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

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Искусственные нейронные сети»

Тема: «Регрессионная модель изменения цен на дома в Бостоне»

Студент гр. 7383 _____ Медведев И.С.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цели.

Реализовать предсказание медианной цены на дома в пригороде бостона в середине 1970-х по таким данным, как уровень преступности, ставка местного имущественного налога и т.д.

Задачи.

- Ознакомиться с задачей регрессии
- Изучить отличие задачи регрессии от задачи классификации
- Создать модель
- Настроить параметры обучения
- Обучить и оценить модели
- Ознакомиться с перекрестной проверкой

Выполнение работы.

Целью задачи классификации является предсказание одной дискретной метки для образца входных данных. Целью задачи регрессии является предсказание не дискретной метки, а значения на непрерывной числовой прямой: например, предсказание температуры воздуха.

1) Была создана и обучена модель искусственной нейронной сети в Keras, код представлен в приложении А.

2) При исследовании разных архитектур и обучение при различных параметрах обучения ИНС было изменено:

- Количество блоков: 4, 6, 8

Для выведения графиков зависимости mean_absolute_error от количества эпох и нахождения точки переобучения, была использована функция smooth_curve, которая каждую оценку заменяет экспоненциальным скользящим средним по предыдущим оценкам. Это сделано для того чтобы получить более гладкую кривую.

Изначально была рассмотрена модель с 4 блоками и со 150 эпохами.

Были выведены графики по всем 4 блокам. Графики представлены на рис. 1-4.

