Gramian Angular Field + CNN

Отчет по дисциплине “Введение в искусственный интеллект”.

Задача по прогнозированию цен на акции с использованием сверточных нейронных сетей.

Выполнили студенты Санкт-Петербургского государственного университета, обучающиеся на направлении “Прикладная математика, фундаментальная информатика и программирование” группы 19.Б06-пу Минченко Никита и Шарова Ангелина.

Описание задачи:
На основе данных о ценах акции на фондовом рынке спрогнозировать направление движение цены (рост, падение, боковик) в следующий период времени. Данные о ценах акций берутся с помощью парсинга сайта Finam.ru, общий размер датасета около 1000000 записей, детализация данных подневная, данные содержат в себе цены открытия, закрытия торгов, максимальную и минимальную цены, объем торгов. Данные разделяются на обучающее, валидационное и тестовое множества по дате: обучающее - данные до 09.05.2018, около 750000 строк, валидационное - с 09.05.2018 до 09.11.2018 (около 50000 строк, тестовое с 09.11.2018 (около 250000 строк). Критерий качества - F1 - мера. Один обучающий пример включает в себя данные за 20 дней, а ответом к нему является направление изменения цены в следующие 5 дней.
Постановка задачи обучения: обучить сверточную нейронную сеть на преобразованных данных так, чтобы она давала применимый на практике результат и превосходила по критерию качества наивный классификатор, всегда предсказывающий 0 (боковик) - самый распространенный класс.
Работа выполнена на языке Python, для ускорения обучения сетей использовался облачный сервис Google Colab. Использовались библиотеки NumPy, Pandas, Matplotlib, PyTorch, PyTS, Scikit-Learn и другие.

Результаты наивного классификатора:

```
accuracy - 0.62
f1 по классам - 0.77, 0, 0
среднее f1 - 0.2552621015874848
```

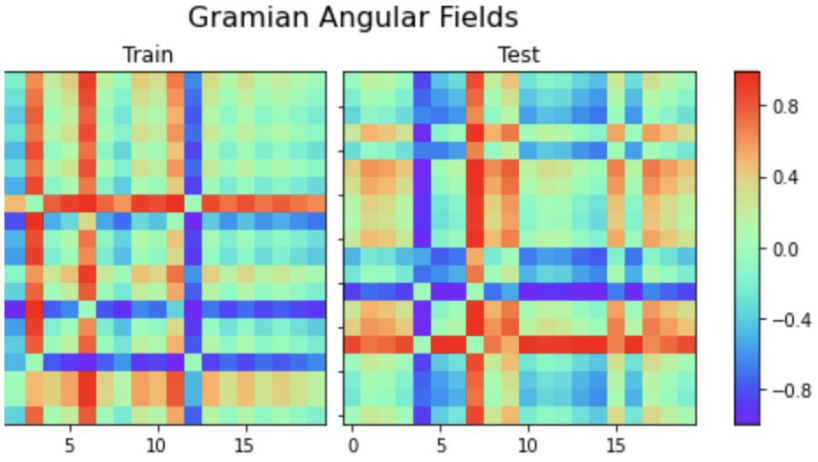
В ходе решения задачи мы сравнивали по критерию качества разные варианты предобработки данных и обучения сетей:

1) Базовая модель (выполнял Минченко Никита)

Предобработка данных:

1. Масштабирование данных путем приведения их к относительным значениям (деление цены каждого дня на предыдущий)
2. Преобразование данных к нужному формату.
   Один обучающий пример - цены закрытия за 20 дней, ответ к нему - одна из трех меток классов, 0 если $0.97 \leq p \leq 1.03,$ где p - произведение цен за следующие 5 дней в относительных значениях, то есть изменение цены не превосходит 3%, 1 класс - $p > 1.03,$ т.е. подъем цены более, чем на 3%, 2 класс - $0.97 < p,$ т.е. спад цены более чем на 3%.
3. Преобразование к двумерному виду, который пригоден для сверточной нейронной сети, с помощью Gramian Angular Difference Field. В процессе этого преобразования данные масштабируются к промежутку [-1, 1].

