



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ

Робототехника и комплексная автоматизация (РК)

КАФЕДРА

Системы автоматизированного проектирования (РК6)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

***«Применение свёрточных нейронных сетей (CNN) для
решения задач технического анализа при управлении
активами на фондовом рынке»***

Студент РК6-86Б

(Подпись, дата)

Онюшев А.А.

И.О. Фамилия

Руководитель курсового проекта

(Подпись, дата)

Витюков Ф.А.

И.О. Фамилия

2024 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой РК6
А.П. Карпенко

«____» _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине _____ Модели и методы анализа проектных решений _____

Студент группы _____ РК6-86Б _____

Онюшев Артем Андреевич
(Фамилия, имя, отчество)

Тема курсового проекта: Применение сверточных нейронных сетей (CNN) для решения задач
технического анализа при управлении активами на фондовом рынке

Направленность КП (учебный, исследовательский, практический, производственный, др.) учебный
Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

Задание. Разработать архитектуру нейронной сети на основе сверток. Разработать датасет для загрузки
необходимых данных в НС. Подобрать необходимые настройки и параметризацию для НС. Обучить
НС и провести исследование её пригодности для анализа биржи и стратегии торговли
брокера

Оформление курсового проекта:

Расчетно-пояснительная записка на 28 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

Дата выдачи задания «8» октября 2023 г.

Руководитель курсовой работы

(Подпись, дата)

Витюков Ф.А.

И.О. Фамилия

Студент

(Подпись, дата)

Онюшев А.А.

И.О. Фамилия

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

АННОТАЦИЯ

В данной работе рассмотрено, что такое сверточная нейронная сеть и чем она отличается от простого линейного перцептрона. Описана работа по настройке и подбору нужных нам функций-оптимизации, функций-потерь и функций активации, а также размера и количества нейронных слоев и сверточных слоев. Описана работа по составлению датасета на основе данных о тикере MSFT. Также в данной работе были проведены тесты с разными настройками и проанализированы результаты, полученные на выходе. Было проведено сравнение с результатами работы линейного перцептрона при схожих настройках.

В расчётно-пояснительной записке 28 страниц, 21 рисунок, 10 графических листов.

СОДЕРЖАНИЕ

АННОТАЦИЯ.....	3
ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ	5
ВВЕДЕНИЕ.....	6
1. Устройство CNN	7
2. Изображение «в глазах» компьютера	9
3. Свертка.....	11
4. Слой подвыборки (пулинга)	13
5. Собственная CNN	14
6. Работа с yahoo finance.....	16
7. Проведение анализов.....	18
ЗАКЛЮЧЕНИЕ	26
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	27

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

НС – Нейронная сеть.

CNN – (Convolutional Neural Networks) Сверточная НС.

MLP – (Multilayer Perceptron) Линейный перцептрон.

Функция-потерь – Функция, показывающая расстояние от предугаданного НС ответа до истинного ответа.

Функция-активации – Функция, отвечающая за изменение весов синапсов.

Тикер – Краткое название в биржевой информации котируемых инструментов (акций, облигаций, индексов).

EMA – (Exponential Moving Average) Экспоненциальная скользящая средняя. Один из показателей, помогающих при техническом анализе.

Stop-Loss – Биржевая заявка трейдера, в которой условием исполнения указано достижение цены, которая хуже, чем текущая рыночная.

Take-Profit – Биржевая заявка трейдера, которая выставляется заранее, чтобы в случае роста рынка зафиксировать прибыль по бумаге.

MSE – (Mean Squared Error) Метрика, используемая для оценки эффективности работы регрессионной модели.

MAE – (Mean Absolute Error) Метрика, используемая для оценки эффективности работы регрессионной модели.

ВВЕДЕНИЕ

В настоящее время область сверточных нейронных сетей (CNN) активно исследуется и применяется в различных областях, включая финансовый рынок и биржевую торговлю. CNN позволяют анализировать и обрабатывать большие объемы данных, что идеально подходит для работы с финансовой информацией, которая обычно является многомерной и динамичной.

Использование CNN для анализа биржи и стратегий торговли брокера может быть эффективным способом прогнозирования рыночных движений и принятия обоснованных решений о покупке или продаже активов. Эти модели имеют преимущество в распознавании сложных закономерностей и паттернов на графиках цен, что может помочь в принятии более точных и выгодных решений.

Данное исследование будет сосредоточено на разработке и оптимизации сверточных нейронных сетей для анализа данных биржи и выработки стратегий торговли для брокеров. В этом исследовании рассмотрены различные подходы к построению моделей CNN, включая архитектуру сети, выбор оптимальных гиперпараметров и оптимизацию процесса обучения. Также будет проведен анализ полученных НС результатов для разных настроек.

Целью данного исследования является анализ пригодности CNN для выведения таких сложных закономерностей, как стоимости тикеров.

1. Устройство CNN

Сверточная нейронная сеть (ConvNet/CNN) — это специальная архитектура искусственных нейронных сетей, которая может принимать входное изображение (матрицу данных), присваивать важность (изучаемые веса и смещения) аспектам или объектам изображения (матрицы) и отличать одно от другого. При этом изображения (матрицы), в сравнении с другими архитектурами НС, требуют гораздо меньше предварительной обработки. В примитивных методах фильтры разрабатываются вручную, но достаточно обученные сети CNN учатся применять эти фильтры/характеристики самостоятельно.

Архитектура CNN аналогична структуре связей нейронов в мозгу человека, учёные черпали вдохновение в организации зрительной коры головного мозга. Отдельные нейроны реагируют на стимулы только в некоторой области поля зрения, также известного как перцептивное поле. Множество перцептивных полей перекрывается, полностью покрывая поле зрения CNN.