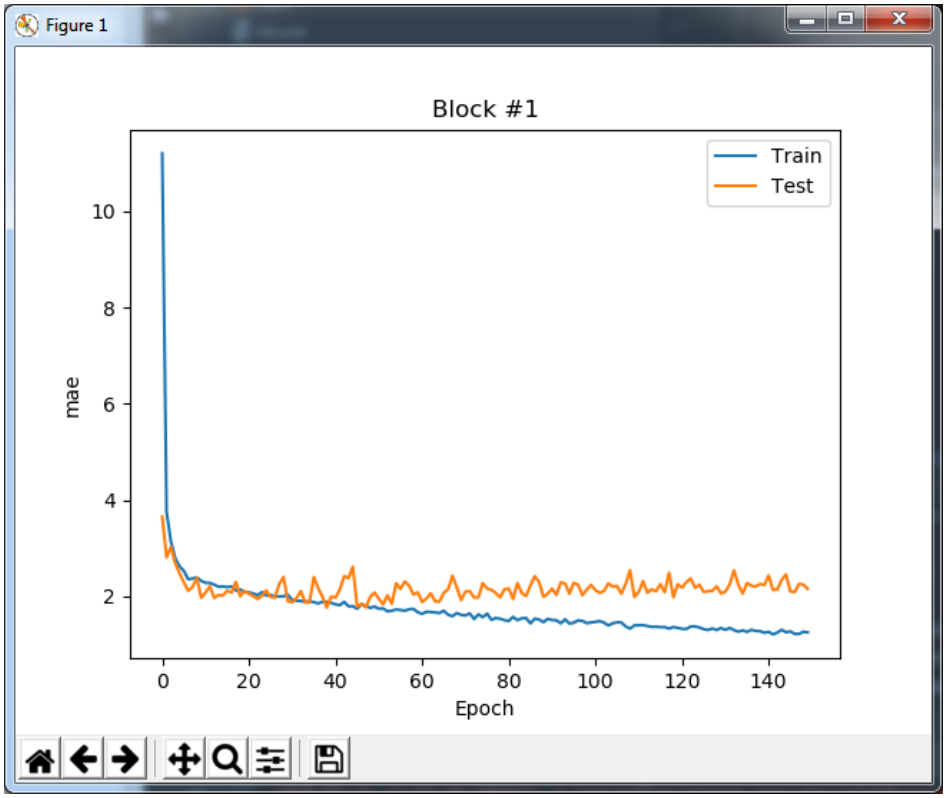

Рисунок 1 – График оценки MAE для 1 блока

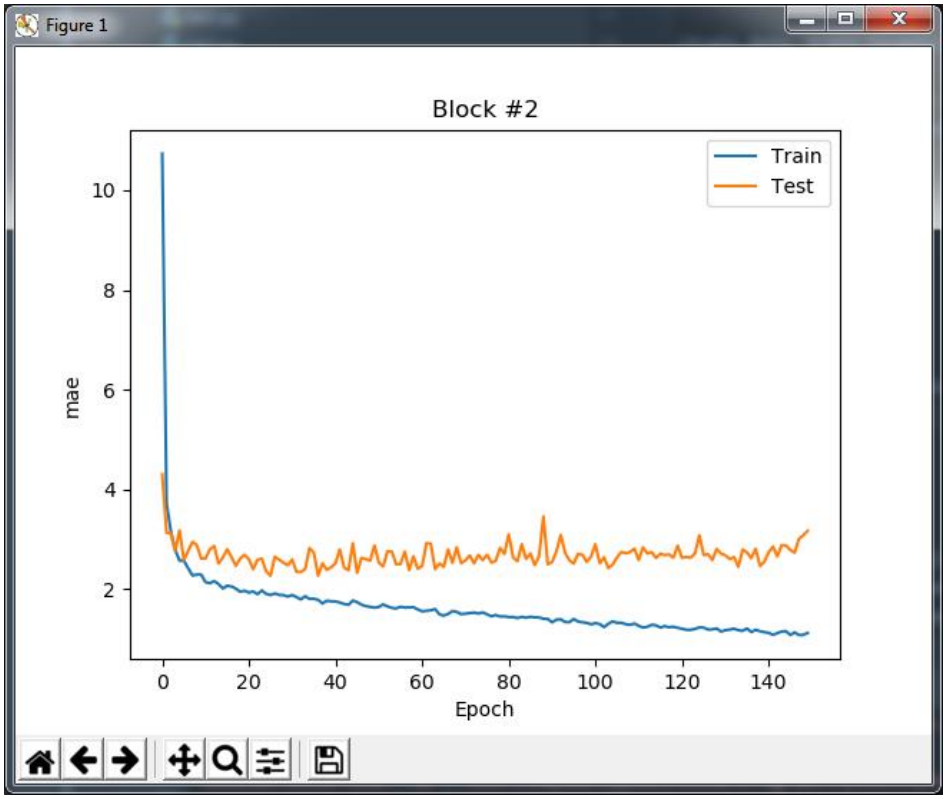

Рисунок 2 – График оценки MAE для 2 блока

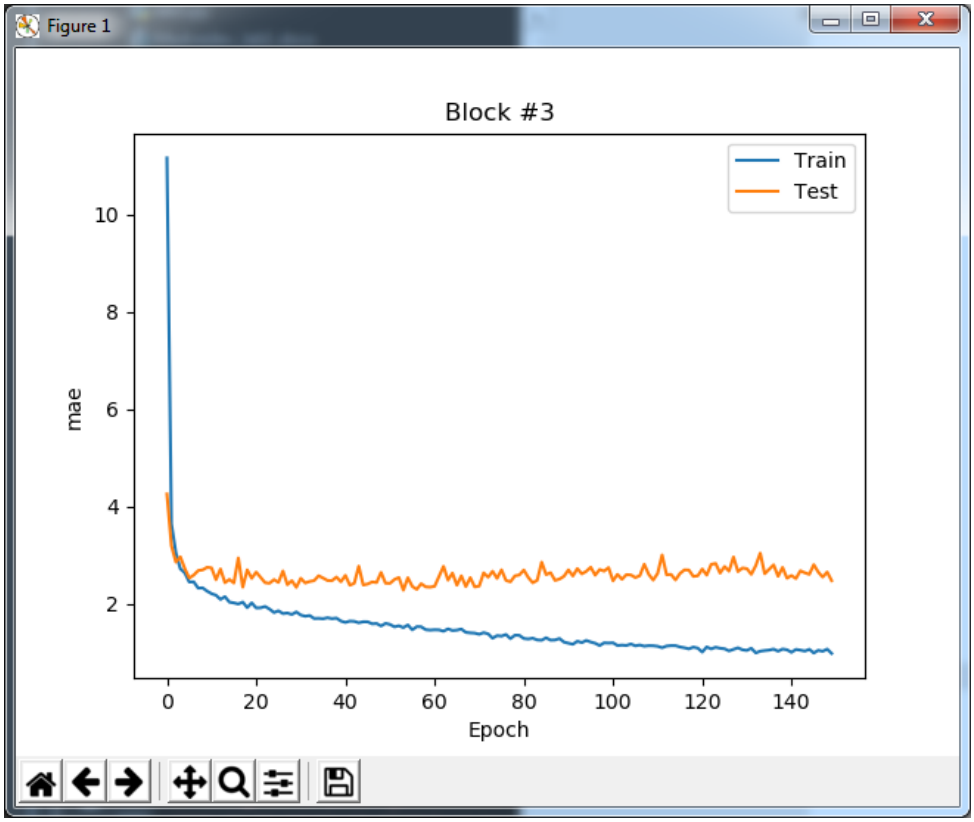

Рисунок 3 – График оценки MAE для 3 блока

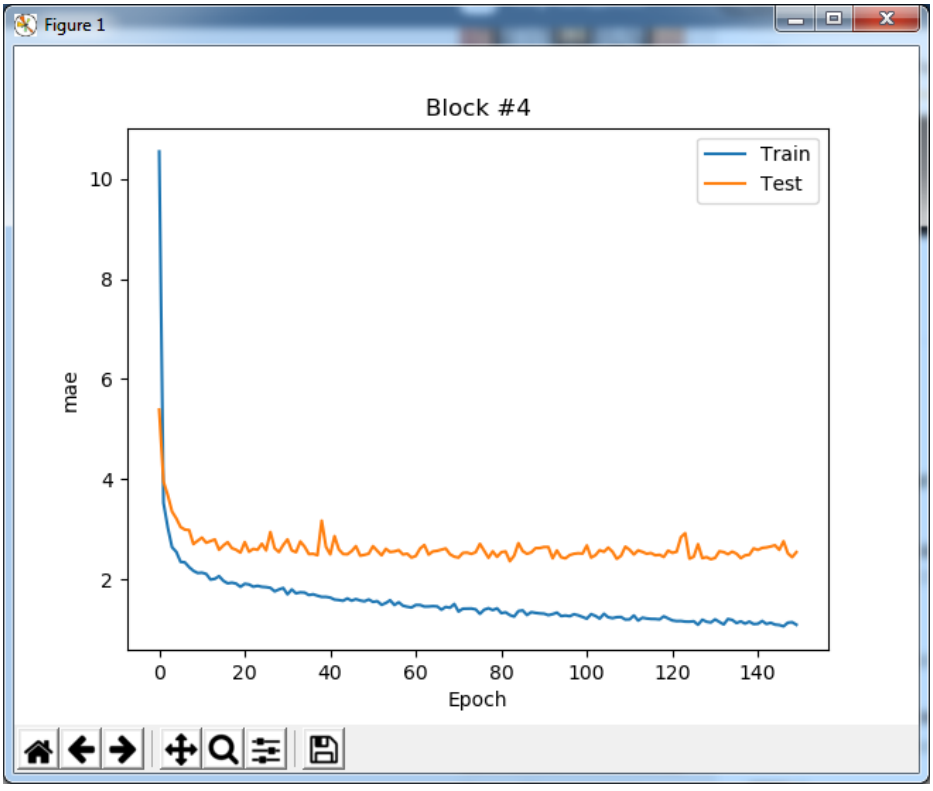

Рисунок 4 – График оценки MAE для 4 блока

