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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Искусственные нейронные сети»

Тема: Регрессионная модель изменения цен на дома в Бостоне

Студентка гр. 8383 _____ Максимова А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

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.

Данный набор содержит относительно немного образцов данных: всего 506, разбитых на 404 обучающих и 102 контрольных образца. И каждый признак во входных данных (например, уровень преступности) имеет свой масштаб. Например, некоторые признаки являются пропорциями и имеют значения между 0 и 1, другие - между 1 и 12 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

Требования

1. Объяснить различия задач классификации и регрессии
2. Изучить влияние количества эпох на результат обучения модели
3. Выявить точку переобучения
4. Применить перекрестную проверку по К блокам при различных К
5. Построить графики ошибки и точности во время обучения для моделей, а также усредненных графики по всем моделям

Основные теоретические положения

Задача классификации: задача, в которой имеется множество объектов, где каждый объект, исходя из его свойств и параметров, можно отнести к конкретному классу. Таким образом, нейронная сеть, получая на вход объект, возвращает на выход вероятность (дискретную величину) его принадлежности

к каждому из классов. В процессе обучения ИНС стремимся достигнуть результата, когда вероятность принадлежности к правильному классу имеет значение единицы, к другим классам - нуля.

Задача регрессии: задача, в которой имеется множество объектов, имеющих различные признаки. На основе этих данных нейронная сеть должна вернуть действительное число, которое может широко изменяться в каком-то непрерывном диапазоне. Так в данной работе имеется множество, включающее в себя 506 объектов, каждый из которых имеет 13 признаков, имеющих различный масштаб, а на выходе прогнозируется цена дома, выраженная в 1000$.

Различия задач классификации и регрессии: при решении задачи классификации нейронная сеть на выходе возвращает вектор размера n, где n - число классов, заполненный значениями из диапазона от 0 до 1 (дискретные значения) - вероятности принадлежности к классам (хотим выяснить к какому классу принадлежит объект), а при решении задачи регрессии получаем непрерывную величину.

Выполнение работы

1. Были импортированы все необходимые для работы классы и функции.

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense              # полносвязанный слой
from tensorflow.keras.models import Sequential         # сеть прямого распространения
from tensorflow.keras.datasets import boston_housing   # данные
```

2. Набор данных, используемый в лабораторной работе, присутствует в составе Keras. Всего имеется 506 объектов, 404 из которых отводятся на обучение, 102 - контрольные образцы. Каждый объект имеет 13 числовых параметров (входных) и 1 выходной для всех объектов.

Загрузка данных была выполнена с помощью функции load_data(), которая возвращает кортеж массивов NumPy: данные для обучения и тестирования.

```
(train_data, train_targets), (test_data, test_targets) = boston_housing.load_data()
print(train_data.shape)
print(test_data.shape)
print(test_targets)
```

Для проверки успешности был выполнен вывод форм матриц с помощью атрибута shape, в которых лежат параметры объектов, а также содержимое вектора, в котором хранятся прогнозируемые цены на дома для тестовых данных.

```
(404, 13)
(102, 13)
[ 7.2 18.8 19.  27.  22.2 24.5 31.2 22.9 20.5 23.2 18.6 14.5 17.8 50.
 20.8 24.3 24.2 19.8 19.1 22.7 12.  10.2 20.  18.5 20.9 23.  27.5 30.1
  9.5 22.  21.2 14.1 33.1 23.4 20.1  7.4 15.4 23.8 20.1 24.5 33.  28.4
 14.1 46.7 32.5 29.6 28.4 19.8 20.2 25.  35.4 20.3  9.7 14.5 34.9 26.6
  7.2 50.  32.4 21.6 29.8 13.1 27.5 21.2 23.1 21.9 13.  23.2  8.1  5.6
 21.7 29.6 19.6  7.  26.4 18.9 20.9 28.1 35.4 10.2 24.3 43.1 17.6 15.4
 16.2 27.1 21.4 21.5 22.4 25.  16.6 18.6 22.  42.8 35.1 21.5 36.  21.9
 24.1 50.  26.7 25. ]
```

3. В данном наборе данных признаки изменяются в разных диапазонах, что мешает нейронной сети определять какие из признаков, вносят наибольшую роль (сеть может адаптироваться к разнородным данным, но это усложнит обучение к тому же на столь небольшом объеме данных). Поэтому было необходимо привести данные к одной шкале, то есть применить нормализацию: для каждого признака во входных данных (столбца в матрице) из каждого значения вычитается среднее по этому признаку, и разность делится на стандартное отклонение, в результате признак центрируется по нулевому значению и имеет стандартное отклонение, равное единице.

```
mean = train_data.mean(axis=0)  #среднее значение
std = train_data.std(axis=0)    #стандартное отклонение
train_data -= mean
train_data /= std
test_data -= mean
test_data /= std
```

4. После была определена функция для создания модели ИНС прямого распространения, состоящей из четырех полносвязанных слоев: первый (входной) содержит 13 нейронов, второй и третий (скрытые) - 64 нейрона, функция активации - Relu: max(0, x); выходной слой (линейный) - 1 нейрон, нелинейная функция активации не используется, так как применение функции активации могло бы ограничить диапазон выходных значений.

Были определены параметры обучения сети: в качестве функции потерь используется "mse" - среднеквадратичная ошибка, вычисляющая квадрат разности между предсказанными и целевыми значениями (часто используется в задачи регрессии), метрика "mae" - средняя абсолютная ошибка, абсолютное значение разности между предсказанными и целевыми значениями.

```
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
```

4. Так как исходный набор данных невелик, то для правдоподобной оценки качества сети можно использовать перекрестную проверку по K блокам. Ее суть заключается в разделении доступных данных на K блоков одинакового размера (обычно K = 4 или 5), создании K идентичных моделей и обучении каждой на K-1 блоках с оценкой по оставшимся блокам. По полученным K оценкам вычисляется среднее значение, принимаемое как оценка модели. Код, реализующий такую проверку, представлен ниже:

```python
#перекрестная проверка по K блокам
k = 4
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []                          #массив оценок

for i in range(k):
    print('processing fold #', i)
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples] #0-24, ..., 75-99
    val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples] #0-24, ..., 75-99

    partial_train_data = np.concatenate([train_data[:i * num_val_samples],
                                         train_data[(i + 1) * num_val_samples:]], axis=0) #пропуск i-ого блока
    partial_train_targets = np.concatenate([train_targets[:i * num_val_samples],
                                            train_targets[(i + 1) * num_val_samples:]], axis=0)
    #обучение
    model = build_model()
    model.fit(partial_train_data, partial_train_targets, epochs=num_epochs, batch_size=1, verbose=0)

    #оценка модели после обучения на тестовых данных
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0) #потери и метрика
    all_scores.append(val_mae)                                          #вставка в конец списка

print(np.mean(all_scores))
```

5. Для проверки работоспособности программы она была запущена. В процессе обучения нейронной сети отображаются четыре величины: среднеквадратичная ошибка сети на обучающих и тестовых данных соответственно - loss и val_loss, средняя абсолютная ошибка - mae и val_mae.

```
Epoch 20/100
303/303 [==============================] - 0s 695us/step - loss: 7.0223 - mae: 1.9277 - val_loss: 8.6402 - val_mae: 1.8778
Epoch 21/100
303/303 [==============================] - 0s 643us/step - loss: 10.4439 - mae: 1.9121 - val_loss: 8.4741 - val_mae: 1.8477
Epoch 22/100
303/303 [==============================] - 0s 740us/step - loss: 8.2824 - mae: 2.0368 - val_loss: 8.5666 - val_mae: 1.8643
Epoch 23/100
303/303 [==============================] - 0s 912us/step - loss: 8.5378 - mae: 1.9938 - val_loss: 7.9608 - val_mae: 1.9889
Epoch 24/100
303/303 [==============================] - 0s 688us/step - loss: 10.7440 - mae: 2.0246 - val_loss: 7.6671 - val_mae: 1.8297
```

6. Для построения графиков среднеквадратичной ошибки и средней абсолютной ошибки в ходе обучения сети были написаны следующие функции:

```python
#Построение графика среднеквадратичной ошибки во время обучения
def plot_mse(mse_, val_mse_):                                   #график среднеквадратичной ошибки
    plt.clf()
    epochs = range(1, len(mse) + 1)
    plt.plot(epochs, mse_, label='Training MSE', linestyle='--', linewidth=2, color="darkmagenta")
    plt.plot(epochs, val_mse_, 'b', label='Validation MSE', color="lawngreen")
    plt.title('Training and validation MSE')                    #оглавление на рисунке
    plt.xlabel('Epochs')
    plt.ylabel('MSE')
    plt.legend()
    plt.show()
```

```python
#Построение графика средней абсолютной ошибки во время обучения
def plot_mae(mae_, val_mae_):                                    #график средней абсолютной ошибки
    plt.clf()
    epochs = range(1, len(mae) + 1)
    plt.plot(epochs, mae_, label='Training MAE', linestyle='--', linewidth=2, color="darkmagenta")
    plt.plot(epochs, val_mae_, 'b', label='Validation MAE', color="lawngreen")
    plt.title('Training and validation MAE')
    plt.xlabel('Epochs')
    plt.ylabel('MAE')
    plt.legend()
    plt.show()
```

7. Для построения графика усредненной абсолютной ошибки на контрольных данных по K моделям были написаны следующие функции:

```python
def avg_val_mae(all_hist_, num_epochs_):
    avg_val_mae = np.zeros(num_epochs_)

    for i in range(len(all_hist_)):
        avg_val_mae += all_hist_[i].history['val_mae']

    avg_val_mae /= len(all_hist_
    plot_avg_mae(avg_val_mae)

