

**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»**

Студентка гр. 8382

\_\_\_\_\_

Бердникова А.А.

Преподаватель

\_\_\_\_\_

Жангиров Т.Р.

Санкт-Петербург

2021

### **Цель работы.**

Написать программу для распознавания объектов на фотографиях (Object Recognition in Photographs) CIFAR-10, научиться классифицировать небольшие изображения по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик.

### **Задачи.**

- Ознакомиться со сверточными нейронными сетями
- Изучить построение модели в Keras в функциональном виде
- Изучить работу слоя разреживания (Dropout)

### **Требования.**

- Построить и обучить сверточную нейронную сеть
- Исследовать работу сеть без слоя Dropout
- Исследовать работу сети при разных размерах ядра свертки

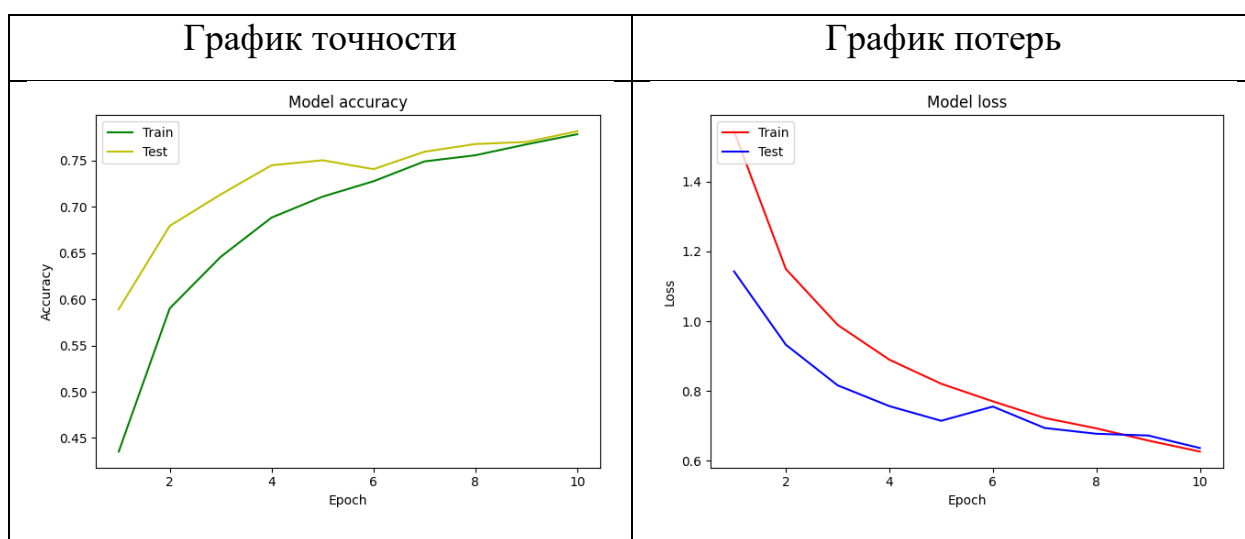
## Ход работы.

В ходе работы была создана и обучена модель искусственной нейронной сети в соответствии с условиями (код представлен в приложении).

