Собственная реализация

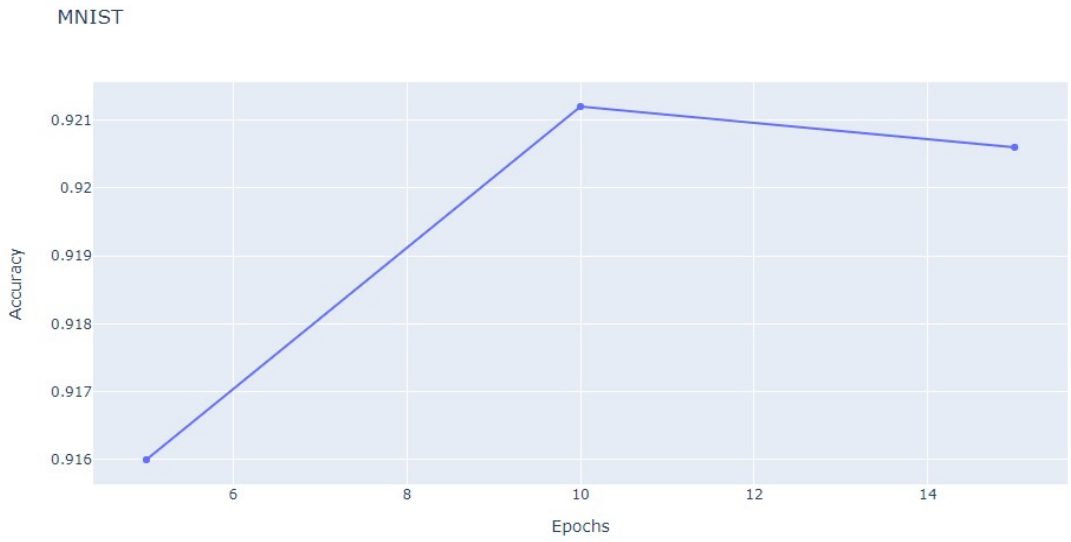
Реализован фреймворк с реализацией класса нейронной сети, позволяющий собирать с помощью передачи параметров перцептрон. (реализованы функции активации – сигмоидная, гиперболический тангенс, softmax).

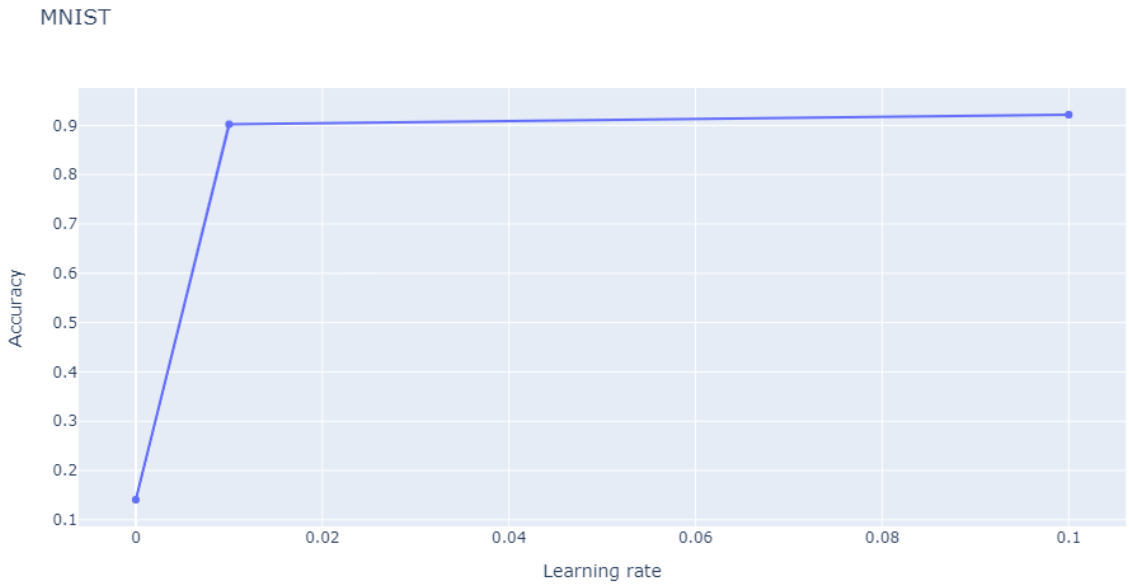
MNIST

Обучение модели происходит следующим образом:

```
small_nns.append(NN(sizes=[784, 10], activations=[softmax], epochs=e, l_rate=1e-1))
small_nns[-1].train(x_train, y_train, x_test, y_test)
```

