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

ОТЧЕТ

по лабораторной работе №5
"Распознавание объектов на фотографиях"
по дисциплине «Искусственные нейронные сети»

Студент гр. 8383 _____ Бабенко Н.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель.

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).

Задание.

- 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

Выполнение работы.

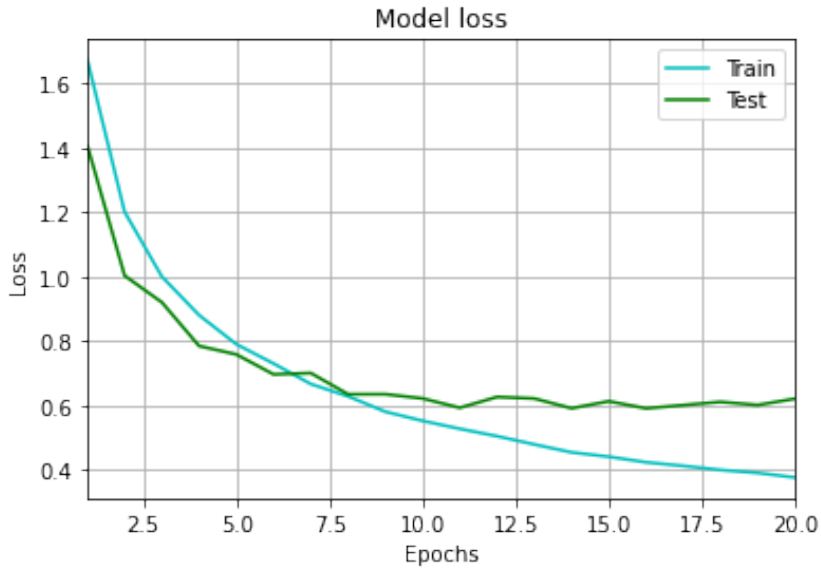
Была построена модель нейронной сети со следующими слоями:

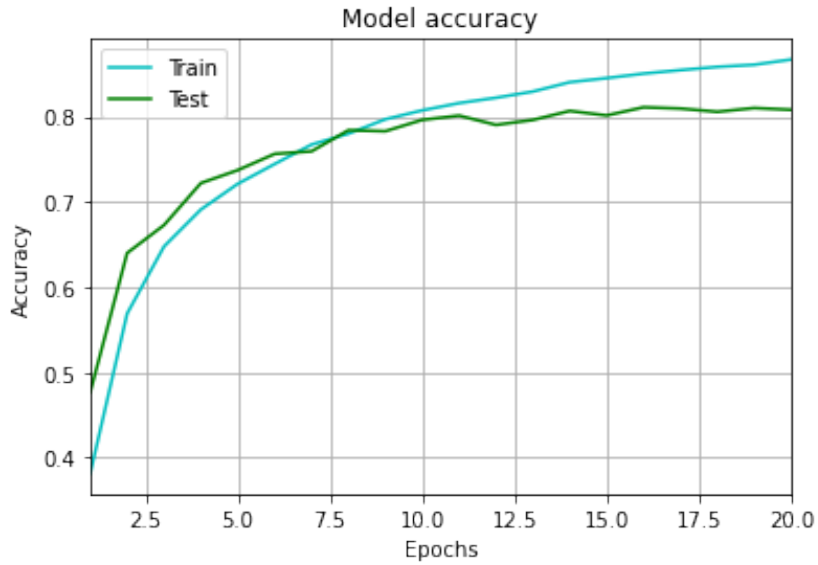
Conv2D (Relu, 32) -> Conv2D (Relu, 32) -> MaxPooling (2x2) -> Dropout (rate = 0.25) -> Conv2D (Relu, 64) -> Conv2D (Relu, 64) - > MaxPooling (2x2) -> Dropout (rate = 0.25) -> Flatten -> Dense (Relu, 512), Dropout (rate=0.5), Dense(Solfmax, 10)

Был выбран компилятор Adam, функция потерь – Categorial Crossentropy, число эпох – 80, batch_size – 100.

Проведем тестирование модели со слоями Dropout и ядром свертки с размером 3x3.

