МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание объектов на фотографиях»

Студент гр. 7382 _____ Государкин Я.С.

Преподаватель _____ Жангиров Т.Р

Санкт-Петербург

2020

Задание

Распознавание объектов на фотографиях (Object Recognition in Photographs)

CIFAR-10 (классификация небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик).

1. Построить и обучить сверточную нейронную сеть
2. Исследовать работу сеть без слоя Dropout
3. Исследовать работу сети при разных размерах ядра свертки

Теоретическая часть

Данная задача — задача многоклассовой классификации изображений. Одно изображение из датасета CIFAR-10, на котором будет обучаться НС — имеет размерность 32 * 32 * 3.

Эти картинки отличаются от датасета MNIST, они трёхканальные, и, немного больше по размеру. В качестве архитектур будем рассматривать архитектуры свёрточных нейронных сетей.

Ход работы

1. Построить и обучить сверточную нейронную сеть

Найденная архитектура даёт точность на тестовых данных ~ 75%.

Оптимизатор: Adam, lr=0.001

batch_size = 100

loss_func: categorical_crossentropy

epochs = 20

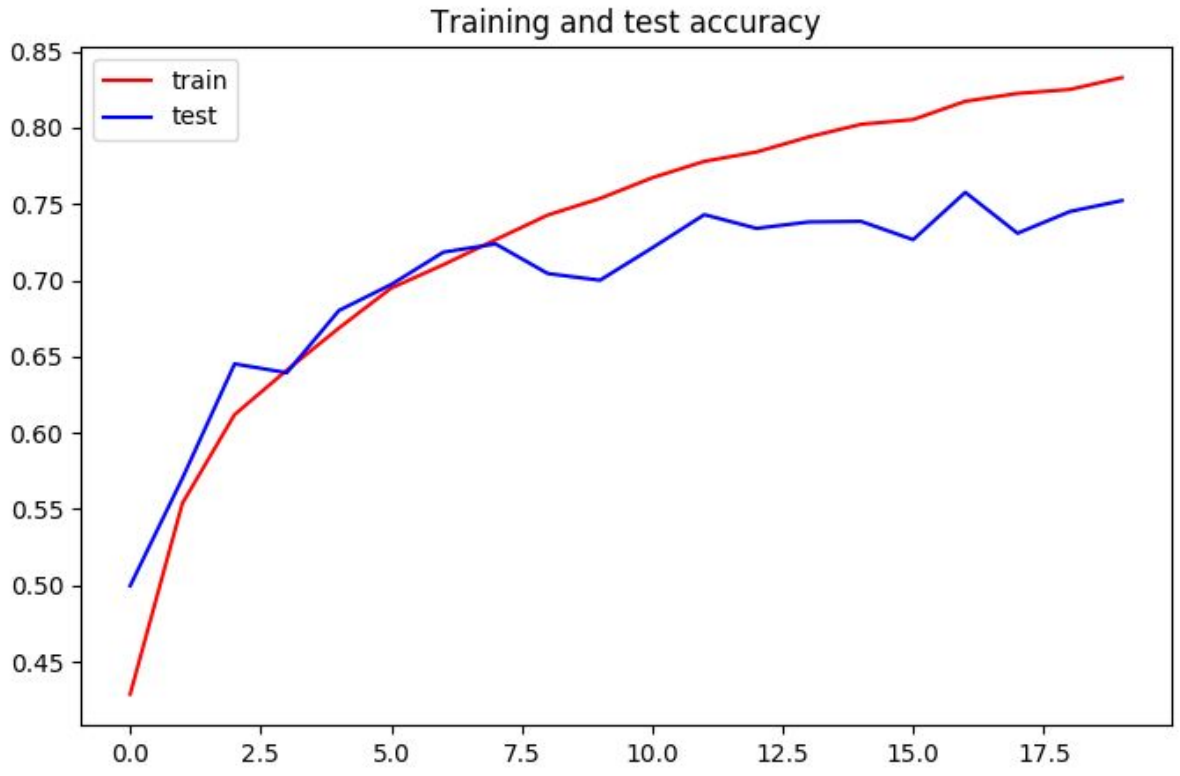
Model:

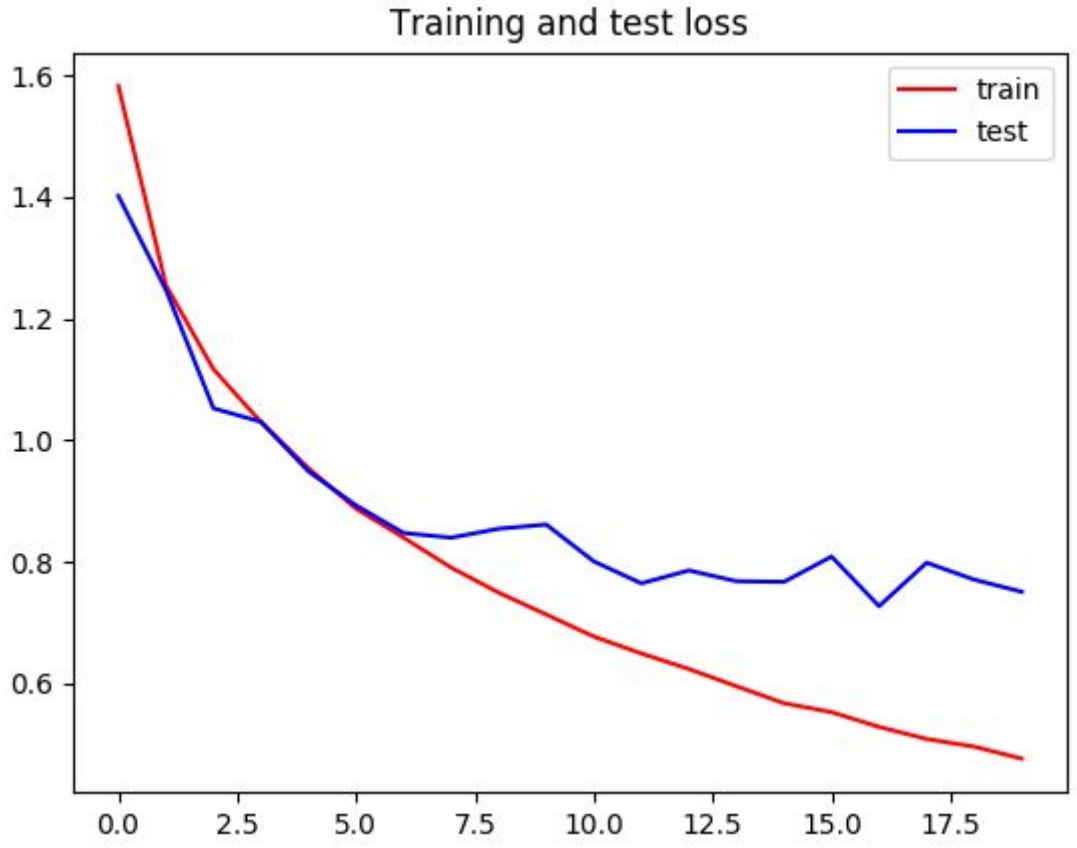
```
self.features = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), strides=(1, 1), activation='relu'),
    Dropout(0.2),
    MaxPool2D(pool_size=2),

    Conv2D(filters=64, kernel_size=(3, 3), strides=(1, 1), activation='relu'),
    Dropout(0.2),
    Conv2D(filters=128, kernel_size=(3, 3), strides=(1, 1), activation='relu'),
    MaxPool2D(pool_size=2),
])

self.lin = Sequential([
    Flatten(),
    Dense(input_dim=6*6*4, units=classes_count, activation='softmax'),
])
```

Графики точности и ошибки см. ниже.

