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. 8383 _____ Сосновский Д. Н.

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).

Задачи.

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

Ход работы.

Согласно методическим указаниям была построена и обучена сверточная нейронная сеть. Исходный код программы приведён в приложении А. Из-за не очень мощного вычислительного железа было принято решение сократить количество эпох с 200 до 20 и batch_size с 32 до 70. При этих и остальных стандартных настройках были получены следующие результаты:

Потери: 253.6493

Точность: 0.5251

Графики точности и потерь приведены на рисунке 1.

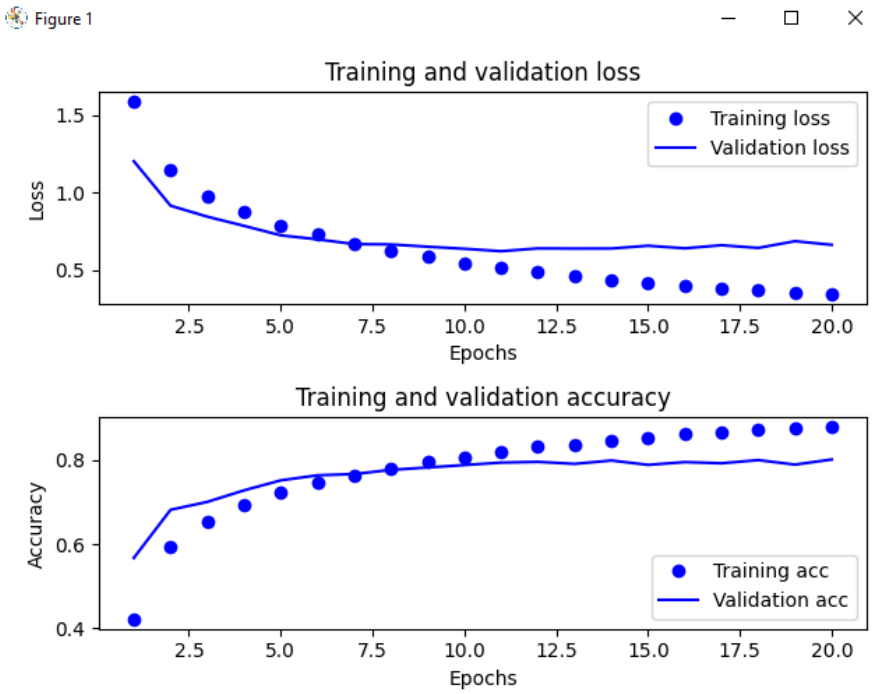

Рисунок 1 – график точности и потерь при первом запуске

Далее были убраны слои Dropout и заново запущена и обучена нейросеть с теми же параметрами. Получены следующие результаты:

Потери: 522.6720

Точность: 0.6019

Графики точности и потерь приведены на рисунке 2.

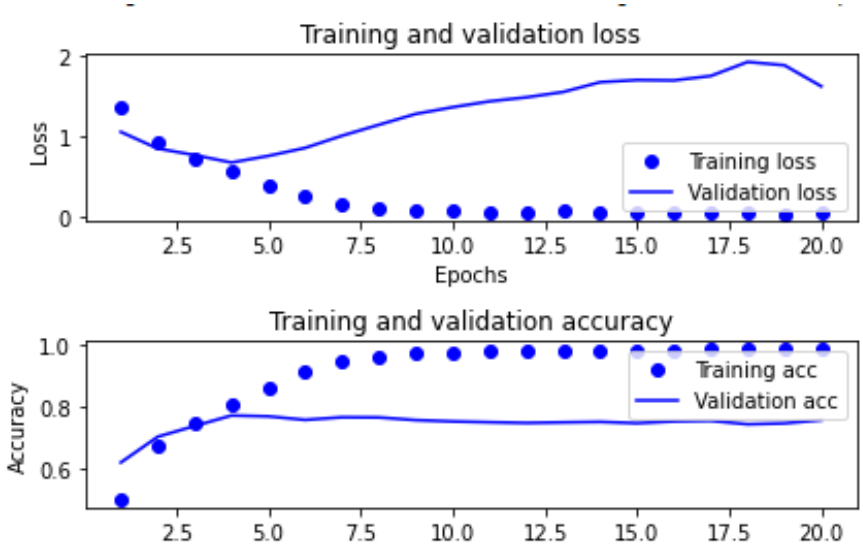

Рисунок 2 – график точности и потерь при втором запуске

Далее тестируем поведение нейросети при разных размерностях ядер свёртки. При следующем запуске указываем kernel_size = 5, что означает, что будет использована матрица свёртки размера 5x5. Полученные результаты:

Потери: 178.1600

Точность: 0.6304

Графики точности и потерь приведены на рисунке 3.

