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

ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание объектов на фотографиях»

Студент гр. 7381 _____ Ильясов А.В.

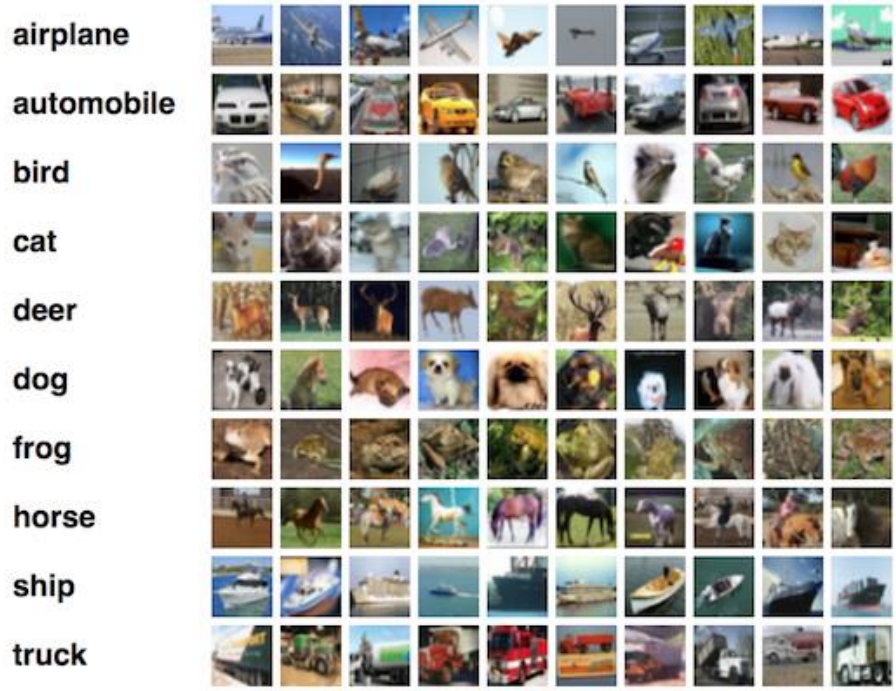
Преподаватель _____ Жукова Н. А.

Санкт-Петербург
2020

Цели.

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.

1) Построить и обучить сверточную нейронную сеть.

Исходная архитектура сети не была изменена, но количество эпох было уменьшено до 20, т.к. обучение на 200 эпохах заняло бы слишком большое количество часов, а после 20-ой эпохи точность и потери сети менялись на тысячные доли.

2) Исследовать работу сети без слоя Dropout.

Затем из модели были исключены слои dropout и также было проведено бучение на 20 эпохах. Результаты обучения с слоями Dropout и без них представлены на рисунках ниже.