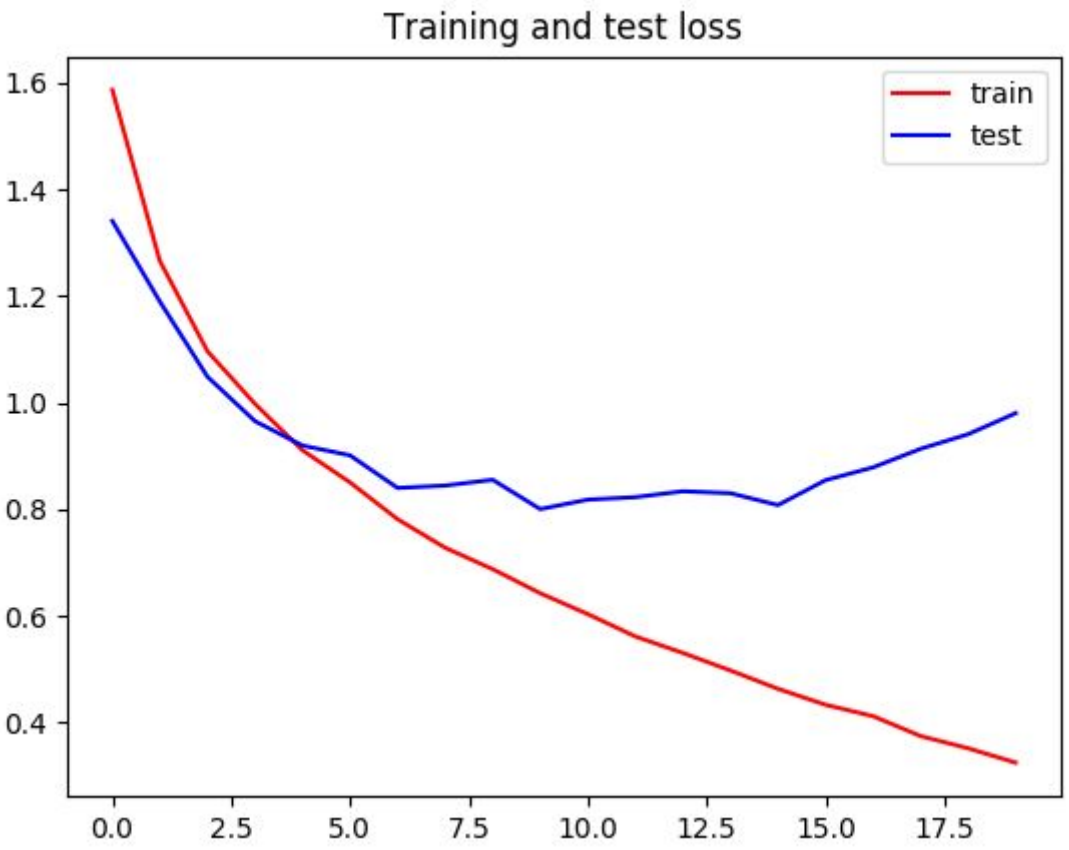


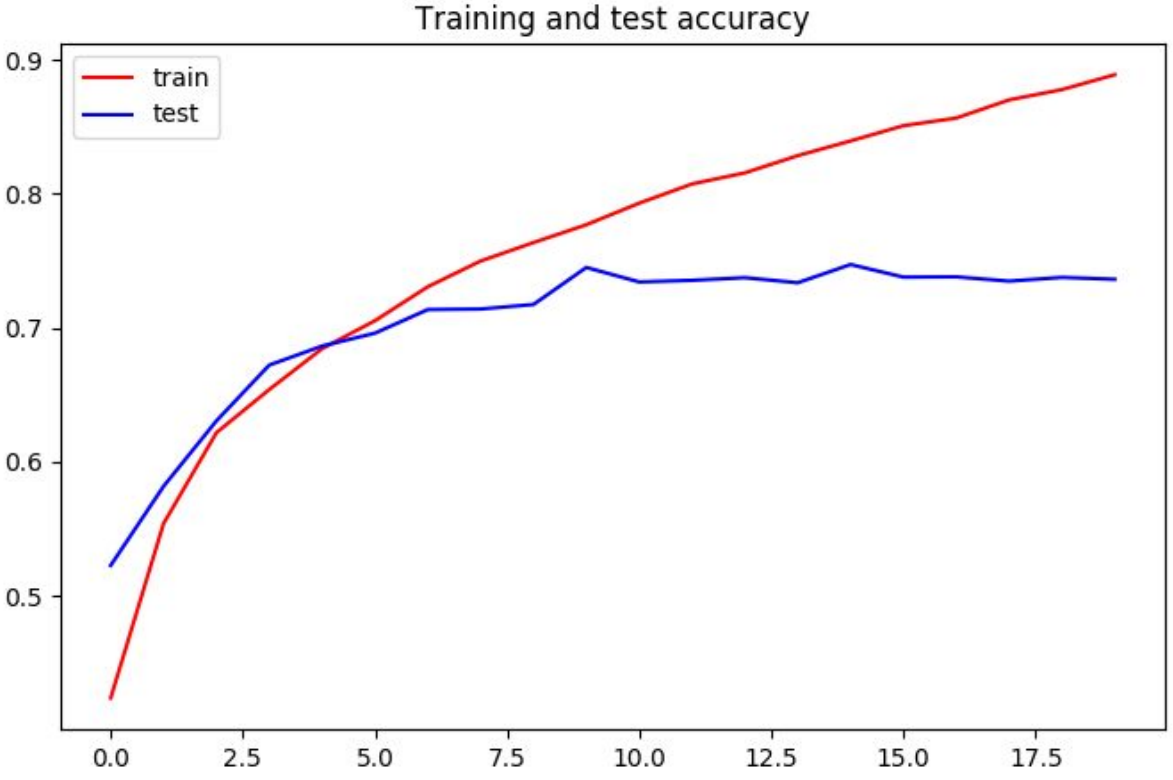


2. Исследовать работу сеть без слоя Dropout

Уберём из моделей слои dropout.

Средняя точность составила 73%. Ошибка на валидации пошла вверх, что говорит о сильном переобучении. Dropout используется для устранения