch 1/10
375/375 [==============================] - 6s 7ms/step - loss: 1.7522 - accuracy: 0.8389 - val_loss: 0.3376 - val_accuracy: 0.9143
Epoch 2/10
375/375 [==============================] - 3s 7ms/step - loss: 0.2780 - accuracy: 0.9272 - val_loss: 0.2641 - val_accuracy: 0.9295
Epoch 3/10
375/375 [==============================] - 3s 7ms/step - loss: 0.1960 - accuracy: 0.9444 - val_loss: 0.2251 - val_accuracy: 0.9427
Epoch 4/10
375/375 [==============================] - 2s 7ms/step - loss: 0.1519 - accuracy: 0.9566 - val_loss: 0.1978 - val_accuracy: 0.9461
Epoch 5/10
375/375 [==============================] - 3s 7ms/step - loss: 0.1347 - accuracy: 0.9605 - val_loss: 0.1781 - val_accuracy: 0.9550
Epoch 6/10
375/375 [==============================] - 1s 4ms/step - loss: 0.1120 - accuracy: 0.9670 - val_loss: 0.1842 - val_accuracy: 0.9502
Epoch 7/10
375/375 [==============================] - 1s 4ms/step - loss: 0.1086 - accuracy: 0.9681 - val_loss: 0.1943 - val_accuracy: 0.9508
Epoch 8/10
375/375 [==============================] - 1s 4ms/step - loss: 0.1018 - accuracy: 0.9706 - val_loss: 0.2008 - val_accuracy: 0.9531
Epoch 9/10
375/375 [==============================] - 1s 4ms/step - loss: 0.0940 - accuracy: 0.9722 - val_loss: 0.1806 - val_accuracy: 0.9564
Epoch 10/10
375/375 [==============================] - 1s 4ms/step - loss: 0.0868 - accuracy: 0.9746 - val_loss: 0.1635 - val_accuracy: 0.9622
```

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

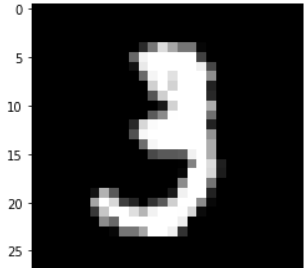



```
score = model.evaluate(test_images, test_labels, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 0.1703936904668808
Test accuracy: 0.960099995136261
```

```
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. 1. 0. 0. 0. 0. 0. 0.]
1/1 [==============================] - 0s 69ms/step
array([[2.2774202e-15, 5.3581234e-10, 3.9536769e-09, 9.9999952e-01,
        8.4330068e-20, 2.5596490e-08, 1.1180695e-20, 2.8880257e-15,
        8.2625057e-08, 3.8702055e-07]], dtype=float32)
```



✓ 0 s ukończono o 15:57