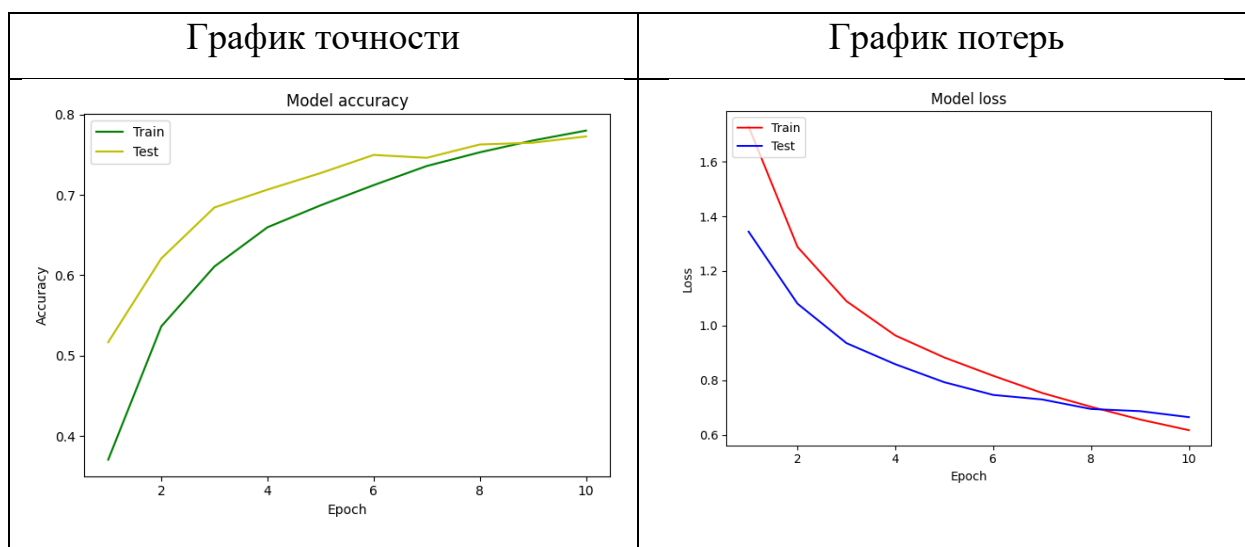
### 1. Выберем модель сети.

Изначально была создана сверточная сеть использующая сверточные слои, слои maxpooling и слои разреживания dropout. Протестировали модель с разным количеством параметра `batch_size`.