переобучения, путём случайного отключения связей или нейронов, таким образом что либо связь выдаёт нулевой сигнал на выход, либо нейрон выдаёт на все свои выходы нулевой сигнал.

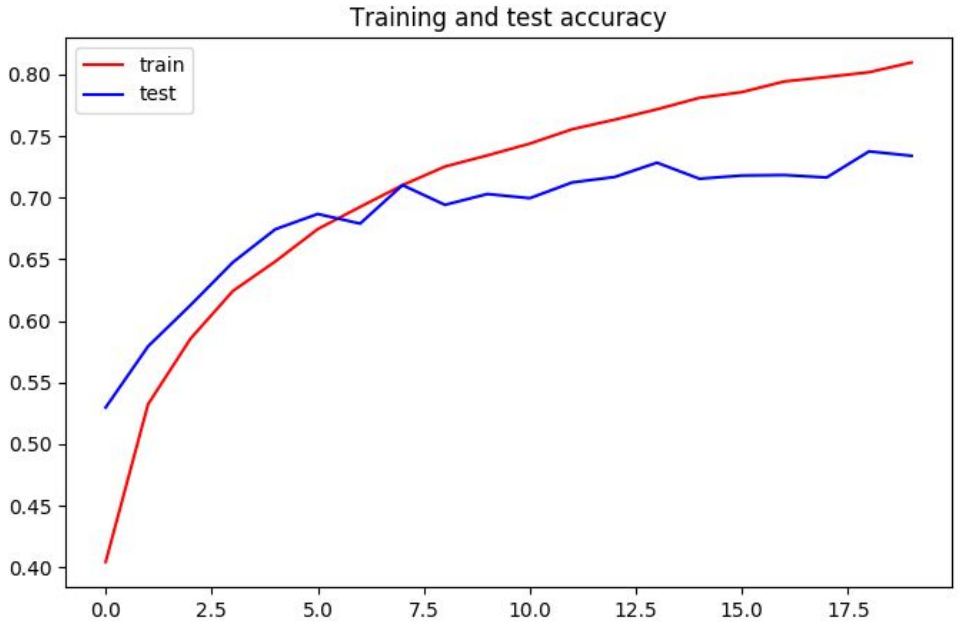

Training and test loss
train
test
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

