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 фотографии»

Студентка гр. 7383 _____ Чемова К.А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель.

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).

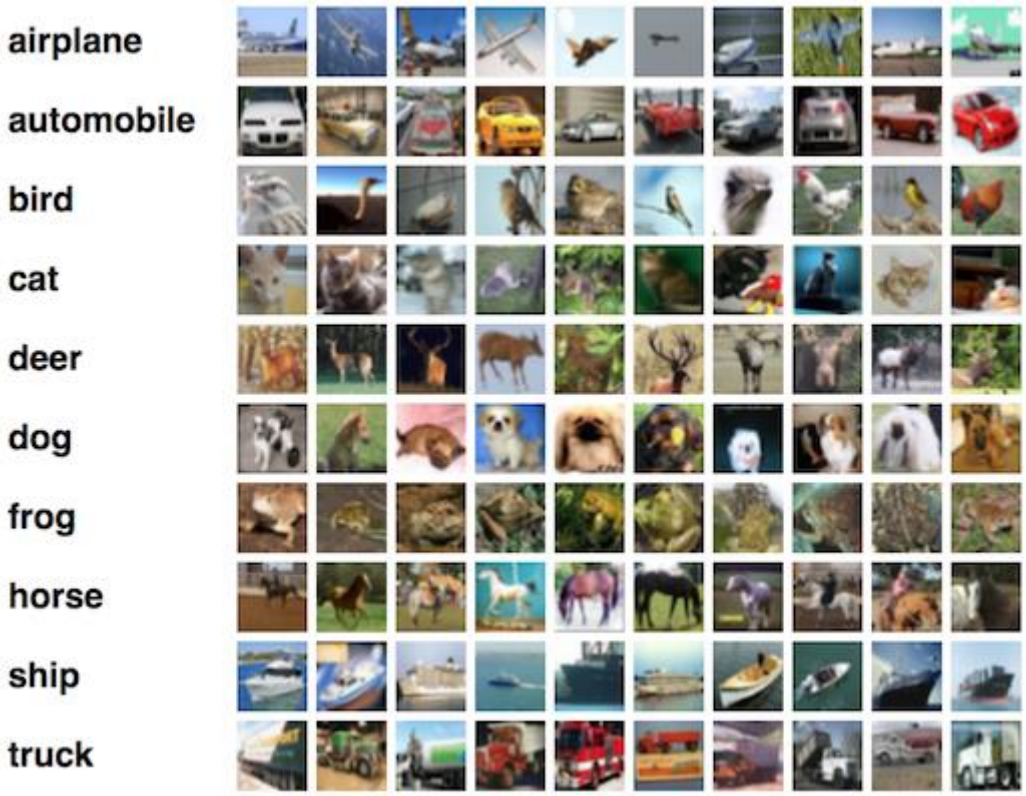

Рисунок 1 – Классификация объектов

Задачи.

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

Требования.

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.

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

CIFAR-10 содержит 32 x 32 x 3 пикселей. В наборе данных содержится 50000 обучающих изображений и 10000 для проверки.

После запуска модели были получены графики точности и ошибок. Графики представлены на рис. 2-3.