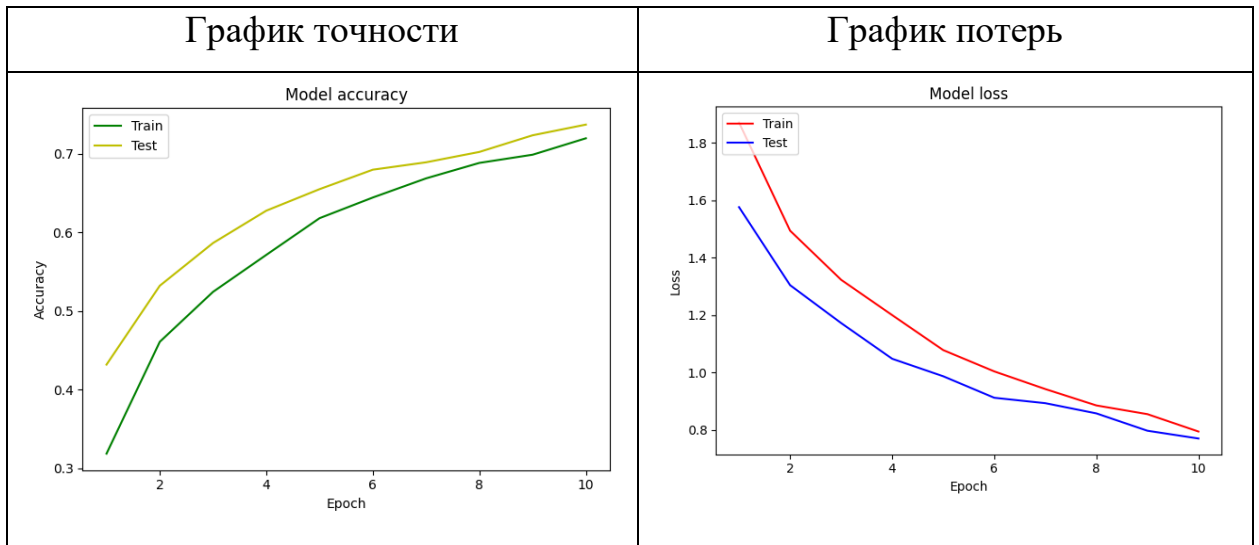
`batch_size = 32`



`batch_size = 256`



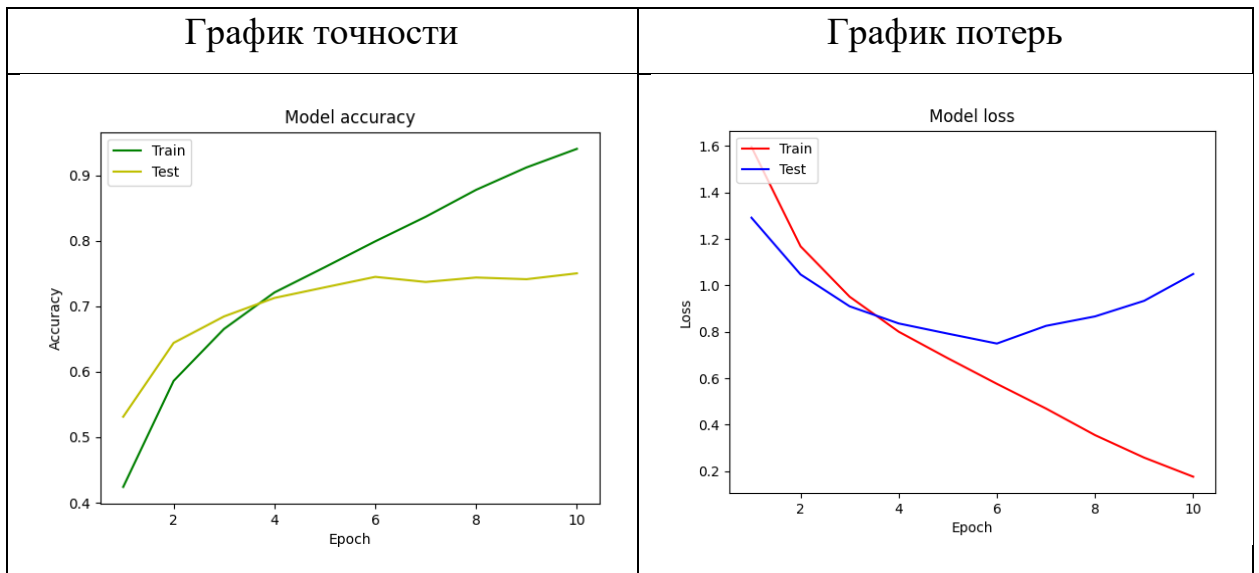
batch\_size = 712



Как видно из графиков выше точность у модели с `batch_size = 256`, поэтому будем в дальнейшем рассматривать ее.