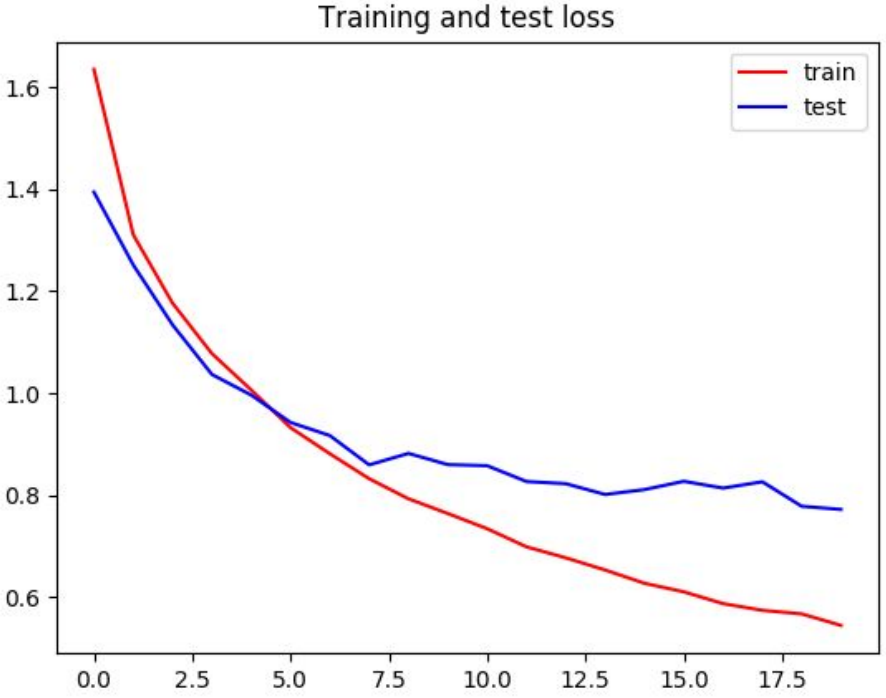

Training and test accuracy
train
test
0.9
0.8
0.7
0.6
0.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

3. Исследовать работу сети при разных размерах ядра свертки

Поменяем размер ядра свёртки на среднем и последнем свёрточном слое с 3x3 на 5x5.

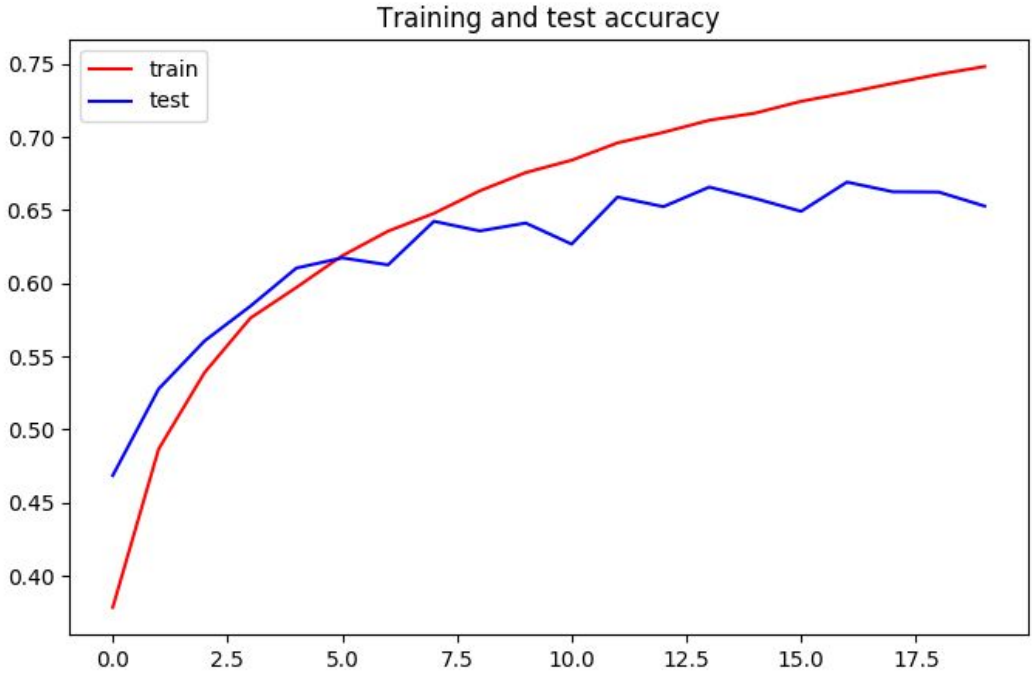
Средняя точность составила на тестовой выборке 74%