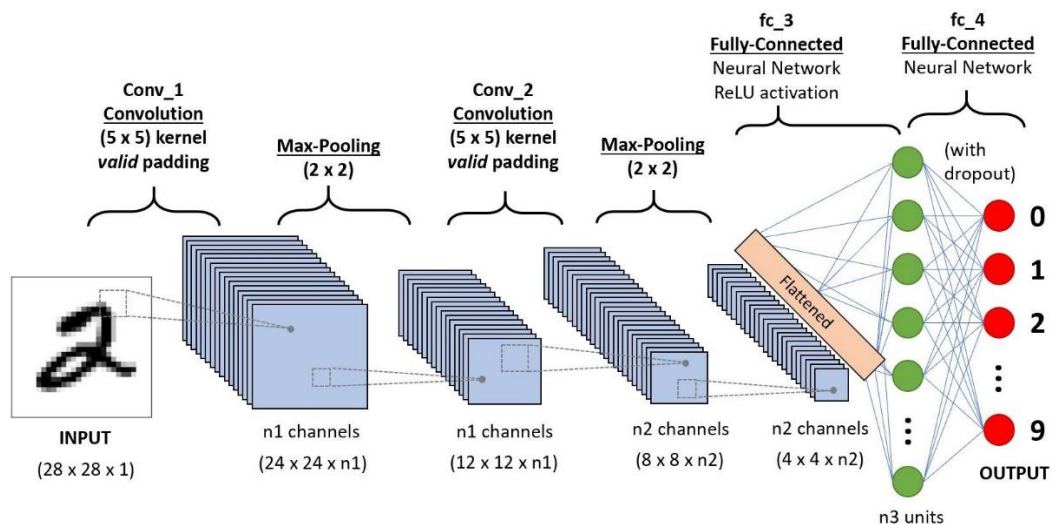


Рис. 1 Архитектура сверточной нейронной сети

Главной особенностью свёрточных сетей является то, что они обычно работают именно с матрицами данных, а потому можно выделить особенности, свойственные именно им. Многослойные персептроны работают с векторами, а потому для них нет никакой разницы, находятся ли какие-то точки рядом или на

противоположных концах, так как все точки равнозначны и считаются совершенно одинаковым образом. Изображения или же матрицы могут обладают локальной связностью. Например, если речь идёт об изображениях человеческих лиц, то вполне логично ожидать, что точки основных частей лица будут рядом, а не разрозненно располагаться на изображении. Поэтому требовалось найти более эффективные алгоритмы для работы с матрицами данных и ими оказались свёрточные сети.

2. Изображение «в глазах» компьютера

Изучая CNN, невольно появляется вопрос «Что такое свертка?». Перед тем, как ответить на него, нужно разобраться, как компьютер «видит» какое-то изображение.

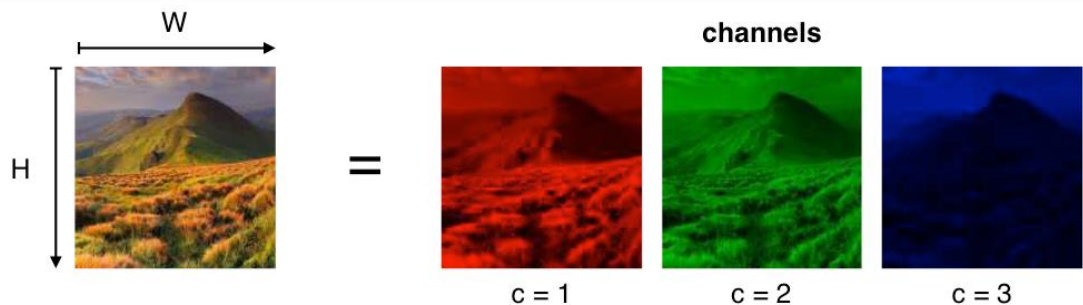


Рис. 2 Как компьютер видит изображение

Изображения в компьютере представляются в виде пикселей, а каждый пиксель – это значения интенсивности соответствующих цветовых каналов. При этом интенсивность каждого из каналов описывается целым числом от 0 до 255. Чаще всего используются цветные изображения, которые состоят из RGB пикселей – пикселей, содержащих яркости по трём каналам: красному, зелёному и синему. Различные комбинации этих цветов позволяют создать любой из цветов всего спектра. Чтобы хранить информацию о всех пикселях удобнее всего использовать тензор – 3D массив чисел, или, проще говоря, массив матриц чисел.

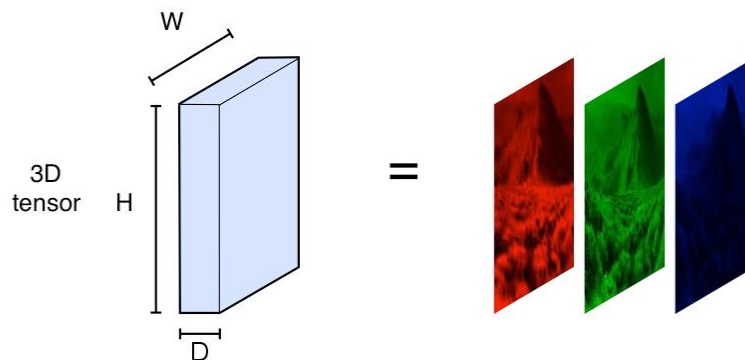


Рис. 3 Преобразование из изображения в тензор

3. Свертка

Слой свёртки, как можно догадаться по названию типа нейронной сети, является самым главным слоем сети. Его основное назначение – выделить признаки на входном изображении и сформировать карту признаков. Карта признаков – это всего лишь очередной тензор (массив матриц), в котором каждая матрица отвечает за какой-нибудь выделенный признак.