Рассмотрены различные количества эпох и шага оптимизатора (learning rate). Выведена зависимость:



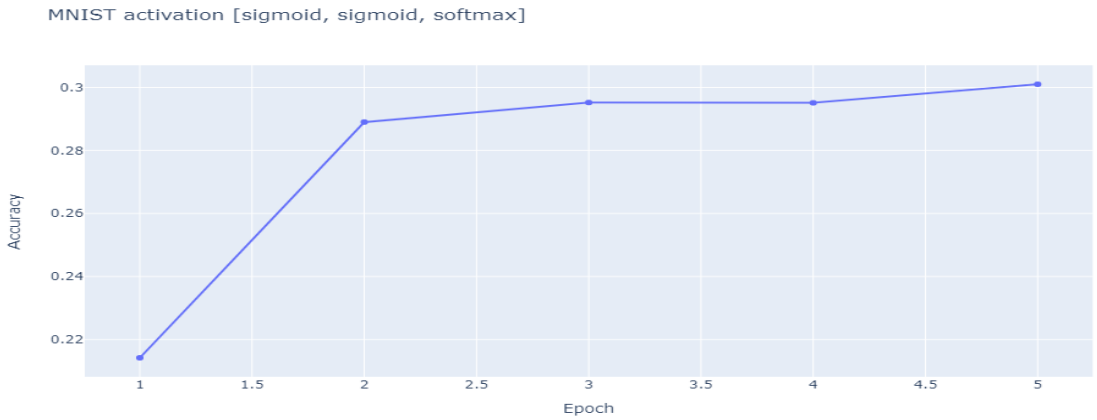


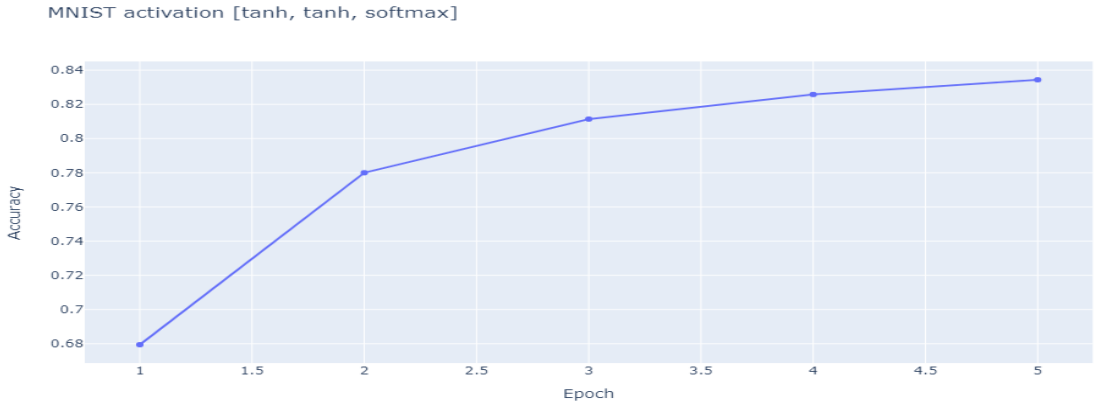
Очевидна зависимость от количества эпох – при увеличении количества эпох увеличивается и качество модели. С другой стороны, слишком маленький шаг показывает очень плохие результаты, сеть недообучена, при увеличении после learning rate = 0.01 особенного улучшения результата не замечено.