def plot_avg_mae(avg_val_mae_):                    #график усредненной абсолютной ошибки
    plt.clf()
    epochs = range(1, len(mae) + 1)
    plt.plot(epochs, avg_val_mae_, 'b', label='Validation AVG MAE', color="indigo")
    plt.title('Validation AVG MAE')
    plt.xlabel('Epochs')
    plt.ylabel('VAL AVG MAE')
    plt.legend()
    plt.show()
```

Изучить влияние количества эпох на результат обучения моделей

Рассмотрим следующие комбинации:

№ эксперимента	1	2	3	4	5
Количество эпох	100	200	50	50	50
Количество блоков	4	4	4	2	8

Эксперимент 1

Параметры обучения и перекрестной проверки:

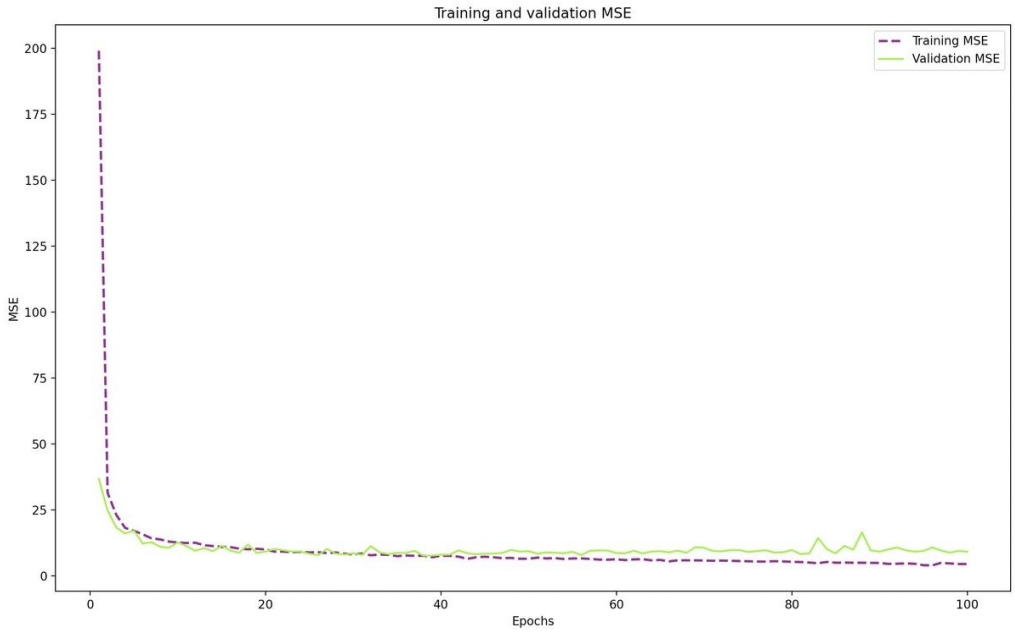
Количество эпох: 100

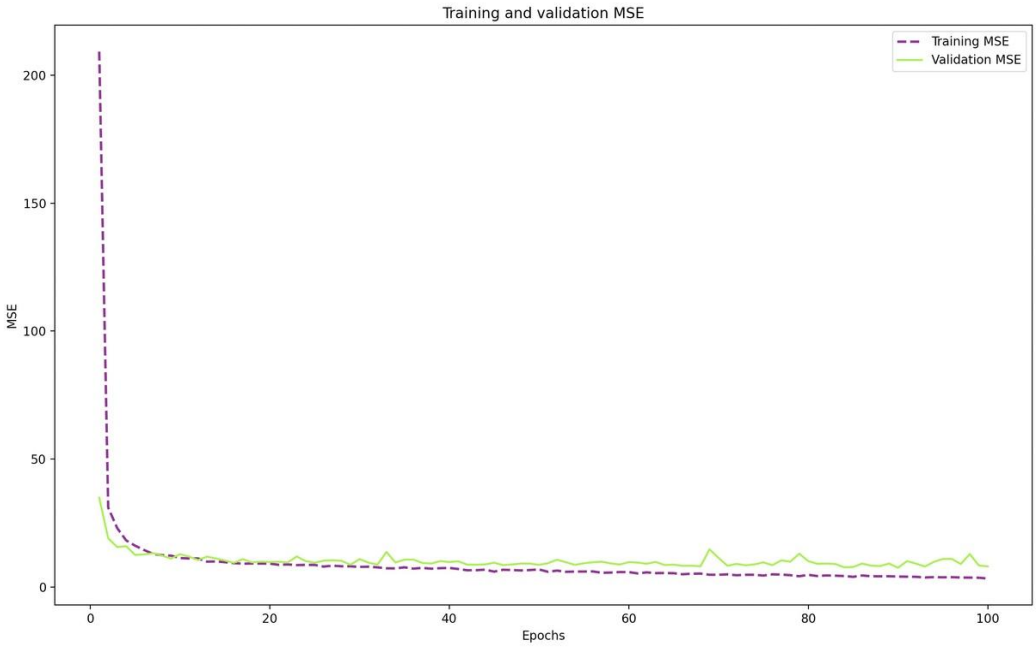
Количество блоков : 4

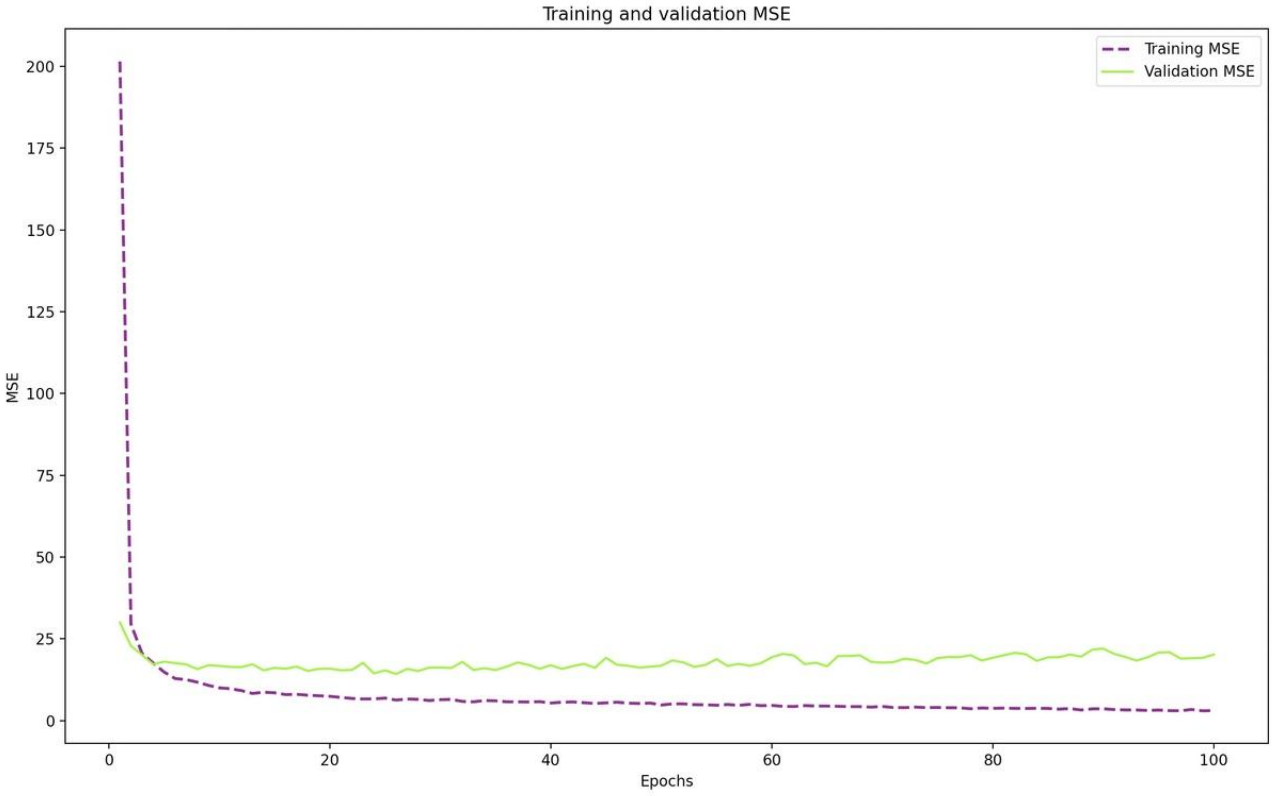
Результаты запуска:

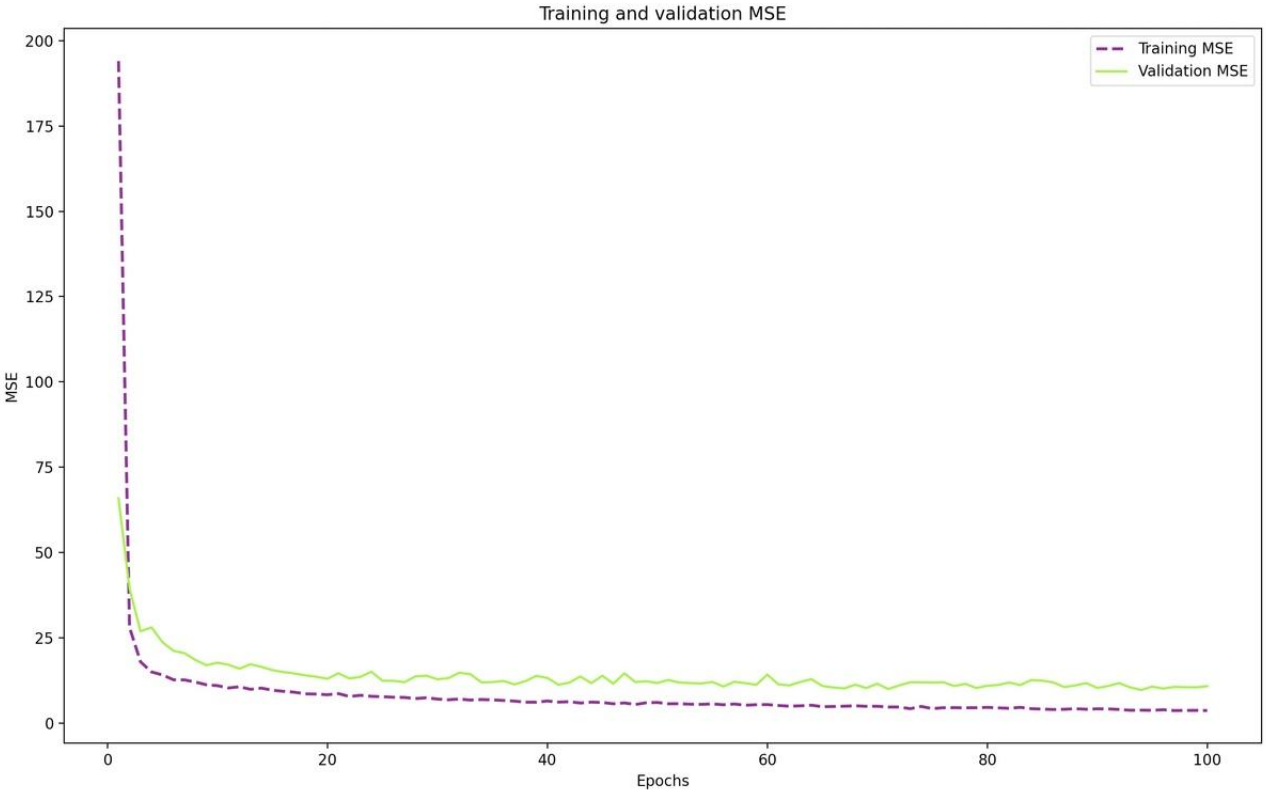
Среднее значение абсолютной ошибки на валидационных данных: 2.47

Графики средних квадратичных ошибок для 4х моделей при обучении и тестировании:

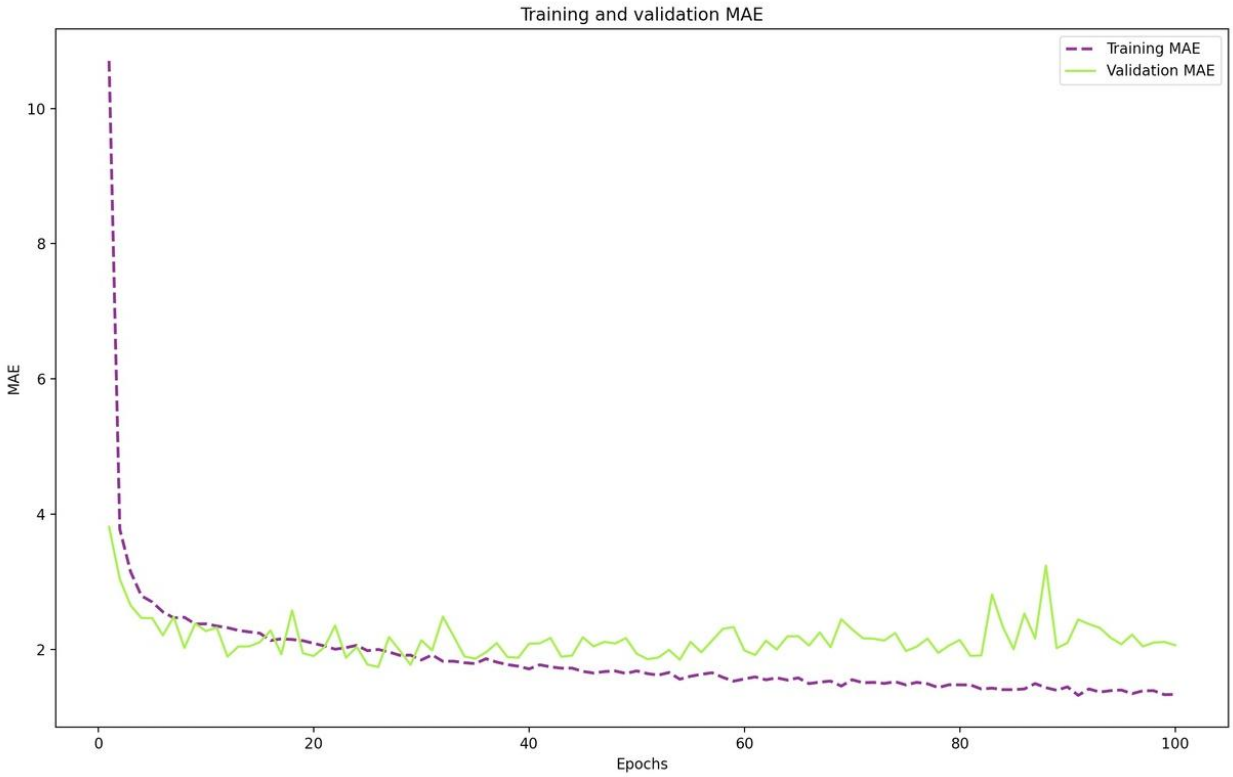


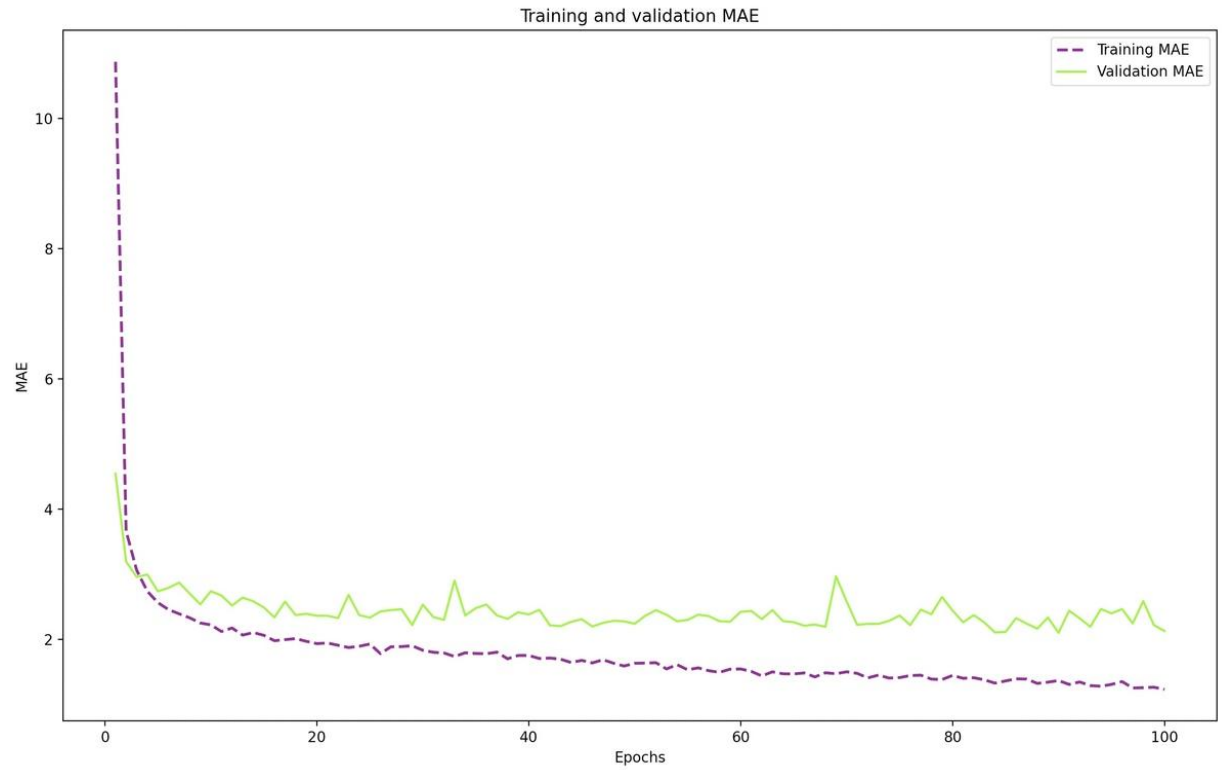


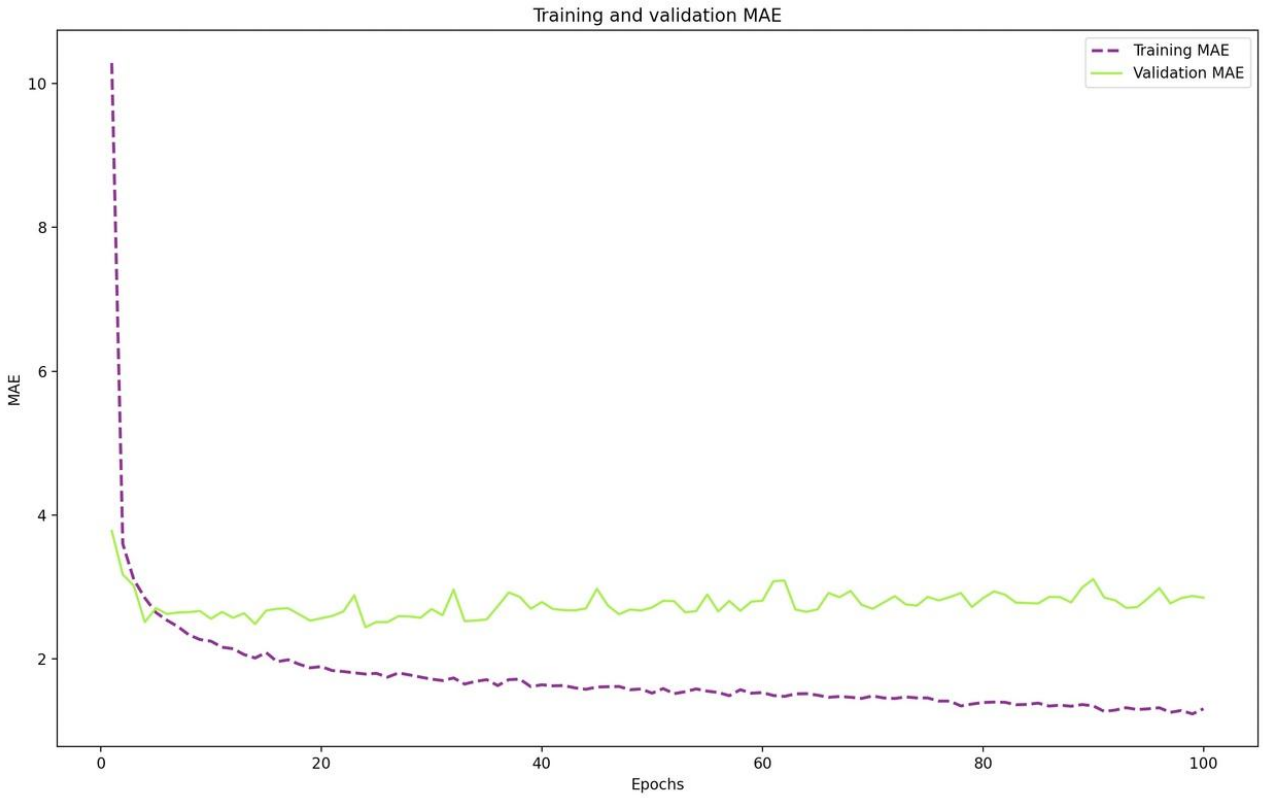

Training and validation MSE
Training MSE
Validation MSE
MSE
Epochs
0
25
50
75
100
125
150
175
200
0
20
40
60
80
100


Training and validation MSE
Training MSE
Validation MSE
MSE