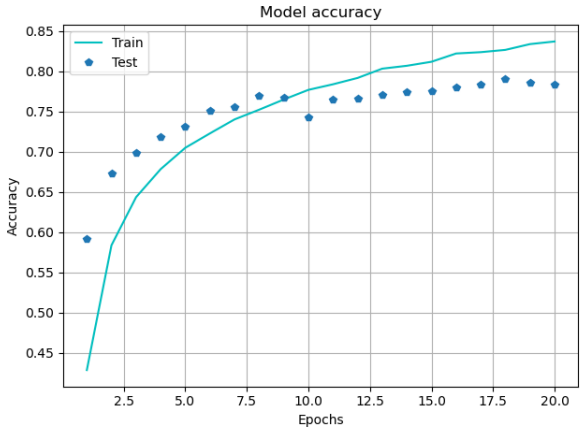

Рисунок 2 – График точности

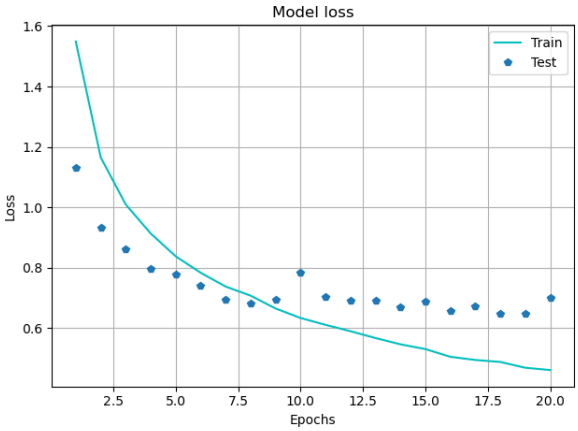

Рисунок 3 – График ошибки

Уберем слой Dropout и проверим работу сети. Графики точности и ошибок представлены на рис. 4-5.

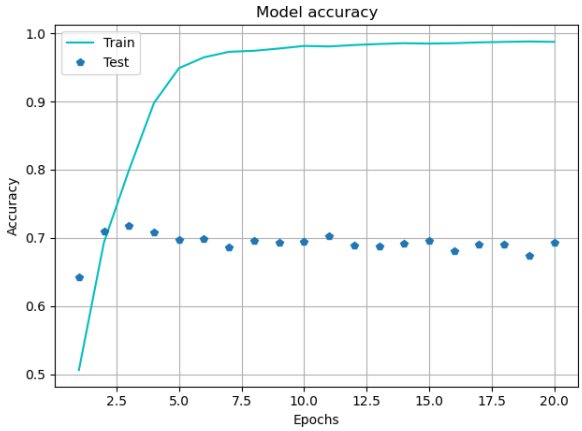

Рисунок 4 – График точности

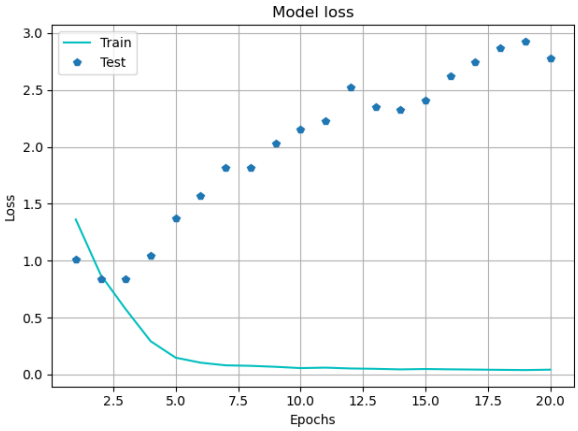

Рисунок 5 – График ошибки

Как видно из графиков точность упала, следовательно, применение слоя Dropout необходимо.

Исследуем работу сети с разными размерами ядра. Для ядра 5×5 графики представлены на рис. 6-7, для 7×7 на 8-9.