Примеры входных данных после обработки GADF



Архитектура нейронной сети: 3 сверточных и 4 полносвязных слоя с функцией активации ReLu и батч-нормализацией

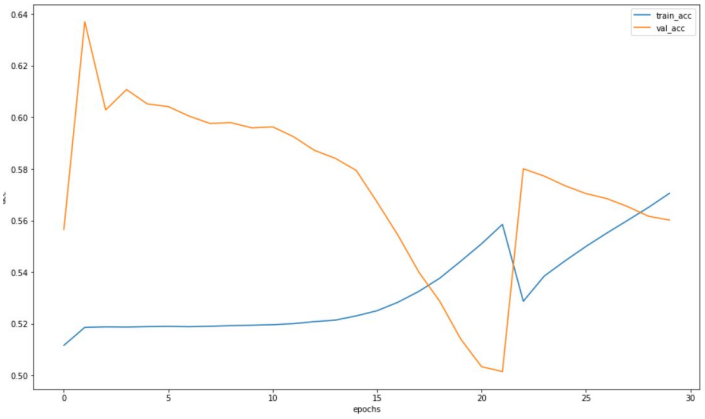
```
ModelCnn(
  (conv1): Sequential(
    (0): Conv2d(1, 32, kernel_size=(4, 4), stride=(2, 2))
    (1): ReLU()
  )
  (conv2): Sequential(
    (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (conv3): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (fc1): Sequential(
    (0): Linear(in_features=512, out_features=1024, bias=True)
    (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (fc2): Sequential(
    (0): Linear(in_features=1024, out_features=1024, bias=True)
    (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (fc3): Sequential(
    (0): Linear(in_features=1024, out_features=1024, bias=True)
    (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (fc4): Sequential(
    (0): Linear(in_features=1024, out_features=3, bias=True)
  )
)
```

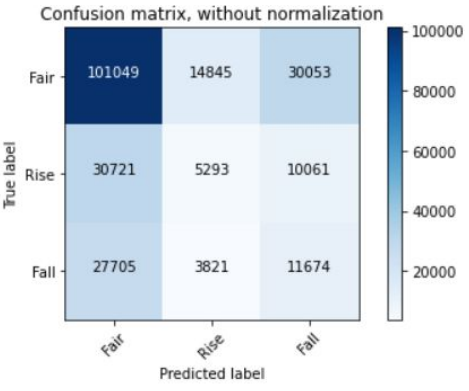
Для обучения мы использовали оптимизатор AdamW, learning rate scheduler ReduceLROnPlateau, начальный learning rate - 0,001, размер батча - 256, функция ошибки - CrossEntropy

Сначала мы обучали сеть на 30 эпохах, потом дообучали на 20.

Результаты обучения после 30 эпох:

График обучения:

