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

Студент гр. 8383 _____ Федоров И.А.

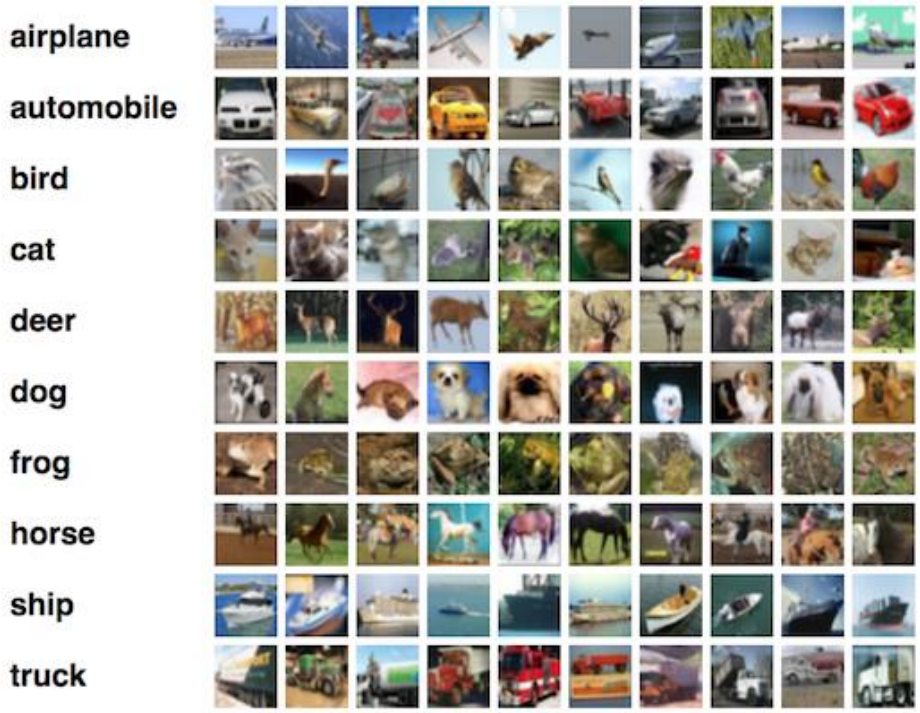
Преподаватель _____ Жангиров Т.Р.

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

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

Выполнение работы.

Были импортированы необходимые для работы класы, модули, функции, а также данные CIFAR-10, которые являются частью библиотеки *keras*.

```
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
```

Были загружены обучающие и тестовые данные, после чего они были преобразованы к виду, удобному для НС (путем нормализации и кодирования меток):

```
(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)
```

Была реализована глубокая сверточная нейронная сеть для классификации изображений из набора CIFAR-10. Сеть состоит из четырех слоев `Convolution_2D` и слоев `MaxPooling2D` после второй и четвертой сверток. В конце находится полносвязная классифицирующая сеть, состоящая из слоя `Flatten` для трансформации входных данных в одномерный вектор, и двух слоев `Dense`. Для регуляризации нашей модели после каждого слоя подвыборки и первого полносвязного слоя применяется слой `Dropout`.

```
inp = Input(shape=(height, width, depth))
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_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)
# классифицирующая часть
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)
```

На рисунке 1 показана схема модели.