Epochs
0
25
50
75
100
125
150
175
200
0
20
40
60
80
100

Графики средних абсолютных ошибок для 4х моделей при обучении и тестировании:






Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs
0
20
40
60
80
100
2
4
6
8
10

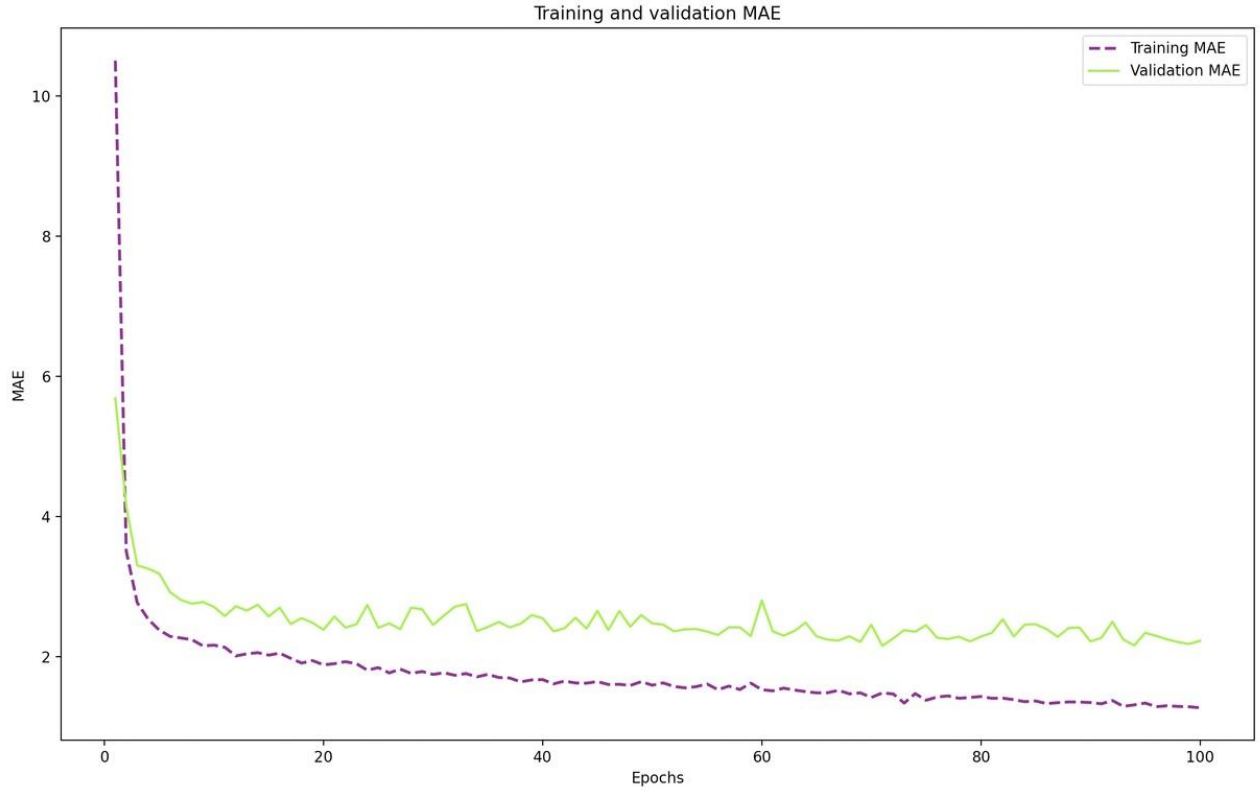
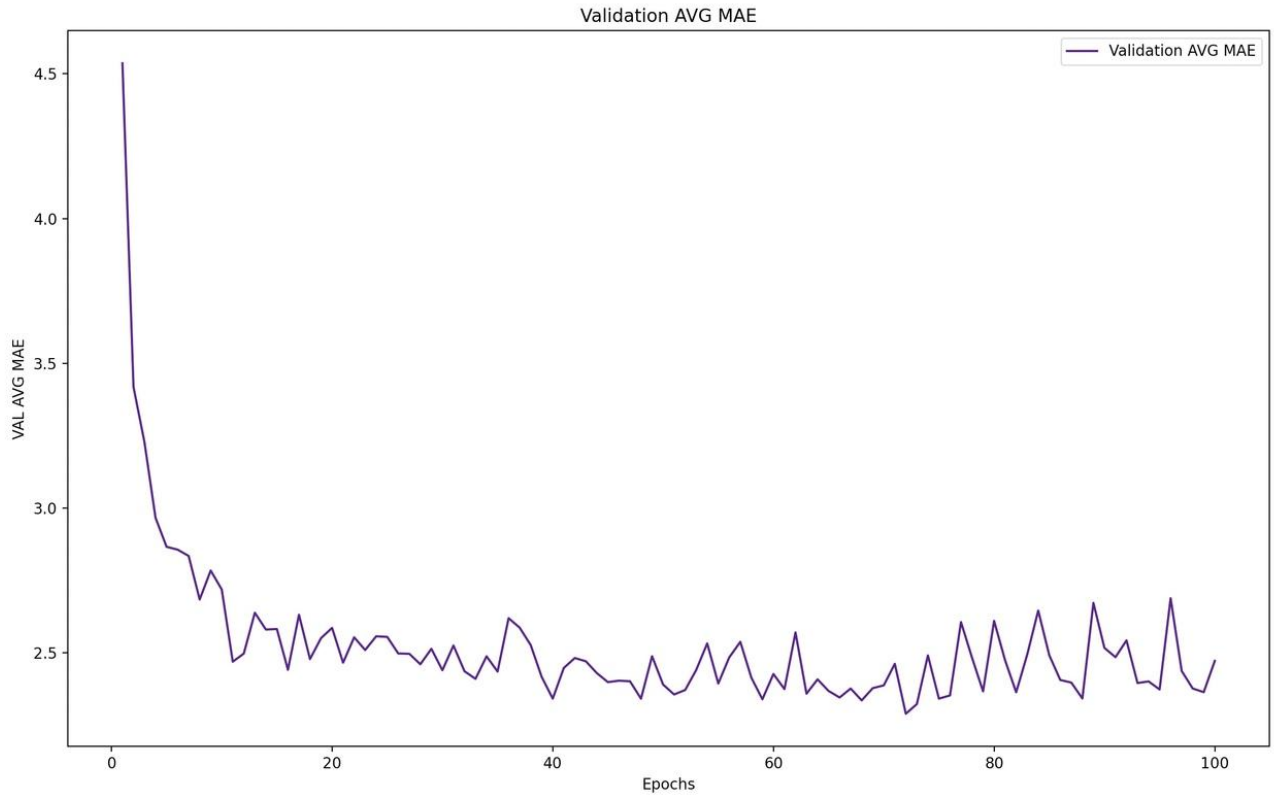

Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs
0
20
40
60
80
100
2
4
6
8
10

График среднего значения средней абсолютной ошибки на контрольных данных по всем моделям:



Выводы:

Из графиков средних квадратичных ошибок можно сделать вывод, что модели ведут себя примерно одинаково. Наименьшее значение средней квадратичной ошибки, впрочем не сильно отличающееся, имеет модель под номером 2.

Из графиков средних абсолютных ошибок видно, что относительно стремительное уменьшение ошибки происходит до 20 эпохи. Наименьшее значение средней абсолютной ошибки достигается на 1 модели, но уже после 40 эпохи это значение начинает расти, приобретая достаточно высокое значение. Наиболее высокие значения ошибки на контрольных данных имеет модель под номером 3.

Из графика среднего значения средней абсолютной ошибки на контрольных данных по всем моделям можно заметить, что значение ошибки достигает наименьшего значения примерно на 70 эпохе, после которой