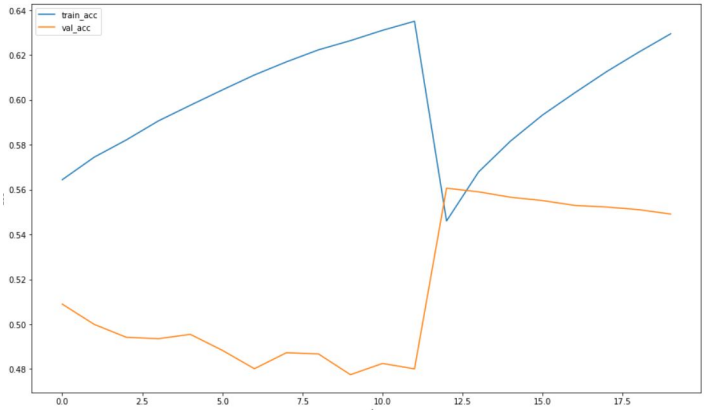


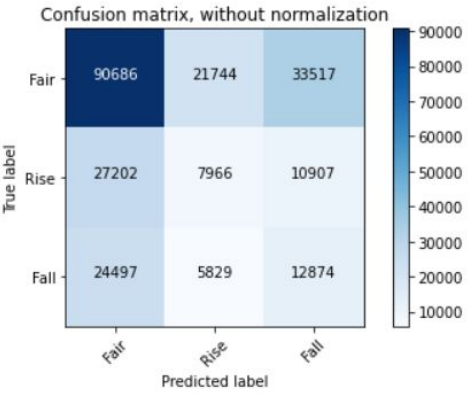



```
f1 по классам - 0.66, 0.15, 0.25
среднее f1 - 0.3528851608039831
```

Результаты обучения после 50 эпох:

График обучения за последние 20 эпох:



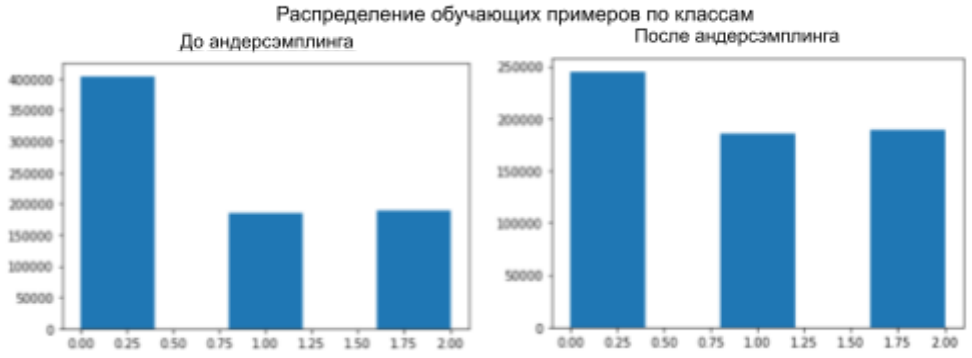



```
f1 по классам - 0.66, 0.15, 0.25
среднее f1 - 0.3528851608039831
```

Как мы можем видеть по confusion matrix, классификатор часто предсказывает класс 0 и плохо предсказывает классы 1 и 2, что может свидетельствовать о переобучении. Для борьбы с ним мы попробовали применить андерсэмплинг.

2) Добавление андерсэмпинга (выполняла Шарова Ангелина)

Предобработка данных: ко всей предобработке был добавлен андерсэмплинг - удаление половины выбранных случайным образом обучающих примеров класса 0, так как этот класс более чем в два раза превосходил остальные классы по объему. После андерсэмплинга в обучающем множестве осталось около 600000 обучающих примеров.

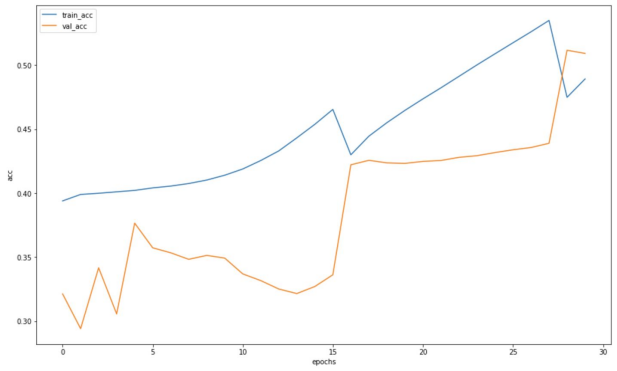


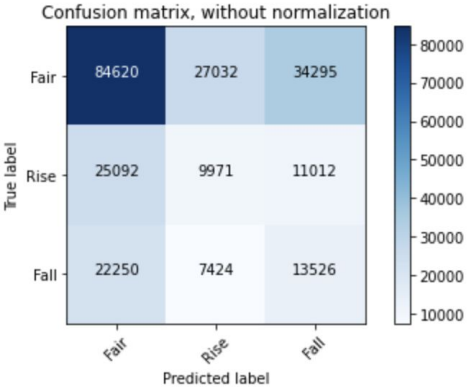
Архитектура нейронной сети осталась без изменений