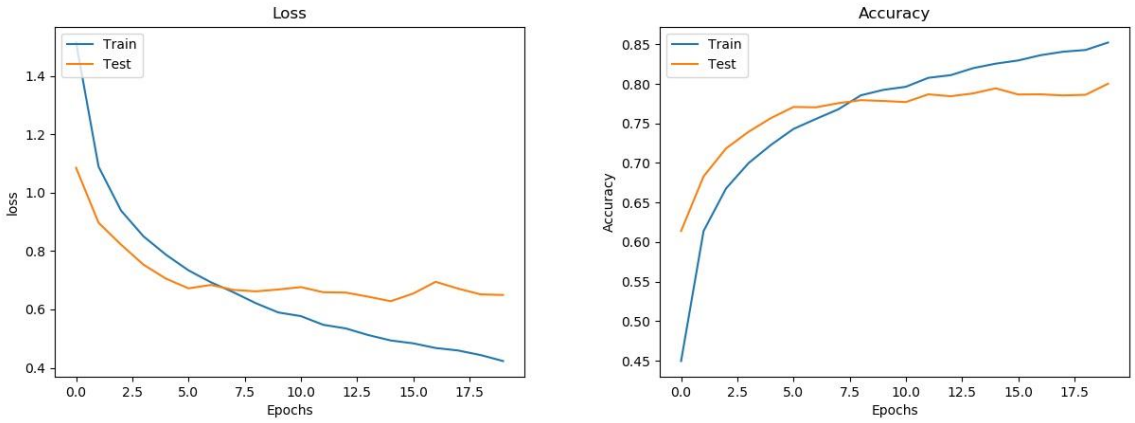


Рисунок 1 – графики функции потерь и точности модели с слоями Dropout

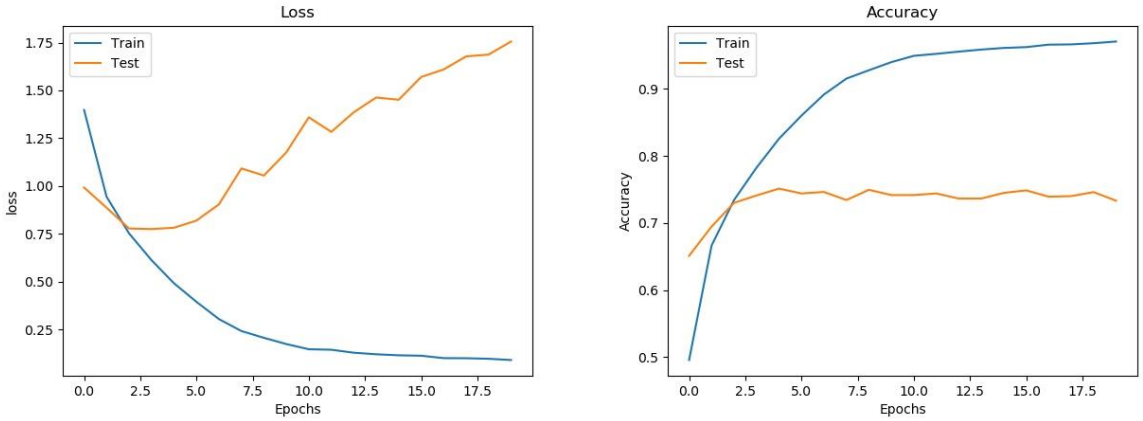


Рисунок 2 – графики функции потерь и точности модели без слоев Dropout

3) Исследовать работу сети при разных размерах ядра свертки.

Обычно используют ядра свертки с нечетными размерами матрицы, поэтому было обучено 3 модели с размерами 3, 5, 7 соотвественно. Результаты обучения для размера 3 показаны на рис. 1, а для размера ядра 5 и 7 представлены на рисунках ниже.