График средних значений MAE по всем блокам представлен на рис. 5.

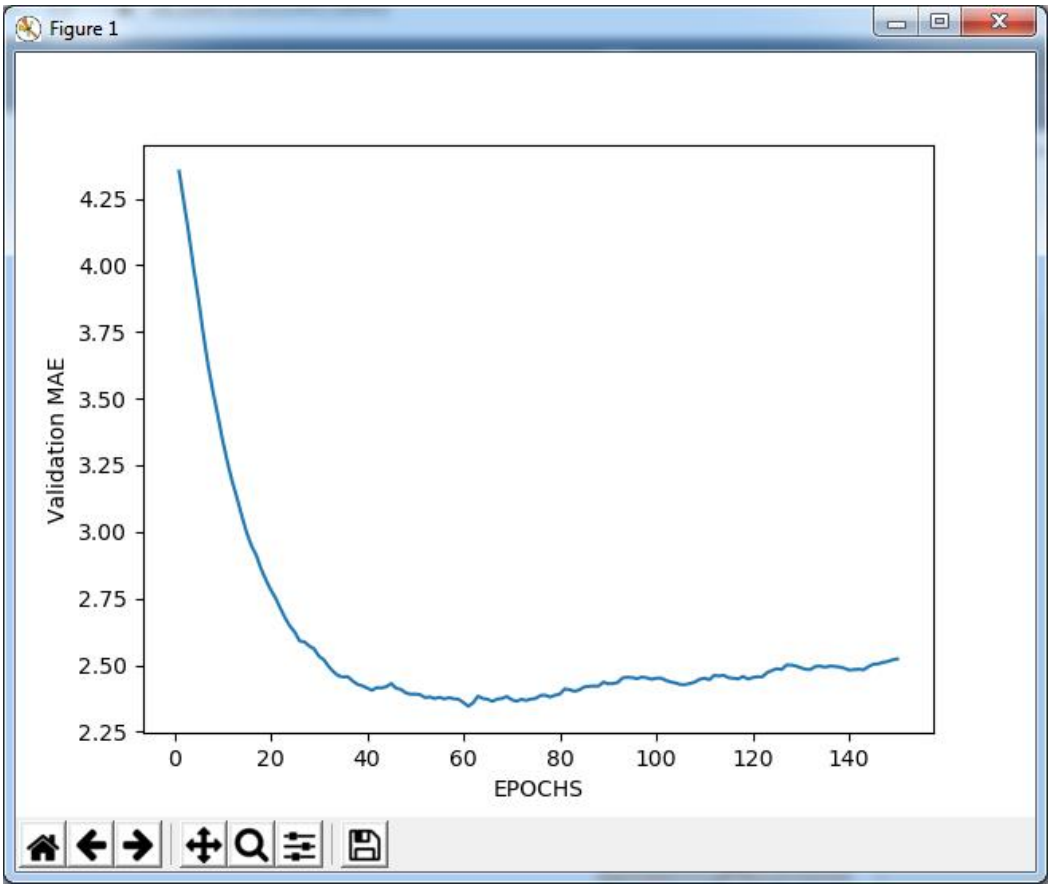

Рисунок 5 – График для среднего значения MAE по 4 блокам

По графику видно, что оценка MAE начинает возрастать после 80 эпохи.

Далее количество блоков было изменено на 6. Графики по отдельным блокам и среднее значение MAE представлены на рис. 6-12.