Матрицы ошибок:

```
Epochs = 5, l_rate = 0.1
[[ 964    0    1    1    0    2    8    1    3    0]
 [   0 1114    4    2    1    1    4    1    7    1]
 [   9   15  905   25    8    5   11   12   31   11]
 [   5    1   16  912    1   40    1    6   16   12]
 [   2    4    3    1  899    0   11    1    5   56]
 [  10    3    4   27   13  778   12    6   31    8]
 [  13    4    4    1    6   23  903    2    2    0]
 [   3   12   20   12    7    2    1  925    4   42]
 [   8   14    4   32    8   42   14    8  832   12]
 [   9    6    0    9   21   14    1   13    6  930]]

Epochs = 5, l_rate = 0.01
[[ 944    0    1    4    2   12   10    1    5    1]
 [   0 1099    4    6    1    2    4    0   19    0]
 [  12   12  868   21   17    3   19   22   44   14]
 [   5    1   24  870    2   47    7   14   27   13]
 [   2    5    2    2  879    0   17    1   11   63]
 [  21    6    7   59   19  692   19   11   42   16]
 [  21    3   13    1    9   13  884    3   11    0]
 [   2   13   26   11    8    1    2  914    7   44]
 [  11    7   10   42   13   33   12   20  806   20]
 [  10    6    7   12   50   10    2   24   16  872]]

Epochs = 5, l_rate = 1e-07
[[  1 747   3  39 126   3  10  48   0   3]
 [154 715 138   0  18   4   0  67  36   3]
 [ 31 904  19   9   5   0  10  41  12   1]
 [ 12 830  63   5  14   0   9  53   6  18]
 [ 50 674   2   0 127   7  34  66  16   6]
 [  5 587 149  21  16   1   3  93   4  13]
 [ 55 716   9  20  15   0  31  78  31   3]
 [ 50 379   4 113  87   6   9 245  16 119]
 [ 12 816  25   0  50   0   0  61   6   4]
 [ 21 543  10   5  72   2   8 308  15  25]]

Epochs = 10, l_rate = 0.1
[[ 944    0    1    4    2   12   10    1    5    1]
 [   0 1099    4    6    1    2    4    0   19    0]
 [  12   12  868   21   17    3   19   22   44   14]
 [   5    1   24  870    2   47    7   14   27   13]
 [   2    5    2    2  879    0   17    1   11   63]
 [  21    6    7   59   19  692   19   11   42   16]
 [  21    3   13    1    9   13  884    3   11    0]
 [   2   13   26   11    8    1    2  914    7   44]
 [  11    7   10   42   13   33   12   20  806   20]
 [  10    6    7   12   50   10    2   24   16  872]]

Epochs = 10, l_rate = 1e-07
[[ 960    0    1    2    1    2   11    1    2    0]
 [   0 1114    3    2    1    1    4    2    8    0]
 [  10   15  903   23    7    5   13   11   35   10]
 [   3    1   15  911    1   36    3   10   17   13]
 [   2    5    2    1  896    0   12    3    8   53]
 [  11    2    1   27   10  781   13    7   33    7]
 [  18    4    3    2    5   17  904    2    3    0]
 [   1   10   17   12    7    0    0  936    7   38]
 [   8   13    5   27    8   36   11    7  843   16]
 [   9    4    0    7   19   14    1   14    9  932]]

Epochs = 15, l_rate = 0.1
[[  1 747   3  39 126   3  10  48   0   3]
 [154 715 138   0  18   4   0  67  36   3]
 [ 31 904  19   9   5   0  10  41  12   1]
 [ 12 830  63   5  14   0   9  53   6  18]
 [ 50 674   2   0 127   7  34  66  16   6]
 [  5 587 149  21  16   1   3  93   4  13]
 [ 55 716   9  20  15   0  31  78  31   3]
 [ 50 379   4 113  87   6   9 245  16 119]
 [ 12 816  25   0  50   0   0  61   6   4]
 [ 21 543  10   5  72   2   8 308  15  25]]

Epochs = 15, l_rate = 0.01
[[ 960    0    1    2    1    2   11    1    2    0]
 [   0 1114    3    2    1    1    4    2    8    0]
 [  10   15  903   23    7    5   13   11   35   10]
 [   3    1   15  911    1   36    3   10   17   13]
 [   2    5    2    1  896    0   12    3    8   53]
 [  11    2    1   27   10  781   13    7   33    7]
 [  18    4    3    2    5   17  904    2    3    0]
 [   1   10   17   12    7    0    0  936    7   38]
 [   8   13    5   27    8   36   11    7  843   16]
 [   9    4    0    7   19   14    1   14    9  932]]

Epochs = 15, l_rate = 1e-07
[[ 950    0    3    2    1   10    9    2    3    0]
 [   0 1100    1    5    1    2    5    2   19    0]
 [  18    7  887   19   15    1   13   19   48    5]
 [   4    0   21  899    1   41    4   15   17    8]
 [   2    4    2    0  897    2   10    3   11   51]
 [  14    4    6   35   20  737   16    8   43    9]
 [  16    3    5    2   15   18  892    0    7    0]
 [   4   14   32    6   10    0    0  918    4   40]
 [   7    8    8   25   16   31   16   17  837    9]
 [   8    4    5   14   39   13    1   28   12  885]]
```