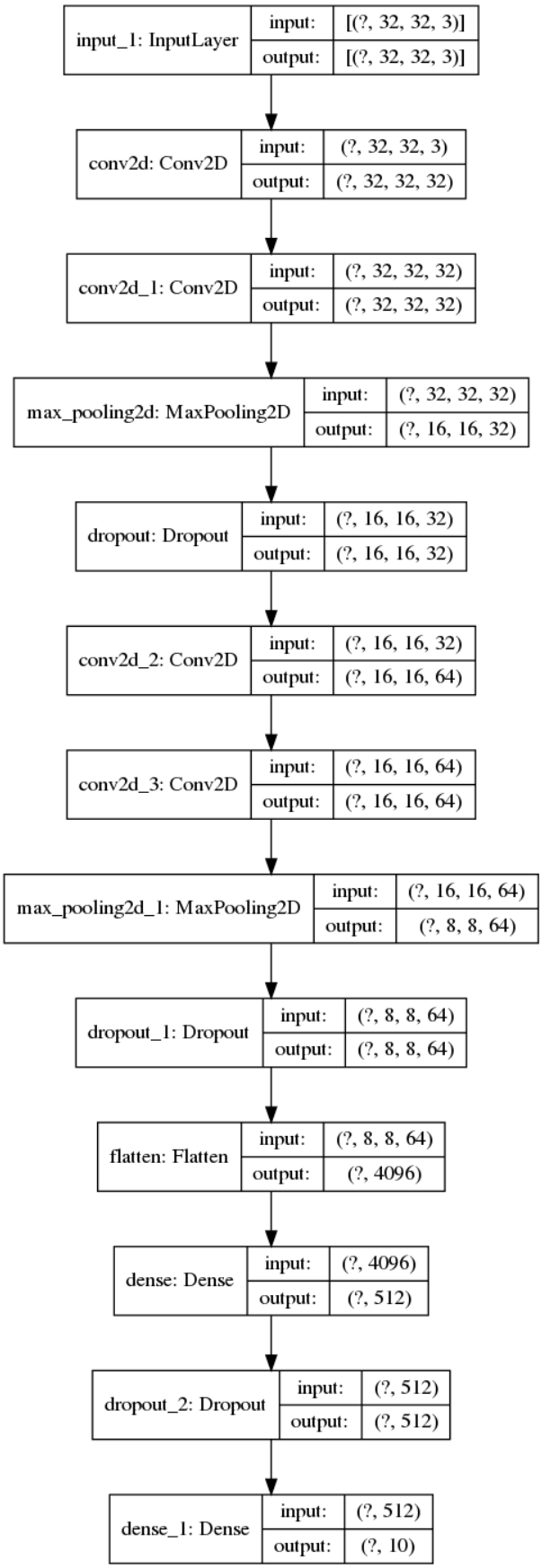


Рисунок 1 - Схема модели

Было проведеное обучение и проверка данной конфигурации сети. Число эпох было сокращено до 15. Графики потерь и точности показаны на рис. 2.

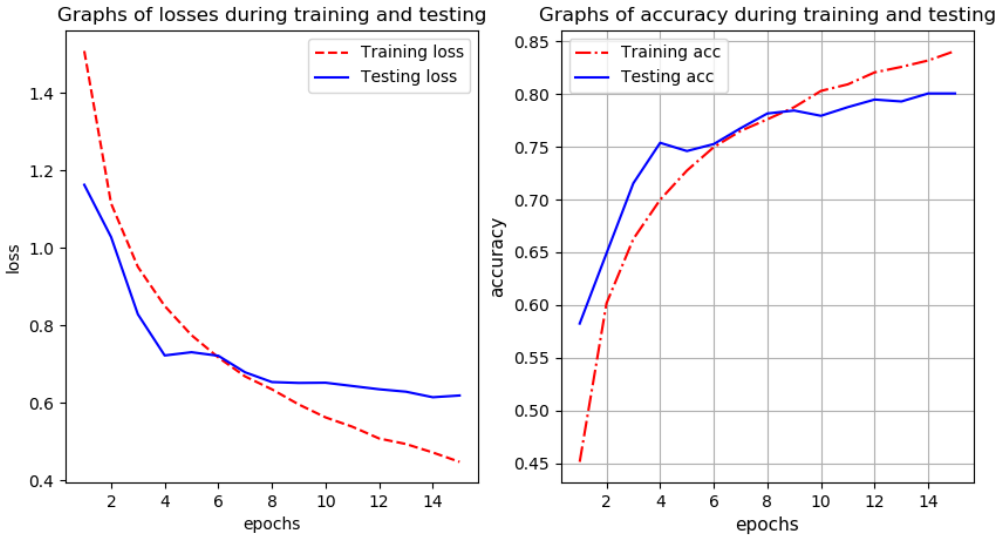


Рисунок 2

Результаты:

Train acc	Validation acc	Test acc
0.8535113	0.787624	0.6183532

Был проведен эксперемент с сетью, после удаления слоев `Dropout`. Графики потерь и точности показаны на рис. 3. Результаты приведены ниже.