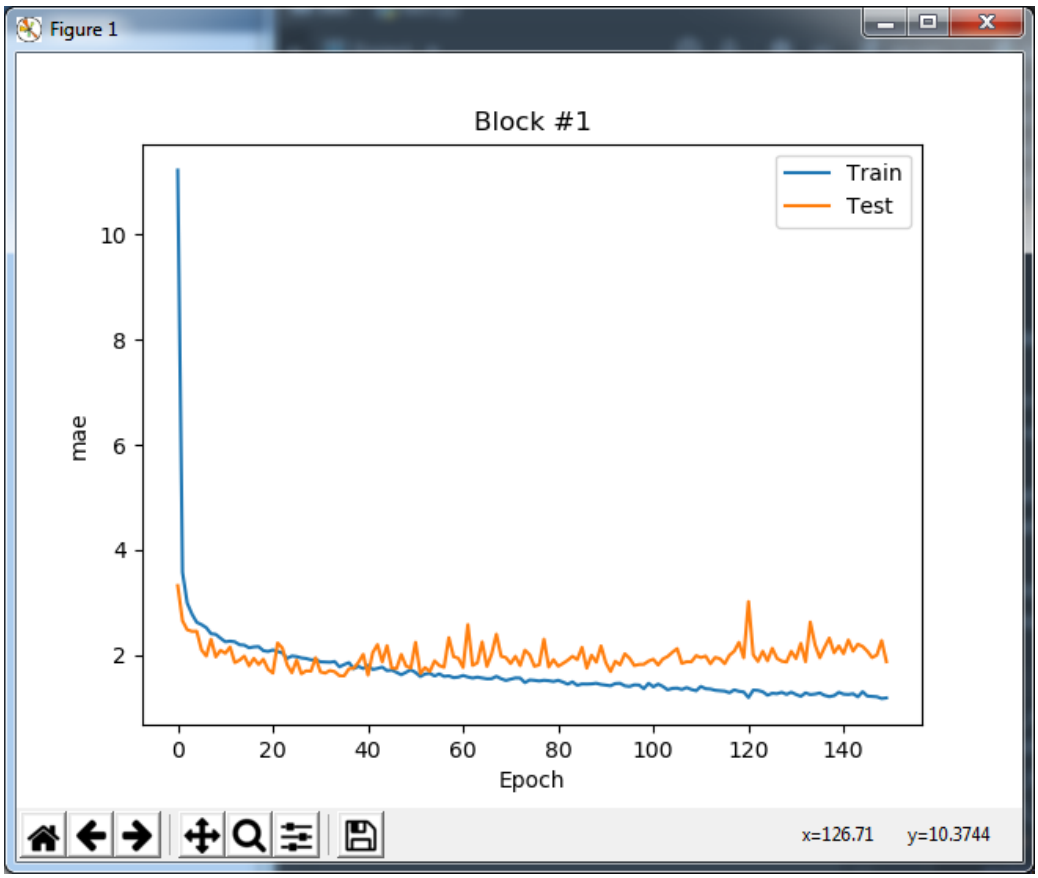

Рисунок 6 – График оценки MAE для 1 блока

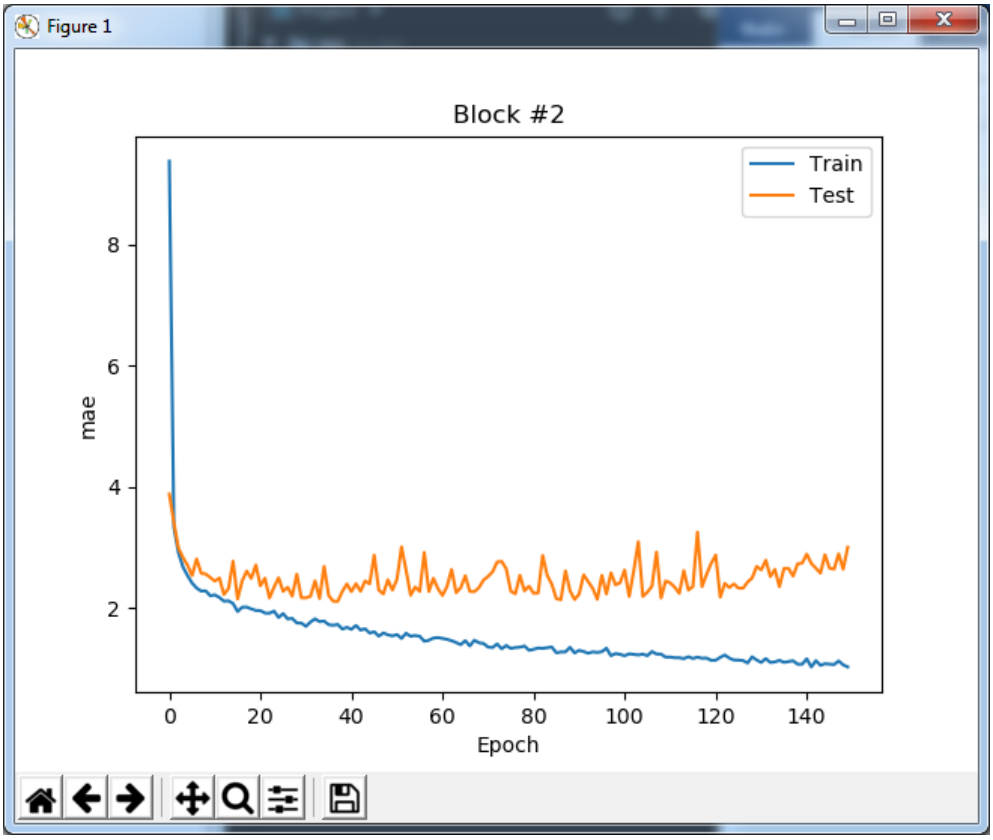

Рисунок 7 – График оценки MAE для 2 блока

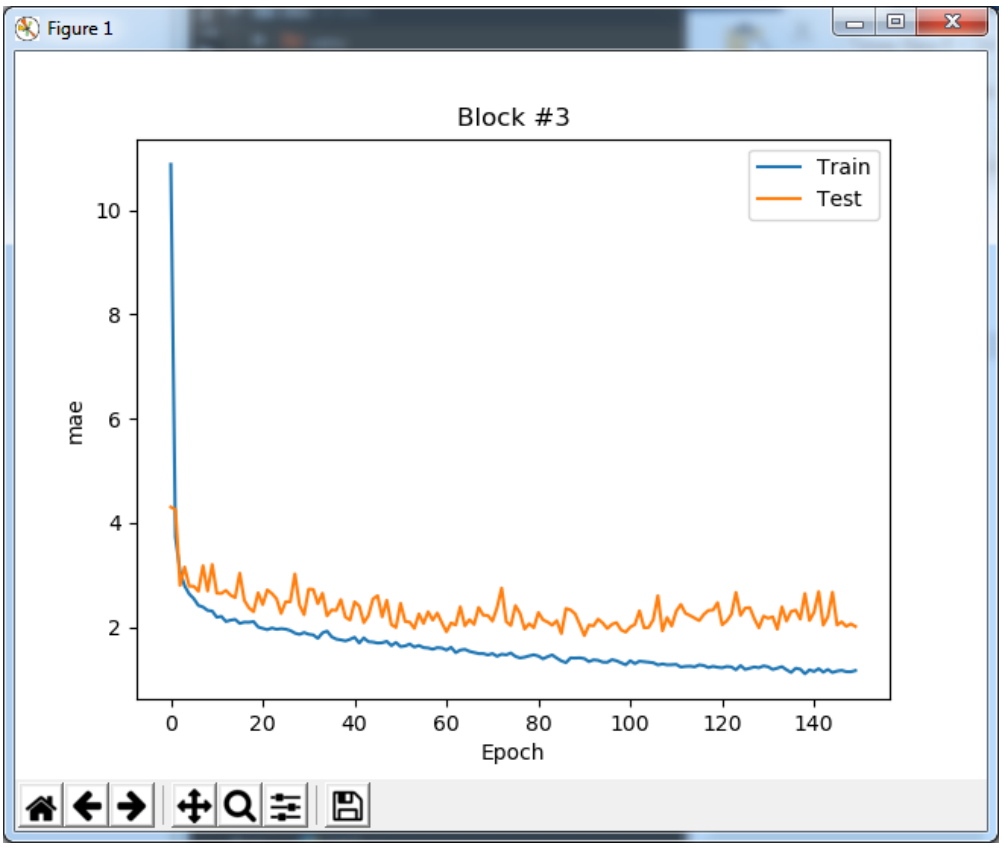

Рисунок 8 – График оценки MAE для 3 блока

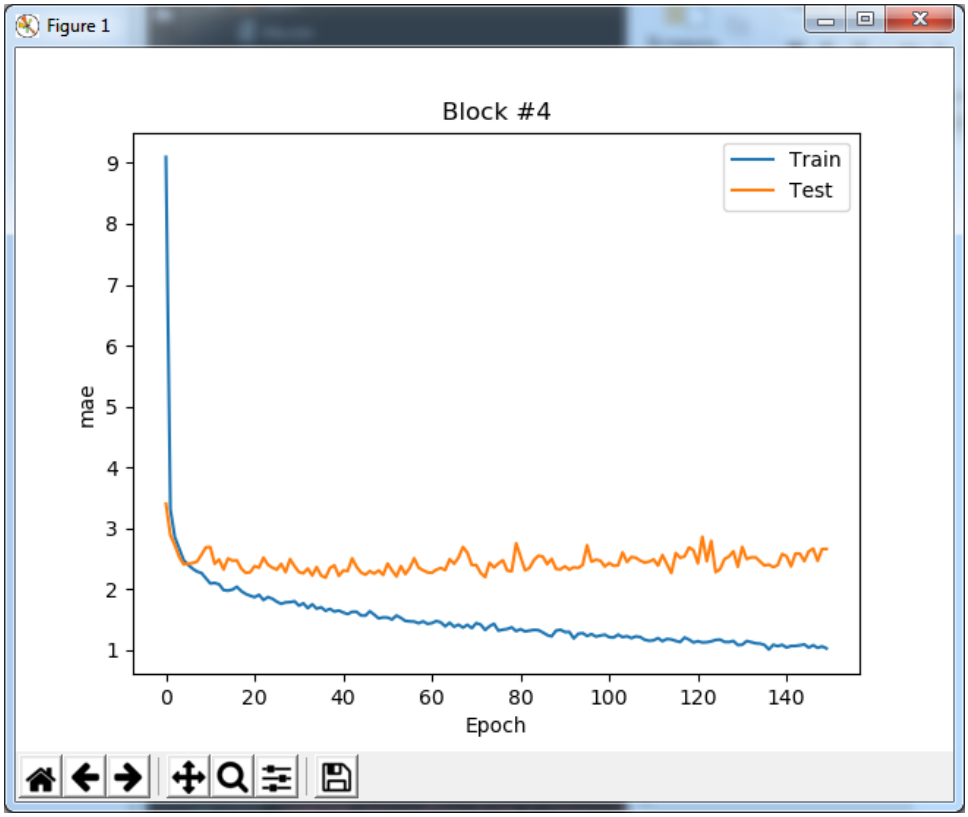

Рисунок 9 – График оценки MAE для 4 блока

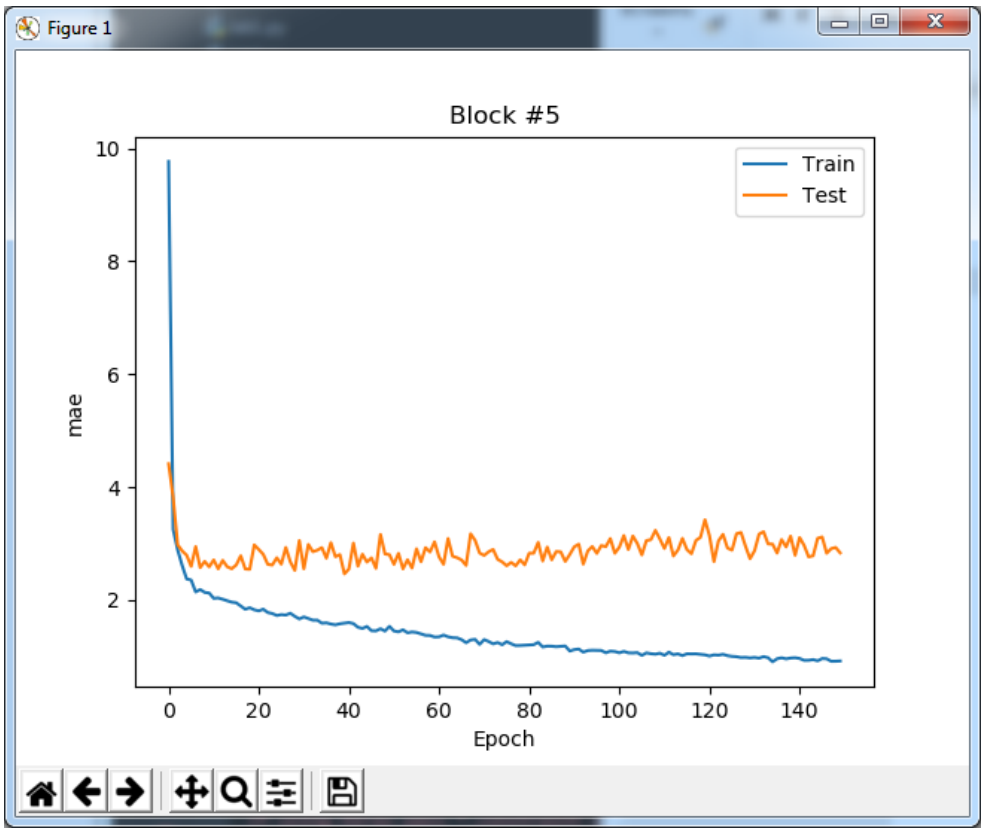

Рисунок 10 – График оценки MAE для 5 блока

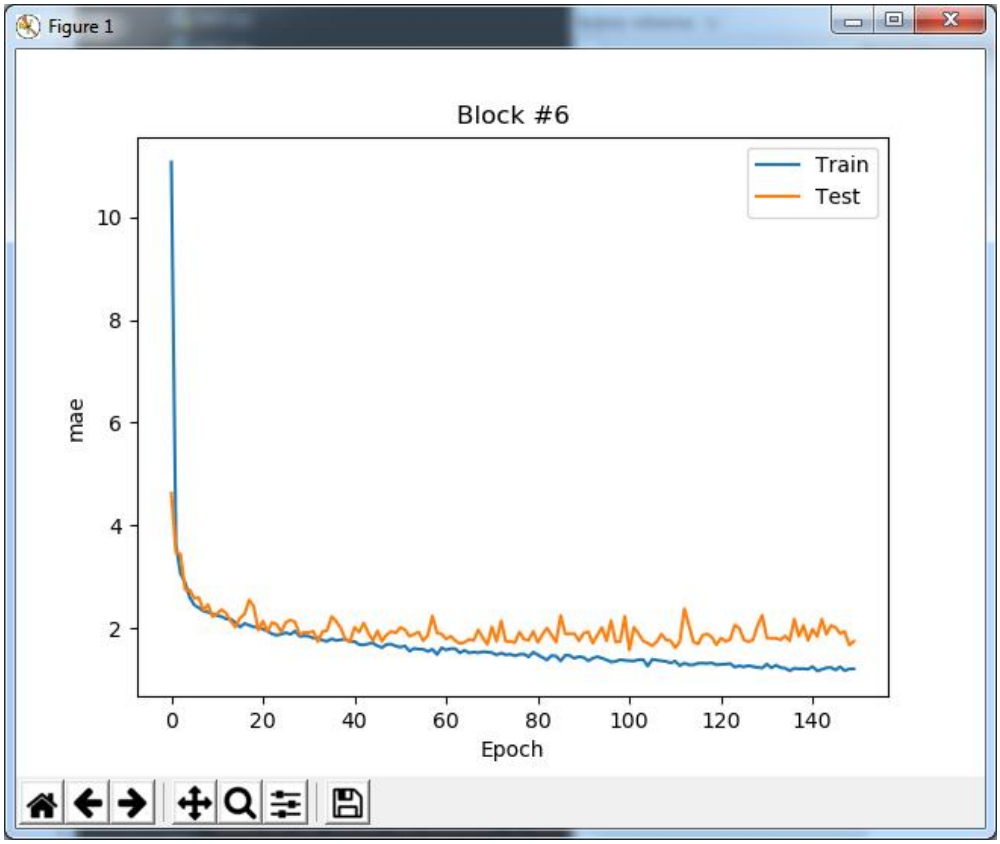