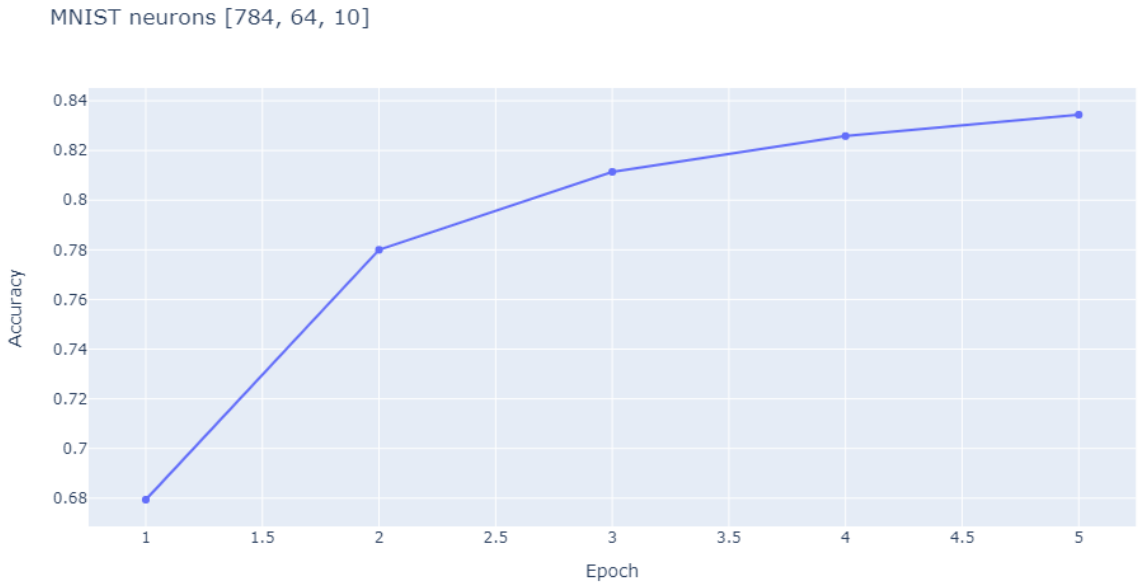
Для трехслойного персептрона рассмотрена зависимость от вида активационной функции.

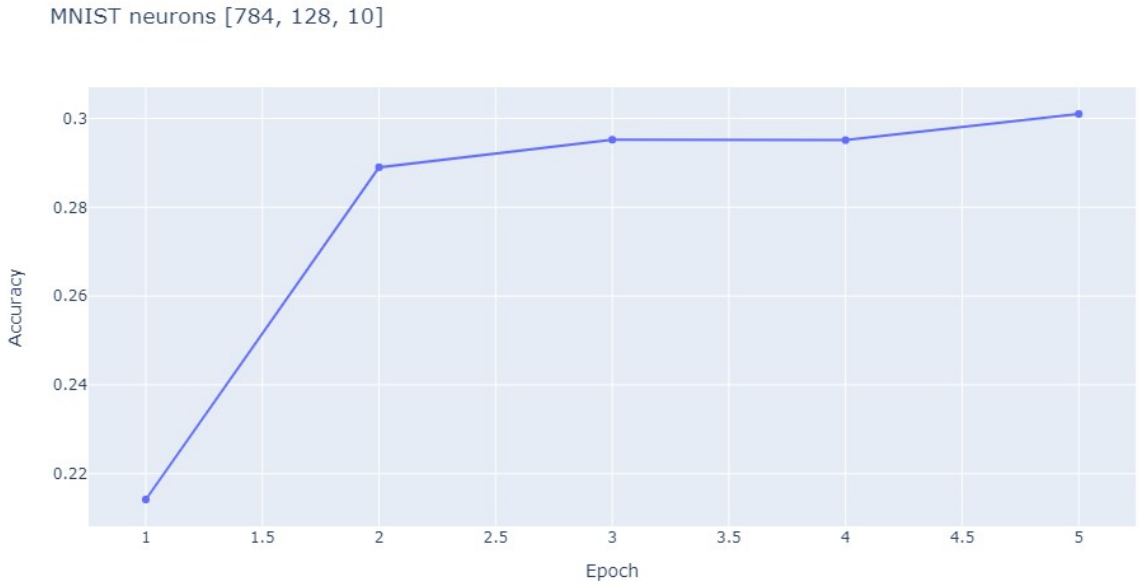




Из этого следует вывод, что функция активация при составлении модели важна.