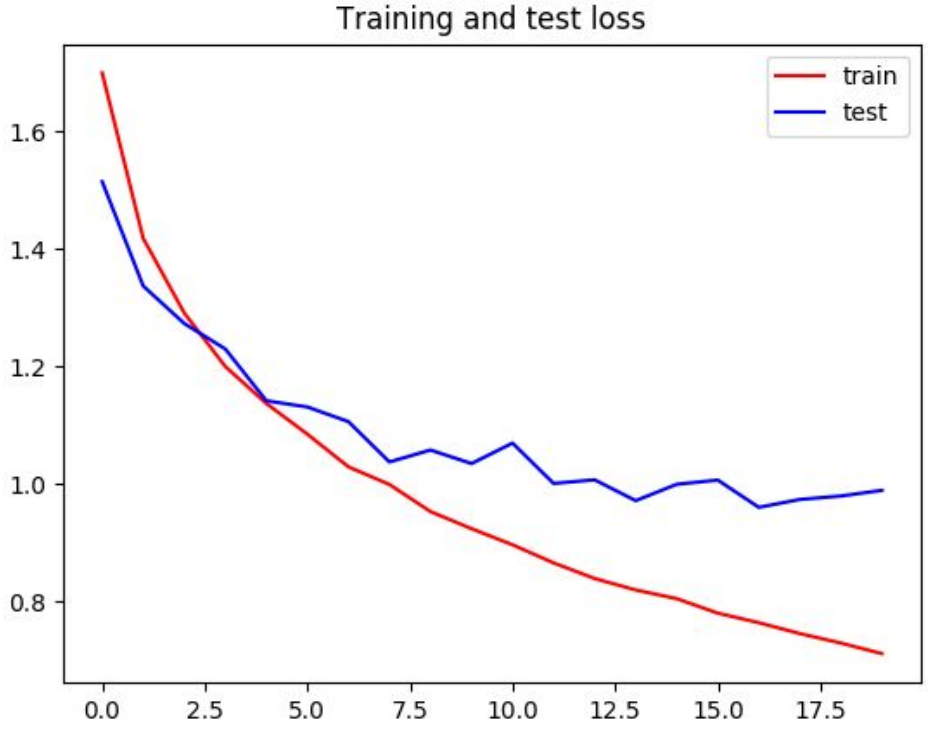




Поменяем размер ядра свёртки на первом слое на 7x7 и сравним с предыдущим.

Средняя точность составила на тестовой выборке 66%, что значительно ниже предыдущих конфигураций.





Выводы.

В ходе лабораторной работы мы ознакомились с задачей классификации изображений из датамета CIFAR-10, подобрали архитектуру, дающую точность на тестовой выборке — 75%. Было показано, что Dropout увеличивает устойчивость сети к отключению части связей и к переобучению. Смена размера ядра свёртки в двух слоях практически не повлияла на конечную точность, однако когда в первом слое было установлено kernel size 7x7, то качество заметно просело. Это произошло из-за того, что 7x7 слишком обобщало признаки сущностей на такой маленькой картинке, в итоге картинка сильнее размывается после прохода через слой и НС выдаёт плохой ответ. Код в приложении А.

ПРИЛОЖЕНИЕ А

```
import tensorflow.keras as keras

from tensorflow.keras import losses
from tensorflow.keras import optimizers
from tensorflow.keras.layers import Conv2D, Dropout, MaxPool2D, Dense, Flatten
from tensorflow.keras.models import Sequential

from tensorflow.keras.utils import to_categorical
from tensorflow.keras import datasets

import matplotlib.pyplot as plt

class ConvClassifier(keras.Model):
    def __init__(self, classes_count = 10):
        super(ConvClassifier, self).__init__()

        self.features = Sequential([
            Conv2D(filters=32, kernel_size=(3, 3), strides=(1,1), activation='relu'),
            Dropout(0.3),
            MaxPool2D(pool_size=2),

            Conv2D(filters=64, kernel_size=(5, 5), strides=(1, 1), activation='relu'),
            Dropout(0.3),
            Conv2D(filters=128, kernel_size=(5, 5), strides=(1, 1), activation='relu'),
            MaxPool2D(pool_size=2),
        ])

        self.lin = Sequential([
            Flatten(),
            Dense(input_dim=6*6*4, units=classes_count, activation='softmax'),
        ])

    def call(self, inputs):
        x = self.features(inputs)
```

```
        x = self.lin(x)
        return x

# constatnts
batch_size = 100
epochs = 20

# load data
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()

print(x_train.shape)

# normalize data
x_train = x_train / 255
x_test = x_test / 255

y_test = to_categorical(y_test, num_classes=10)
y_train = to_categorical(y_train, num_classes=10)

# init model
classifier = ConvClassifier()
optimizer = optimizers.Adam(lr=0.001)
loss = losses.CategoricalCrossentropy()

classifier.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
H = classifier.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.1)



# testing
test_loss, test_acc = classifier.evaluate(x_test, y_test)
print('test_acc:', test_acc)

# plot
plt.figure(1,figsize=(8,5))
plt.title("Training and test accuracy")
plt.plot(H.history['acc'], 'r', label='train')
plt.plot(H.history['val_acc'], 'b', label='test')
plt.legend()
plt.show()
plt.clf()
```

```
plt.figure(1,figsize=(8,5))
plt.title("Training and test loss")
plt.plot(H.history['loss'], 'r', label='train')
plt.plot(H.history['val_loss'], 'b', label='test')
plt.legend()
plt.show()
plt.clf()
```