Для того, чтобы слой мог выделять признаки, в нём имеются так называемые фильтры (или ядра). Фильтр – квадратная матрица небольшого размера (обычно 3×3 или 5×5), заполненная определенным образом. Чем больше будет таких фильтров – тем больше признаков удастся выделить и тем больше будет глубина каналов у выходного тензора слоя

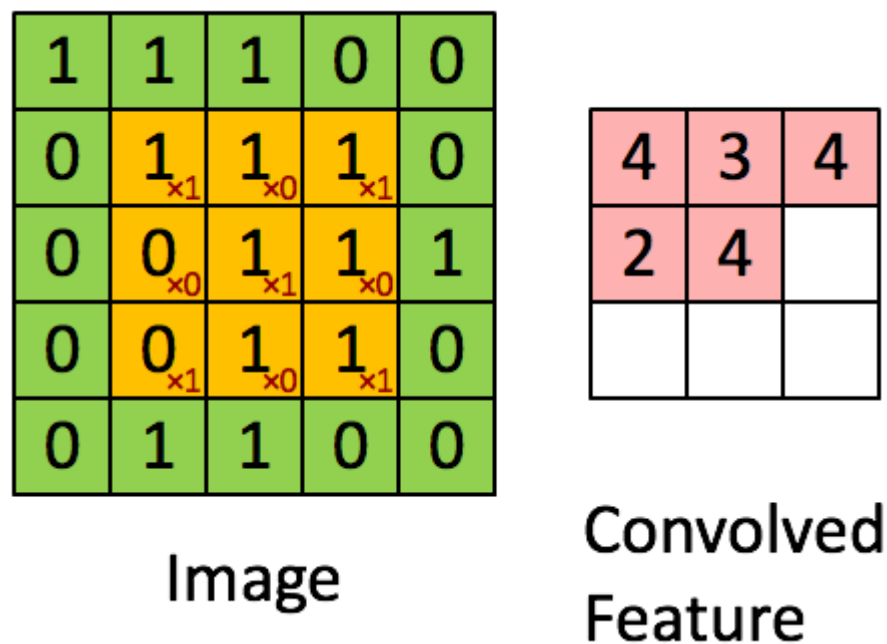


Рис. 5 Свертка изображения $5 \times 5 \times 1$ с фильтром $3 \times 3 \times 1$ для получения признака $3 \times 3 \times 1$

Данный фильтр поочередно накладывается на изображение начиная с левого верхнего угла и пока не дойдет до правого нижнего угла. На каждом шаге числа в матрице фильтра перемножаются с соответствующими числами на изображении, полученные результаты складываются и записываются в матрицу признака. Лучше понять это можно, посмотрев на рисунок 5. На нем желтым цветом изображен фильтр размером $3 \times 3 \times 1$, а красные цифры в правом углу –

числа матрицы фильтра. Зеленым цветом изображена матрица изображения $3 \times 3 \times 1$, а красным полученный признак. Таким образом на 5-ом шаге итерации мы получим:

$$1 * 1 + 0 * 1 + 1 * 1 + 0 * 0 + 1 * 1 + 0 * 1 + 1 * 0 + 0 * 1 + 1 * 1 = 4$$

Полученные в итоге матрицы признаков могут иметь такой же размер, что и исходное изображение, либо размер меньше (как в нашем случае). Это зависит от заданных размера шага и начального заполнения. В [данном](https://github.com/vdumoulin/conv_arithmetic/tree/master/gif) репозитории (См. https://github.com/vdumoulin/conv_arithmetic/tree/master/gif) можно найти множество разных GIF-файлов, которые помогут лучше понять, как заполнение и длина шага работают.

4. Слой подвыборки (пулинга)

Так как сверточные НС используют так же и простую сеть прямого распространения, нам необходимо уменьшить кол-во выходных параметров из слоя свертки и при этом не потерять важную информацию. Для этого используют слои подвыборки.

Данный слой позволяет уменьшить пространство признаков, сохраняя наиболее важную информацию. Существует несколько разных версий слоя пулинга, среди которых максимальный пулинг, средний пулинг и пулинг суммы. Наиболее часто используется именно слой макспулинга.

Feature Map				Max Pooling		Average Pooling		Sum Pooling	
6	6	6	6	6	6	5.25	5.25	21	21
4	5	5	4	4	4	3	3	12	12
2	4	4	2						
2	4	4	2						

Рис. 6 Преобразования слоя подвыборки

Действия этого слоя идентичны действиям слоя свертки, только используются другие операции (для макспулинга – берется максимальное число, для среднего пулинга – берется среднее арифметическое от чисел).

5. Собственная CNN

Изучив, как устроена сверточная нейронная сеть, можно приступить к написанию собственной.

При разработке архитектуры нейронной сети требуется учитывать, что на вход должны передаваться данные некоторого размера, которые и будут проанализированы НС. В данной работе на вход НС передается информация о тикерах в формате: Массив[N] со значениями High, массив[N] со значениями Low, массив[N] со значениями ЕМА 200, где N – количество исследуемых дней от 40 до 2000, а также передается массив[N] со значениями asset – число отвечающее за наличие данного тикера в портфеле. Все эти 4 массива по N значений образуют матрицу $4 \times N$. Её мы и будем использовать, как входные данные.

Проанализировав данные, поданные на вход НС, она возвращает некий результат своего анализа – выходные данные. При разработке архитектуры важно учесть правильность выходных данных. В данной работе выходными данными должно быть число в диапазоне от -0.5 до 1.5, означающие биржевые заявки трейдеру – take-profit или stop-loss.

Преобразуем нашу матрицу $4 \times N$ в удобный для обычных линейных нейронов формат, используя три слоя свертки с ядрами 2×2 и итоговым выходным количеством сверток 1×64 . Далее с помощью скрытых слоев обычных линейных нейронов проанализируем получившиеся свертки. Было использовано 4 скрытых линейных слоя для лучшего анализа.