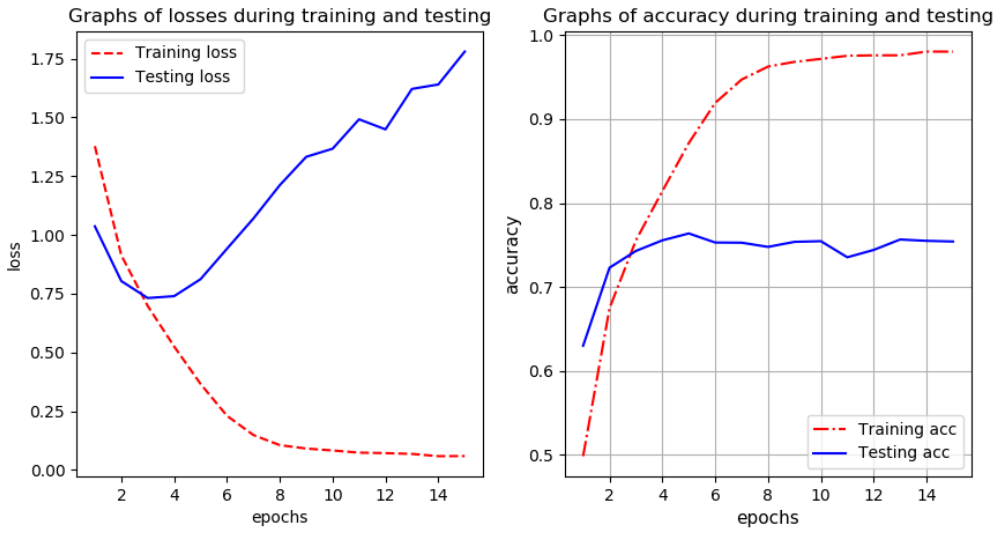


Рисунок 3

Результаты:

Train acc	Validation acc	Test acc
0.9804567	0.754231	0.5834374

Видно, что без слоев прореживания модель начинает переобучаться, точность на тренировочных данных резко возросла, в то время как точность на тестовых и проверочных данных упала.

Был проведен эксперимент с изначальной моделью сети, ядро свертки было увеличено до размера 5x5. График потерь и точности приведен на рис. 4. Результаты приведены ниже.