Рисунок 11 – График оценки MAE для 6 блока

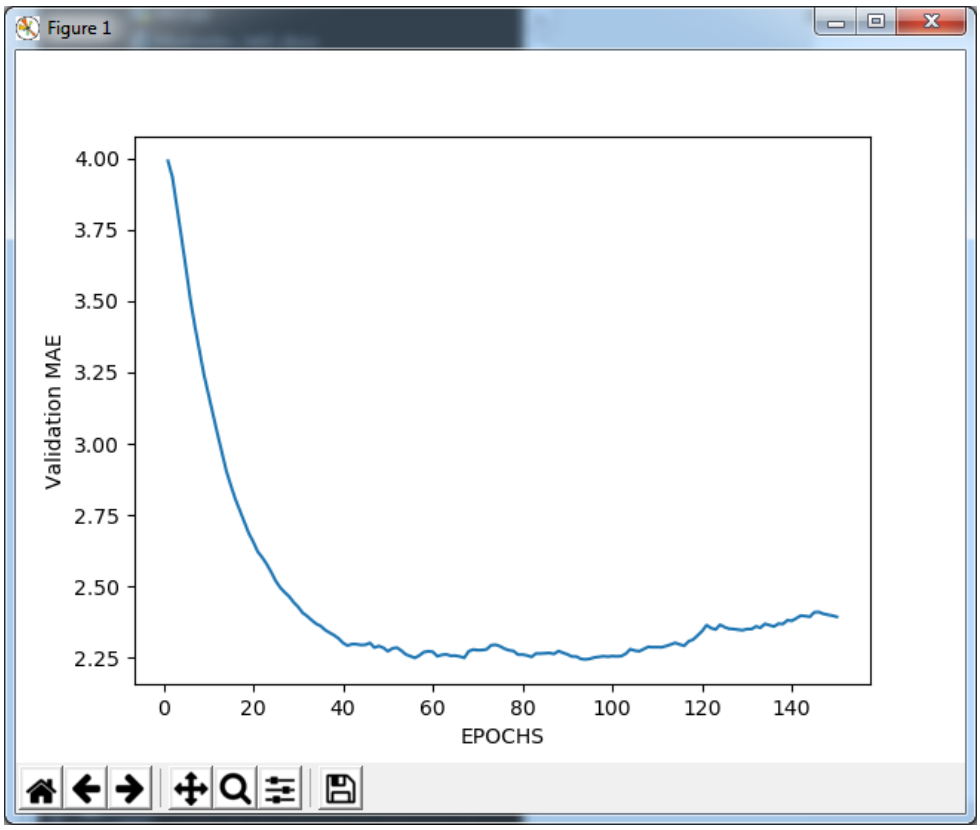

Рисунок 12 – График для среднего значения MAE по 6 блокам

На рис. 12 можно заметить, что оценка MAE начала возрастать после 100-ой эпохи. Если сравнить графики на рис. 12 и рис. 5, то можно заметить, что на 6 блоках, до переобучения, достигаются меньшие значения MAE.

Далее количество блоков было увеличено до 8. Графики представлены на рис. 13-21.