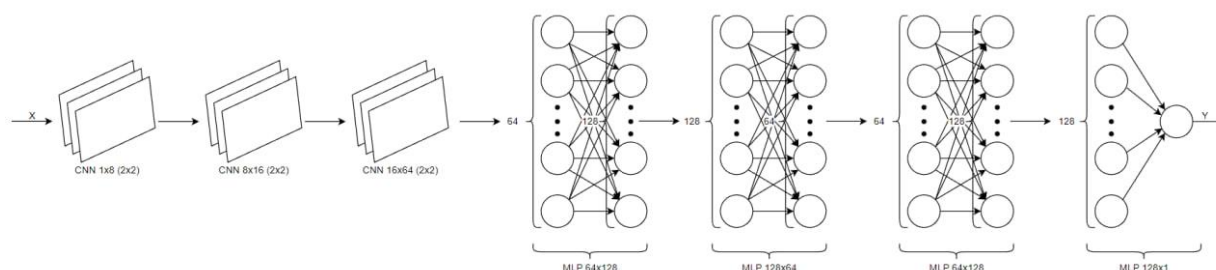


Рис. 7 Архитектура НС.

Реализация данной архитектуры в коде:

Листинг 1 – Класс нейронной сети.

```
1 class ConvNet(nn.Module):
2     def __init__(self):
3         super().__init__()
4
5         self.act = nn.LeakyReLU(0.18) #функция активации
6         self.conv0 = nn.Conv2d(1, 8, 2, stride=1, padding=0)
7         # сверточный слой со сверточным ядром 2 на 2
8         self.conv1 = nn.Conv2d(8, 16, 2, stride=1, padding=0)
9         self.conv2 = nn.Conv2d(16, 64, 2, stride=1, padding=0)
10
11         self.adaptivepool = nn.AdaptiveAvgPool2d((1, 1))
12         self.flatten = nn.Flatten()
13         self.linear1 = nn.Linear(64, 128)
14         # линейный слой 64 на 128
15         self.extra_linear_1 = nn.Linear(128, 64)
16         # линейный слой 128 на 64
17         self.extra_linear_2 = nn.Linear(64, 128)
18         # линейный слой 64 на 128
19         self.linear2 = nn.Linear(128, 1)
20         # линейный слой 128 на 1
21
22
23     def forward(self, x):
24         # функция описывающая одну итерацию обучения НС
25         out = self.conv0(x)
26         out = self.act(out)
27         out = self.conv1(out)
28         out = self.act(out)
29         out = self.conv2(out)
30         out = self.act(out)
31
32         out = self.adaptivepool(out)
33         out = self.flatten(out)
34         out = self.linear1(out)
35         out = self.act(out)
36
37         out = self.extra_linear_1(out)
38         out = self.act(out)
39         out = self.extra_linear_2(out)
40         out = self.act(out)
41
42         out = self.linear2(out)
43         return out
```

6. Работа с yahoo finance

Для получения необходимой информации о дневных свечах тикера, необходимо обратиться к API yahoo finance, но так как они прекратили его поддержку в 2016 году, воспользуемся модулем ufinance.

Чтобы выбрать конкретный тикер, создадим переменную ticker. Чтобы выбрать количество дней информацию по которым мы хотим получить, создадим переменную days. Чтобы выбрать количество дней для ЕМА введем переменную ЕМА_N. Реализация данного кода на python:

Листинг 2 – Функция загрузки информации с yahoo finance

```
1 Days = 200 # Кол-во дней в датасете
2 ЕМА_N = 200 # Кол-во дней в ЕМА
3 num_candle = Days+ЕМА_N
4
5 ticker = "MSFT" # Названия тикера который отслеживаем
6 hist = fin.download(ticker, period=f'{num_candle}d', interval='1d')
```

Функция fin.download() загрузит всю необходимую нам информацию о свечах, а код указанный в листинг 3 создаст столбец с информацией о ЕМА.

Листинг 3 – Создание столбца с ЕМА

```
1 hist[f'ЕМА{ЕМА_N}'] = hist['Close'].ewm(span=ЕМА_N,
    adjust=False).mean() # Считаем нужный ЕМА
```

Итоговая таблица будет иметь вид, указанный на рисунке 7.

	Date	Open	High	Low	Close	ЕМА200	Assets	Take_profit	Stop_loss
0	07.12.2021	331,64	335,8	330,1	334,92	334,92			
1	08.12.2021	335,31	335,5	330,8	334,97	334,9205			
2	09.12.2021	334,41	336,49	332,12	333,1	334,9024			
3	10.12.2021	334,98	343	334,79	342,54	334,9784			
4	13.12.2021	340,68	343,79	339,08	339,4	335,0224			
5	14.12.2021	333,22	334,64	324,11	328,34	334,9559			
6	15.12.2021	328,61	335,19	324,5	334,65	334,9529			
7	16.12.2021	335,71	336,76	323,02	324,9	334,8528			
8	17.12.2021	320,88	324,92	317,25	323,8	334,7428			
9	20.12.2021	320,05	322,8	317,57	319,91	334,5953			
10	21.12.2021	323,29	327,73	319,8	327,29	334,5226			
11	22.12.2021	328,3	333,61	325,75	333,2	334,5094			
12	23.12.2021	332,75	336,39	332,73	334,69	334,5112			
13	27.12.2021	335,46	342,48	335,43	342,45	334,5902			
14	28.12.2021	343,15	343,81	340,32	341,25	334,6565			
15	29.12.2021	341,3	344,3	339,68	341,95	334,729			

Рисунок 8. Вид полученной таблицы датасета

Получив такую таблицу, можно приступать к заполнению колонок «Stop-Loss» и «Take-Profit». Данные колонки заполняются брокером от руки в зависимости от принятой им стратегии ведения торгов. После заполнения этих колонок, наша таблица будет готова.