Для двухслойного персептрона рассмотрена зависимость от количества нейронов в слоях:





FashionMNIST

Получены результаты обучения и матрица ошибок:

```
small_nns.append(NN(sizes=[784, 128, 128, 10], activations = [tanh, tanh, softmax], epochs=e, l_rate=l))
small_nns[-1].train(x_train, y_train, x_test, y_test)
```

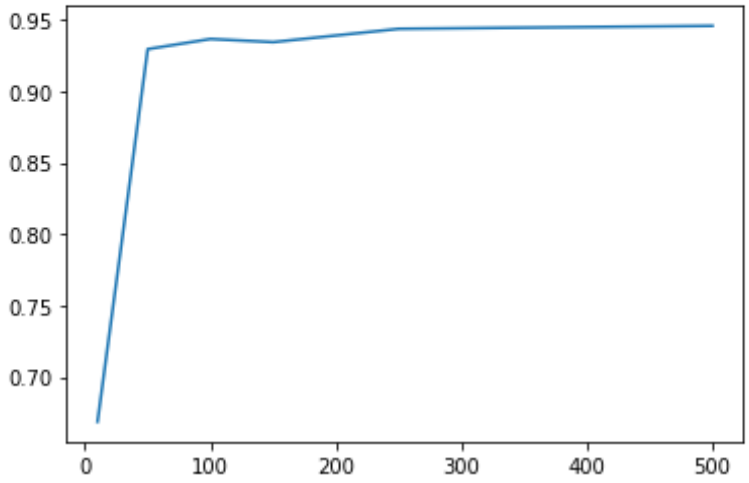
```
Epoch: 1, Time Spent: 113.48s, Accuracy: 78.11%
Epoch: 2, Time Spent: 245.94s, Accuracy: 90.09%
Epoch: 3, Time Spent: 375.25s, Accuracy: 92.09%
Epoch: 4, Time Spent: 507.09s, Accuracy: 92.96%
Epoch: 5, Time Spent: 646.05s, Accuracy: 93.67%
Epoch: 6, Time Spent: 758.97s, Accuracy: 94.27%
Epoch: 7, Time Spent: 864.99s, Accuracy: 94.58%
Epoch: 8, Time Spent: 971.91s, Accuracy: 94.83%
Epoch: 9, Time Spent: 1076.54s, Accuracy: 95.14%
Epoch: 10, Time Spent: 1182.88s, Accuracy: 95.36%
```

Реализация Pytorch

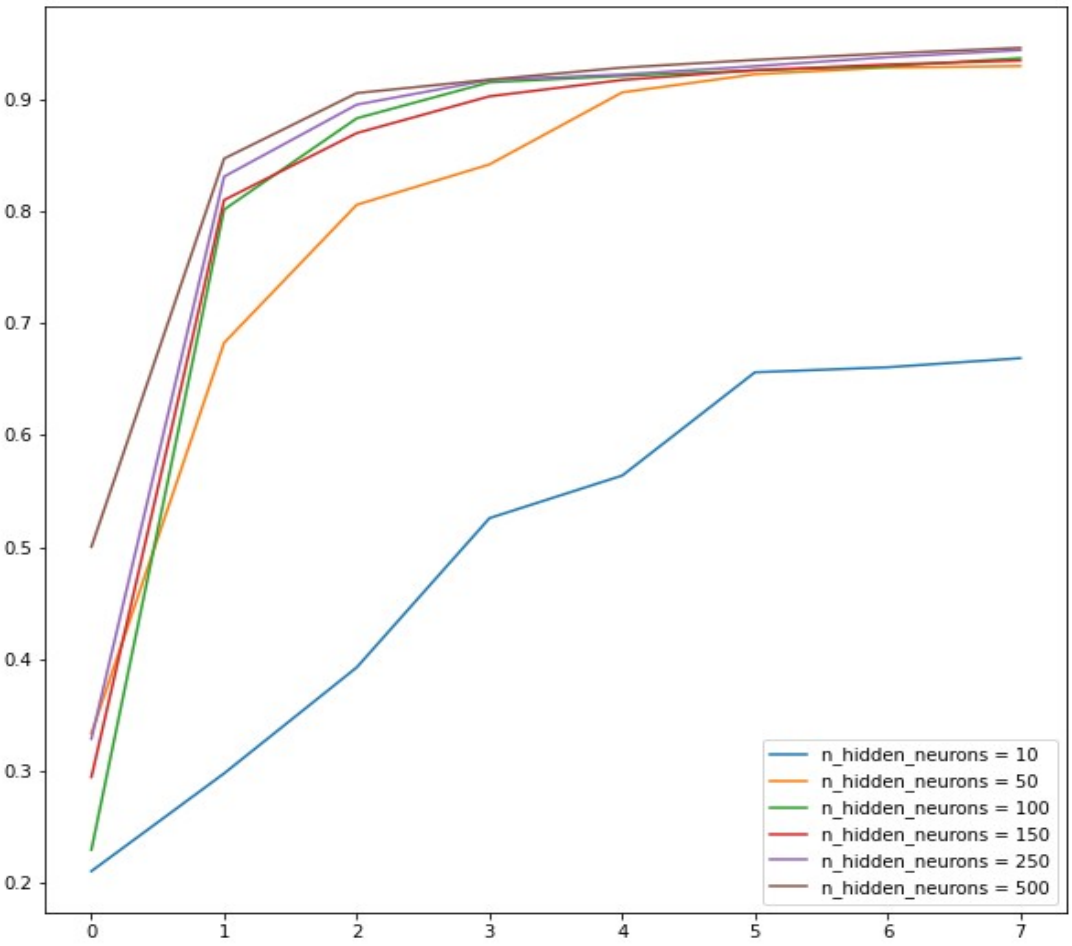
MNIST

```
# Зависимость точности от количества нейронов
import matplotlib.pyplot as plt
plt.plot(hiddens_neurons,accs)
```

```
[<matplotlib.lines.Line2D at 0x7f46dd739890>]
```