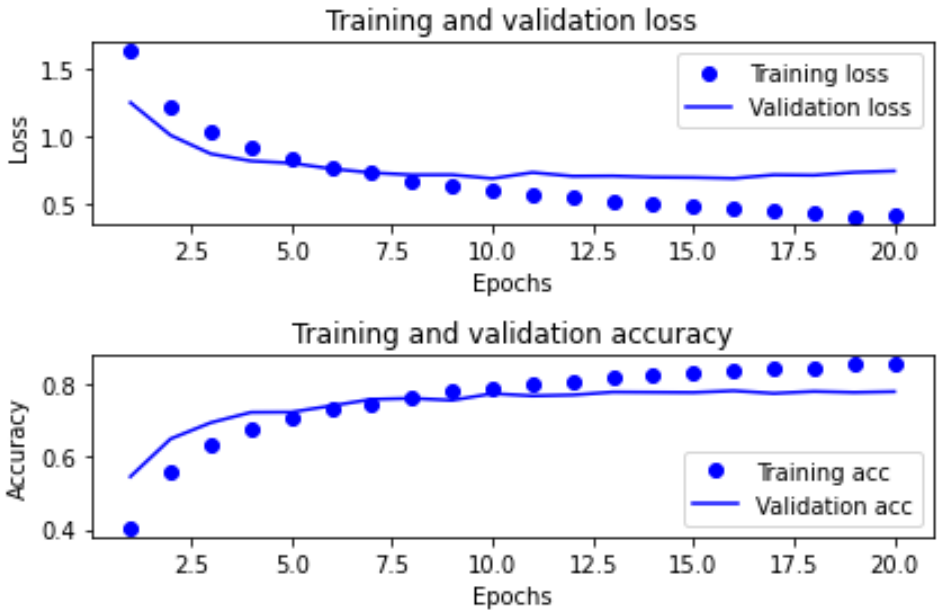

Рисунок 3 – график точности и потерь при третьем запуске

Попробуем изменить kernel_size на 2, тем самым перейдя к матрице 2x2. Полученные результаты:

Потери: 529.0072

Точность: 0.3314

Графики точности и потерь приведены на рисунке 4.

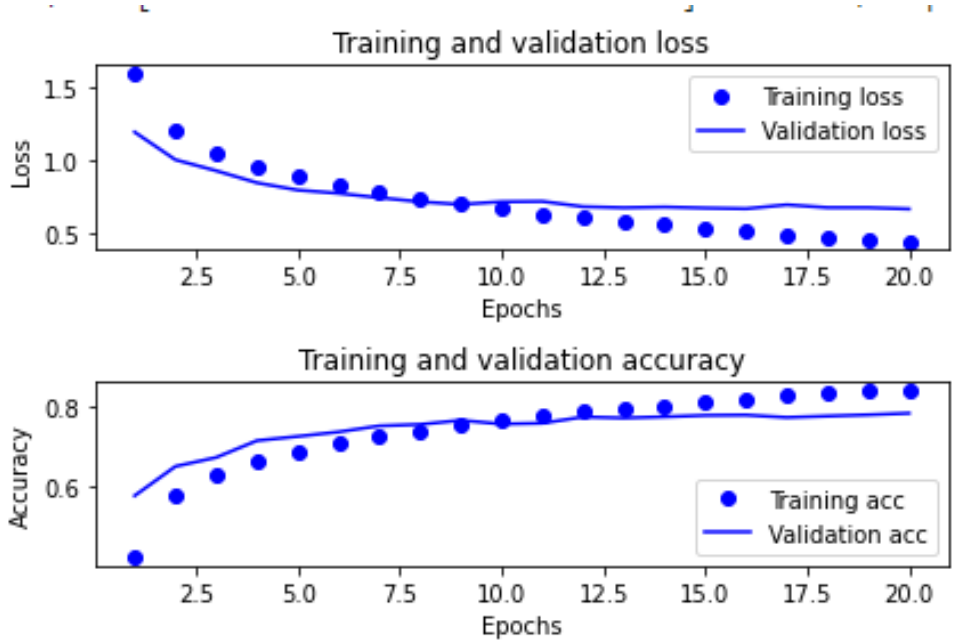

Рисунок 4 – график точности и потерь при четвёртом запуске

Исходя из графиков видно, что при отсутствии dropout слоя сеть очень быстро приходит к переобучению, из-за очень большого количества признаков. Изменения размерностей матрицы свёртки привели к незначительным ухудшениям точности модели. Оптимальным из всех запусков является размер ядра матрицы свертки kernel_size = 3.

Вывод.

В ходе работы была реализована сверточная нейронная сеть для распознавания объектов на фотографиях. Были проведены исследования зависимости работы модели от наличия слоя Dropout и от разных размерностей ядра свертки.

ПРИЛОЖЕНИЕ А

В этом приложении приведён исходный код программы.

```
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 import utils

# Глобальные параметры
batch_size = 70
num_epochs = 20
kernel_size = 3 # we will use 3x3 kernels throughout
pool_size = 2 # we will use 2x2 pooling throughout
conv_depth_1 = 32 # we will initially have 32 kernels per conv. layer...
conv_depth_2 = 64 # ...switching to 64 after the first pooling layer
drop_prob_1 = 0.25 # dropout after pooling with probability 0.25
drop_prob_2 = 0.5 # dropout in the dense layer with probability 0.5
hidden_size = 512 # the dense layer will have 512 neurons

# Данные для сети
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, height, width, 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 /= np.max(X_train)
X_test /= np.max(X_train)
Y_train = utils.np_utils.to_categorical(y_train, num_classes)
Y_test = utils.np_utils.to_categorical(y_test, num_classes)

# Входной слой
inp = Input(shape=(height, width, depth))

# Слои Свертки и пуллинга
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)

# Слой Dropout
# drop_1 = Dropout(drop_prob_1)(pool_1)

# Еще слои свертки и пуллинга
conv_3 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), padding='same',
activation='relu')(pool_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)

# Еще слой Dropout
# drop_2 = Dropout(drop_prob_1)(pool_2)


flat = Flatten()(pool_2)
hidden = Dense(hidden_size, activation='relu')(flat)
```

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

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

H = model.fit(X_train, Y_train,
              batch_size=batch_size, epochs=num_epochs,
              verbose=1, validation_split=0.1)


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

#Получение ошибки и точности в процессе обучения
loss = H.history['loss']
val_loss = H.history['val_loss']
acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
epochs = range(1, len(loss) + 1)

#Построение графика ошибки
plt.subplot(2, 1, 1)
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

#Построение графика точности
plt.subplot(2, 1, 2)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```