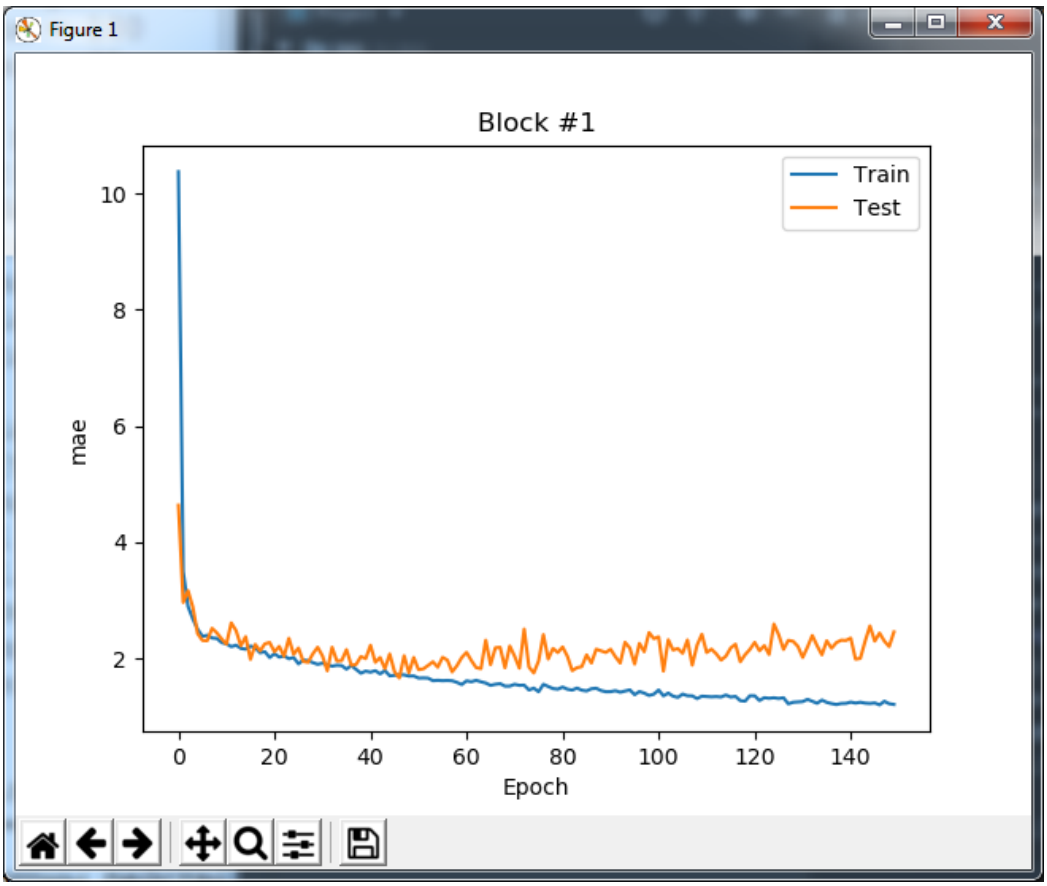

Рисунок 13 – График оценки MAE для 1 блока

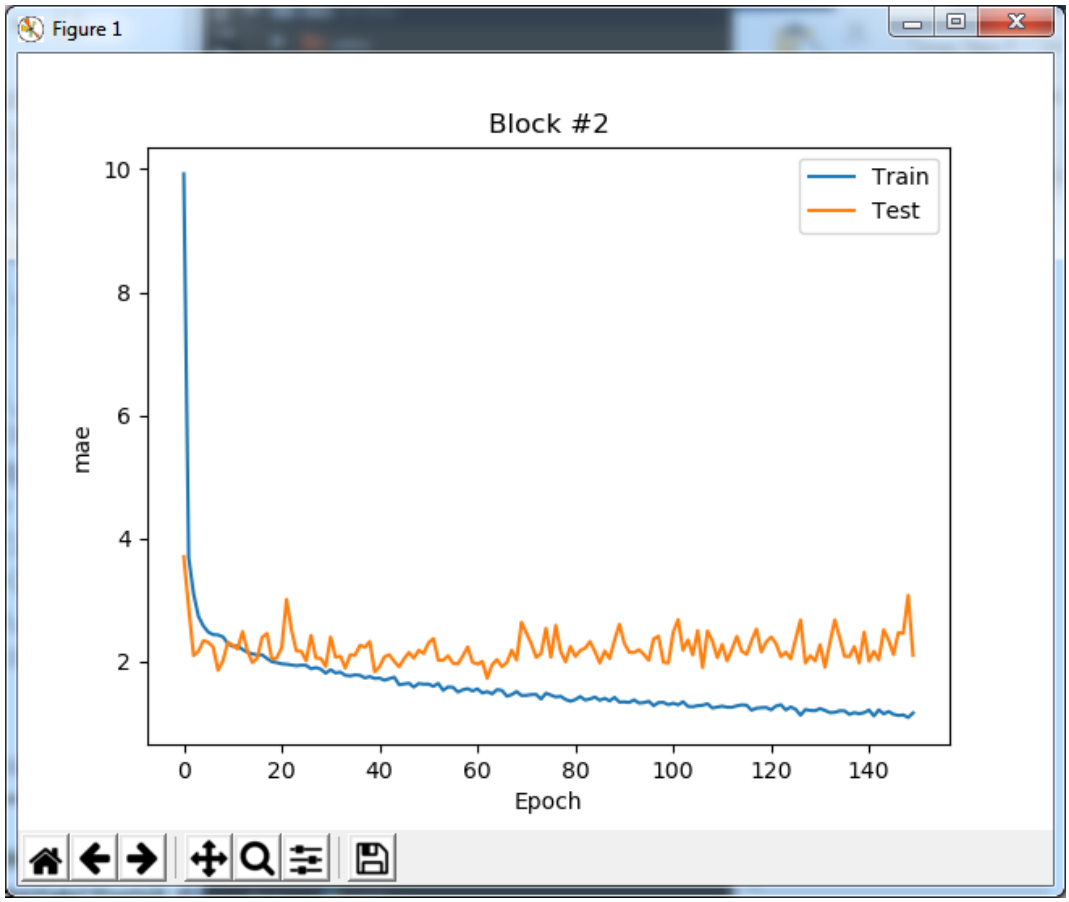

Рисунок 14 – График оценки MAE для 2 блока

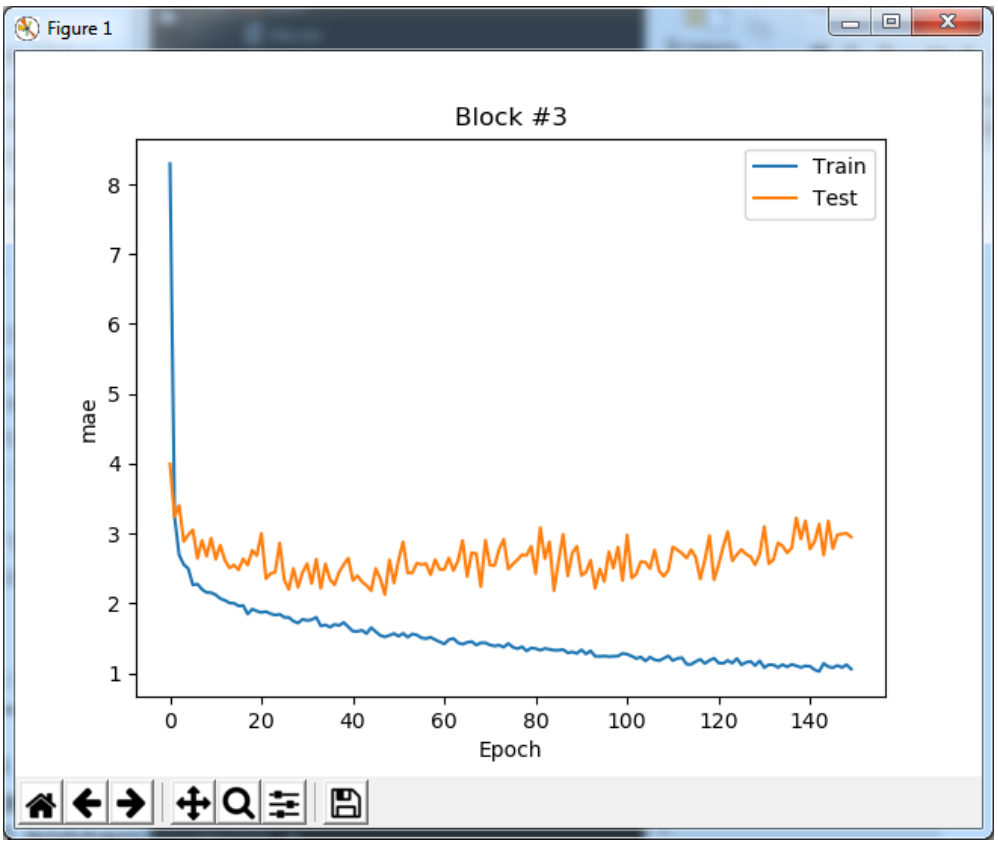

Рисунок 15 – График оценки MAE для 3 блока

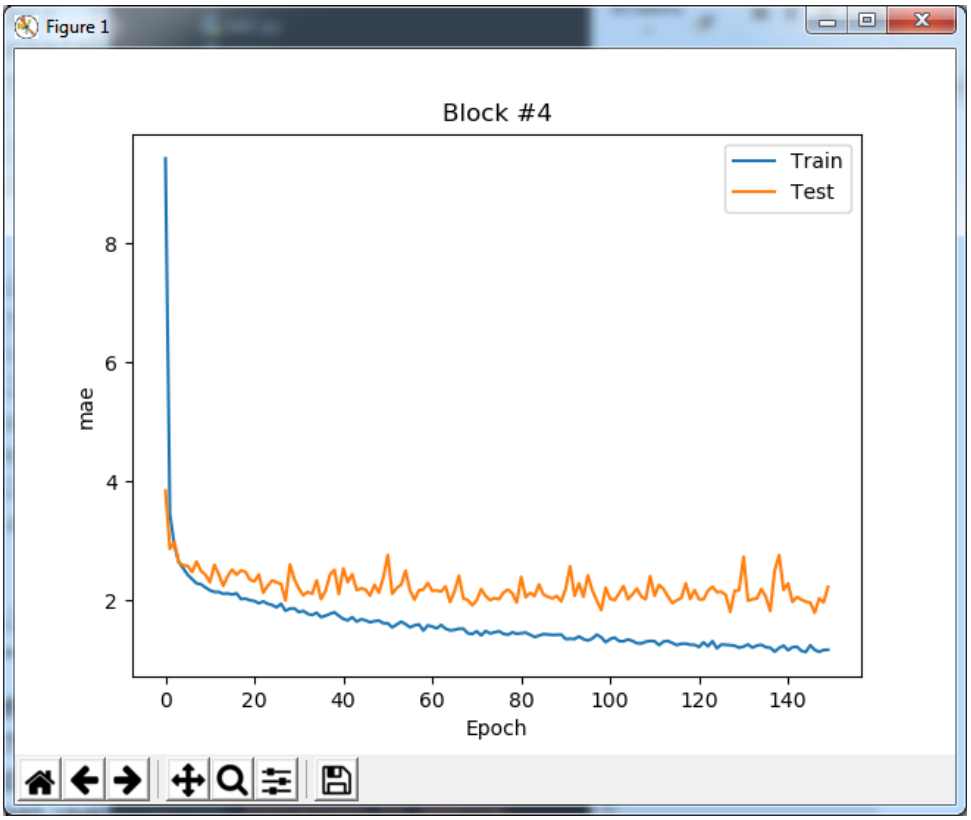

Рисунок 16 – График оценки MAE для 4 блока

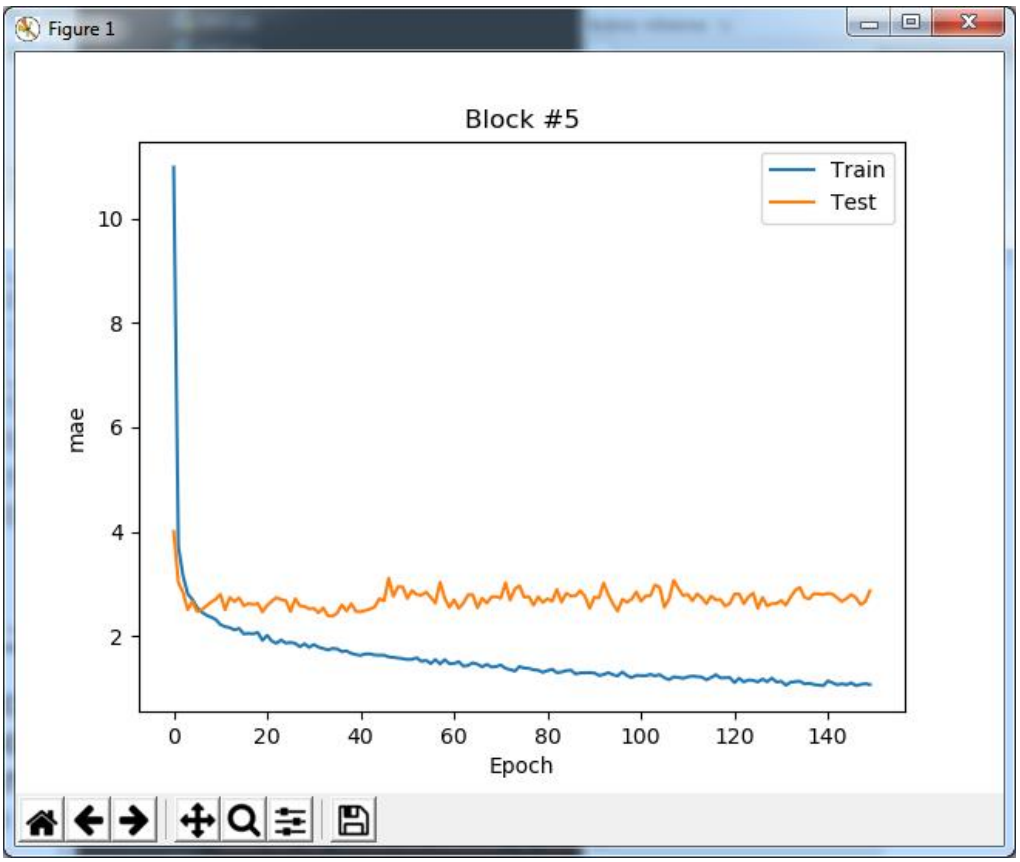

Рисунок 17 – График оценки MAE для 5 блока

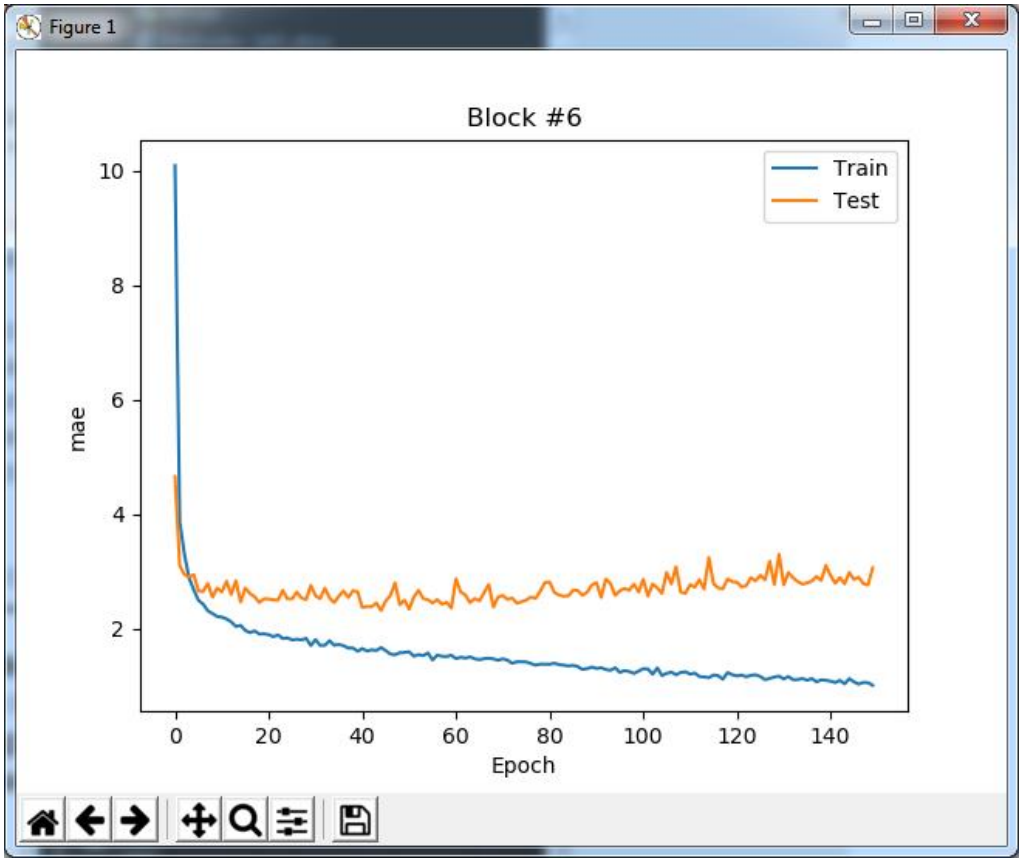

Рисунок 18 – График оценки MAE для 6 блока

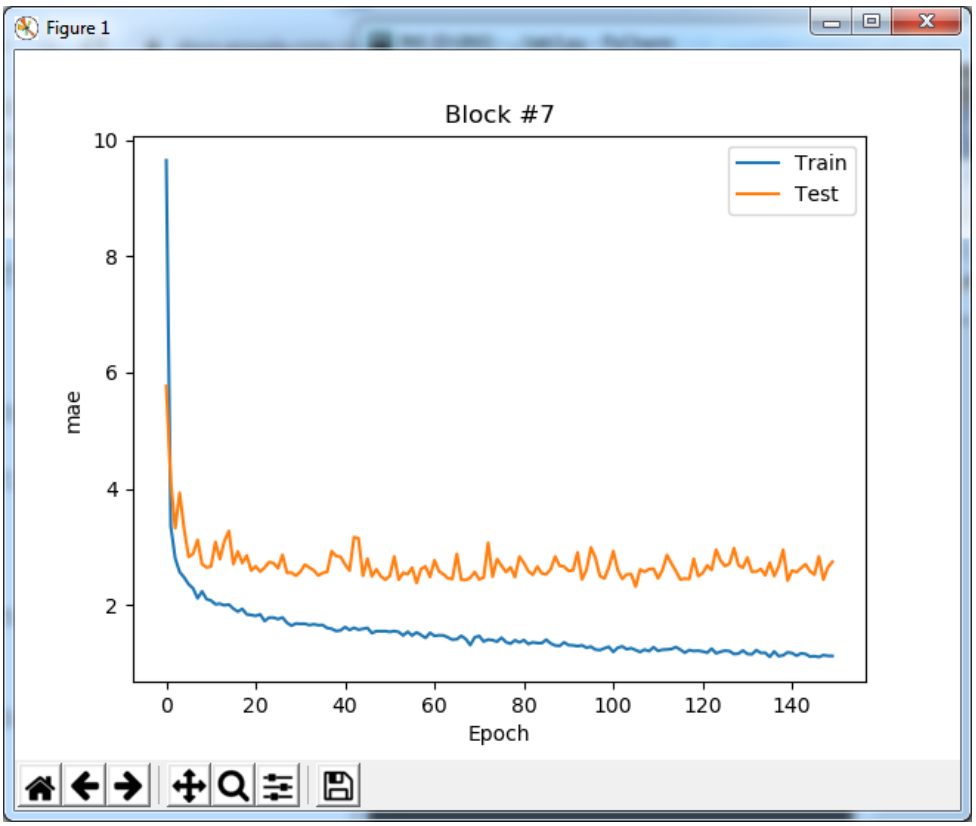

Рисунок 19 – График оценки MAE для 7 блока

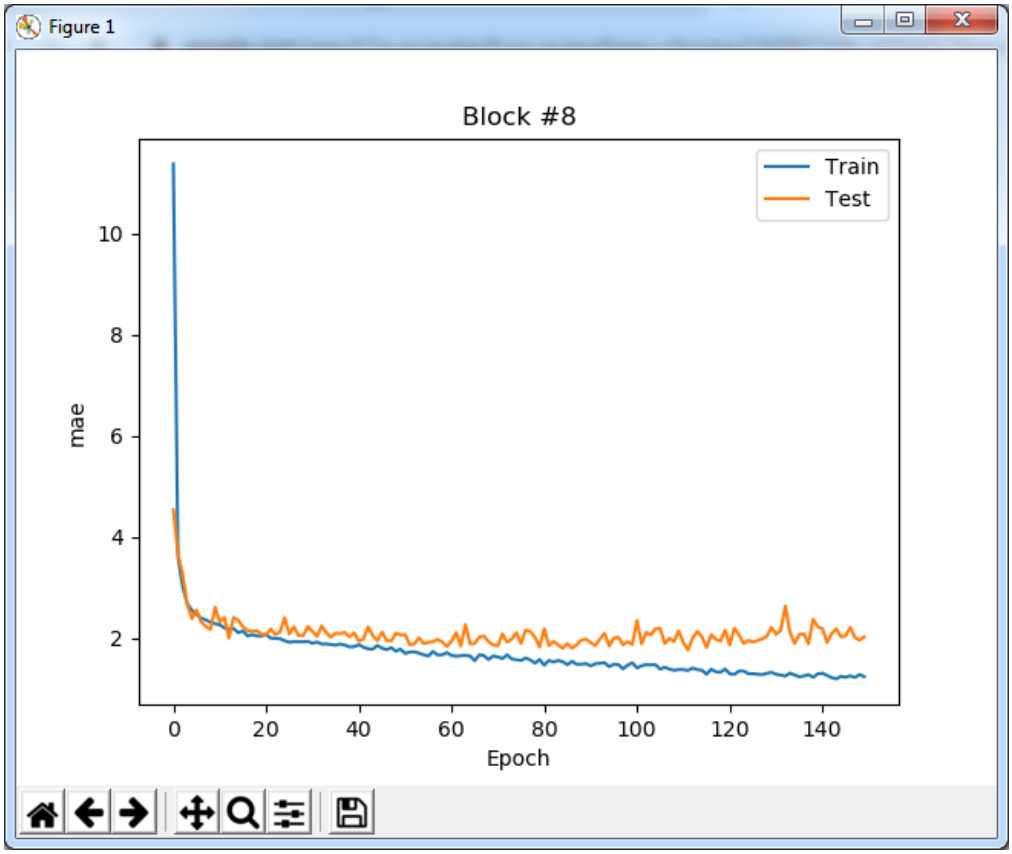

Рисунок 20 – График оценки MAE для 8 блока

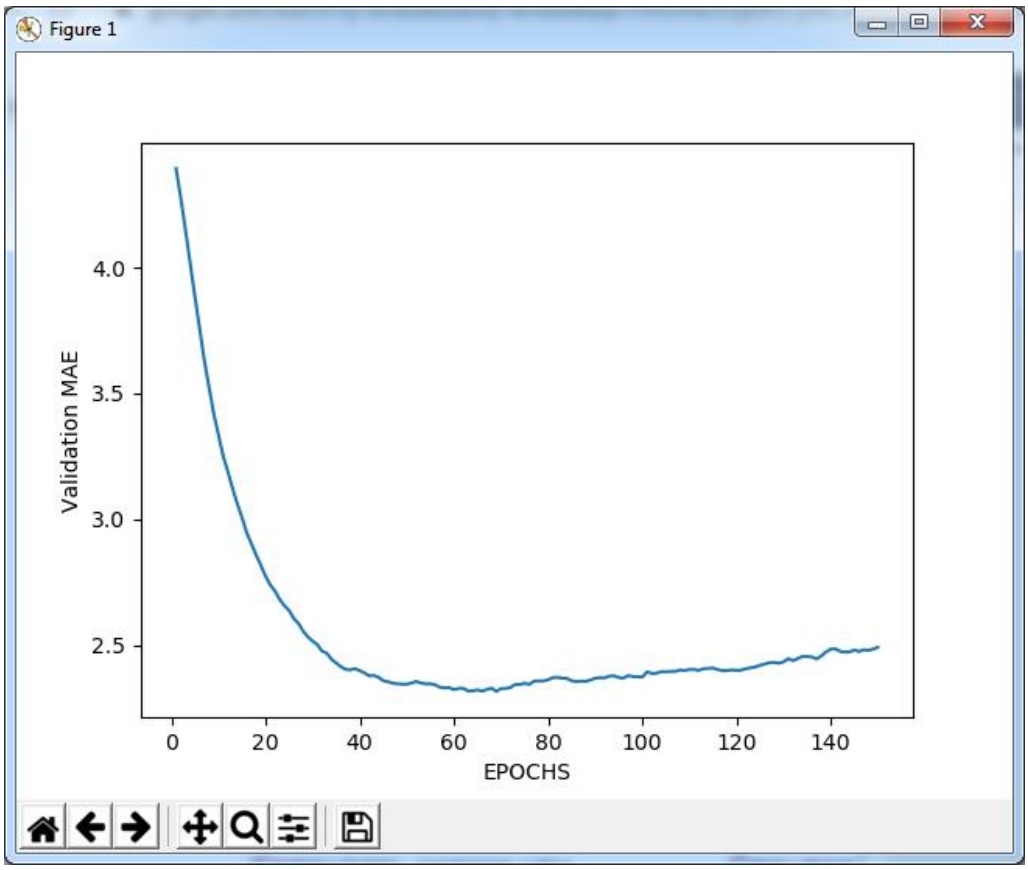

Рисунок 12 – График для среднего значения MAE по 8 блокам

По рис. 12 видно, что переобучение начинается с 60 эпохи и значение оценки MAE хуже, чем в предыдущих случаях, следовательно, оптимальным вариантом будет модель с 6-ю блоками и 80-ю эпохами.

