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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание объектов на фотографиях»

Студентка гр. 7383 _____ Маркова А.В.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

Распознавание объектов на фотографиях (Object Recognition in Photographs).

Реализовать классификацию небольших изображений (32 × 32) по десяти классам: самолёт, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, судно и грузовик.

Порядок выполнения работы.

1. Ознакомиться со свёрточными нейронными сетями;
2. Изучить построение модели в Keras в функциональном виде;
3. Изучить работу слоя разреживания (Dropout).

Требования.

1. Построить и обучить свёрточную нейронную сеть;
2. Исследовать работу сети без слоя Dropout;
3. Исследовать работу сети при разных размерах ядра свёртки.

Ход работы.

Классификация – один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект – значит, указать номер (или наименование класса), к которому относится данный объект.

CIFAR-10 – набор данных для оценки моделей машинного обучения по задаче классификации объектов на фотографии.

Каждое цветное изображение представляет собой квадрат 32 на 32 пикселей. Стандартный набор данных используется для оценки и сравнения моделей, где 50000 изображений используется для обучения модели (тренировочный датасет), а отдельный набор из 10000 изображений используется для её проверки.

Была построена нейронная сеть, разработанный код представлен в приложении А.

Архитектура сети: оптимизатор «SGD» со скорость обучения = 0.1, batch_size = 100, nb_epoch = 20, функция потерь «categorical_crossentropy». Модель нейронной сети представлена на рис. 1.

```
# Создание модели
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)
drop_1 = Dropout(drop_prob_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)
drop_2 = Dropout(drop_prob_1)(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)
drop_3 = Dropout(drop_prob_1)(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)
```

Рисунок 1 – Модель нейронной сети.

Значения параметров: conv_depth_1 = 32, kernel_size = 3, pool_size = 2, drop_prob_1 = 0.2, conv_depth_2 = 64, conv_depth_3 = 128, hidden_size = 512, drop_prob_2 = 0.5.

Графики потери и точности обучения текущей архитектуры приведены на рис. 2 - 3 соответственно.

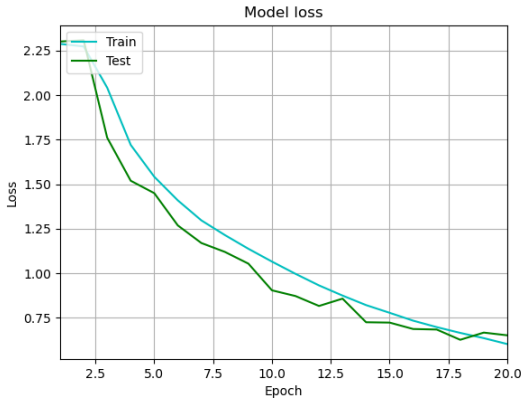

Рисунок 2 – График потери во время обучения модели.

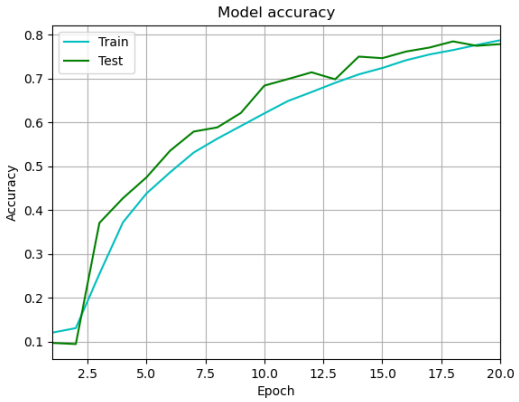

Рисунок 3 – График точности во время обучения модели.

Точность обучения модели равна 0.7626 при заданных параметрах архитектуры.

Исследуем работу сети без слоя разреживания (Dropout). Результаты показаны на рис. 4 - 5.