Результаты тестирования приведены на графиках:





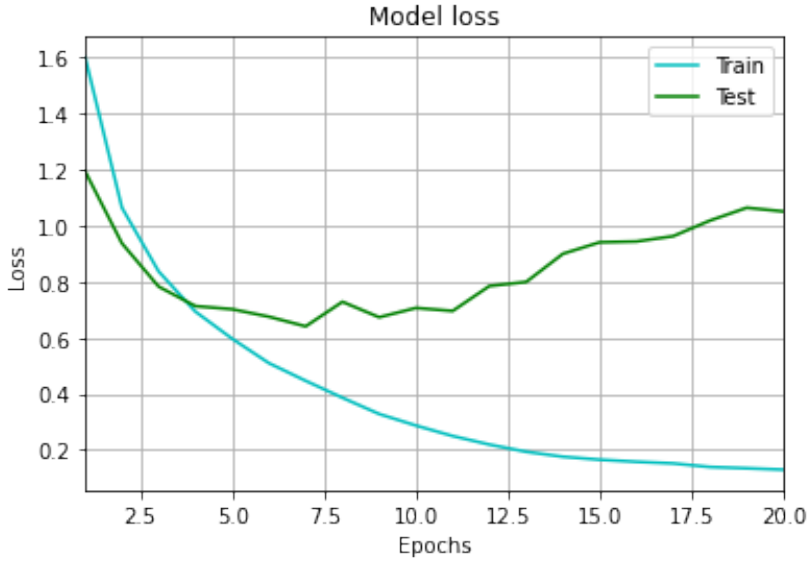
После обучения

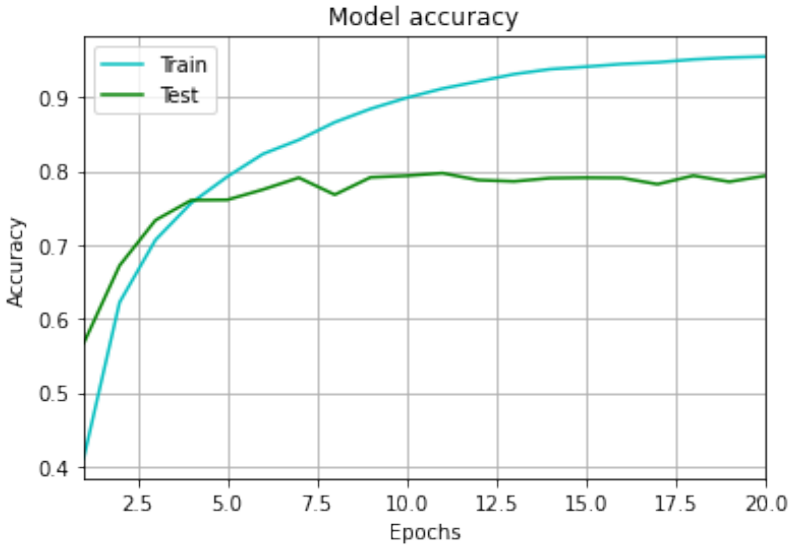
- точность: 0.8083000183105469
- потери: 0.6204720735549927

После 16 эпохи наблюдается переобучение.

Проведем тестирование модели без слоев Dropout и ядром свертки с размером 3x3.

Результаты тестирования приведены на графиках:





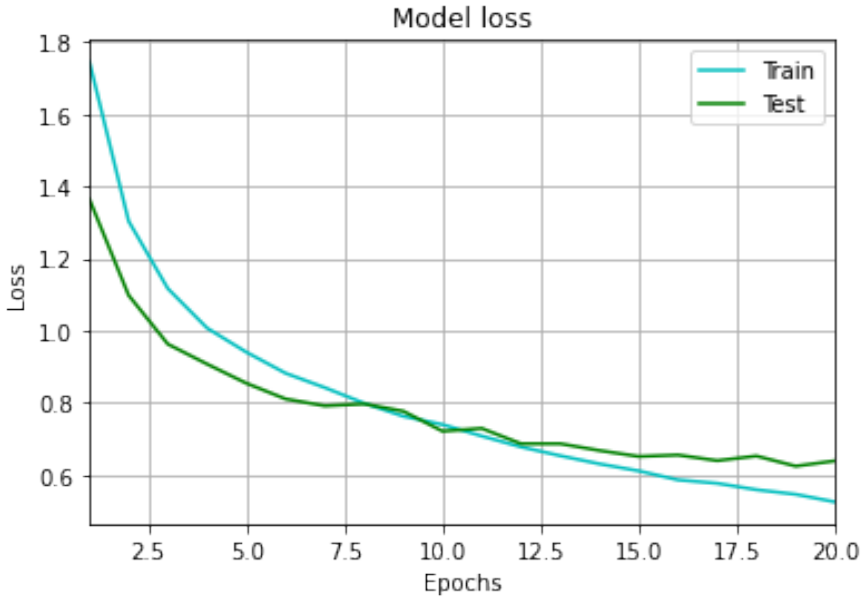
После обучения

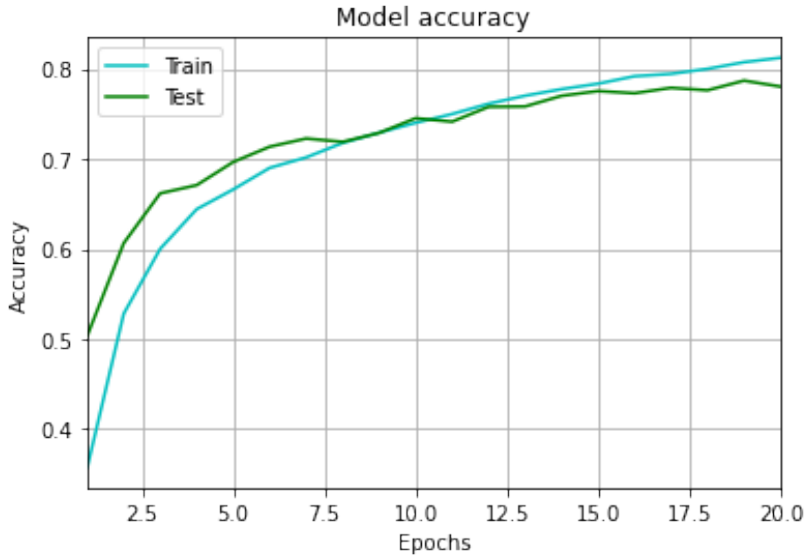
- точность: 0.7598999738693237
- потери: 0.9462826251983643

Без использования слоев Dropout, которые на каждом прогоне с определенной вероятностью «выключают» нейроны, переобучение наблюдается уже после 7 эпохи. Потери значительно выше, чем в сети со слоями Dropout, что показывает их эффективность для борьбы с переобучением.

Проведем тестирование модели со слоями Dropout и ядром свертки с размером 2x2.

Результаты тестирования приведены на графиках:





После обучения

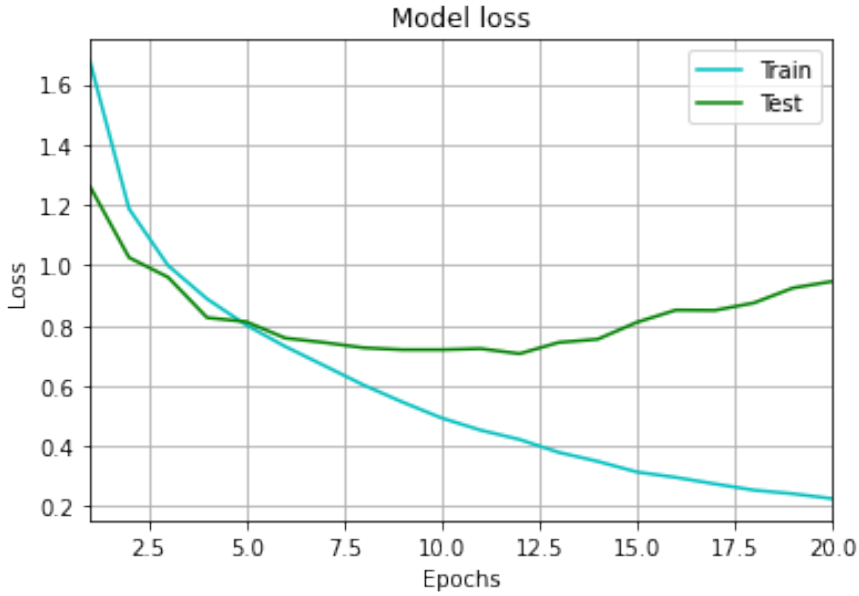
- точность: 0.7804999947547913
- потери: 0.6387689709663391