	Date	Open	High	Low	Close	EMA200	Assets	Take_profit	Stop_loss
249	02.12.2022	249,82	256,06	249,69	255,02	265,245	1	1,0908879	0,671586
250	05.12.2022	252,01	253,82	248,06	250,2	265,0953	1	1,0908879	0,680956
251	06.12.2022	250,82	251,86	243,78	245,12	264,8965	1	1,0908879	0,671586
252	07.12.2022	244,83	246,16	242,21	244,37	264,6923	1	1,0908879	0,671586
253	08.12.2022	244,84	248,74	243,06	247,4	264,5202	1	1,0908879	0,711408
254	09.12.2022	244,7	248,31	244,16	245,42	264,3302	1	1,0908879	0,739518
255	12.12.2022	247,45	252,54	247,17	252,51	264,2126	1	1,0908879	0,762942
256	13.12.2022	261,69	263,92	253,07	256,92	264,14	0	0,14993076	1,011487
257	14.12.2022	257,13	262,59	254,31	257,22	264,0711	0	0,14993076	1,011487
258	15.12.2022	253,72	254,2	247,34	249,01	263,9213	0	0,14993076	1,011487
259	16.12.2022	248,55	249,84	243,51	244,69	263,7299	0	0,14993076	1,011487
260	19.12.2022	244,86	245,21	238,71	240,45	263,4983	0	0,14993076	1,011487
261	20.12.2022	239,4	242,91	238,42	241,8	263,2824	0	0,14993076	1,011487
262	21.12.2022	241,69	245,62	240,11	244,43	263,0948	0	0,14993076	1,011487
263	22.12.2022	241,26	241,99	233,87	238,19	262,847	0	0,14993076	1,011487
264	23.12.2022	236,11	238,87	233,94	238,73	262,607	0	0,14993076	1,011487

Рисунок 9. Итоговый вид таблицы датасета

7. Проведение анализов

Чтобы НС хорошо работала требуется обучить её на данных из всего датасета. Прогоняя эти данные повторно множество раз, мы повышаем вероятность более правильного ответа НС. Прогон всего датасета через НС называется – эпоха. Совсем не обязательно в одной эпохе обучать НС всем датасетом за раз, можно разбить его на меньшие части. Такие части называют – батч данных.

Для исследования был составлен пробный датасет, который содержал 200 сущностей для обучения и 50 сущности для тестов. Размер батча равен 25 сущностей. Используемый код для обучения НС на Python:

Листинг 4 – Код обучающего цикла НС.

```
1  for epoch in range(epochs):
2      loss_val = 0
3      acc_val = 0
4      for sample in train_loader:
5          info, lbl = sample['info'], sample['label']
6          # переносим требуемые данные на то устройство, где будем
          обучать НС (CPU или GPU)
7          info = info.to(device)
8          lbl = lbl.to(device)
9          optimizer.zero_grad()
10
11         with autocast(use_amp):
12             pred = CNNNet(info)
13             loss = loss_fn(pred, lbl)
14             # Считаем функцию-потерь
15             scaler.scale(loss).backward()
16             loss_item = loss.item()
17             loss_val += loss_item
18
19             # Обновляем веса
20             scaler.step(optimizer)
21             scaler.update()
22
23             acc_current = accuracy_v3(pred.cpu().float(),
24 lbl.cpu().float(), epsilon=epsilon)
25             acc_val += acc_current
26
27         print(f"Epoch : {epoch+1}")
28         print(f"Loss : {loss_val / len(train_loader)}")
29         print(f"Acc : {acc_val / (len(train_loader)*batch_size)}")
30     print(f'Full time learning : {time.time() - start_time}')
```

Исследуемая задача более всего похожа на задачу регрессии, основываясь на этом, максимально логичным решением будет использовать функцию MSELoss (Mean Squared Error Loss) или MAELoss (Mean Absolute Error Loss), как функцию-потерь.

Формула MSE выглядит так:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

MSE = mean squared error

n = number of data points

Y_i = observed values

\hat{Y}_i = predicted values

Рисунок 9. Формула подсчета MSE

Формула MAE выглядит так:

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - \hat{y}_1|}{n}$$

Рисунок 10. Формула подсчета MAE

После выбора подходящей функции-потерь, необходимо подобрать функцию-оптимизатор. На данный момент одной из самых популярных и эффективных функций-оптимизаторов является Adam (adaptive moment), он отлично подойдет и для этой задачи. Lr (шаг поиска) достаточно будет указать равным 10^{-3} . Отличный пример сравнения разных функций-оптимизаций приведен на данном сайте: <https://emiliendupont.github.io/2018/01/24/optimization-visualization/>

Подобрав все возможные необходимые настройки, можно приступить к тестированию.

Тестирование будем проводить и на CNN архитектуре, и на MLP. В качестве функций-потерь будем тестировать MSELoss и MAELoss. Также

рассмотрим разное количество эпох (50, 100, 200, 400, 800, 1600, 3200, 6400, 10000). Все предложенные параметры нужны нам, чтобы четко понять как архитектура НС, функция-потерь и количество эпох влияют на итоговый результат.

Полученные результаты представлены ниже. График красного цвета – требуемый (истинный) результат. График синего цвета – результат, предугаданный НС. По оси X – номера сущностей из тестового датасета. По оси Y – “stop loss”.

Используя 50 эпох:

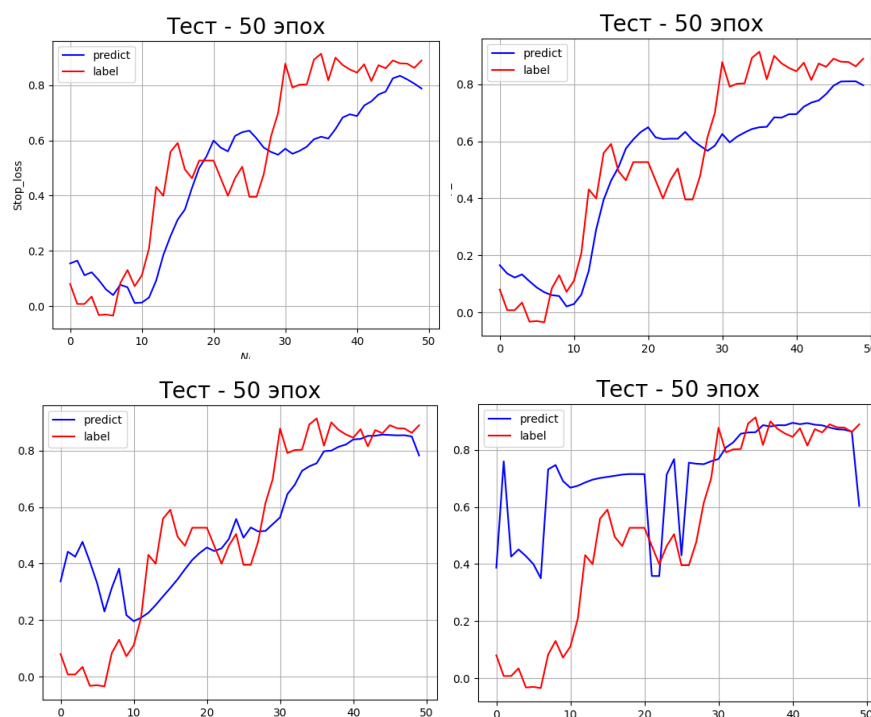
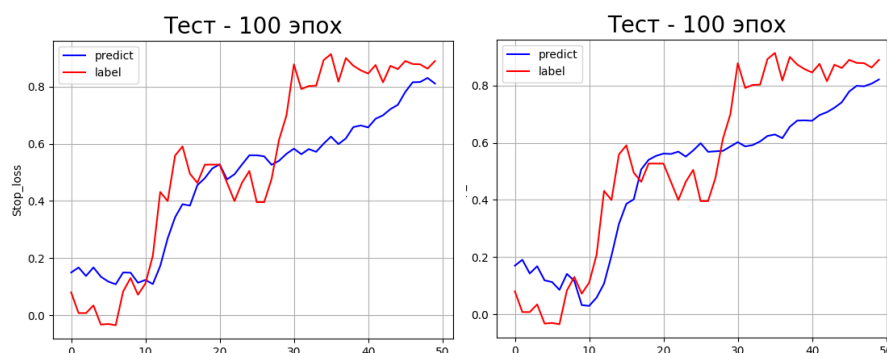


Рисунок 11. Тест с 50 эпохами (слева – MSE, справа – MAE; сверху – MLP, Снизу – CNN).

Используя 100 эпох:



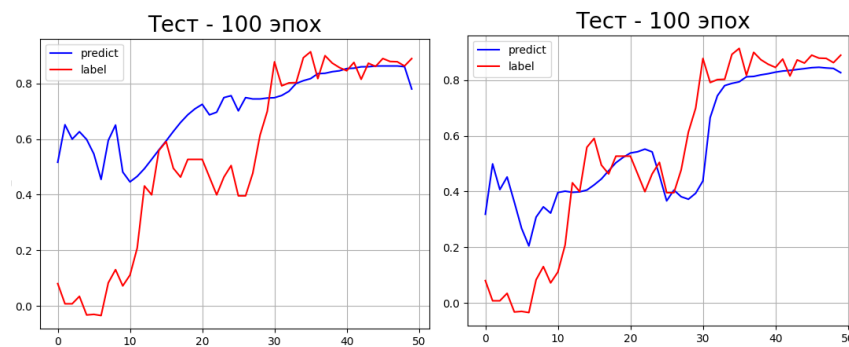


Рисунок 12. Тест с 100 эпохами (слева – MSE, справа – MAE; сверху – MLP, Снизу – CNN).

Используя 200 эпох:

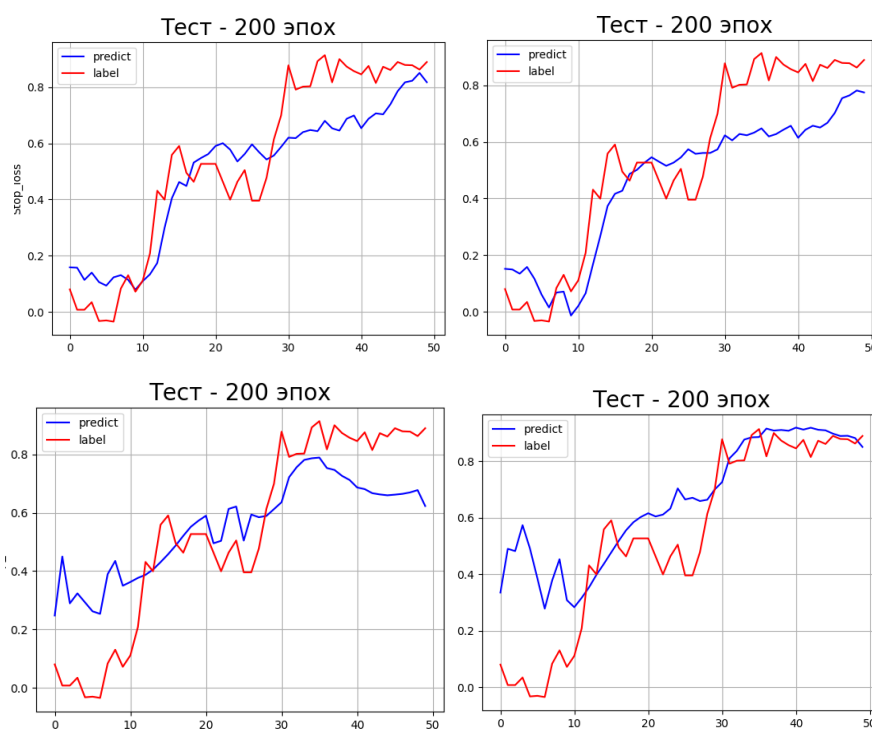
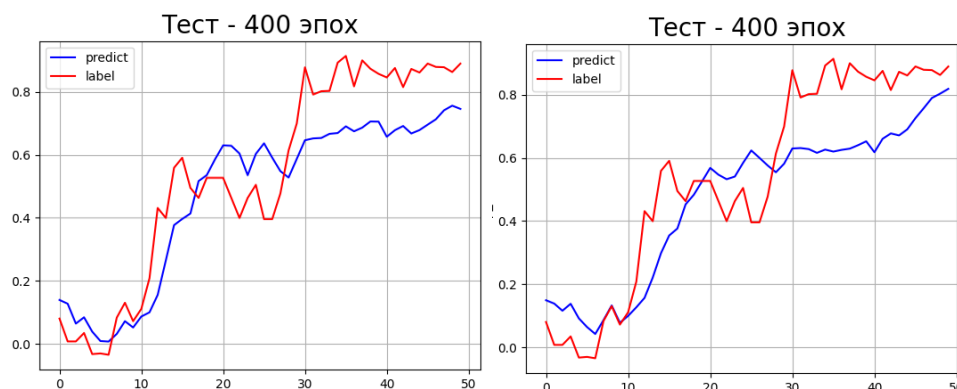