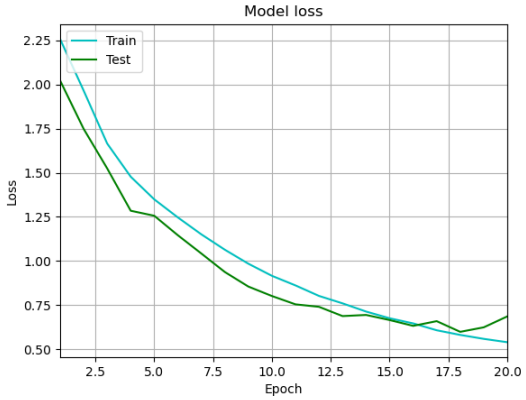

Рисунок 4 – График потери во время обучения модели без Dropout.

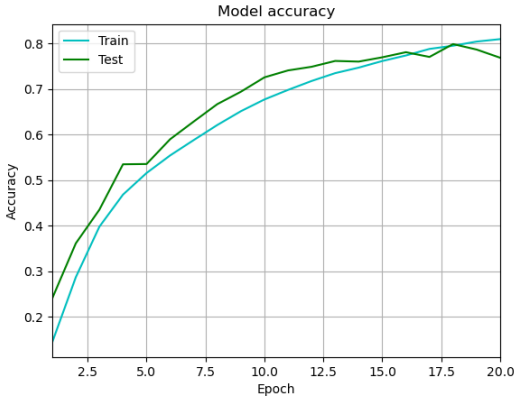

Рисунок 5 – График точности во время обучения модели без Dropout.

Из графиков видно, что примерно после 18 эпохи начинается переобучение. Точность немного понизилась (0.7589). Dropout – метод регуляризации искусственных нейронных сетей, предназначенный для уменьшения переобучения сети за счёт предотвращения сложных коадаптаций отдельных нейронов на тренировочных данных во время обучения. Для этого исключают определённый процент случайных нейронов на разных итерациях

во время обучения сети. Из всего вышесказанного следует, что использование слоя разреживания в данной задаче необходимо.

Исследуем работу сети при разных размерах ядра свёртки. Результаты обучения с ядрами размера 2 × 2, 5 × 5 и 8 × 8 продемонстрированы на рис. 6 – 11

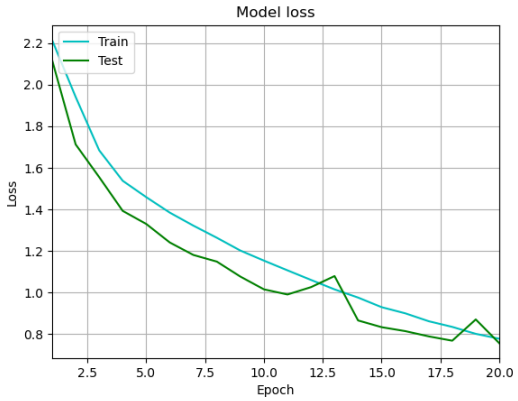

Рисунок 6 – График потери при размере ядра свёртки 2 × 2.

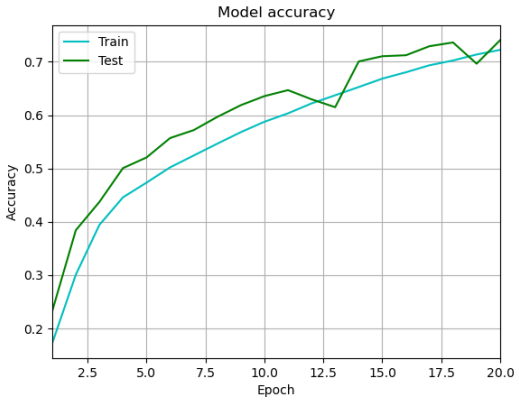

Рисунок 7 – График точности при размере ядра свёртки 2 × 2.

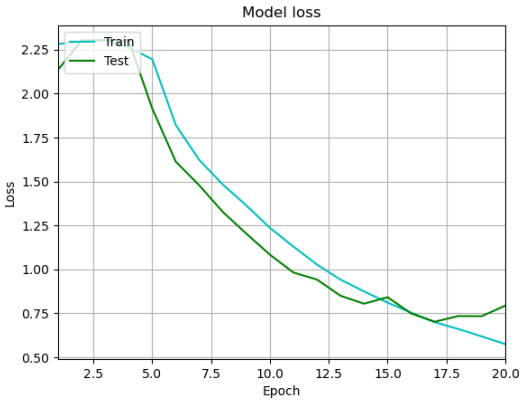

Рисунок 8 – График потери при размере ядра свёртки 5 × 5.