значение ошибки снова начинает возрастать, что может говорить о переобучение нейронной сети.

Чтобы удостовериться в том, что сеть склонна к переобучение при относительно высоком значение количества эпох, увеличим начальное значение в два раза.

Эксперимент 2

Параметры обучения и перекрестной проверки:

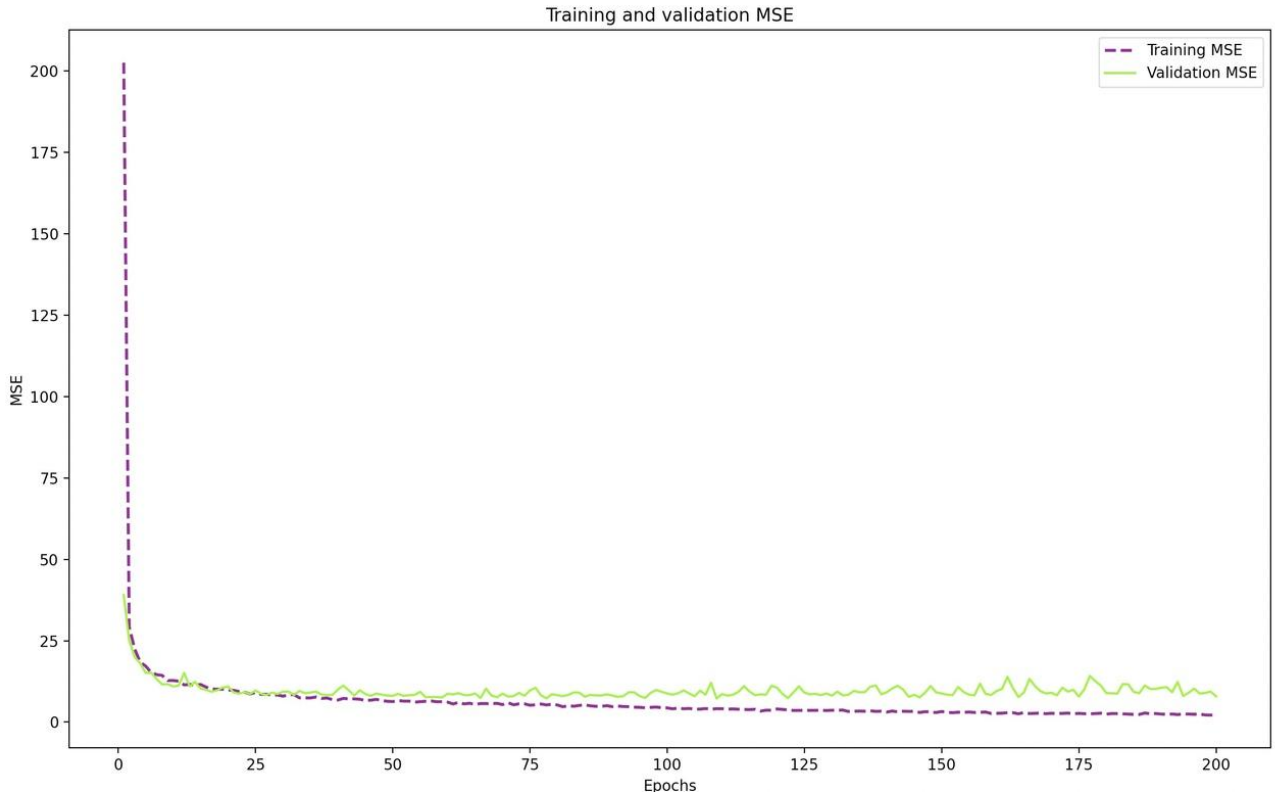
Количество эпох: 200

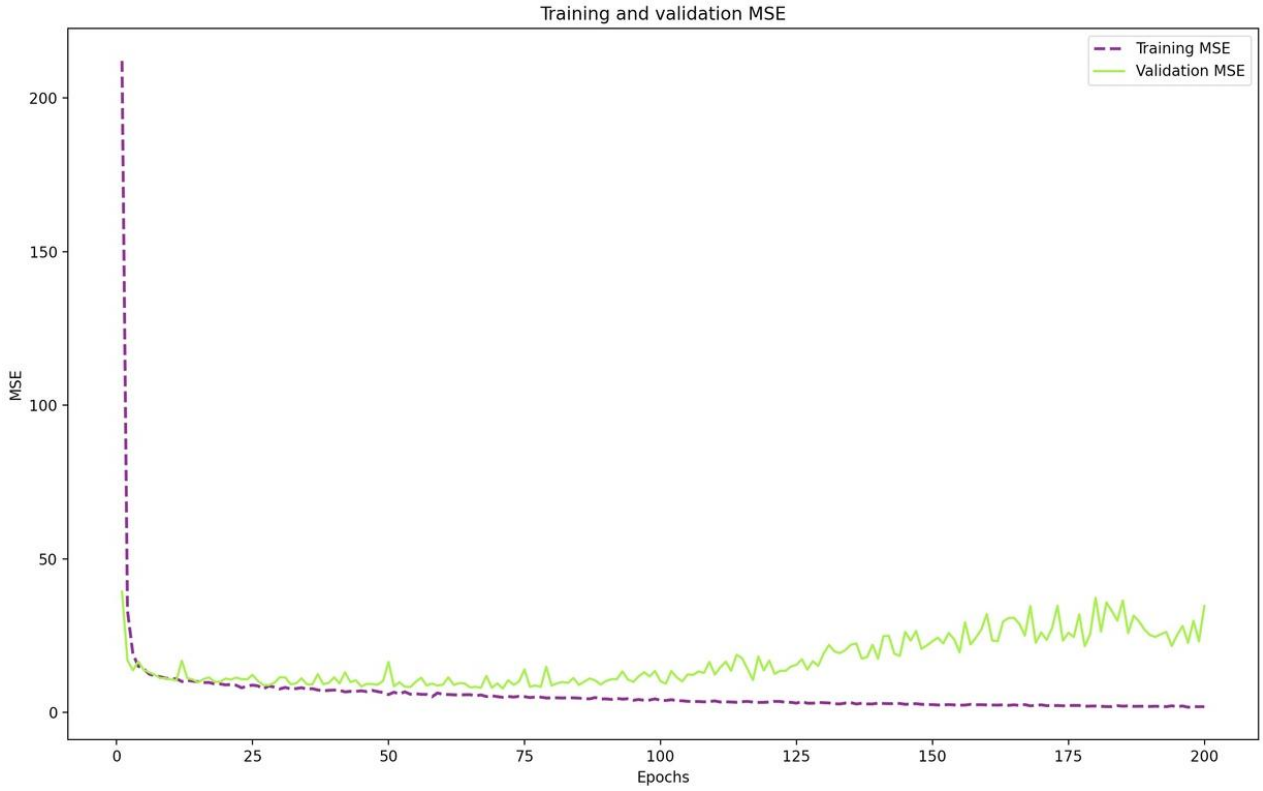
Количество блоков : 4

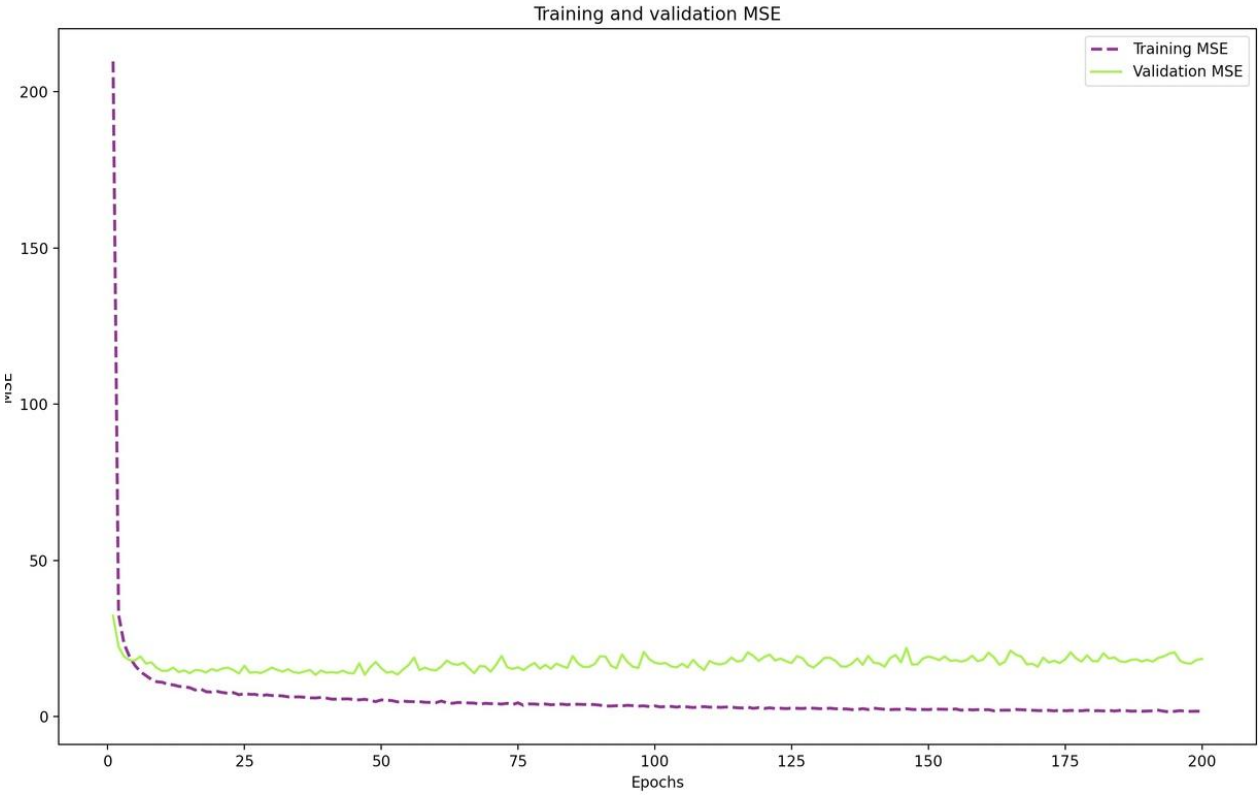
Результаты запуска:

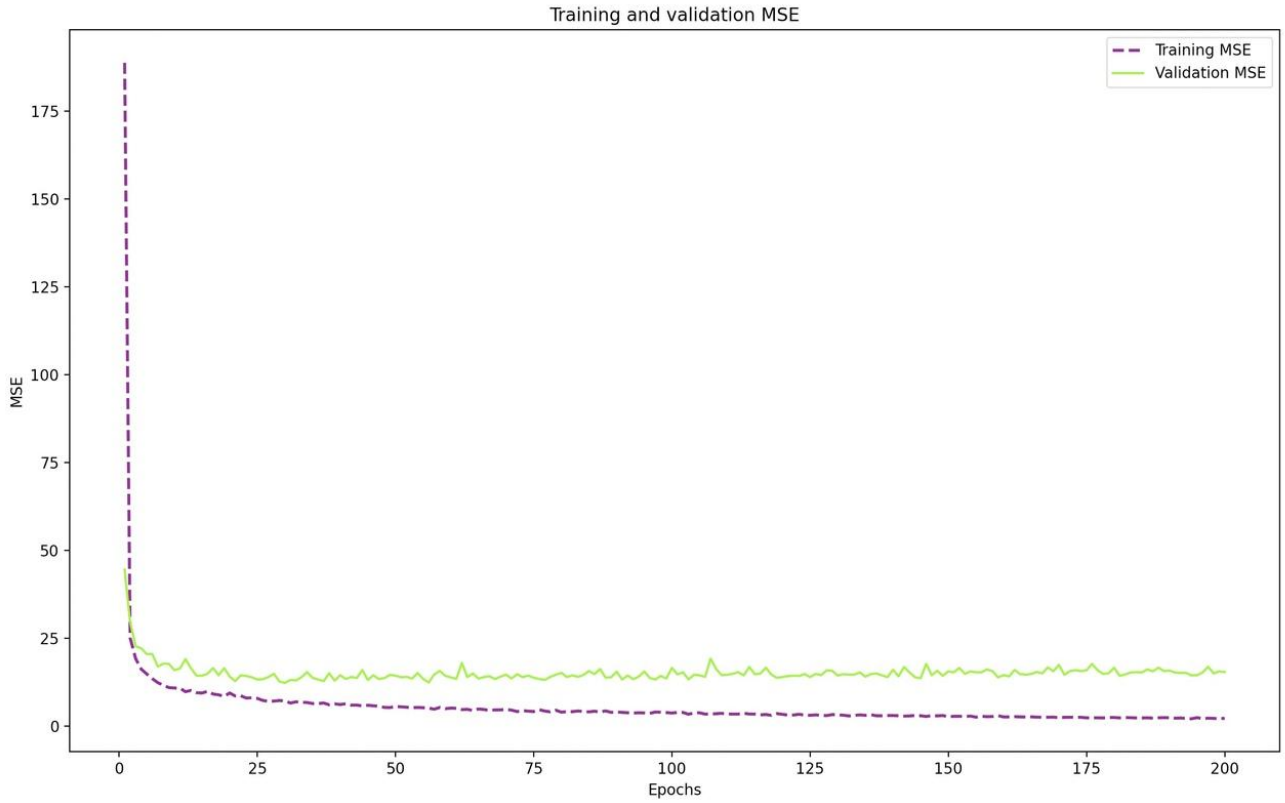
Среднее значение абсолютной ошибки на валидационных данных: 2.62

Графики средних квадратичных ошибок для 4х моделей при обучении и тестировании:

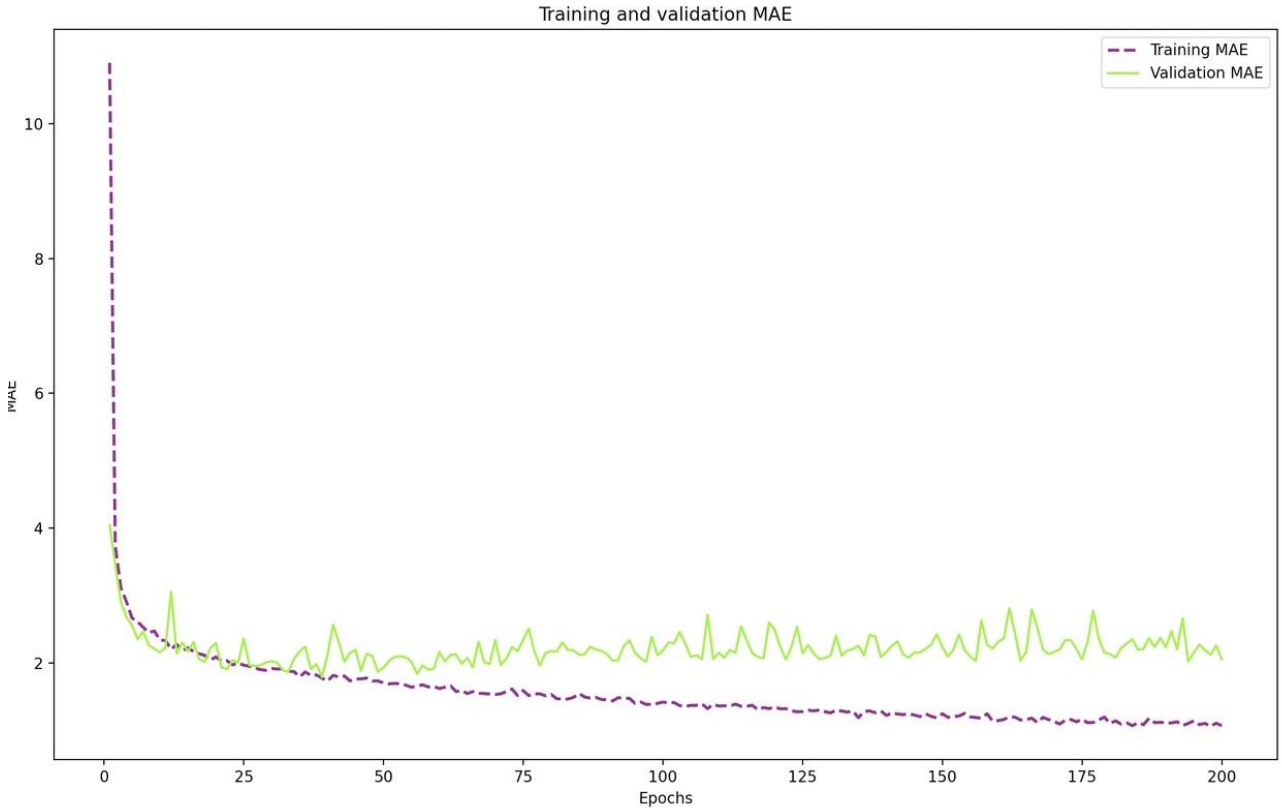


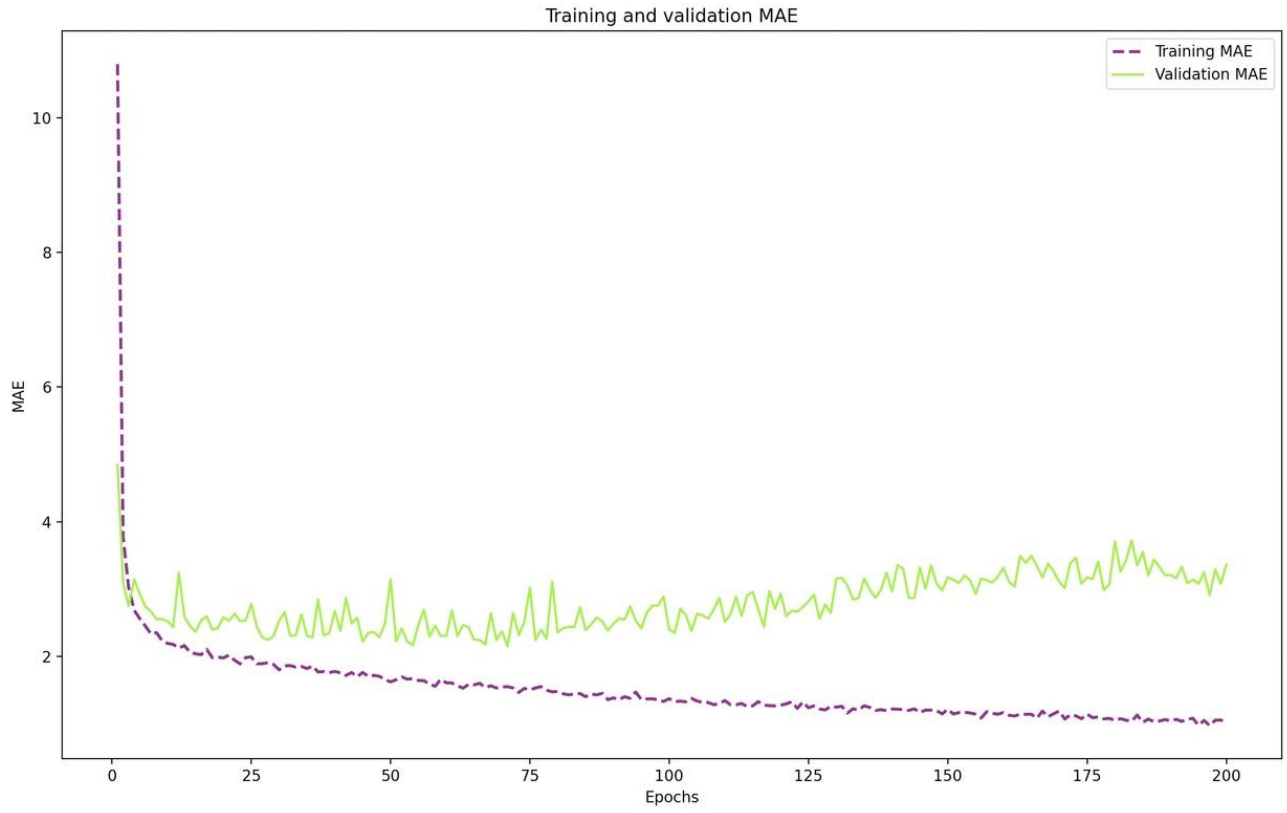

Training and validation MSE
Training MSE
Validation MSE
MSE
Epochs
0
50
100
150
200
0
25
50
75
100
125
150
175
200

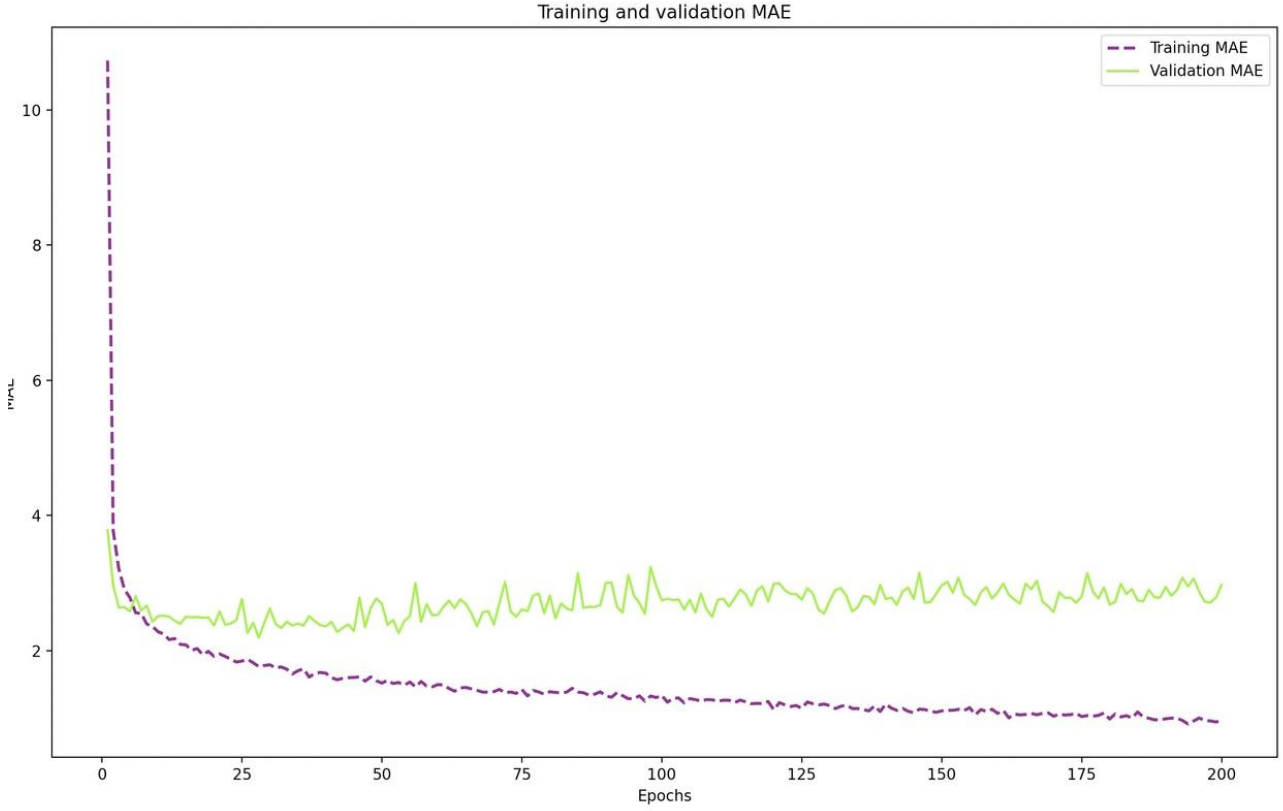

Training and validation MSE
Training MSE
Validation MSE
Epochs
0
50
100
150
200
0
25
50
75
100
125
150
175
200



Графики средних абсолютных ошибок для 4х моделей при обучении и тестировании:




Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs
0
25
50
75
100
125
150
175
200
2
4
6
8
10


Training and validation MAE
Training MAE
Validation MAE
Epochs