Сначала мы обучали сеть на 30 эпохах, потом дообучали на 20 дважды

Результаты обучения после 30 эпох:

График обучения:

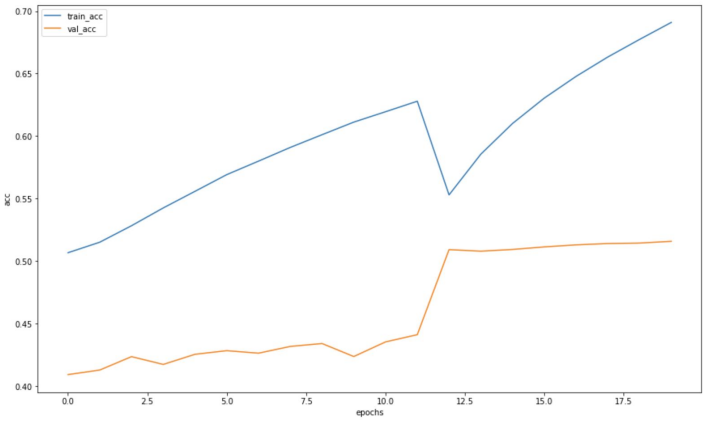


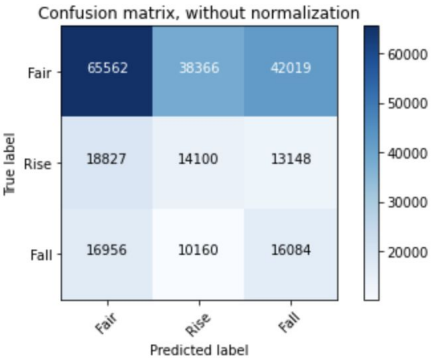



```
accuracy - 0.45963812908656504
f1 по классам - 0.61, 0.22, 0.27
среднее f1 - 0.3648183167042889
```

Результаты обучения после 50 эпох:

График обучения за последние 20 эпох:

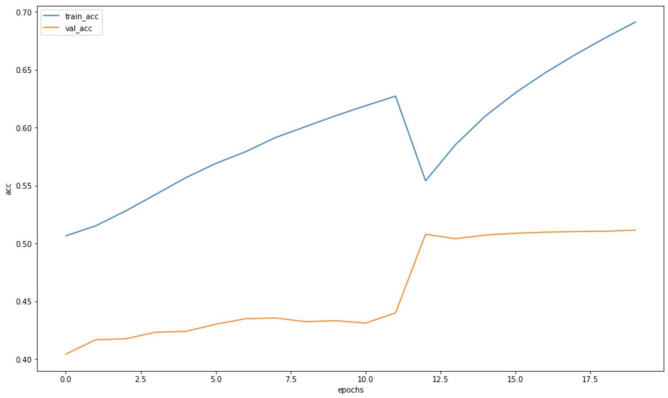


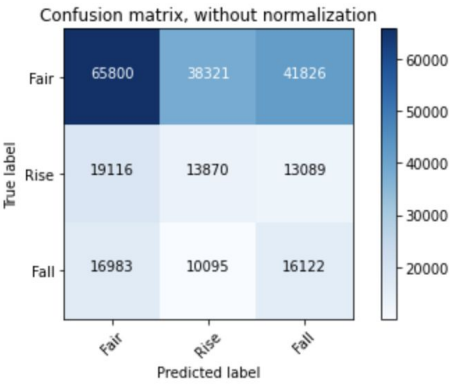



```
accuracy - 0.40704525937199754
f1 по классам - 0.53, 0.26, 0.28
среднее f1 - 0.35691010097804154
```

Результаты обучения после 70 эпох:

График обучения:



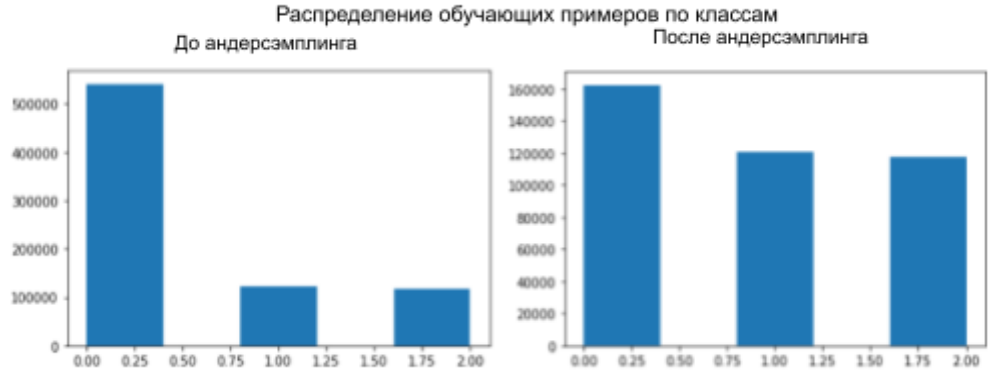



```
accuracy - 0.4072408193111189
```

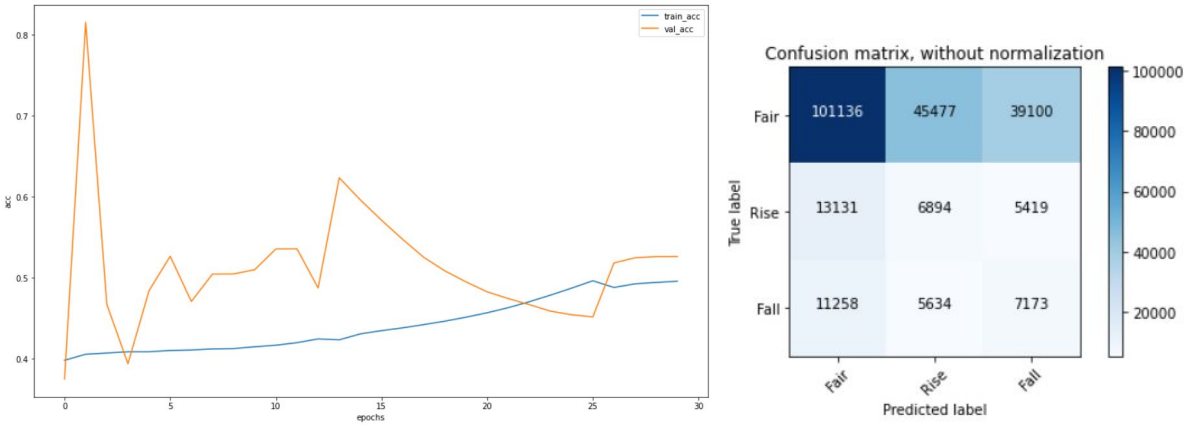
```
f1 по классам - 0.53, 0.26, 0.28
среднее f1 - 0.35640878395639614
```

Как мы видим, обучение на 30 эпохах дает чуть лучший результат в среднем, но уступает результату сети, обученной на 50 эпохах, по результатам на более малочисленных классах 1 и 2, а обучение на 70 эпохах не улучшает результаты, полученные на 50 эпохах, поэтому в дальнейшем мы будем обучать сети на 30 эпохах, дообучать до 50 эпох и сравнивать, где результаты получатся лучше. По сравнению с предыдущим случаем сеть стала лучше предсказывать классы 1 и 2, так что можно сделать вывод, что использование андерсэмплинга было оправданным. Однако качество классификации все еще оставляет желать лучшего.

3)Изменение разделения данных на классы (выполняла Шарова Ангелина)
Из предположения, что классы подъема и спада цены (1 и 2) будут распознаваться лучше, если в этих классах будут более характерные примеры, которые будут сильнее отличаться от боковика, мы пробовали изменить разбиение на классы, повысив границу разбиения на классы по отклонению цены с 3% до 5%. Классы при этом получились еще более несбалансированными, поэтому пришлось применять андерсэмплинг.