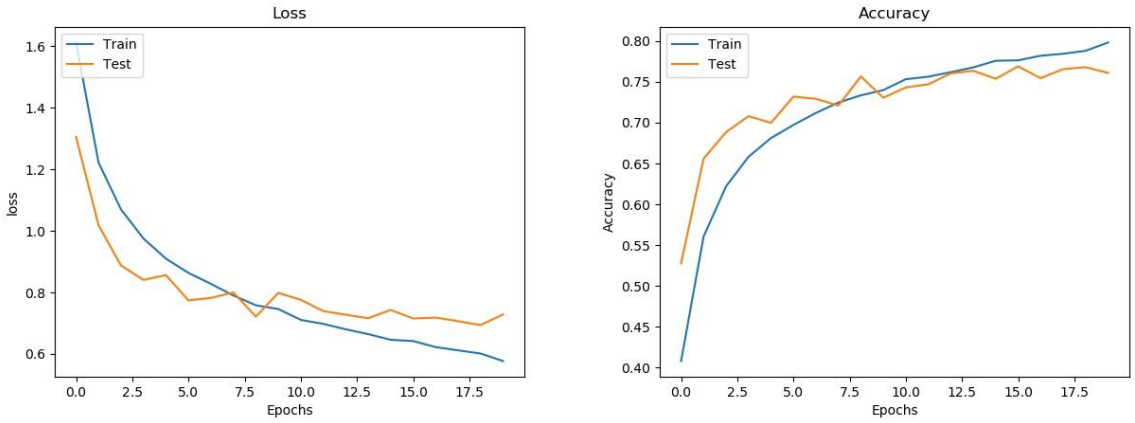


Рисунок 3 – графики функции потерь и точности модели с размером ядра 5

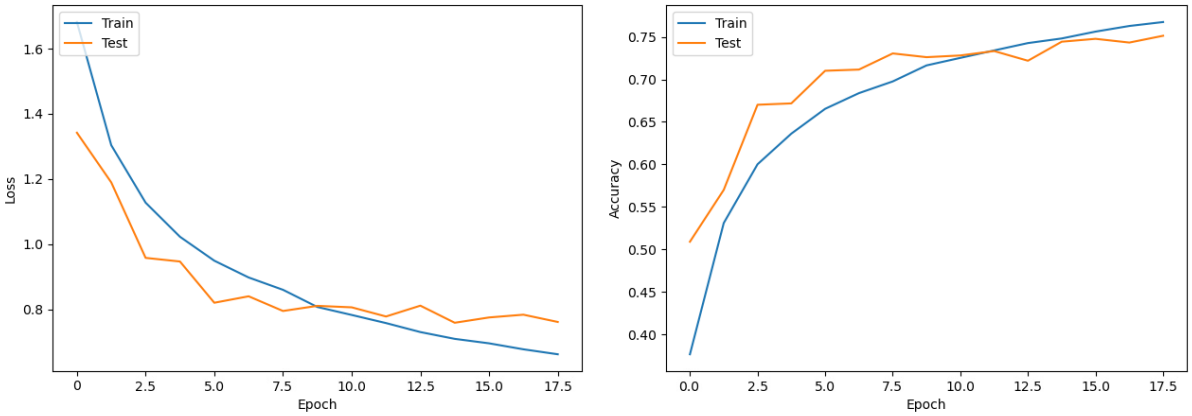


Рисунок 4 – графики функции потерь и точности модели с размером ядра 7

Из рисунков 1, 3 и 4 видно, что с увеличением размера ядра точность модели уменьшается.

Вывод.

В ходе выполнения данной работы было произведено ознакомление со сверточными нейронными сетями: изучено построение модели в Keras в функциональном виде и изучена работа слоя разреживания (Dropout).

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
from keras.layers import Dense
from keras.models import Sequential

from keras.datasets import boston_housing

import numpy as np
import matplotlib.pyplot as plt

import numpy as np
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import boston_housing


def build_model():
    model = Sequential()
    model.add(Dense(64, activation='relu',
input_shape=(train_data.shape[1],)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model


if __name__ == '__main__':
    (train_data, train_targets), (test_data, test_targets) =
boston_housing.load_data()

    mean = train_data.mean(axis=0)
    train_data -= mean
    std = train_data.std(axis=0)
    train_data /= std

    test_data -= mean
    test_data /= std

    k = 4
    num_val_samples = len(train_data) // k

    num_epochs = 70
    #all_scores = []
    mean_loss = []
    mean_mae = []
    mean_val_loss = []
    mean_val_mae = []
```

```
    for i in range(k):
        val_data = train_data[i * num_val_samples: (i + 1) *
num_val_samples]
        val_targets = train_targets[i * num_val_samples: (i + 1) *
num_val_samples]

        partial_train_data = np.concatenate([train_data[:i *
num_val_samples],
                                             train_data[(i + 1) *
num_val_samples:]], axis=0)
        partial_train_target = np.concatenate([train_targets[: i *
num_val_samples],
                                               train_targets[(i + 1)
* num_val_samples:]], axis=0)
        model = build_model()
        history = model.fit(partial_train_data,
partial_train_target, epochs=num_epochs, batch_size=1,
                            validation_data=(val_data, val_targets),
verbose=0)


mean_val_mae.append(history.history['val_mean_absolute_error'])
        mean_mae.append(history.history['mean_absolute_error'])

        plt.plot(history.history['mean_absolute_error'])
        plt.plot(history.history['val_mean_absolute_error'])
        title = 'Model accuracy' + ', i = ' + str(i+1)
        plt.title(title)
        plt.ylabel('mae')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Test'], loc='upper left')
        plt.show()

        mean_val_loss.append(history.history['val_loss'])
        mean_loss.append(history.history['loss'])

        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        title = 'Model loss' + ', i = ' + str(i+1)
        plt.title(title)
        plt.ylabel('loss')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Test'], loc='upper left')
        plt.show()

    plt.plot(np.mean(mean_mae, axis=0))
    plt.plot(np.mean(mean_val_mae, axis=0))
    title = 'Mean model mean absolute error'
    plt.title(title)
    plt.ylabel('mae')
```

```
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

plt.plot(np.mean(mean_loss, axis=0))
plt.plot(np.mean(mean_val_loss, axis=0))
title = 'Mean model loss'
plt.title(title)
plt.ylabel('loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```