0
25
50
75
100
125
150
175
200
2
4
6
8
10

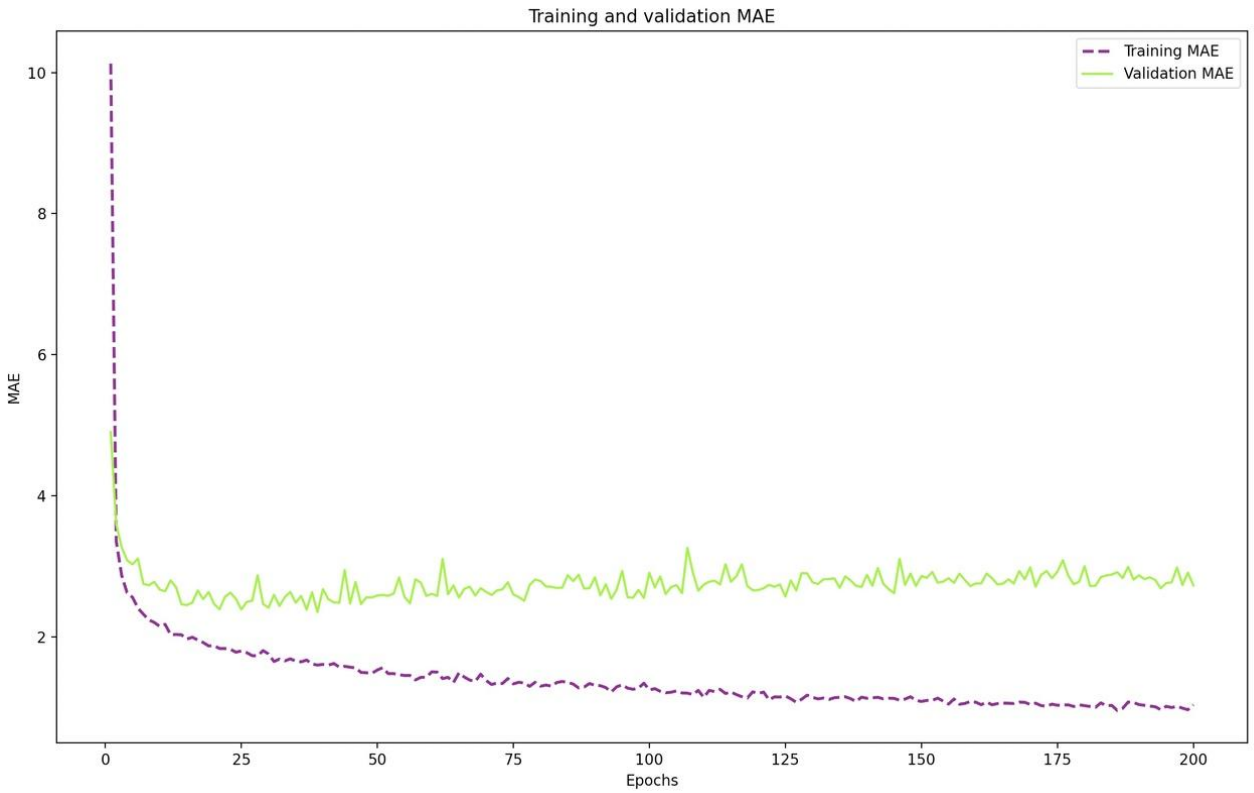
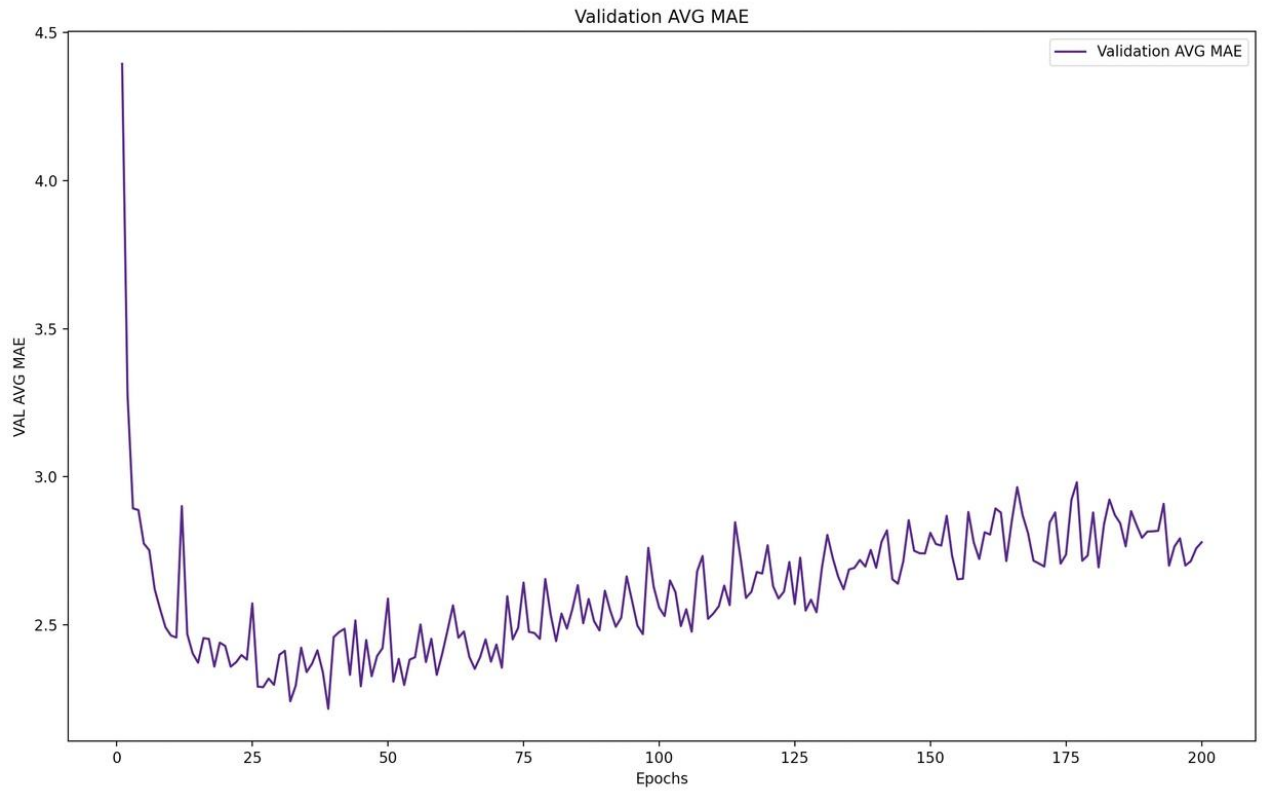


График среднего значения средней абсолютной ошибки на контрольных данных по всем моделям:



Выводы:

Среднее значение абсолютной ошибки на валидационных данных возросло практически на 2 десятых, по сравнению с предыдущими данными.

Как видно из графиков, значения средних квадратичных и средних абсолютных ошибок на данных для обучения уменьшились, что логично, так как количество эпох на обучение увеличилось вдвое. При этом, те же величины на контрольных данных увеличились. Наибольшее значения ошибок достигается на модели под номером 2.

Из графика среднего значения средней абсолютной ошибки на контрольных данных по всем моделям можно сделать вывод, что в диапазоне с 35-45 эпоху происходит переобучение сети - значение ошибки начинает стремительно возрастать.

Таким образом, как и ожидалось, увеличение количества эпох для обучения негативно сказалось на работе нейронной сети - происходит переобучение.

Уменьшим значение количества эпох в 2 раза относительно начальной величины.

Эксперимент 3

Параметры обучения и перекрестной проверки:

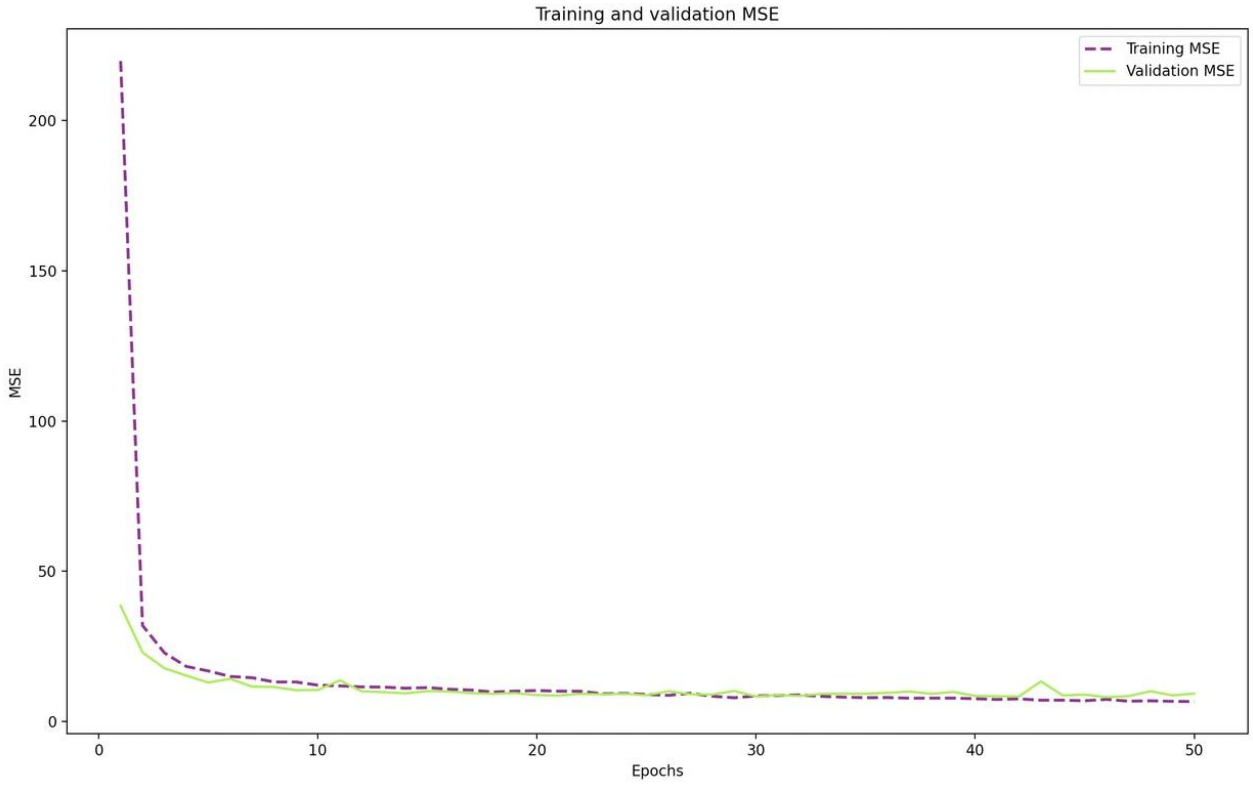
Количество эпох: 50

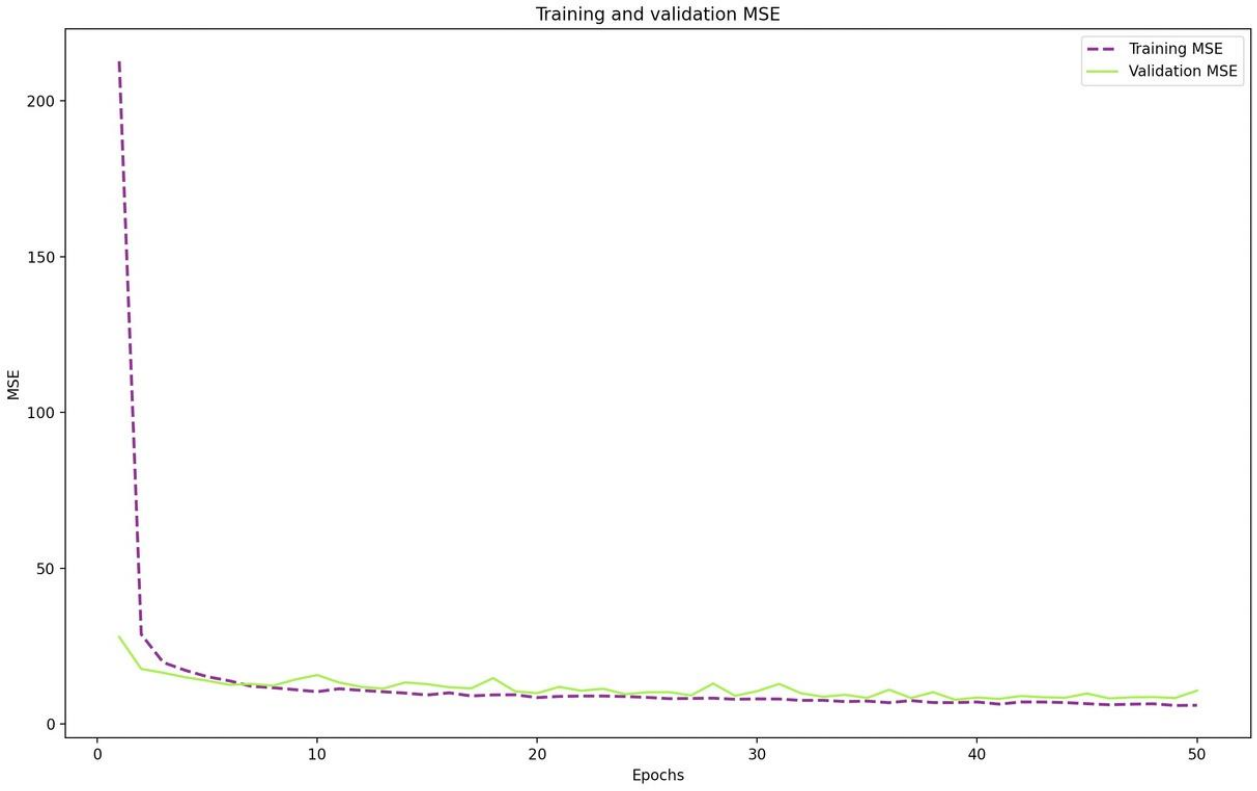