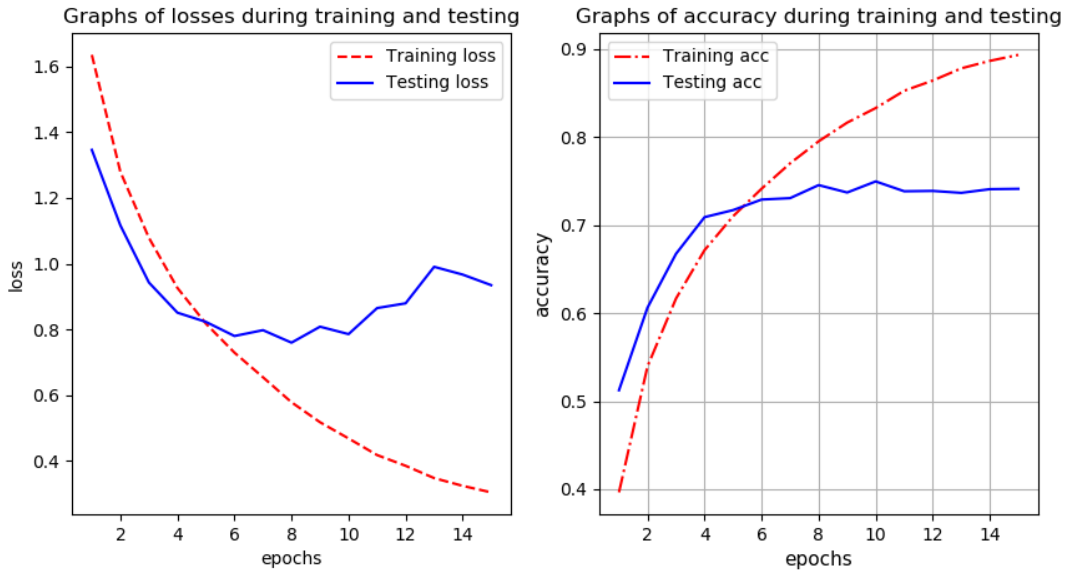


Рисунок 4

Результаты:

Train acc	Validation acc	Test acc
0.89420031	0.731233	0.569882

Увелечение размера ядра свертки негативно сказалось на качестве модели, модель начинает переобучаться, кроме этого упала точность на проверочных и тестовых данных.

Был проведен эксперимент с изначальной моделью сети, ядро свертки было увеличено до размера 8x8. График потерь и точности приведен на рис. 5. Результаты приведены ниже.

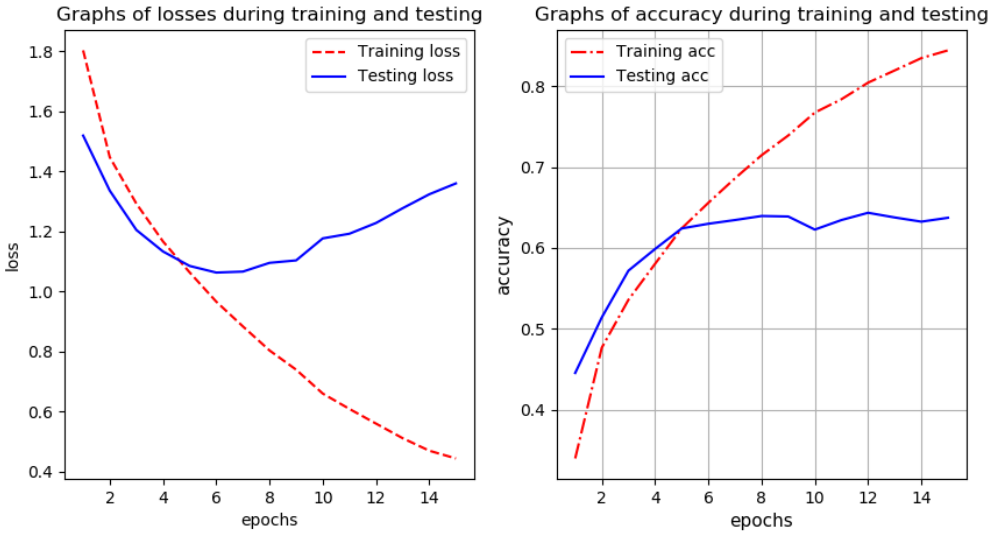


Рисунок 5

Результаты:

Train acc	Validation acc	Test acc
0.84432465	0.6374523	0.4345032

Результат аналогичен предыдущему.

Был проведен эксперимент с изначальной моделью сети, ядро свертки было уменьшено до размера 2x2. График потерь и точности приведен на рис. 6. Результаты приведены ниже.