Вывод.

В ходе выполнения данной работы была изучена задача регрессии с помощью библиотеки Keras и ее отличие от задачи классификации. Была изучена перекрестная проверка.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А: ИСХОДНЫЙ КОД

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

from tensorflow.keras.datasets import boston_housing

import numpy as np
import matplotlib.pyplot as plt


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



(train_data, train_targets), (test_data, test_targets) =
boston_housing.load_data()

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

train_data -= mean
train_data /= std

test_data -= mean
test_data /= std

k = 8
num_val_samples = len(train_data) // k
num_epochs = 150
all_mae_histories = []
mean_val_mae = []
for i in range(k):
    print(i)
    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'])
    plt.plot(history.history['mean_absolute_error'])
    plt.plot(history.history['val_mean_absolute_error'])
    title = 'Block #' + str(i+1)
    plt.title(title)
    plt.ylabel('mae')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'])
    plt.show()



def smooth_curve(points, factor=0.9):
    smoothed_points = []
    for point in points:
        if smoothed_points:
            prev = smoothed_points[-1]
            smoothed_points.append(prev*factor+point*(1-
factor))
        else:
            smoothed_points.append(point)
    return smoothed_points

average_mae_history = [np.mean([x[i] for x in mean_val_mae])
for i in range(num_epochs)]
smooth_mae_history = smooth_curve(average_mae_history)
plt.plot(range(1, len(smooth_mae_history)+1),
smooth_mae_history)
plt.xlabel('EPOCHS')
plt.ylabel("Validation MAE")
plt.show()
```