Количество блоков : 4

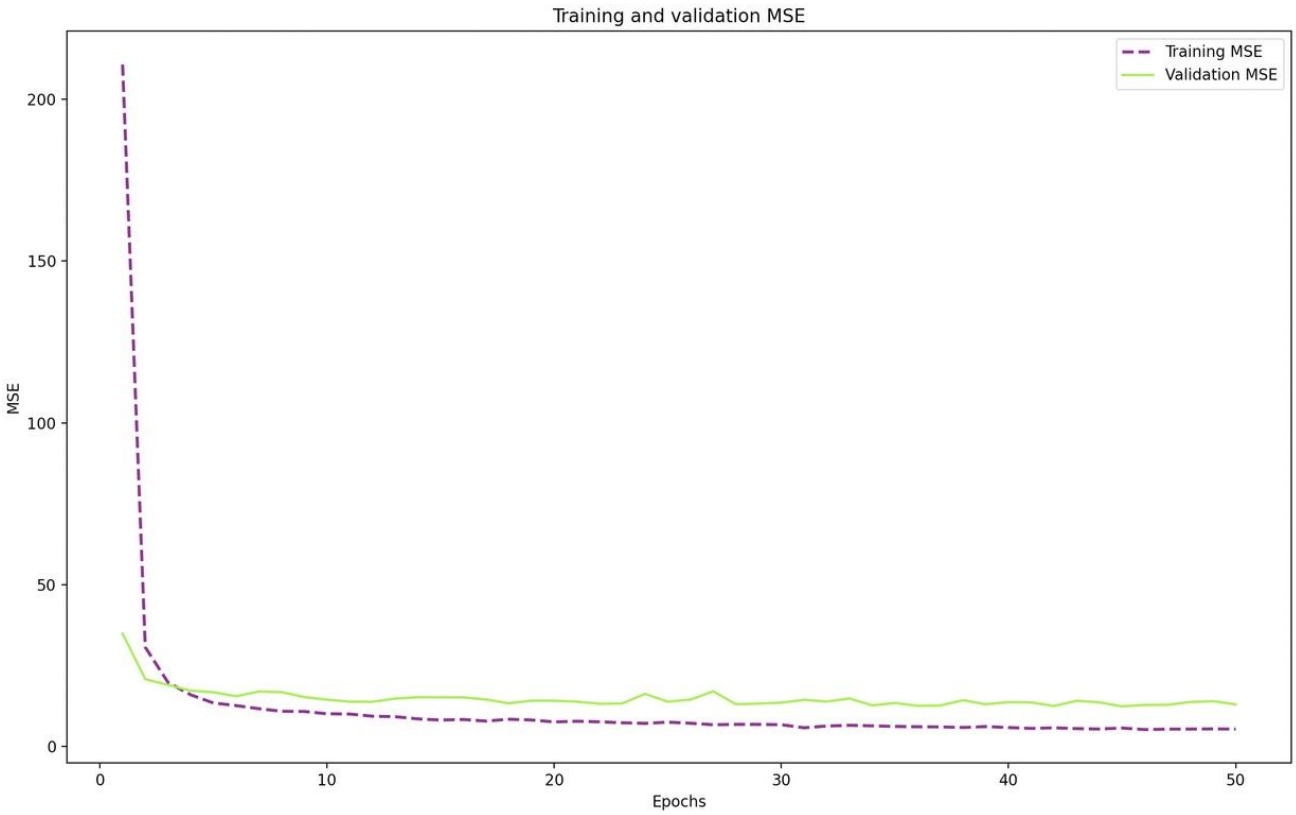
Результаты запуска:

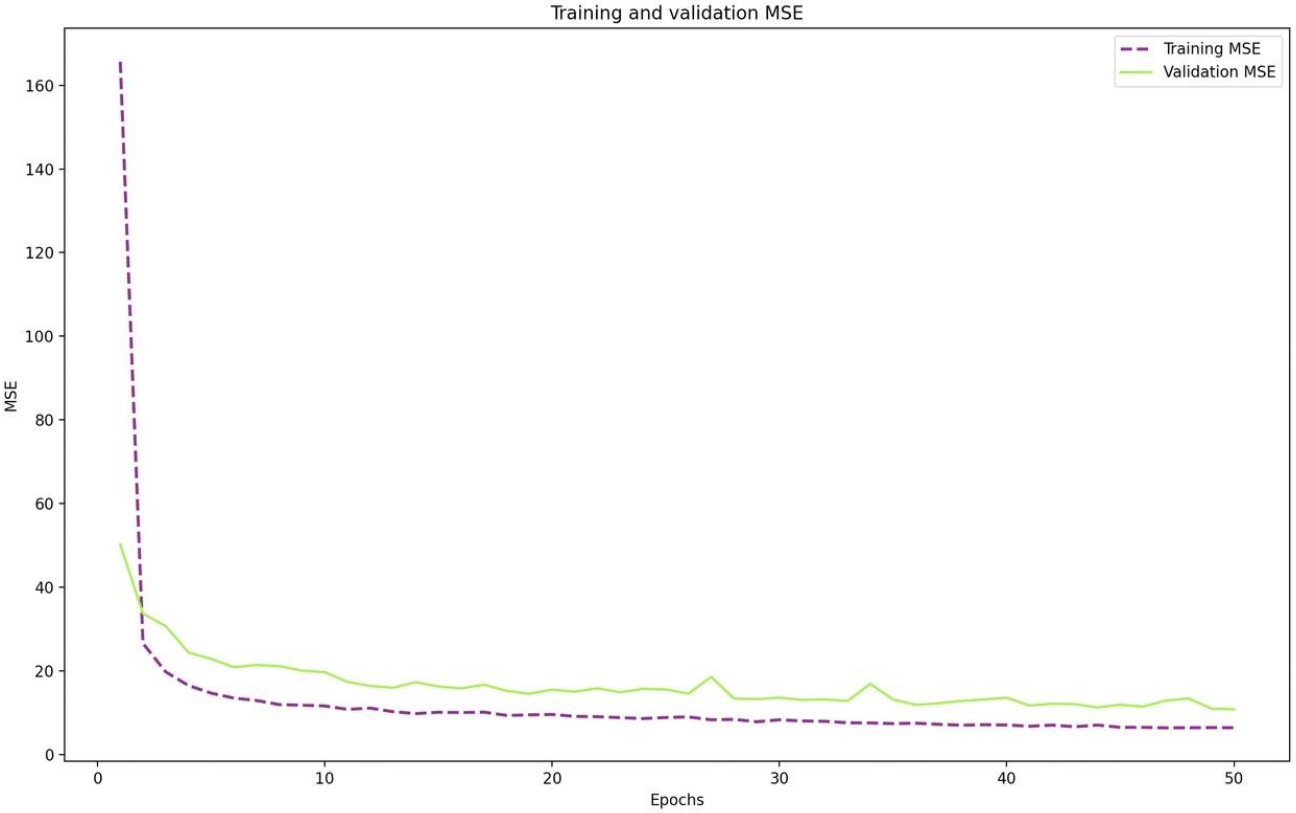
Среднее значение абсолютной ошибки на валидационных данных: 2.48

Графики средних квадратичных ошибок для 4х моделей при обучении и тестировании:

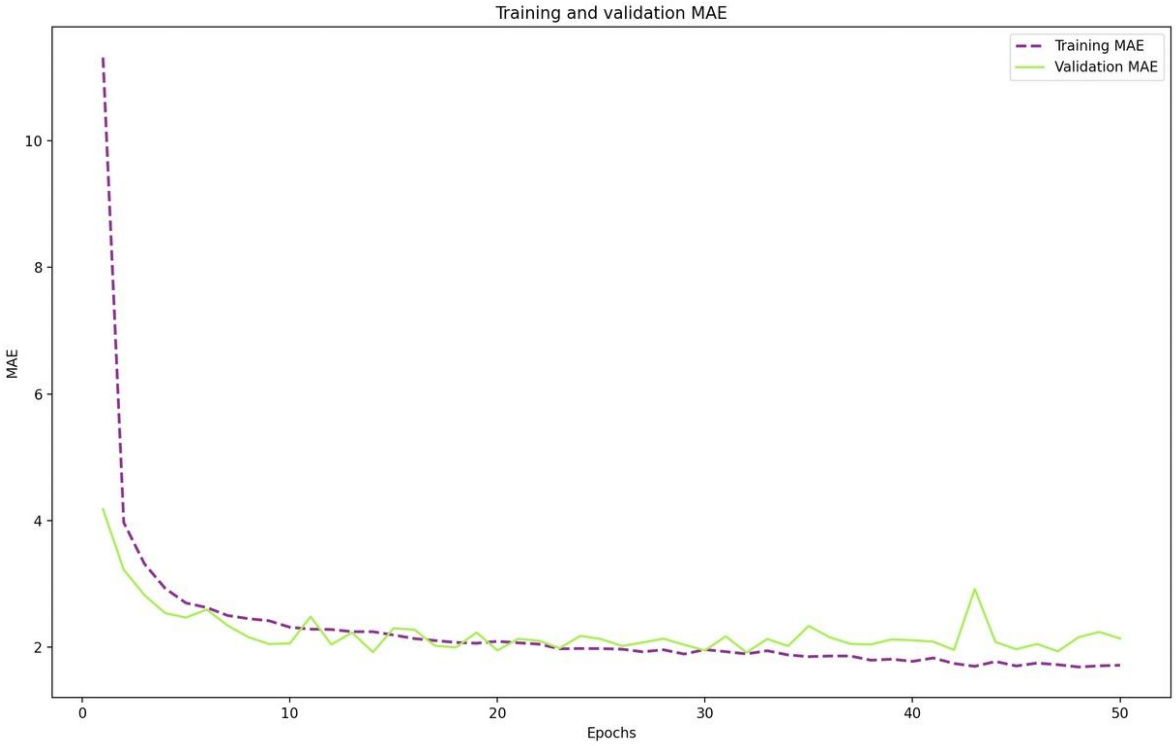


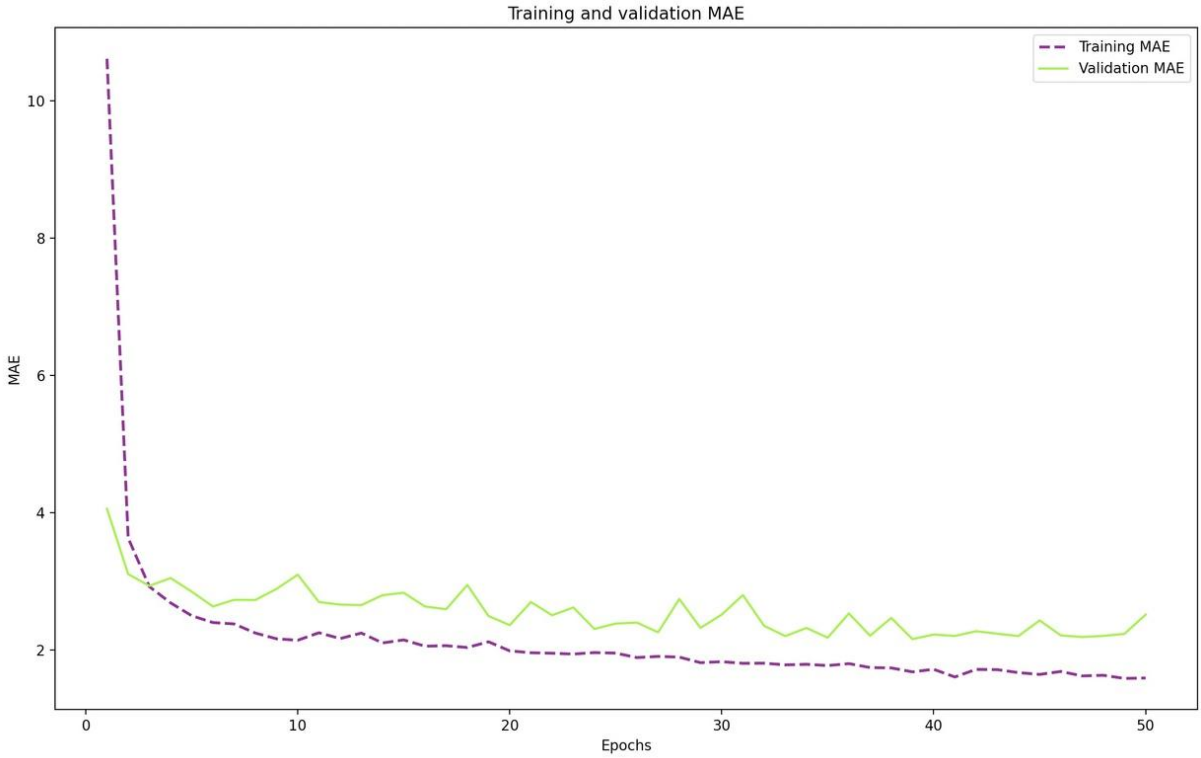


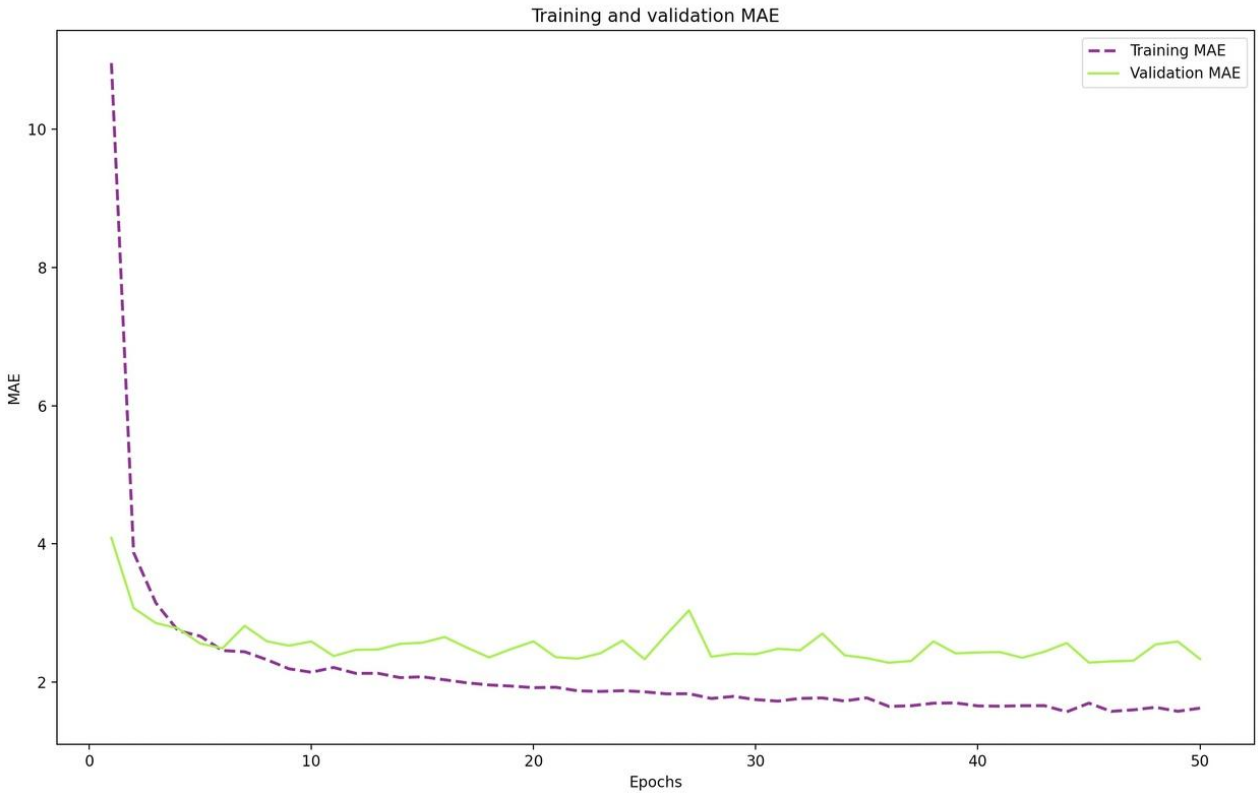

Training and validation MSE
Training MSE
Validation MSE
MSE
Epochs
0
50
100
150
200
0
10
20
30
40
50


Training and validation MSE
Training MSE
Validation MSE
MSE
Epochs
0
20
40
60
80
100
120
140
160
0
10
20
30
40
50

Графики средних абсолютных ошибок для 4х моделей при обучении и тестировании:






Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs
0
10
20
30
40
50
2
4
6
8
10

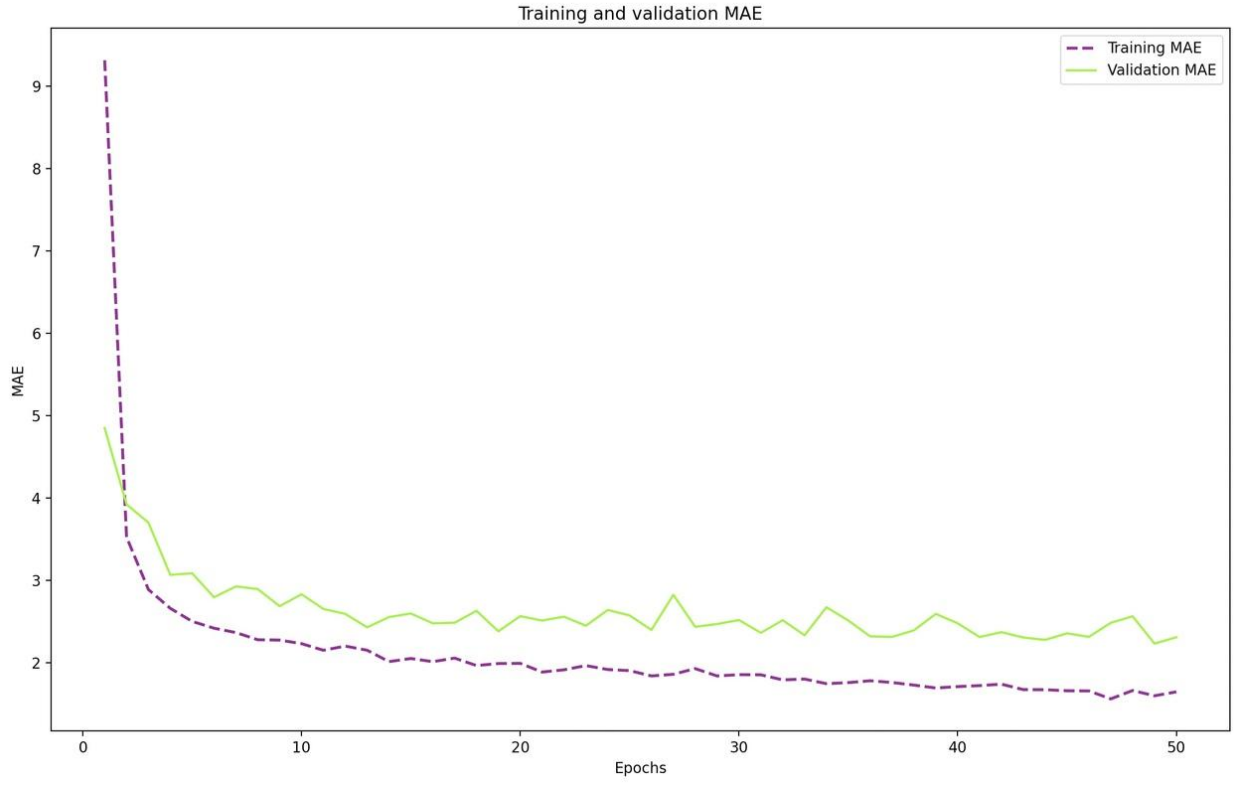
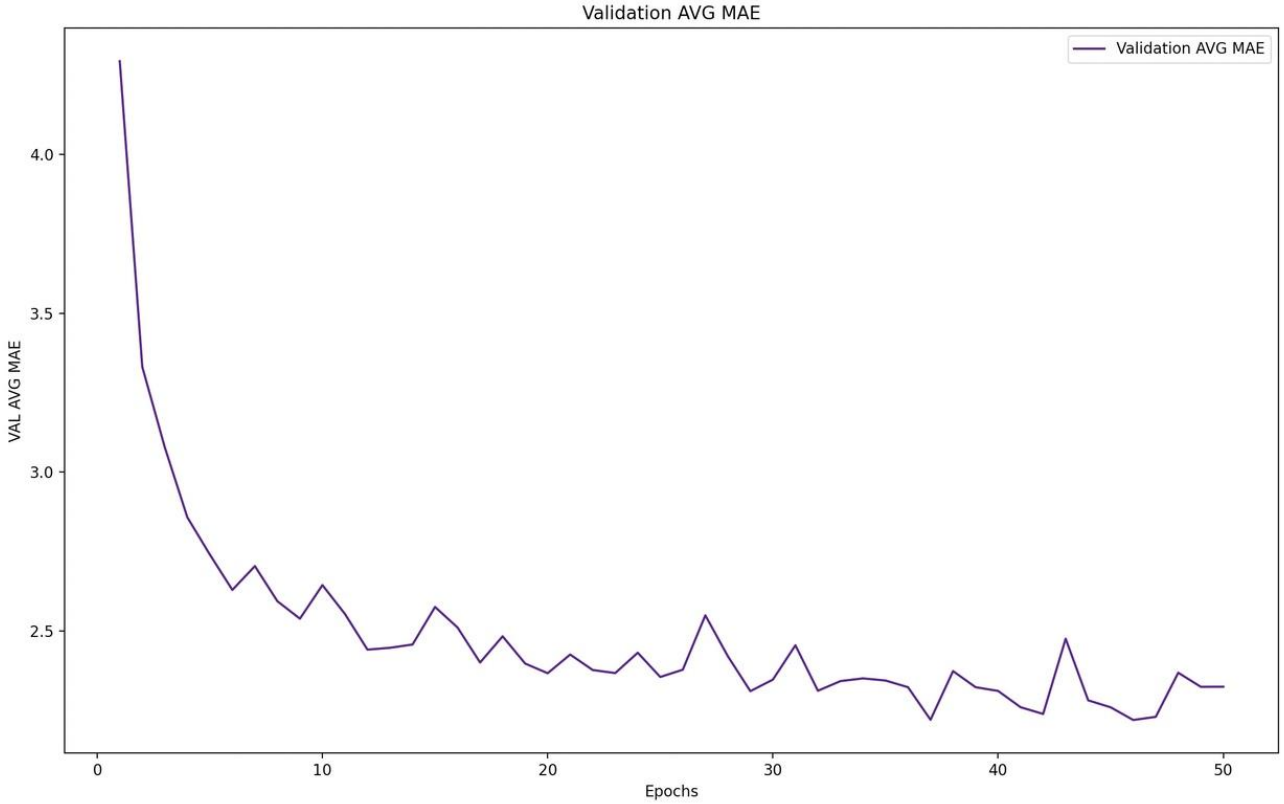

Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs
0
10
20
30
40
50
2
3
4
5
6
7
8
9

График среднего значения средней абсолютной ошибки на контрольных данных по всем моделям:



Выводы:

Среднее значение абсолютной ошибки на валидационных данных по сравнению с тем же показателем при 100 эпохах уменьшилось на одну десятую.

Из графиков средних квадратичных ошибок видно, что значения ошибок уменьшились по сравнению с моделями из проведенных ранее экспериментов.