Точность на разных эпохах при разном количестве скрытых нейронов

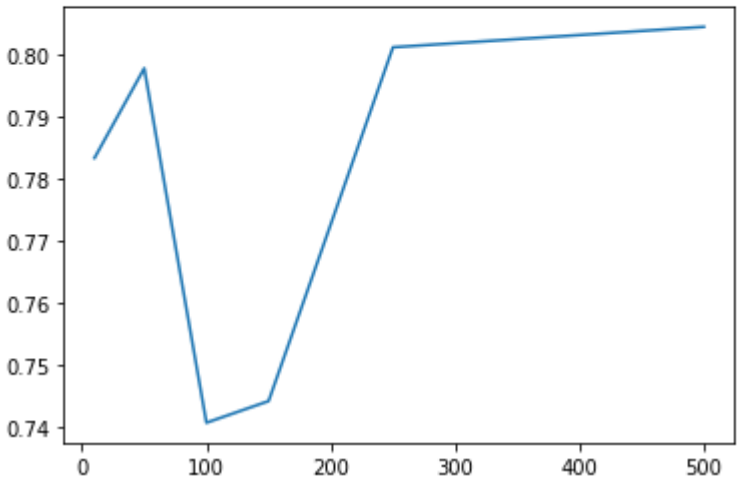


Смотря на графики можно сделать вывод, что после 100 нейронов в каждом скрытом слое точность не особо сильно возрастает. Касательно второго графика хочется заметить, что сеть с 500 нейронами в каждом скрытом слое на каждой эпохе достигает лучшей метрики. При этом начиная примерно с 3 эпохи рост метрики значительно замедляется для все сетей с количеством нейронов в скрытых слоях равных 100, 150, 250, 500. В конце хочется заметить, что большая часть рассмотренных сетей достигают неплохого показателя метрики. Это указывает на неплохую применимость архитектуры перцептронов для решения задачи классификация датасета MNIST