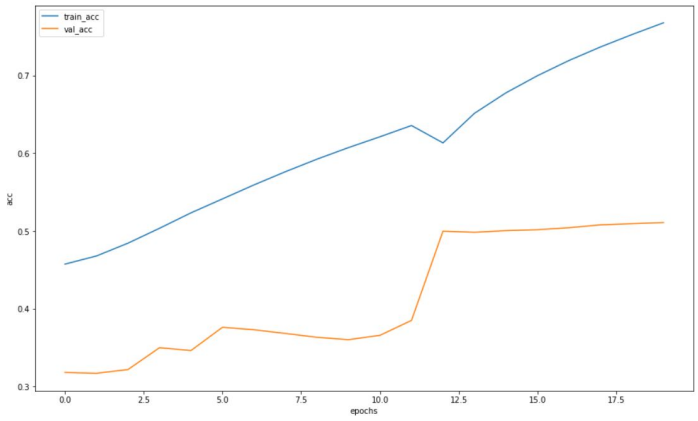


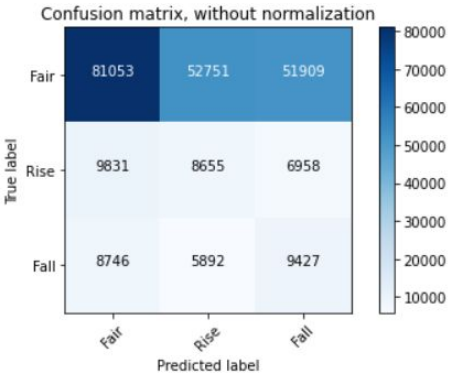
Архитектура сети осталась прежней.
Сначала мы обучали сеть на 30 эпохах, потом дообучали на 20.
Результаты обучения после 30 эпох:
График обучения:




```
f1 по классам - 0.65, 0.17, 0.19
```

```
среднее f1 - 0.33483008009948784
```
Результаты обучения после 50 эпох:
График обучения за последние 20 эпох:





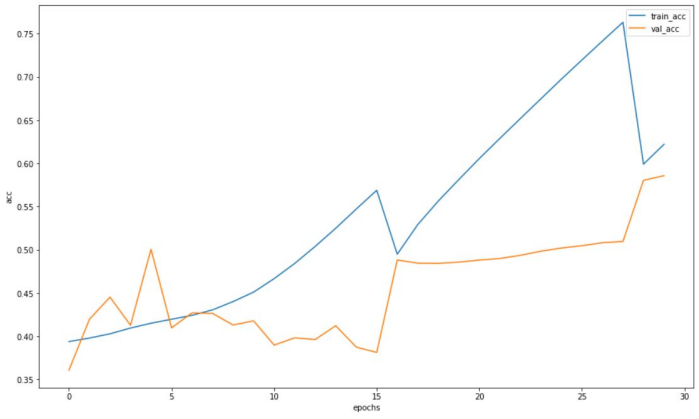

```
f1 по классам - 0.57, 0.19, 0.2
среднее f1 - 0.31963142817059914
```
Как можно видеть из confusion matrix, классификатор часто предсказывает класс роста или падения для примеров из класса боковик. Скорее всего это свидетельствует об ошибочности предположения и о том, что алгоритм выделил некоторые закономерности из случаев с большим изменением цены и относит случаи с меньшим изменением цены, но похожими закономерностями, к классам с изменением цены. Таким образом, повышение границы оказалось избыточным.