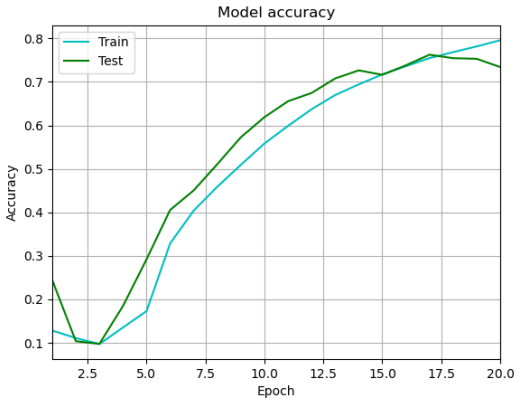

Рисунок 9 – График точности при размере ядра свёртки 5 × 5.

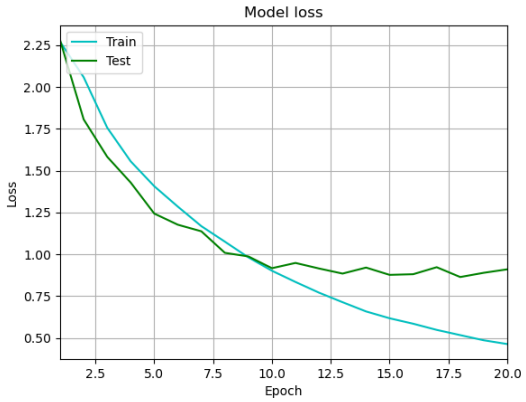

Рисунок 10 – График потери при размере ядра свёртки 8 × 8.

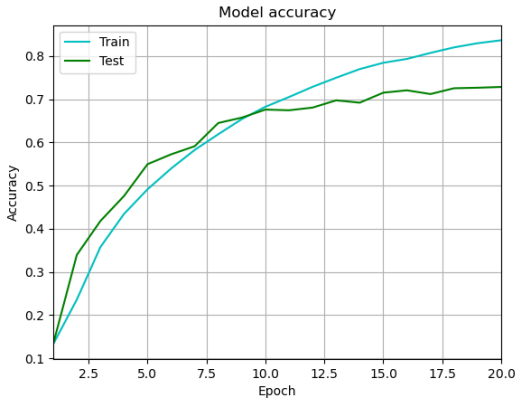

Рисунок 11 – График точности при размере ядра свёртки 8 × 8.

Из графиков видно, что с увеличением размера ядра свёртки при неизменных других параметрах сети переобучение возникает раньше, а точность падает (0.7218, 0.7197 и 0.709 соответственно).

Примеры тестовых изображений представлены на рис. 12.