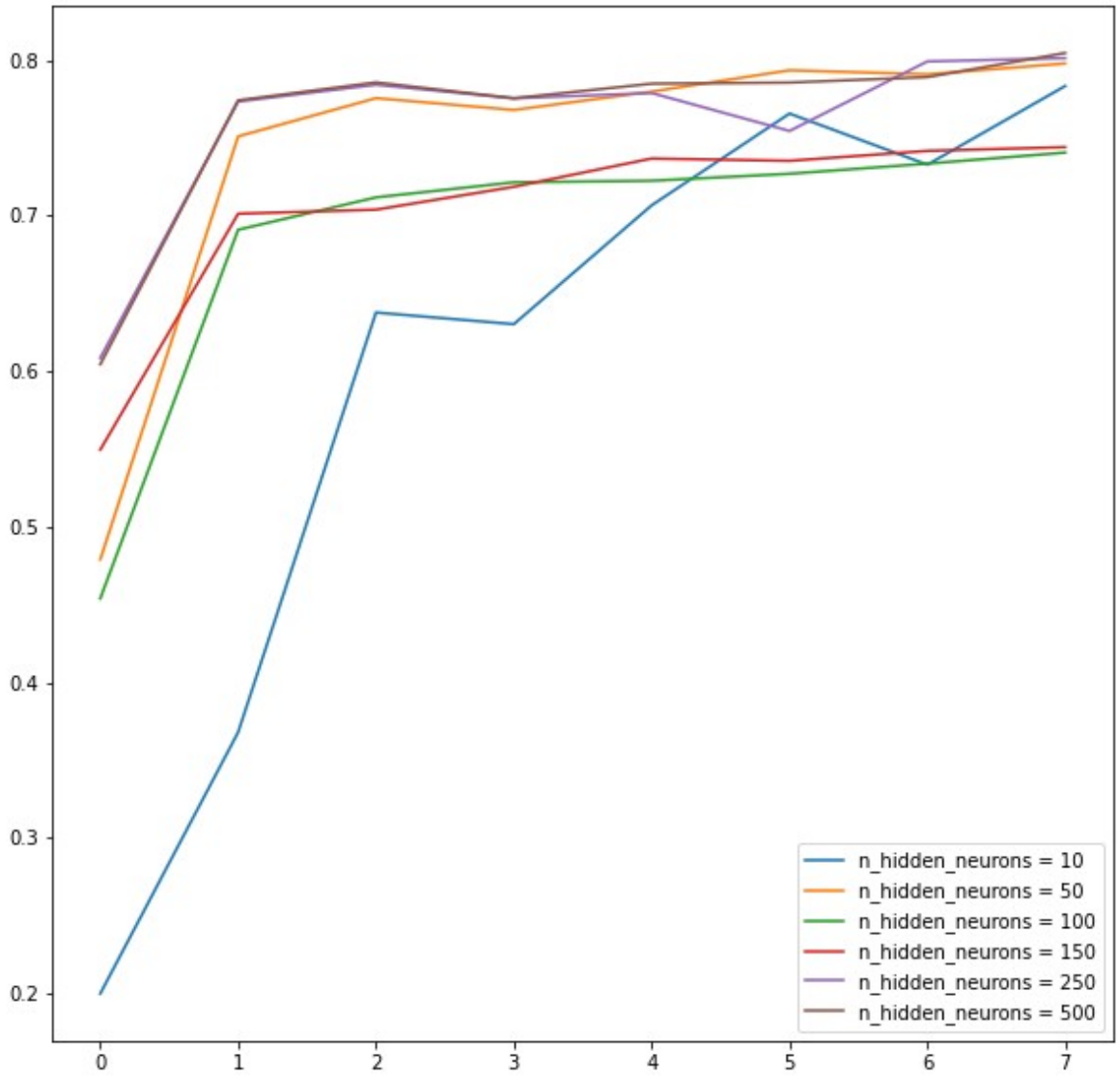
FashionMNIST

```
: 1 # Зависимость точности от количества нейронов
  2 import matplotlib.pyplot as plt
  3 plt.plot(hiddens_neurons,accs)
```

```
: [<matplotlib.lines.Line2D at 0x7f46dcf72d90>]
```



Точность на разных эпохах при разном количестве скрытых нейронов

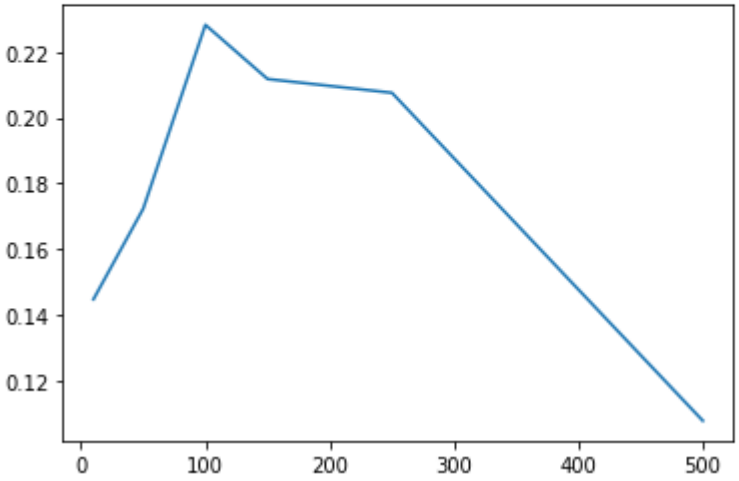


Смотря на графики можно сделать вывод, что почему от 100 до 150 нейронов наблюдается падения метрики. Это действительно странная аномалия(не критично падает, но все-таки падает). При этом далее от 200 до 500 нейронов идет постепенный медленный рост метрики. Касательно второго графика хочется заметить, что сеть с 500 нейронами в каждом скрытом слое на каждой эпохе достигает лучшей метрики. При этом начиная примерно с 1 эпохи рост метрики значительно замедляется для все сетей с количеством нейронов в скрытых слоях равных 50,100, 150, 250, 500. В конце хочется заметить, что большая часть рассмотренных сетей достигают неплохого показателя метрики. Это указывает на неплохую применимость архитектуры перцептронов для решения задачи классификация датасета Fmnist