Рисунок 13. Тест с 200 эпохами (слева – MSE, справа – MAE; сверху – MLP, Снизу – CNN).

Используя 400 эпох:



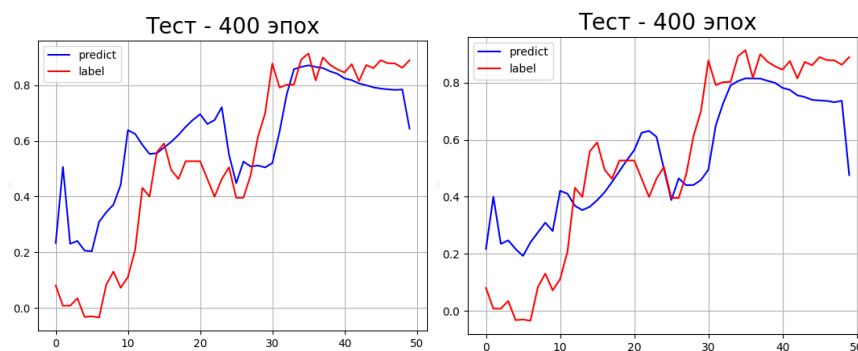


Рисунок 14. Тест с 400 эпохами (слева – MSE, справа – MAE; сверху – MLP, Снизу – CNN).

Используя 800 эпох:

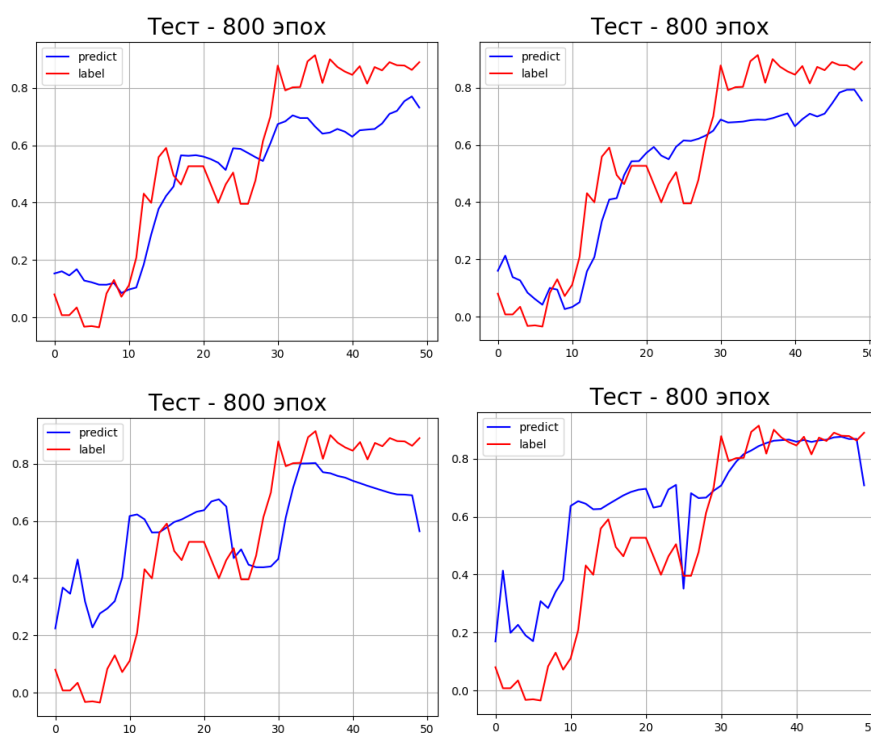
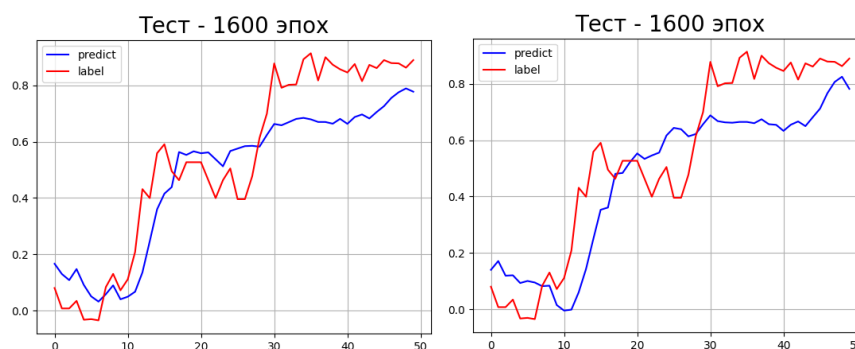


Рисунок 15. Тест с 800 эпохами (слева – MSE, справа – MAE; сверху – MLP, Снизу – CNN).