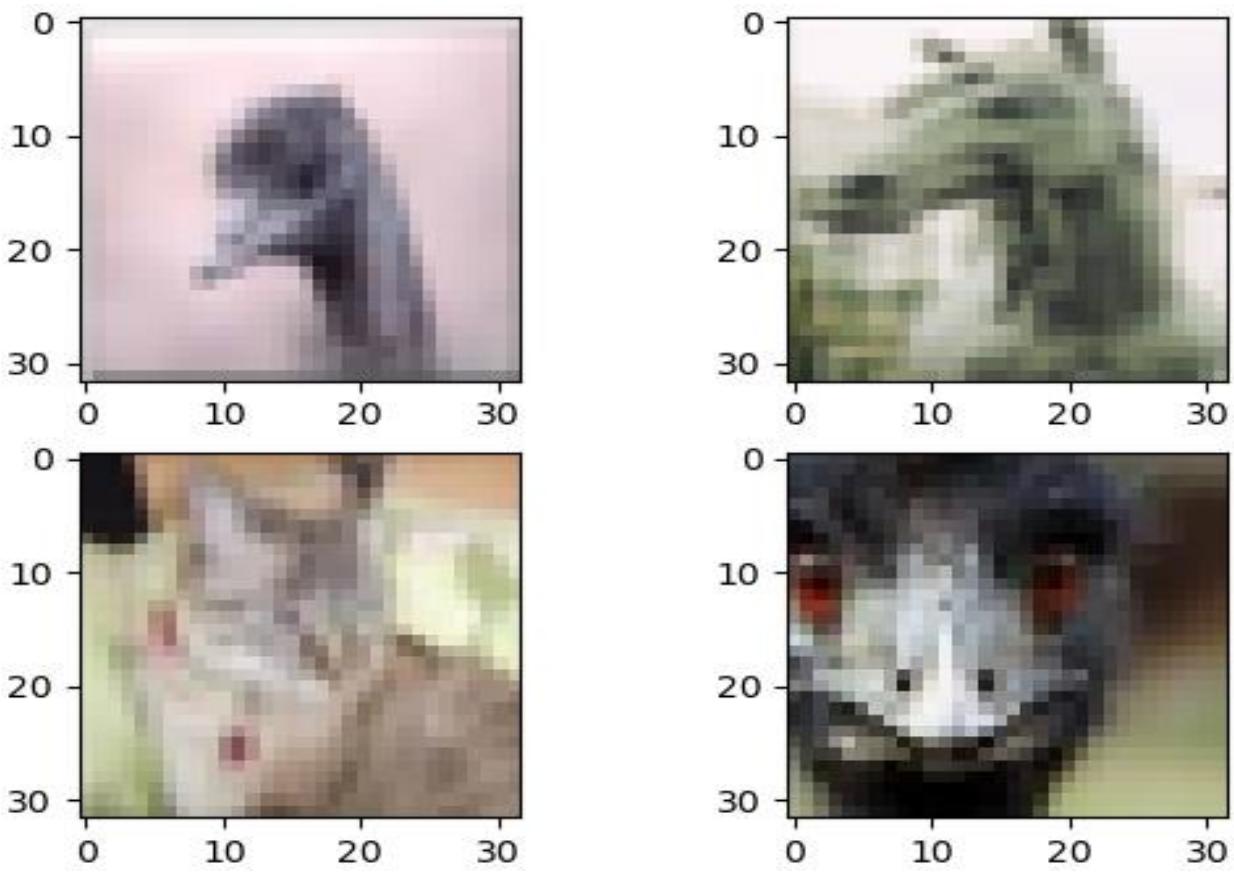

Рисунок 12 – Изображения для тестирования.

Выводы.

В ходе выполнения лабораторной работы было изучено представление и обработка графических данных, цветных изображений из базы данных CIFAR-10. Было изучено влияние слоя разреживания (Dropout) на результат обучения сети. Dropout увеличивает устойчивость сети к переобучению. Также было выявлено, что при изменении размера ядра свёртки необходимо корректировать параметры всей модели.

ПРИЛОЖЕНИЕ А

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

# Загрузка набора данных CIFAR-10, который содержит 60000 цветных
изображений 32x32
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# Вывод изображений для тестов
# plt.subplot(221)
# plt.imshow(X_test[84])
# plt.subplot(222)
# plt.imshow(X_test[145])
# plt.subplot(223)
# plt.imshow(X_test[8])
# plt.subplot(224)
# plt.imshow(X_test[35])
# plt.show()

pool_size = 2
kernel_size = 3
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

num_train, width, height, depth = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]

# Нормализация
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
```

```
      X_train /= 255.0
      X_test /= 255.0

      # Горячее кодирование значений класса, преобразовывая вектор
целых чисел класса в двоичную матрицу
      Y_train = to_categorical(y_train, num_classes)
      Y_test = to_categorical(y_test, num_classes)

      # Создание модели
      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)
      drop_1 = Dropout(drop_prob_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)
      drop_2 = Dropout(drop_prob_1)(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)
      drop_3 = Dropout(drop_prob_1)(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)

      # Инициализация параметров обучения
      optimizer = SGD(0.1)
      model.compile(loss='categorical_crossentropy',
optimizer=optimizer, metrics=['accuracy'])

      # Обучение сети
      hist = model.fit(X_train, Y_train, batch_size=batch_size,
epochs=num_epochs, verbose=1, validation_split=0.1)
```

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

# Построение графика ошибки
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()

# Построение графика точности
plt.clf()
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']
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()
```