## 2. Исследуем работу сети без слоя *Dropout*

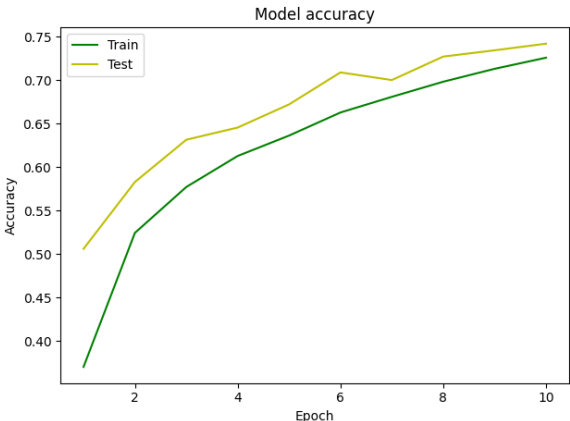
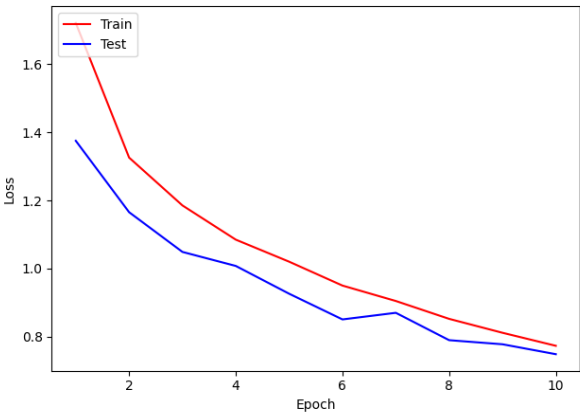
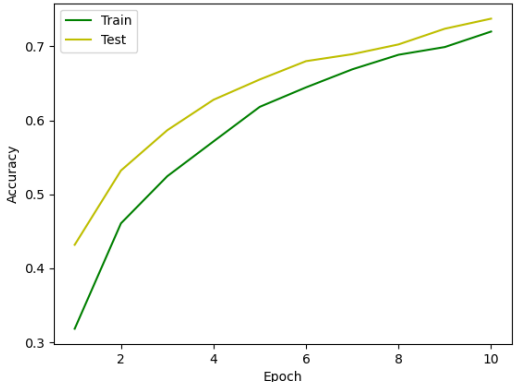
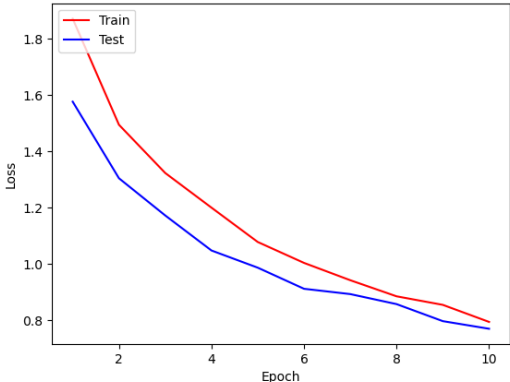
Исследовали выбранную модель без слоя `Dropout`.

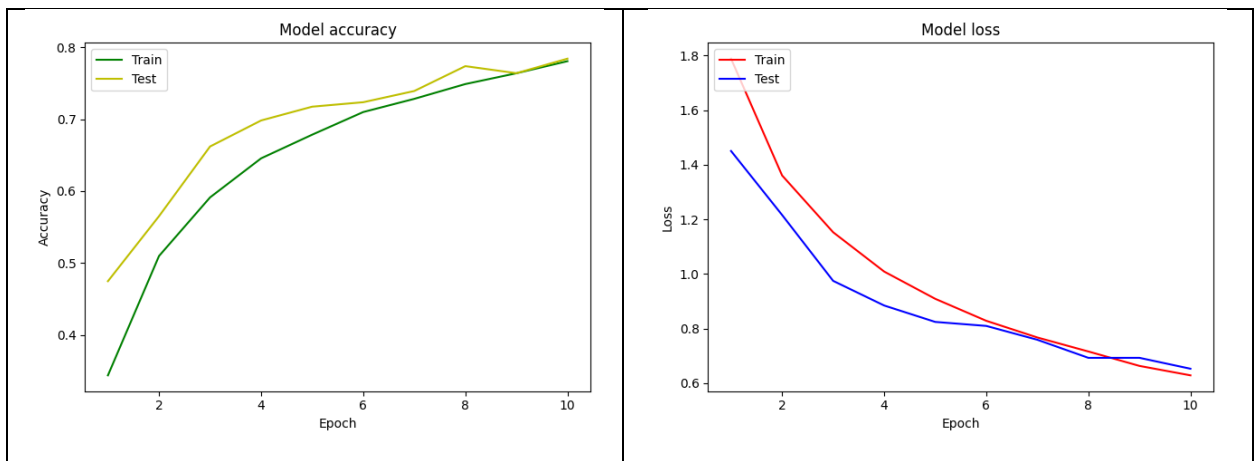


Из графиков видно, что после 4 эпохи потери начали расти, а точность перестала повышаться, из чего можно сделать вывод о необходимости слоев прореживания.

### 3. Исследуем работу сети при разных размерах ядра свертки

Были изучены архитектуры, у которых в сверточных слоях размер ядра свертки имеет форму (2,2), (3,3) и (5,5) соответственно. Результаты представлены в следующей таблице.