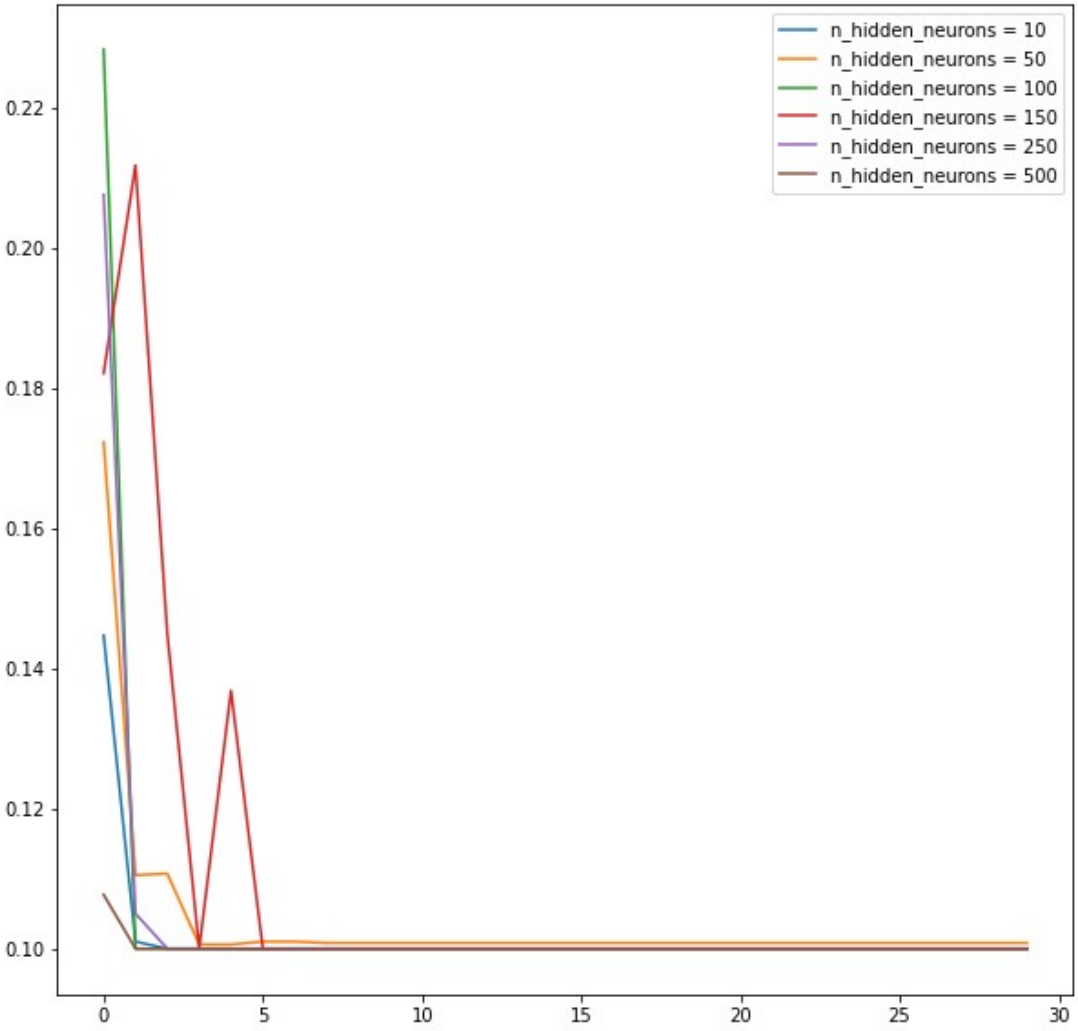
CIFAR10

```
]:  1 # Зависимость точности от количества нейронов
    2 import matplotlib.pyplot as plt
    3 plt.plot(hiddens_neurons,accs)
```

```
]: [<matplotlib.lines.Line2D at 0x7f46db27d210>]
```



Точность на разных эпохах при разном количестве скрытых нейронов



Для датасета CIFAR10 лучший показатель метрики достигается при количестве нейронов в каждом скрытом слое равном 100. Касательно зависимости метрики от количества эпох обучения хочется заметить, что метрика начинает уменьшаться после 1-3 эпох. Данный факт может говорить о возможном переобучении модели. Также метрика достигает всего-то 0.22 максимально. Это указывает о плохом потенциале использования архитектуры перцептрона для решения задачи классификации датасета CIFAR10. Метрику могло бы поднять добавление сверхточных слоев и макспулингов.