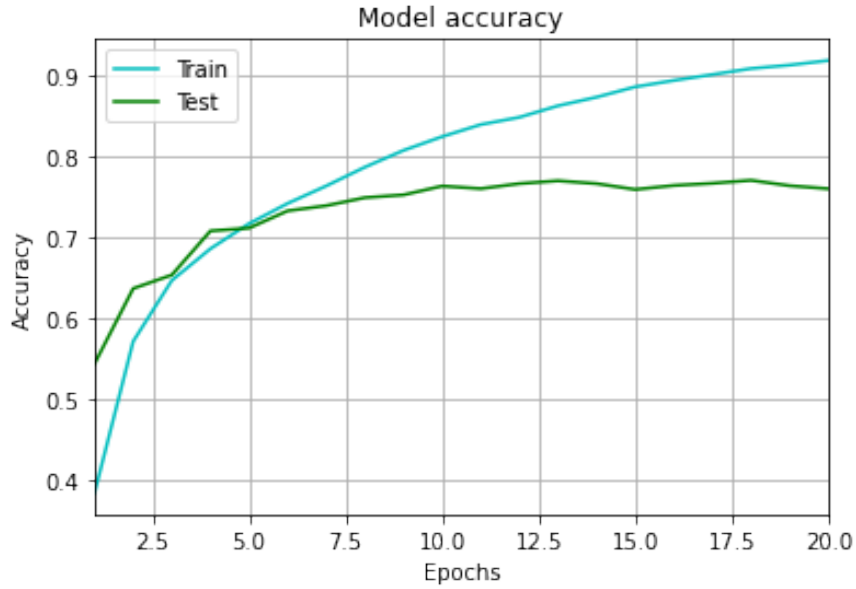
Переобучение не наблюдается. Точность оказалась ниже, чем при использовании ядра свертки 3x3 (если в обоих случаях используются слои Dropout). В

целом, обучение более медленное, чем в случае ядра свертки 3x3, но можно обучать более 20-ти эпох, так как нет переобучения. Скорее всего, ядро свертки 3x3 является более подходящим для данного датасета.

Проведем тестирование модели без слоев Dropout и ядром свертки с размером 2x2.

Результаты тестирования приведены на графиках:





После обучения

- точность: 0.7598999738693237
- потери: 0.9462826251983643

Наблюдается переобучение на 12 эпохе. Точность и потерь схожи с полученными в модели без Dropout и с ядром свертки 3x3.

Основные наблюдения:

- Применение слоя Dropout позволяет избежать быстрого переобучения модели, хоть и не исключает возможность его появления
- Модель с ядром свертки 3x3 обучается быстрее (за меньше количество эпох), и достигает более высокой точности, чем модель с ядром свертки 2x2

Выводы.

В ходе выполнения лабораторной работы было изучено решение задачи классификации цветных изображений с помощью сверточной нейронной сети. Было изучено влияние слоев Dropout на показатели сети во время обучения, а также влияние размера ядра свертки слоев Conv2D.

ПРИЛОЖЕНИЕ А

Исходный код программы. Файл main.py

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

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

pool_size = 2
kernel_size = 2
num_epochs = 20
batch_size = 100
drop_prob_1 = 0.2
drop_prob_2 = 0.5
conv_depth_1 = 32
conv_depth_2 = 64
hidden_size = 512
conv_depth_3 = 128
dropout = False

num_train, width, height, depth = 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 /= 255.0
X_test /= 255.0

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

inp = Input(shape=(width, height, depth))

conv_1 = Convolution2D(conv_depth_1, (kernel_size, kernel_size), padding='same',
strides=(1, 1), 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',
strides=(1, 1), activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), padding='same',
strides=(1, 1), 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
conv_5 = Convolution2D(conv_depth_3, (kernel_size, kernel_size), padding='same',
strides=(1, 1), activation='relu')(drop_2)
conv_6 = Convolution2D(conv_depth_3, (kernel_size, kernel_size), padding='same',
strides=(1, 1), activation='relu')(conv_5)
pool_3 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_6)
if (dropout):
    drop_3 = Dropout(drop_prob_1)(pool_3)
else:
    drop_3 = pool_3
flat = Flatten()(drop_3)
hidden = Dense(hidden_size, activation='relu')(flat)
drop_4 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_4)
model = Model(inp, out)

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

hist = model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs,
verbose=1, validation_data=(X_test, Y_test))

result = model.evaluate(X_test, Y_test, verbose=0)
print('[Loss, Accuracy]: ', result)

x = range(1, num_epochs+1)
history_dict = hist.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'c', label='Train')
plt.plot(epochs, val_loss_values, 'g', label='Test')
plt.title('Model loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.xlim(x[0], x[-1])
```

```
plt.legend()
plt.grid()
plt.show()

plt.clf()
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(epochs, acc_values, 'c', label='Train')
plt.plot(epochs, val_acc_values, 'g', label='Test')
plt.title('Model accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.xlim(x[0], x[-1])
plt.legend()
plt.grid()
plt.show()
```