

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

Звіт до лабораторної роботи №6

з курсу

«Машинне навчання»

студента 2 курсу
групи ІТ-02
Макарова Іллі Сергійовича

Викладач:
Оніщенко В.

Тема: Проектування та навчання штучної нейронної мережі для задач

класифікації

Виконання:

Итак, у роботи есть три части, пожалуй отчет я тоже так разделю. Первая часть, это класификация цифр **mnist**, далее **cifar10** и наконец **fashion-mnist**.

ЧАСТЬ ПЕРВАЯ

Ну что, тут нужно было просто повторить действия из лабы, так что тут я особо комментировать не буду.

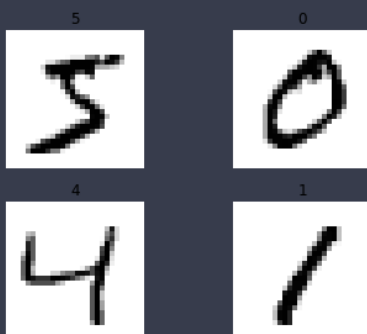
```
In [9]: import cv2
import matplotlib.pyplot as plt
from keras.datasets import mnist
from keras import layers
from keras.utils import to_categorical
from keras.models import Sequential, load_model

In [24]: (train_images, train_labels), (test_images, test_labels) = mnist.load_data()

In [31]: figure, axes = plt.subplots(nrows=2, ncols=2, figsize=(6, 4))

for axes, image, target in zip(axes.ravel(), train_images, train_labels):
    axes.imshow(image, cmap=plt.cm.gray_r)
    axes.set_title(target)
    axes.axis('off')

plt.tight_layout()
```



```
In [4]: train_images = train_images.reshape((60000, 28 * 28))
        train_images = train_images.astype('float32') / 255

        test_images = test_images.reshape((10000, 28 * 28))
        test_images = test_images.astype('float32') / 255
```

```
In [5]: train_labels = to_categorical(train_labels)
        test_labels = to_categorical(test_labels)
```

Building model

```
In [6]: model = Sequential()
        model.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
        model.add(layers.Dense(10, activation='softmax'))
```

```
In [7]: model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
In [26]: model.fit(x=train_images, y=train_labels, epochs=10, batch_size=128)
```

```
In [27]: model.evaluate(x=test_images, y=test_labels)

313/313 [=====] - 4s 11ms/step - loss: 0.0773 - accuracy: 0.9804

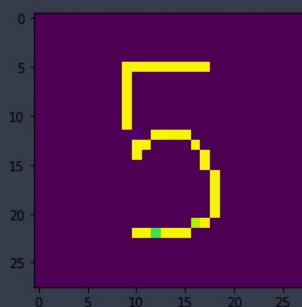
[0.07731213420629501, 0.980400025844574]
```

```
In [28]: model.save('models/digits_classifier.h5')
```

```
In [10]: model = load_model('models/digits_classifier.h5')
```

```
In [19]: image_5 = 255 - cv2.imread('images/5.png', 0)
        image_5 = cv2.resize(image_5, (28, 28))

        plt.imshow(image_5)
        plt.show()
```



```
In [20]: image_5 = image_5.reshape((1, 28 * 28))
        image_5 = image_5.astype('float32') / 255
```

```
In [21]: result = model.predict(image_5)
```

```
In [22]: print(f'Prediction result: {result.argmax()}')
```

```
Prediction result: 5
```

ЧАСТЬ ДВА

Ну тут уже интереснее, cifar10 это уже не просто черно-белые картинки чисел, это большой датасет цветных фотографий, что можно разделить на 10 классов, тут уже обычной нейронкой не обойтись.