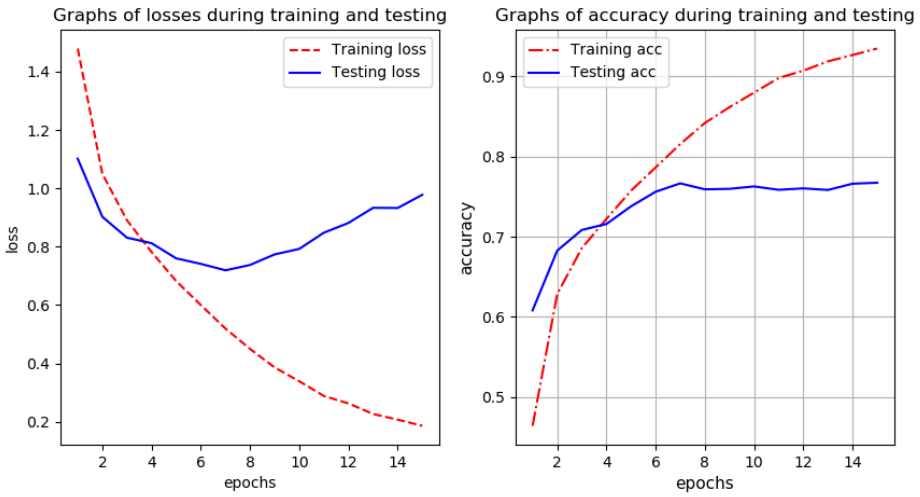


Рисунок 6

Результаты:

Train acc	Validation acc	Test acc
0.930247	0.767243	0.50071

Можно сделать вывод, что оптимальным размером ядра свертки в данном случае является 3x3.

Выводы.

В ходе выполнения данной работы была реализована ИНС для распознавания объектов на фотографиях CIFAR-10 (классификация небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик). Были проведены эксперементы с разными конфигурации сети (до и после удаления слоев Dropout, разные размеры ядра свертки). Были построены необходимые графики и приведены необходимые результаты.

Приложение

```
import numpy as np
import pandas
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

#from tensorflow.keras import utils

# графики потерь и точности при обучении и тестирования
def plot_model_loss_and_accuracy(history, figsize_=(10,5)):
    plt.figure(figsize=figsize_)
    train_loss = history.history['loss']
    test_loss = history.history['val_loss']
    train_acc = history.history['acc']
    test_acc = history.history['val_acc']
    epochs = range(1, len(train_loss) + 1)

    plt.subplot(121)
    plt.plot(epochs, train_loss, 'r--', label='Training loss')
    plt.plot(epochs, test_loss, 'b-', label='Testing loss')
    plt.title('Graphs of losses during training and testing')
    plt.xlabel('epochs')
    plt.ylabel('loss')
    plt.legend()
    plt.subplot(122)
    plt.plot(epochs, train_acc, 'r-.', label='Training acc')
    plt.plot(epochs, test_acc, 'b-', label='Testing acc')
    plt.title('Graphs of accuracy during training and testing')
    plt.xlabel('epochs', fontsize=11, color='black')
    plt.ylabel('accuracy', fontsize=11, color='black')
    plt.legend()
    plt.grid(True)

    plt.show()


batch_size = 50
num_epochs = 15
kernel_size = 3          # размер окна
pool_size = 2            # max-pooling
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25     # dropout
drop_prob_2 = 0.5      # dropout probability
hidden_size = 512


(X_train, y_train), (X_test, y_test) = cifar10.load_data()      # y_train = [
[3] [4] [2] ]
num_train, height, width, depth= X_train.shape    # 50000 training examples
num_test = X_test.shape[0]                        # 10000 test examples
num_classes = np.unique(y_train).shape[0]         # 10 image classes

X_train = X_train.astype('float32')           # подготовавливаем данные
X_test = X_test.astype('float32')
X_train /= np.max(X_train)                     # нормализуем на [0, 1]
X_test /= np.max(X_train)
```

```
Y_train = utils.np_utils.to_categorical(y_train, num_classes) # One-hot
кодирование
Y_test = utils.np_utils.to_categorical(y_test, num_classes)# [3] = [0 0 0 1 0
... ]

# создание модели
inp = Input(shape=(height, width, depth))
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_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)

# классифицирующая часть
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,
                            verbose=1, validation_split=0.1)


test_score = model.evaluate(X_test, Y_test, verbose=1)
print('Test result: ')
print(test_score)

# сохранение модели
model.save('8383_Fedorov_LR5.h5')
plot_model_loss_and_accuracy(history)
utils.plot_model(model, to_file='model.png', show_shapes=True)
```