Размер ядра свертки 2*2	
График точности	График потерь
	
Размер ядра свертки 3*3	
График точности	График потерь
	
Размер ядра свертки 5*5	
График точности	График потерь



По графикам видно, что при для данной модели точность наибольшая при размере ядра  $5 \times 5$ , потери тестовых данных тоже меньше, чем при размере ядра  $3 \times 3$ , поэтому для данной модели ядро размером  $5 \times 5$  подходит больше.

### **Выводы.**

В ходе выполнения данной работы была создана сеть для классификации изображений, были более подробно изучены сверточные сети, влияние слоев разреживания и ядра свертки на результаты обучения.

## ПРИЛОЖЕНИЕ

### Исходный код

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Convolution2D,
MaxPooling2D, Dense, Dropout, Flatten
from tensorflow.python.keras.utils import np_utils

batch_size = 256
num_epochs = 10
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512

def load_data():
    (X_train, y_train), (X_test, y_test) = cifar10.load_data()
    num_train, depth, height, width = X_train.shape
    num_test = X_test.shape[0]
    num_classes = np.unique(y_train).shape[0]

    X_train = X_train.astype('float32')
    X_test = X_test.astype('float32')

    X_train /= np.max(X_train)
    X_test /= np.max(X_train)

    Y_train = np_utils.to_categorical(y_train, num_classes)
    Y_test = np_utils.to_categorical(y_test, num_classes)

    return X_train, Y_train, X_test, Y_test, num_train, depth, height,
width, num_test, num_classes

def build_model(kernel_size = 3, dropout = True):
    inp = Input(shape=(depth, height, width))

    conv_1 = Convolution2D(conv_depth_1, (kernel_size, kernel_size),
```

```

        padding='same', activation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1, (kernel_size, kernel_size),
        padding='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)

if dropout:
    drop_1 = Dropout(drop_prob_1)(pool_1)
else:
    drop_1 = pool_1

conv_3 = Convolution2D(conv_depth_2, (kernel_size, kernel_size),
        padding='same', activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, (kernel_size, kernel_size),
        padding='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)

if dropout:
    drop_2 = Dropout(drop_prob_1)(pool_2)
else:
    drop_2 = pool_2

flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)

if dropout:
    drop_3 = Dropout(drop_prob_2)(hidden)
else:
    drop_3 = hidden

out = Dense(num_classes, activation='softmax')(drop_3)

model = Model(inputs=inp, outputs=out)
model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
return model

```

```

def create_graphics(history, label):
    # графики потерь
    loss = history.history['loss']
    val_loss = history.history['val_loss']

```



```

epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'r')
plt.plot(epochs, val_loss, 'b')
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.savefig(label + '_loss.png')
plt.show()

# графики точности
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, 'g')
plt.plot(epochs, val_acc, 'y')
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.savefig(label + '_acc.png')
plt.show()

if __name__ == '__main__':
    X_train, Y_train, X_test, Y_test, num_train, depth, height, width,
    num_test, num_classes = load_data()
    print("Введите, что вы хотите сделать:\n"
          "1 - Исходная сеть\n"
          "2 - Сеть без слоя Dropout\n"
          "3 - Исследование сети при разных размерах ядра свертки\n")
    num = input()

    if num == '1':
        model = build_model()
        history = model.fit(X_train, Y_train,
                           batch_size=batch_size, epochs=num_epochs,
                           verbose=1, validation_split=0.1)
        res = model.evaluate(X_test, Y_test, verbose=1)
        print(res)
        create_graphics(history, 'best')

    if num == '2':
        model = build_model(dropout=False)

```

```

        history = model.fit(X_train, Y_train,
                             batch_size=batch_size, epochs=num_epochs,
                             verbose=1, validation_split=0.1)
        create_graphics(history, 'drop')

    if num == '3':
        for kernels in [2, 5, 7]:
            model = build_model(kernels, True)
            history = model.fit(X_train, Y_train,
                                batch_size=batch_size,
epochs=num_epochs,
                                verbose=1, validation_split=0.1)
            create_graphics(history, str(kernels))

    res = model.evaluate(X_test, Y_test, verbose=1)
    print(res)

```