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. 8383 _____ Кормщикова А.О.

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.

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

Ход работы

Были загружены обучающие и тестовые данные из набора CIFAR-10, данные были нормализованы, нормализация выполнена путем деления всех элементов на максимальное значение обучающего множества.

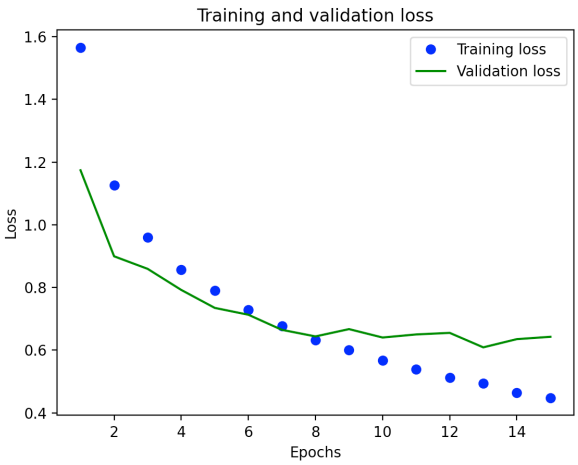
```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train) # Normalise data to [0, 1] range
X_test /= np.max(X_train) # Normalise data to [0, 1] range
Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)
```

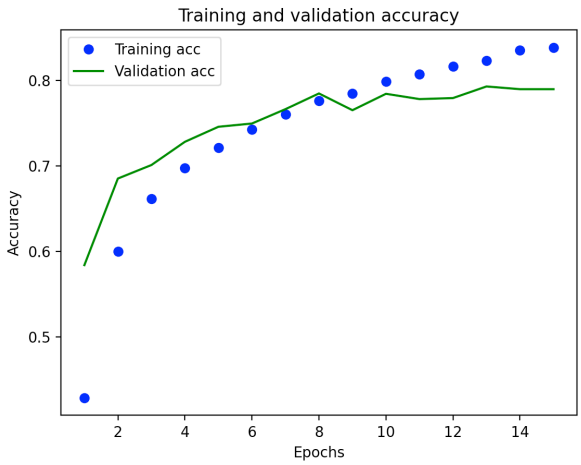
Была реализована сверточная нейронная сеть, состоящая из четырех слоев Convolution_2D и слоев MaxPooling2D, после второй и четвертой сверток. После выходное изображение слоя подвыборки трансформируется в одномерный вектор и проходит два полносвязных слоя. Для регуляризация модели после каждого слоя применяется слой Dropout

```
inp = Input(shape=(depth, height, width))
# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling
layer)
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)
drop_1 = Dropout(drop_prob_1)(pool_1)
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling
layer)
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)
drop_2 = Dropout(drop_prob_1)(pool_2)
# Now flatten to 1D, apply Dense -> ReLU (with dropout) ->
softmax
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)
```

Для обучение было выбрано 15 эпох. Графики точности и потерь для данной конфигурации:

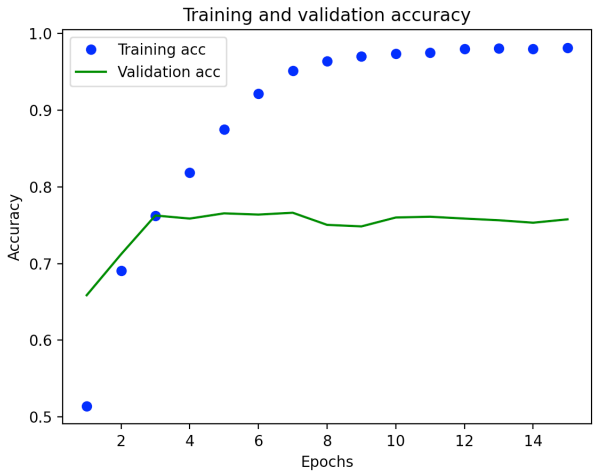




Полученная точность:

Train accuracy	Validation accuracy	Test accuracy
0.8413	0.7898	0.5683

Были удалены слои Dropout:

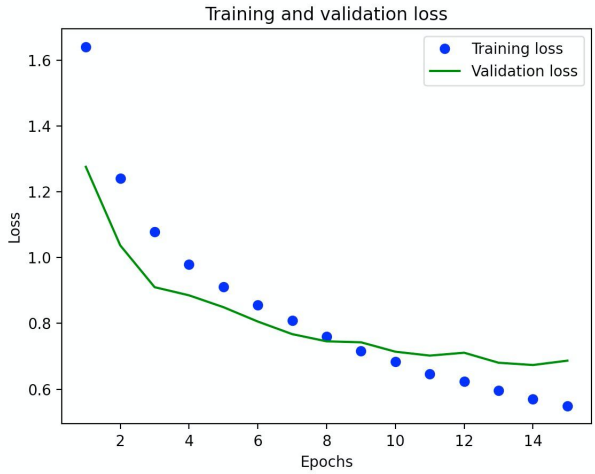


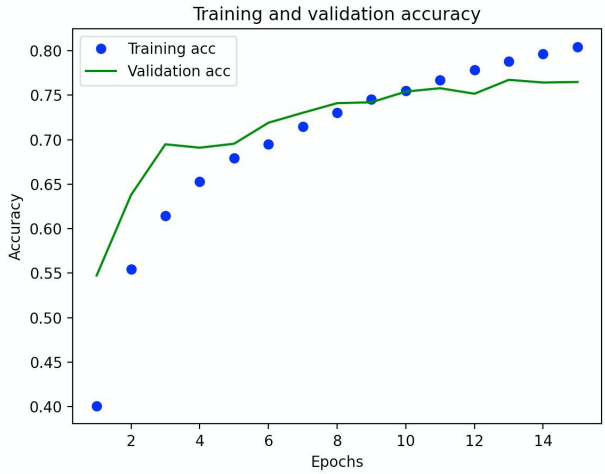
Полученная точность:

