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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание объектов на фотографиях»

Студент гр. 8382 Ершов М.И.
Преподаватель Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.

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).

Порядок выполнения работы.

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

Требования.

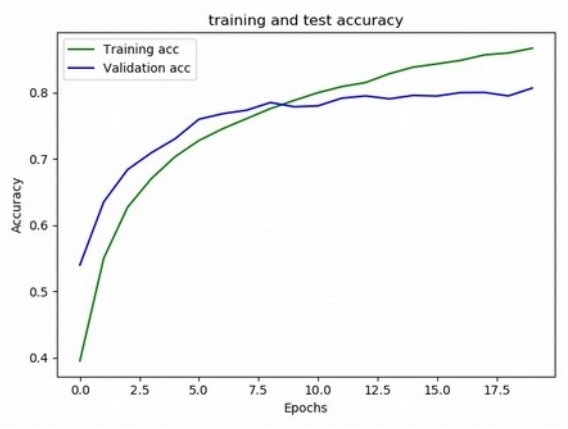
- Построить и обучить сверточную нейронную сеть
- Исследовать работу сеть без слоя Dropout
- Исследовать работу сети при разных размерах ядра свертки

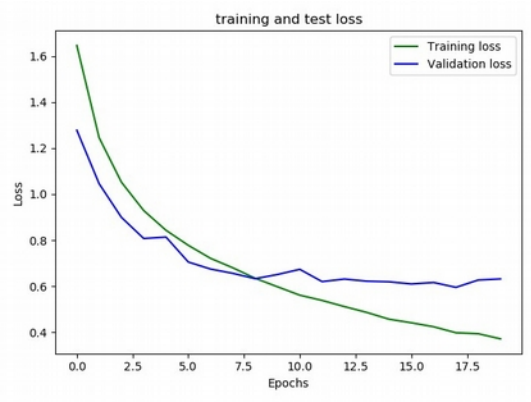
Ход работы.

Набор данных СИФАР-10 состоит из 60'000 цветных рисунков следующих десяти классов: самолеты, легковые автомобили, птицы, кошки, олени, собаки, лягушки, лошади, корабли, грузовики; размер каждого образа – 32*32 пикселей. В обучающую выборку входят 50'000 рисунков, а 10'000 – в тестовую.

1. Была построена сверточная нейронная сеть, использующая слои maxpooling и dropout со следующей архитектурой:

- Оптимизатор – adam
- `batch_size=128`
- `loss='categorical_crossentropy'`
- `epochs=20` Точность ~80%



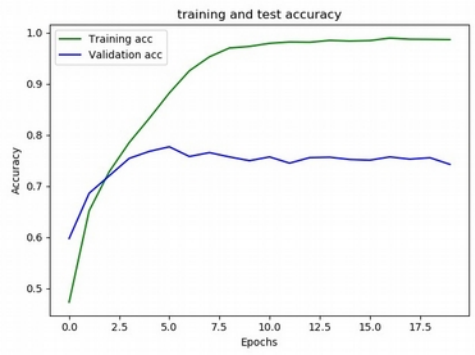


а б

Рисунок 1 – Графики точности и потерь данной архитектуры 2.

Исследуем работу сети без слоя Dropout:

а б



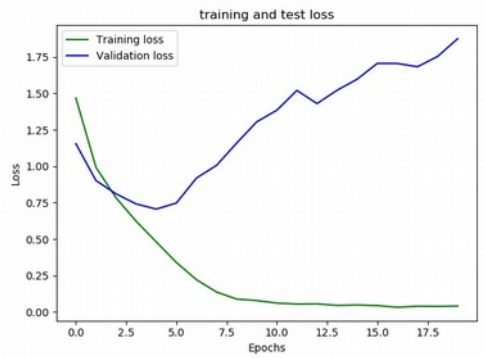
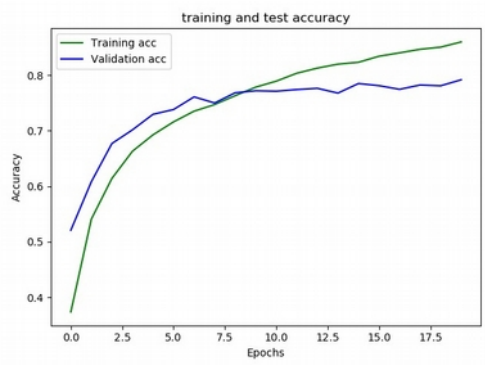


Рисунок 2 – Графики точности и потерь без слоев разреживания.

Видим, что наблюдается переобучение после ~5 эпохи. Dropout как раз используется для решения этой проблемы путем случайного исключения нейронов во время итераций.

3. Исследуем работу сети при разных размерах ядра свертки. Рассмотрим размеры 5x5 и 7x7:

а б



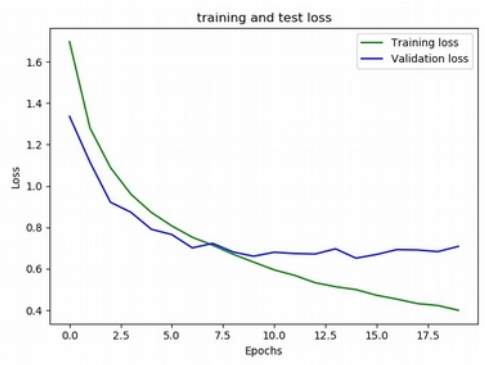


Рисунок 3 – Графики точности и потерь с размером ядра 5x5.

а б

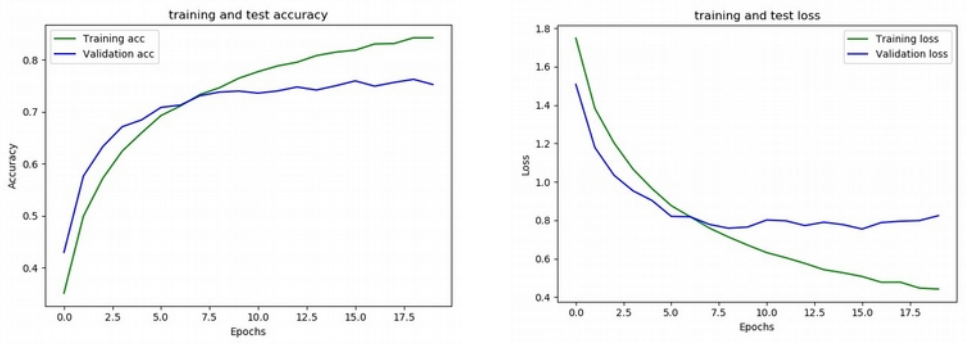


Рисунок 4 – Графики точности и потерь с размером ядра 7x7. Как видим, при увеличении размера ядра переобучение начинает возникать немного раньше, а точность уменьшается.

Выводы.

В ходе выполнения данной работы ознакомились со сверточными нейронными сетями и на их основе получили представление о распознавании объектов на фотографиях. Было исследовано влияние слоя разреживания и размера ядра свертки на нейронную сеть.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

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

batch_size = 128
num_epochs = 20
kernel_size = 3
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512

(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)


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

conv_1 = Convolution2D(conv_depth_1, (kernel_size, kernel_size), padding='same', acti-
vation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1, (kernel_size, kernel_size), padding='same', acti-
vation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)

conv_3 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), padding='same', acti-
vation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), padding='same', acti-
vation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
drop_2 = Dropout(drop_prob_1)(pool_2)

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

```
drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)

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

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs, ver-
bose=1, validation_split=0.1)
history_dict = history.history
model.evaluate(X_test, Y_test, verbose=1)

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'g', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.clf()
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'g', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```