Начало все тоже, импорты, собираем данные, делим их:

```
In [1]: import cv2
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Sequential
from keras.datasets import cifar10
from keras import layers
from keras.utils import to_categorical
from keras.optimizers import Adam

In [2]: (train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

In [3]: train_images[0].shape

(32, 32, 3)

In [4]: train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0

In [5]: train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

Еще я тут сразу напишу функцию, что по входному категориальному вектору выведет мне клас к которому относиться ответ:

```
In [6]: def label_array_to_labels_name(label_arr):
image_index = {
    0: 'PLANE',
    1: 'AUTOMOBILE',
    2: 'BIRD',
    3: 'CAT',
    4: 'DEER',
    5: 'DOG',
    6: 'FROG',
    7: 'HORSE',
    8: 'SHIP',
    9: 'TRUCK'
}

return image_index[label_arr.argmax()]
```

Сразу это дело и проверим:

```
In [7]: plt.imshow(train_images[0])
plt.axis('off')
print(f'This is: {label_array_to_labels_name(train_labels[0])}')

This is: FROG
```

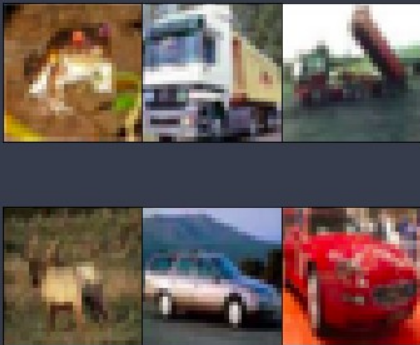


Вроде все работает, теперь давайте выведем несколько фотографий из датасета, чтоб вообще посмотреть на них.

```
In [8]: _, axes = plt.subplots(2, 3, figsize=(6, 6))
axes = axes.flatten()

for image, axe in zip(train_images, axes):
    axe.imshow(image)
    axe.set_xticks([])
    axe.set_yticks([])

plt.subplots_adjust(wspace=0, hspace=0)
plt.show()
```



Ну что же, звучит хайпово, теперь надо создать модельку. Как я уже описав выше, тут просто нейронка работать не будет, поэтому я написал сверточную (CNN), два сверточных слоя, после каждого MaxPooling, что уменьшает в два раза, и после еще два полносвязных слоя.

Build a model

```
In [10]: model = Sequential([
    layers.Conv2D(
        filters=32, kernel_size=(3, 3),
        activation='relu', padding='same', input_shape=(32, 32, 3)
    ),
    layers.MaxPool2D(pool_size=(2, 2), strides=2),
    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same'),
    layers.MaxPool2D(pool_size=(2, 2), strides=2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

In [85]: model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

In [86]: model.fit(x=train_images, y=train_labels, epochs=10, batch_size=32)
```

Давайте посмотрим на результаты

```
In [87]: model.evaluate(x=test_images, y=test_labels)

313/313 [=====] - 14s 44ms/step - loss: 1.0090 - accuracy: 0.7185

[1.009033441543579, 0.718500018119812]
```

Нуууу, не то чтоб очень хорошо, 70%, с другой стороны, при слепом выборе, очевидно было бы 10%, поэтому я думаю что пойдет.

Давайте глянем, как хорошо наша нейронка работает на картинке из интернета:

```
In [135]: img = cv2.imread('images/plane.jpg')
    img = cv2.resize(img, (32, 32)).astype('float32') / 255.0

In [136]: plt.imshow(img)
    plt.axis('off')
    plt.show()
```



```
In [137]: img = np.expand_dims(img, 0)
          result = model.predict(x=img)

In [138]: print(f'The result of prediction is: {label_array_to_labels_name(result)}')

The result of prediction is: PLANE
```

Ну что, вроде норм.

ЧАСТЬ ТРИ

И наконец последняя часть, fashion-mnist, тут картинки снова двухмерные, то есть не цветные. Посему тут вначале все тоже самое, только еще внизу мы насильно добавляем нашим данным еще одну размерность

```
In [34]: import cv2
          import matplotlib.pyplot as plt
          import numpy as np
          from keras import layers
          from keras.models import Sequential
          from keras.datasets import fashion_mnist
          from keras.utils import to_categorical
          from keras.optimizers import Adam

In [9]: (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

In [10]: train_images = train_images.astype('float32') / 255.0
          test_images = test_images.astype('float32') / 255.0

In [11]: train_images = np.expand_dims(train_images, axis=3)
          test_images = np.expand_dims(test_images, axis=3)