Train accuracy	Validation accuracy	Test accuracy
0.9825	0.7576	0.6665

Из графиков можно сделать вывод что модель переобучается после 4 эпохи - точность на тренировочных данных растет и вырастает, но не изменяется на проверочных данных. По сравнению с предыдущей моделью точность на тренировочных данных увеличилась с 84% до 98%, однако проверочные данных показывают худший результат, что говорит о том, что нейросеть работает хуже для новых данных.

Были возвращены слои Dropout, размер ядра был изменен на 2.



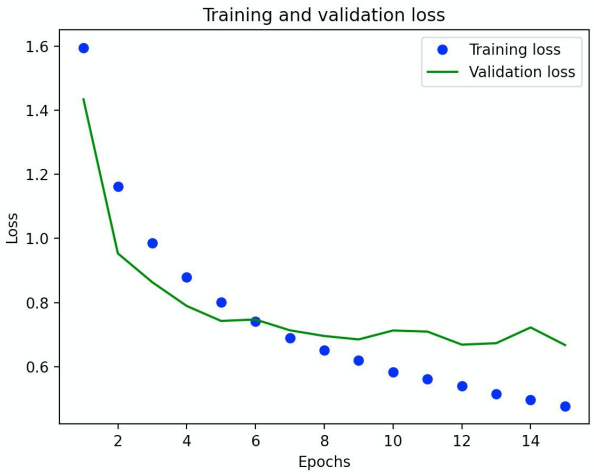


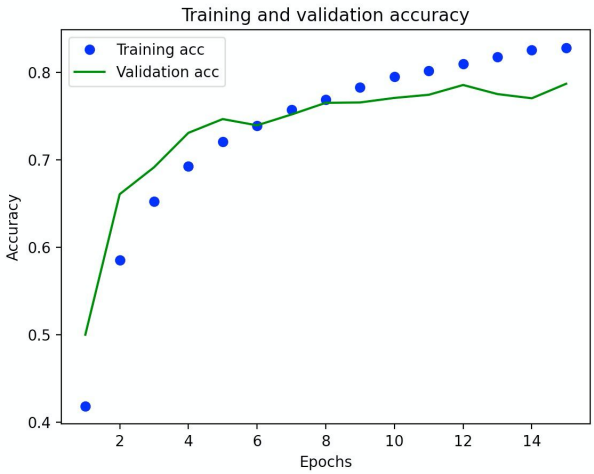
Полученная точность:

Train accuracy	Validation accuracy	Test accuracy
0.8083	0.7648	0.3919

Точность на тестовых данных значительно упала, на тренировочных и проверочных показатели незначительно ухудшились

Также нейросеть была обучена с размером ядра 5:





Полученная точность:

Train accuracy	Validation accuracy	Test accuracy
0.8321	0.7872	0.5996

Увеличение размера не дало улучшений, однако время обучение значительно увеличилось.

Из полученных данных стоит сделать вывод, что наилучшим значением размера ядра свертки в данном случае является 3x3.

Выводы.

Во время выполнения лабораторной работы была реализована сверточная нейронная сеть для распознавания объектов на фотографиях из CIFAR-10. Изучено построение модели в функциональном виде, исследована работа сети без слоя dropout, изучена работа сети при разных размерах ядра свертки.

ПРИЛОЖЕНИЕ А

```
import matplotlib.pyplot as plt  #импорт модуля для графиков
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 = 60 # in each iteration, we consider 32 training
examples at once
num_epochs = 15 # we iterate 200 times over the entire training set
kernel_size = 5 # 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
(X_train, y_train), (X_test, y_test) = cifar10.load_data() # fetch
CIFAR-10 data
num_train, depth, height, width = X_train.shape # there are 50000
training examples in CIFAR-10
num_test = X_test.shape[0] # there are 10000 test examples in
CIFAR-10
num_classes = np.unique(y_train).shape[0] # there are 10 image
classes
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train) # Normalise data to [0, 1] range
X_test /= np.max(X_train) # Normalise data to [0, 1] range
Y_train = np_utils.to_categorical(y_train, num_classes) # One-hot
encode the labels
Y_test = np_utils.to_categorical(y_test, num_classes) # One-hot
encode the labels
inp = Input(shape=(depth, height, width))
# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling
layer)
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)
drop_1 = Dropout(drop_prob_1)(pool_1)
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling
```

```
layer)
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)
drop_2 = Dropout(drop_prob_1)(pool_2)
# Now flatten to 1D, apply Dense -> ReLU (with dropout) -> softmax
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) # To define a model, just
specify its input and output layers
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

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

model.evaluate(X_test, Y_test, verbose=1) # Evaluate the trained
model on the test set!

# график ошибки
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()
# график точности
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()
```