Из графиков средних абсолютных ошибок видно, что в целом возрастания ошибок не происходит, хотя некоторые скачки имеют место быть.

В целом, обращаясь к графику среднего значения средней абсолютной ошибки на контрольных данных по всем моделям, можно сделать вывод, что значение ошибки убывает и на данный момент проведенный эксперимент показывает наилучшие результаты. Точка переобучения выявлена не была.

При уменьшении количества эпох до 25 явных улучшений выявлено не было. Оставим значение 50 в качестве оптимального. Попробуем изменить количество блоков для перекрестной проверки.

Эксперимент 4

Параметры обучения и перекрестной проверки:

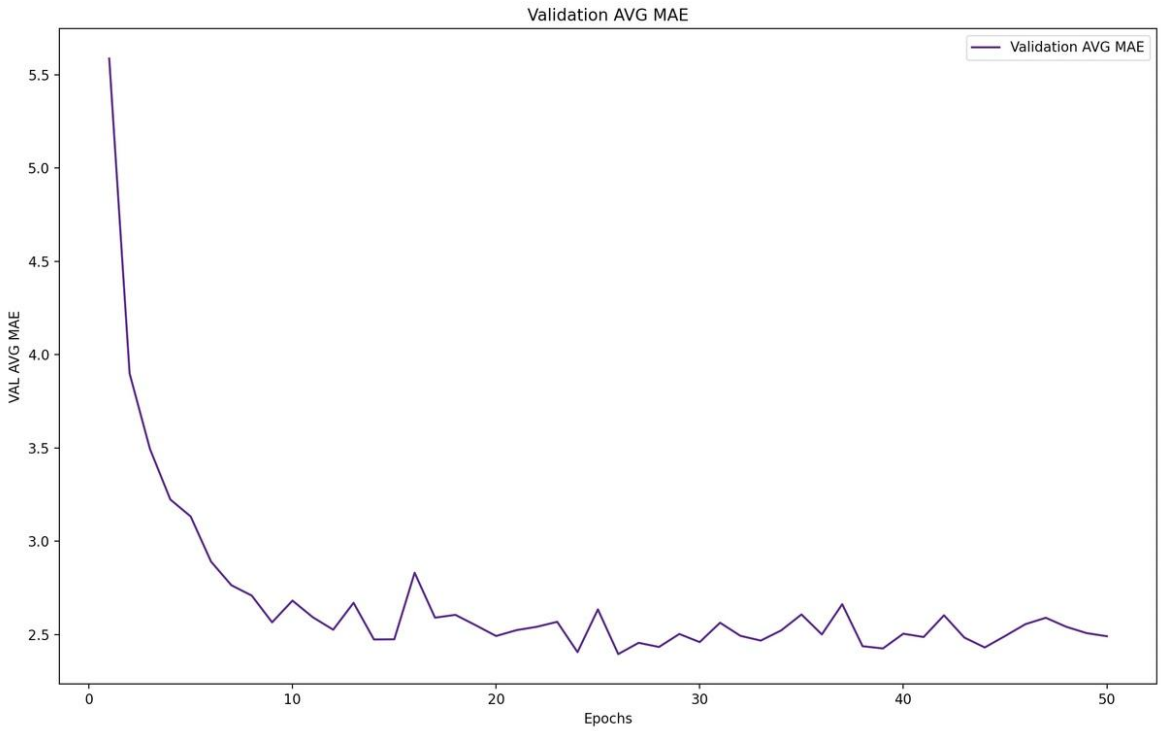
Количество эпох: 50

Количество блоков : 2

Результаты запуска:

Среднее значение абсолютной ошибки на валидационных данных: 2.68

График среднего значения средней абсолютной ошибки на контрольных данных по всем моделям:



Выводы:

Среднее значение средней абсолютной ошибки на контрольных данных увеличилось на 2 десятые. Ухудшение результатов связано, вероятно, с тем, что 50% данных недостаточно для полноценного обучения нейронной сети.

Эксперимент 5

Параметры обучения и перекрестной проверки:

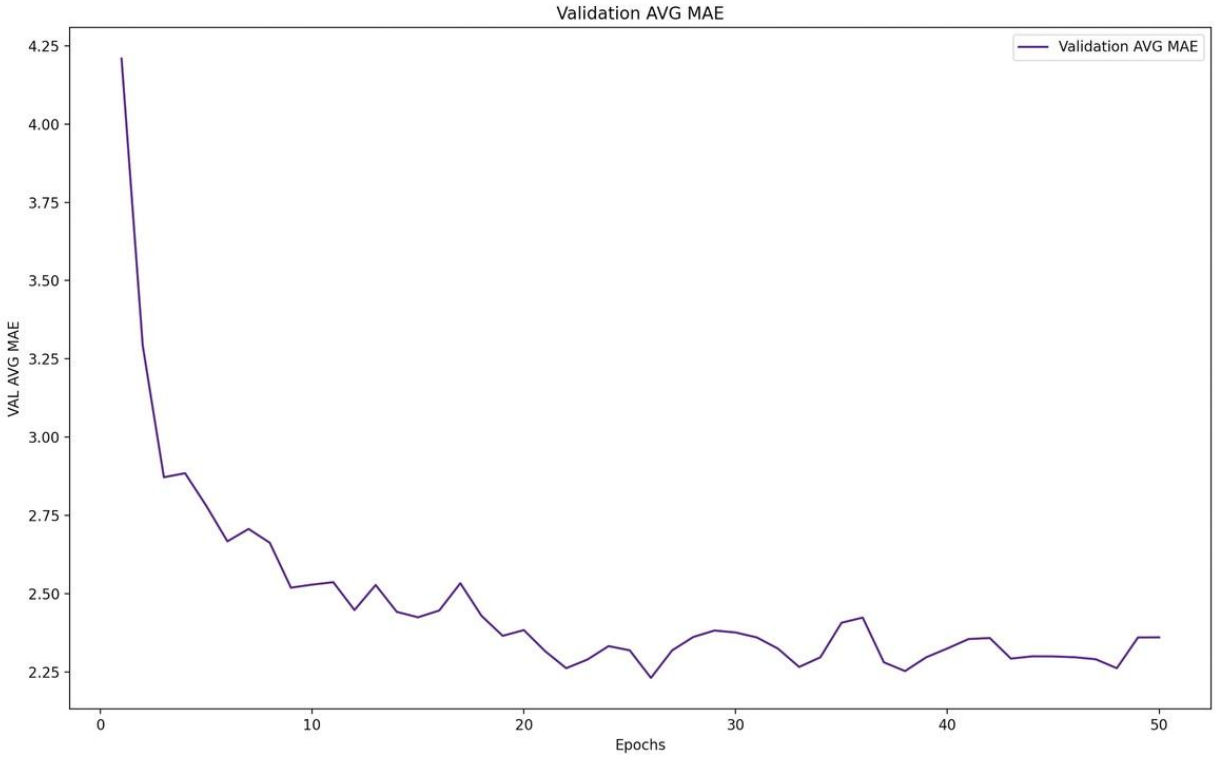
Количество эпох: 50

Количество блоков : 8

Результаты запуска:

Среднее значение абсолютной ошибки на валидационных данных: 2.46

График среднего значения средней абсолютной ошибки на контрольных данных по всем моделям:



Выводы:

Среднее значение абсолютной ошибки на контрольных данных уменьшилось на 2 сотые, что вряд ли можно считать наиболее успешным результатом. Достоверность результата является сомнительной, так как количество валидационных данных составляет всего лишь 0.125% от всех данных (изначальный объем которых невелик).

Таким образом, количество блоков равное четырем является оптимальным из имеющихся.

Выводы

В результате выполнения лабораторной работы был реализован механизм предсказания медианной цены на дома в пригороде Бостона в середине 1970-х. Была изучена задача регрессии и выявлены ее основные отличия от задачи классификации. Было изучено влияние количества эпох на результат обучения моделей, определена точка переобучения, применена перекрестная проверка по K блокам. Были построены соответствующие графики ошибок и точности во время обучения моделей, а также усредненные графики по всем моделям. Была выбрана оптимальная модель, дающая наилучшие результаты (50 эпох для обучения и 4 блока для перекрестной проверки).