In [12]: train_images.shape

(60000, 28, 28, 1)
```

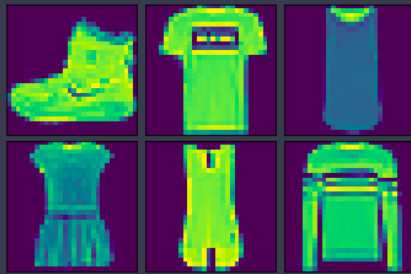
Что тут добавить даже не знаю, далее мы делаем все тоже самое что и раньше, в прошлой части, визуализируем несколько изображений, строим модель, evaluate ее, и тестируем на реальной картинке.

Не знаю, можно ли тут что добавить, так что вот просто скрины.

```
In [13]: _, axes = plt.subplots(2, 3)
         axes = axes.flatten()

         for image, axe in zip(train_images, axes):
             axe.imshow(image)
             axe.set_xticks([])
             axe.set_yticks([])

         plt.subplots_adjust(wspace=0.05, hspace=0.05)
         plt.show()
```



```
In [14]: train_labels = to_categorical(train_labels)
         test_labels = to_categorical(test_labels)
```

Ах, да, тут я тоже написал такую функцию

```
In [15]: def label_array_to_labels_name(label_arr):
         image_index = {
             0: 'T-shirt/top',
             1: 'Trouser',
             2: 'Pullover',
             3: 'Dress',
             4: 'Coat',
             5: 'Sandal',
             6: 'Shirt',
             7: 'Sneaker',
             8: 'Bag',
             9: 'Ankle boot'
         }

         return image_index[label_arr.argmax()]
```

```
In [29]: # Lets build a model

         model = Sequential([
             layers.Conv2D(
                 filters=32, kernel_size=(3, 3), padding='same',
                 activation='relu', input_shape=(28,28,1)
             ),
             layers.MaxPool2D(pool_size=(2,2), strides=2),
             layers.Conv2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu'),
             layers.MaxPool2D(pool_size=(2,2), strides=2),
             layers.Flatten(),
             layers.Dense(128, activation='relu'),
             layers.Dense(10, activation='softmax')
         ])
```



```
In [30]: model.summary()
```

```
Model: "sequential_2"
-----
Layer (type)                 Output Shape              Param #
-----
conv2d_4 (Conv2D)            (None, 28, 28, 32)        320
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 32)        0
conv2d_5 (Conv2D)            (None, 14, 14, 64)       18496
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 64)         0
flatten (Flatten)            (None, 3136)              0
dense_4 (Dense)              (None, 128)               401536
dense_5 (Dense)              (None, 10)                1290
-----
Total params: 421,642
Trainable params: 421,642
Non-trainable params: 0
-----
```

```
In [31]: model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

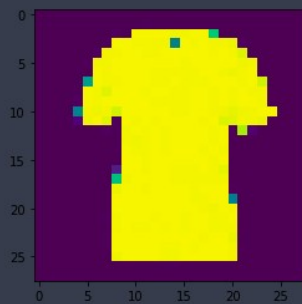
```
In [32]: model.fit(x=train_images, y=train_labels, epochs=10, batch_size=30)
```

```
In [33]: model.evaluate(x=test_images, y=test_labels)
```

```
313/313 [=====] - 9s 28ms/step - loss: 0.2961 - accuracy: 0.9163

[0.2961329221725464, 0.9162999987602234]
```

```
In [64]: img = cv2.imread('images/tshirt.jpg')
img = img.astype('float32') / 255
img = cv2.resize(img, (28, 28))
img = np.delete(img, (0, 1), axis=2)
plt.imshow(img)
plt.show()
```



```
In [65]: img = np.expand_dims(img, axis=0)
result = model.predict(x=img)
```

```
In [66]: print(f'Predicted result is: {label_array_to_labels_name(result)}')
```

```
Predicted result is: T-shirt/top
```

Вот как то так, не знаю, надо ли писать какое то заключение, но что то думаю написать все же стоит. Сегодня я тут было вспомнил что такое CNN, как они работают, и как их вообще строить. Та вот и все :3