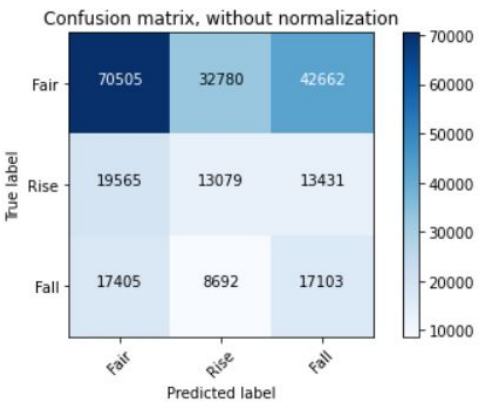
4)Обучение на всех признаках (выполнял Минченко Никита)
Во всех предыдущих случаях мы использовали для обучения только данные о ценах закрытия. Теперь попробуем использовать все признаки, преобразовав каждый в двухмерную матрицу, соединив их по оси канала и подавая на вход сверточной нейросети пятиканальные изображения.
Для предобработки так же используется масштабирование, разбиение данных с границей отклонения 3% и андерсэмплинг.
Архитектуру нейронной сети пришлось немного изменить в связи с изменением числа каналов.

```
ModelCnn5ch(
  (conv1): Sequential(
    (0): Conv2d(5, 64, kernel_size=(4, 4), stride=(2, 2))
    (1): ReLU()
  )
  (conv2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (conv3): Sequential(
    (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (fc1): Sequential(
    (0): Linear(in_features=1024, out_features=1024, bias=True)
    (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (fc2): Sequential(
    (0): Linear(in_features=1024, out_features=1024, bias=True)
    (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (fc3): Sequential(
    (0): Linear(in_features=1024, out_features=1024, bias=True)
    (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (fc4): Sequential(
    (0): Linear(in_features=1024, out_features=3, bias=True)
  )
)
```

Результаты обучения после 30 эпох:

График обучения:

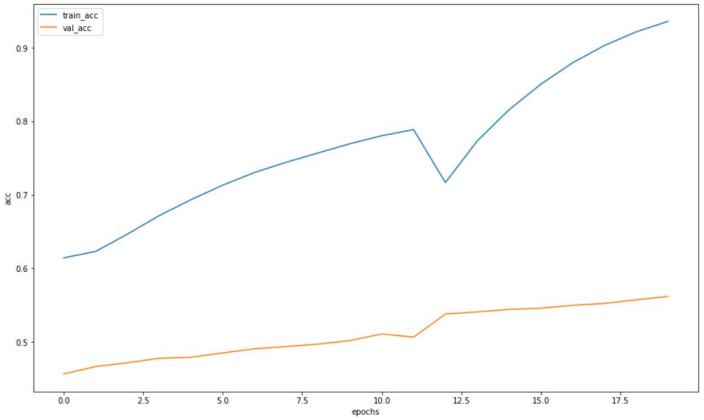


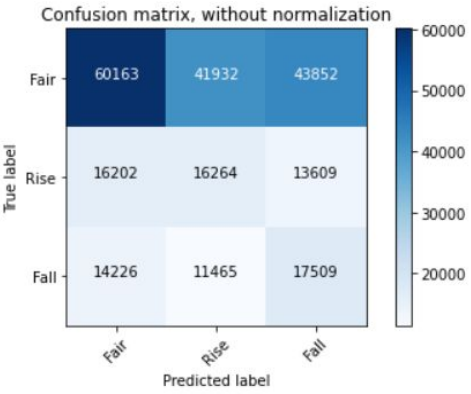



```
f1 по классам - 0.55642367, 0.2599527 , 0.29387608
среднее f1 - 0.3700841490350613
```

Результаты обучения после 50 эпох:

График обучения за последние 20 эпох:






```
f1 по классам - 0.51, 0.28, 0.3
среднее f1 - 0.3620284988348054
```

Добавление данных дало свои результаты и качество немного улучшилось. Таким образом, из всех рассмотренных вариантов решения этот оказался наилучшим.

Из результатов проделанной работы можно сделать вывод, что хоть нейросетевой классификатор превзошел по критерию качества наивный классификатор, и некоторые взаимосвязи и закономерности все таки были выделены из данных.
Машинное обучение развивается изо дня в день - появляются новые методы, поэтому нужно постоянно обучаться. Лучший способ для этого - создавать интересные проекты, например, прогноз цен на акции. И хотя наша модель недостаточна хороша для реальной торговли на бирже, но она дала фундамент, который возможно пригодится в будущем для разработки другой модели.