```
import numpy as np
from keras import layers
from keras import models
import tensorflow as tf

(X_train, y_train), (X_test, y_test) =
tf.keras.datasets.mnist.load_data()

# состав датасета
print(f'|type = {type(X_train)}| shape = {X_train.shape}|')
print(f'|type = {type(y_train)}| shape = {y_train.shape}|')
```

```
|type = <class 'numpy.ndarray'>| shape = (60000, 28, 28)|
|type = <class 'numpy.ndarray'>| shape = (60000,)|
```

```
# Нормализуем
X_train = (X_train)/255 - 0.5
X_test = X_test/255 - 0.5

# размерность отдельного изображения
X_train[0].shape
```

```
(28, 28)
```

```
# Построим нейроесть из 3х Dense слоев (128,128,10)
from keras.engine.sequential import Sequential
model = Sequential()
# входной слой в соответсвии с размерностью данных
model.add(layers.Input(shape = X_train[0].shape))
# преобразование входного слоя
model.add(layers.Flatten())
# 1 Dense слой 128 нейронов
model.add(layers.Dense(128, activation='relu'))
# 2 Dense слой 128 нейронов
model.add(layers.Dense(128, activation = 'relu'))
# 3 Dense слой 10 нейронов он же выходной, (10 цифр => 10 размерность
выходного слоя)
model.add(layers.Dense(10, activation='softmax'))

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

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_6 (Flatten)         (None, 784)               0

 dense_18 (Dense)            (None, 128)               100480

 dense_19 (Dense)            (None, 128)               16512
```

```
 dense_20 (Dense)            (None, 10)                1290

=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

```python
# 118,282 - это 118282

# обучим модель
model.fit(X_train, tf.keras.utils.to_categorical(y_train), epochs =
20,
          validation_data=(X_test,
tf.keras.utils.to_categorical(y_test)))
predictions = model.predict(X_test[:10])
```

```
Epoch 1/20
1875/1875 [==============================] - 5s 2ms/step - loss:
0.3061 - accuracy: 0.9067 - val_loss: 0.1933 - val_accuracy: 0.9377
Epoch 2/20
1875/1875 [==============================] - 4s 2ms/step - loss:
0.1493 - accuracy: 0.9537 - val_loss: 0.1193 - val_accuracy: 0.9636
Epoch 3/20
1875/1875 [==============================] - 5s 2ms/step - loss:
0.1136 - accuracy: 0.9648 - val_loss: 0.1248 - val_accuracy: 0.9611
Epoch 4/20
1875/1875 [==============================] - 4s 2ms/step - loss:
0.0940 - accuracy: 0.9700 - val_loss: 0.0896 - val_accuracy: 0.9717
Epoch 5/20
1875/1875 [==============================] - 5s 2ms/step - loss:
0.0794 - accuracy: 0.9745 - val_loss: 0.0982 - val_accuracy: 0.9707
Epoch 6/20
1875/1875 [==============================] - 4s 2ms/step - loss:
0.0717 - accuracy: 0.9770 - val_loss: 0.1059 - val_accuracy: 0.9669
Epoch 7/20
1875/1875 [==============================] - 4s 2ms/step - loss:
0.0639 - accuracy: 0.9790 - val_loss: 0.0964 - val_accuracy: 0.9713
Epoch 8/20
1875/1875 [==============================] - 5s 2ms/step - loss:
0.0597 - accuracy: 0.9803 - val_loss: 0.0921 - val_accuracy: 0.9712
Epoch 9/20
1875/1875 [==============================] - 4s 2ms/step - loss:
0.0500 - accuracy: 0.9841 - val_loss: 0.0893 - val_accuracy: 0.9720
Epoch 10/20
1875/1875 [==============================] - 5s 2ms/step - loss:
0.0481 - accuracy: 0.9842 - val_loss: 0.0962 - val_accuracy: 0.9743
Epoch 11/20
1875/1875 [==============================] - 4s 2ms/step - loss:
```

```
0.0448 - accuracy: 0.9849 - val_loss: 0.0806 - val_accuracy: 0.9763
Epoch 12/20
1875/1875 [==============================] - 4s 2ms/step - loss:
0.0401 - accuracy: 0.9865 - val_loss: 0.0978 - val_accuracy: 0.9737
Epoch 13/20
1875/1875 [==============================] - 4s 2ms/step - loss:
0.0385 - accuracy: 0.9869 - val_loss: 0.0863 - val_accuracy: 0.9761
Epoch 14/20
1875/1875 [==============================] - 5s 3ms/step - loss:
0.0365 - accuracy: 0.9883 - val_loss: 0.0978 - val_accuracy: 0.9762
Epoch 15/20
1875/1875 [==============================] - 5s 2ms/step - loss:
0.0343 - accuracy: 0.9889 - val_loss: 0.1006 - val_accuracy: 0.9757
Epoch 16/20
1875/1875 [==============================] - 5s 2ms/step - loss:
0.0321 - accuracy: 0.9896 - val_loss: 0.0980 - val_accuracy: 0.9770
Epoch 17/20
1875/1875 [==============================] - 5s 3ms/step - loss:
0.0294 - accuracy: 0.9901 - val_loss: 0.1084 - val_accuracy: 0.9749
Epoch 18/20
1875/1875 [==============================] - 5s 3ms/step - loss:
0.0308 - accuracy: 0.9899 - val_loss: 0.0978 - val_accuracy: 0.9762
Epoch 19/20
1875/1875 [==============================] - 5s 3ms/step - loss:
0.0268 - accuracy: 0.9909 - val_loss: 0.1013 - val_accuracy: 0.9779
Epoch 20/20
1875/1875 [==============================] - 5s 3ms/step - loss:
0.0282 - accuracy: 0.9907 - val_loss: 0.0988 - val_accuracy: 0.9773
1/1 [==============================] - 0s 39ms/step
```

```
# проверим точность
import keras
class_num = 10
y_test = keras.utils.to_categorical(y_test, class_num)

score = model.evaluate(X_test, y_test, verbose=0)
print('loss:', score[0])
print('acc:', score[1])
```

```
loss: 0.09884334355592728
acc: 0.9772999882698059
```