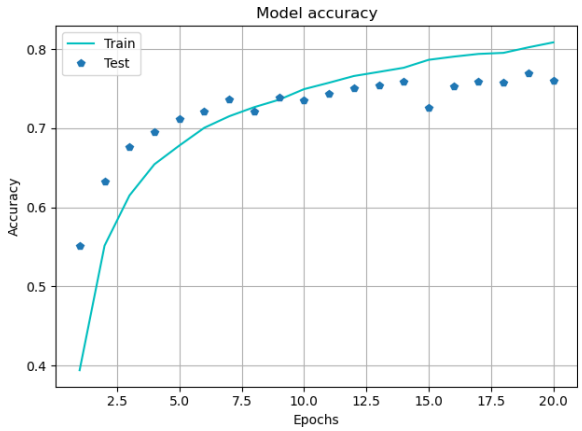

Рисунок 6 – График точности

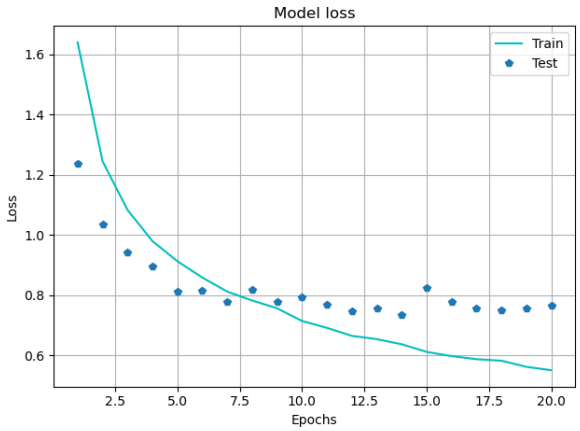

Рисунок 7 – График ошибки

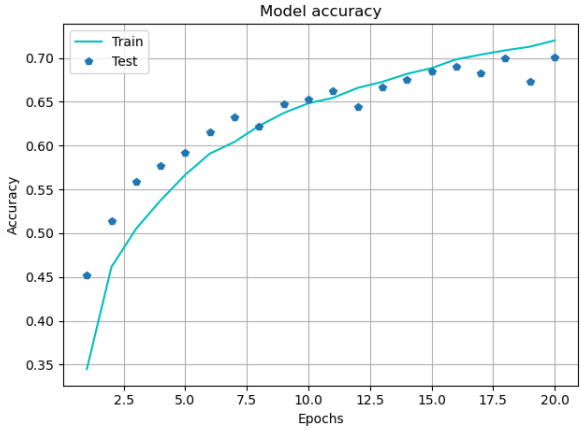

Рисунок 8 – График точности

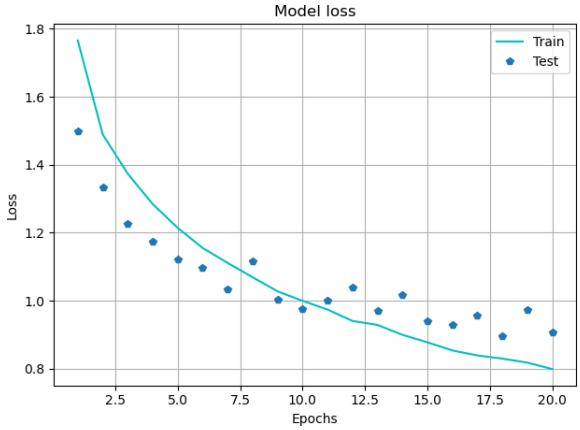

Рисунок 3 – График ошибки

Из графиков видно, что при увеличении ядра свертки переобучение начинается раньше, и точность вычислений падает.

Выводы.

В ходе выполнения лабораторной работы были изучены сверточная нейронная сеть и работа слоя Dropout. Была создана модель сверточной нейронной сети и изучена ее работа со слоем Dropout и без него. Были построены графики ошибок и точности в ходе обучения.

ПРИЛОЖЕНИЕ А

```
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 matplotlib.pyplot as plt
import numpy as np

batch_size = 32
num_epochs = 200
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, border_mode='same',
activation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size, border_mode='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, border_mode='same',
activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size, border_mode='same',
activation='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(input=inp, output=out)
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
history = model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=num_epochs,
verbose=1, validation_split=0.1)
model.evaluate(X_test, Y_test, verbose=1)

x = range(1, num_epochs+1)
history_dict = history.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, 'p', label='Test')
plt.title('Model loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.xlim(x[0], x[-1])
plt.legend()
plt.grid()
plt.show()
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, 'p', label='Test')
plt.title('Model accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.xlim(x[0], x[-1])
plt.legend()
plt.grid()
plt.show()
```