Используя 1600 эпох:



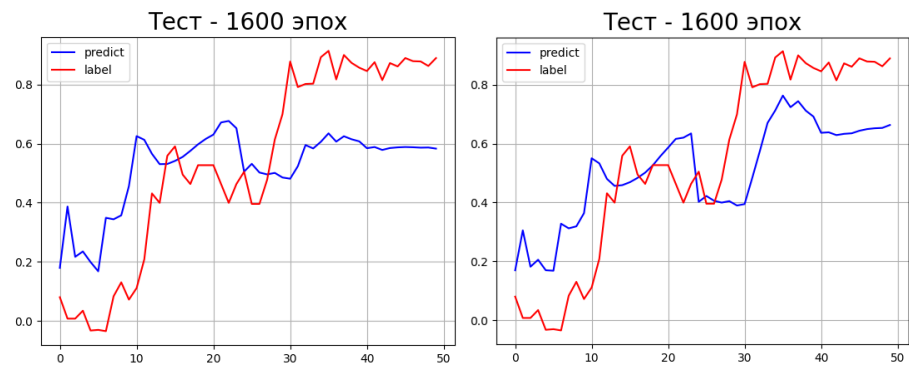


Рисунок 16. Тест с 1600 эпохами (слева – MSE, справа – MAE; сверху – MLP, Снизу – CNN).

Используя 3200 эпох:

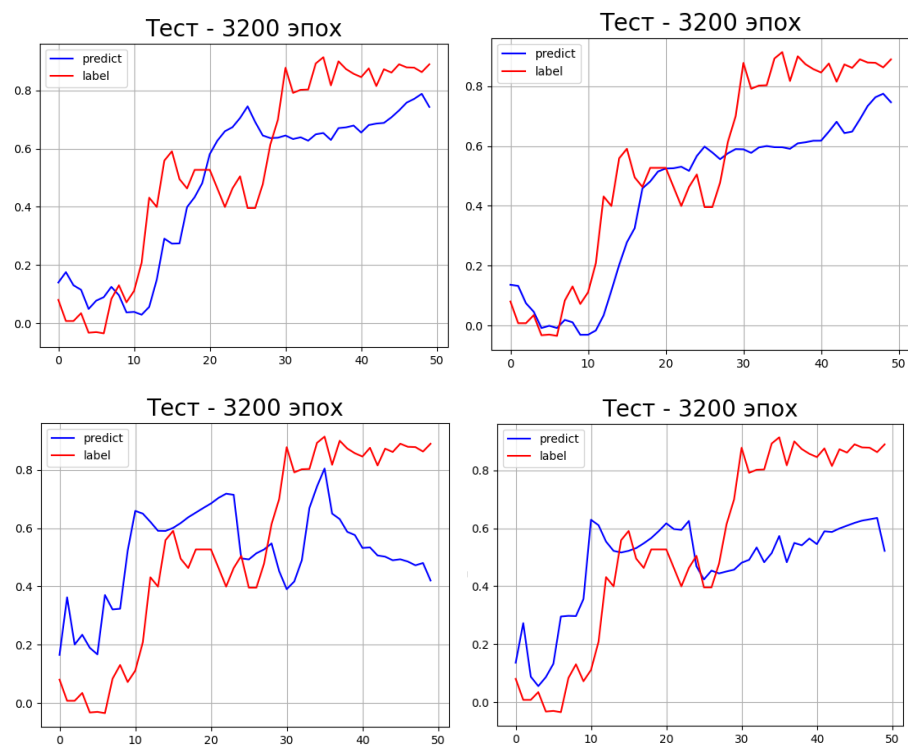


Рисунок 17. Тест с 3200 эпохами (слева – MSE, справа – MAE; сверху – MLP, Снизу – CNN).

Используя 6400 эпох:

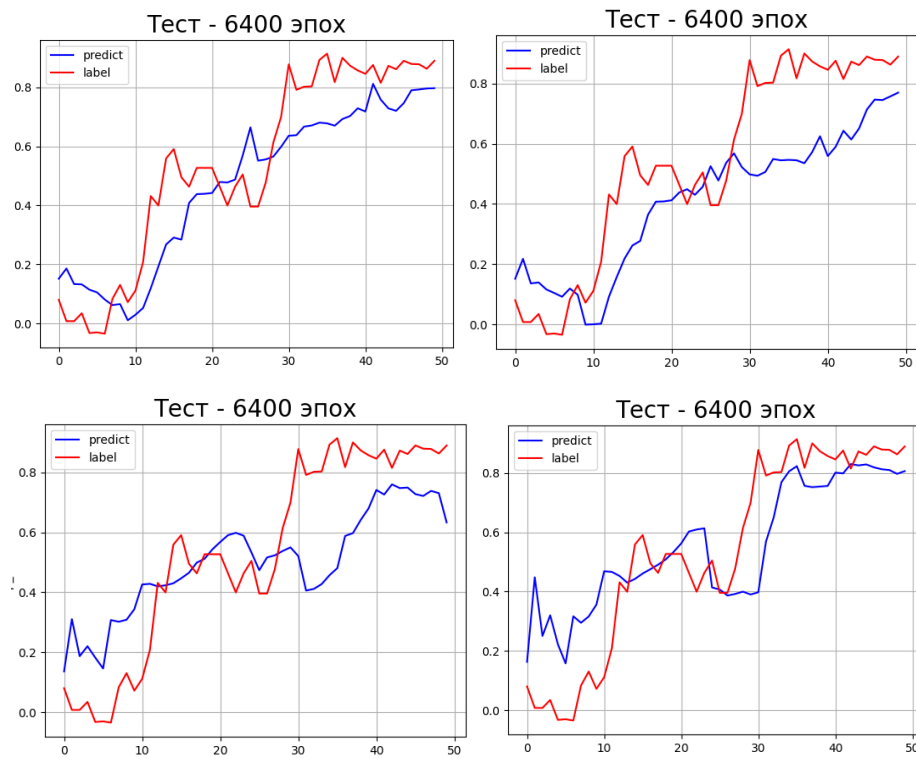


Рисунок 18. Тест с 6400 эпохами (слева – MSE, справа – MAE; сверху – MLP, Снизу – CNN).

Используя 10000 эпох:

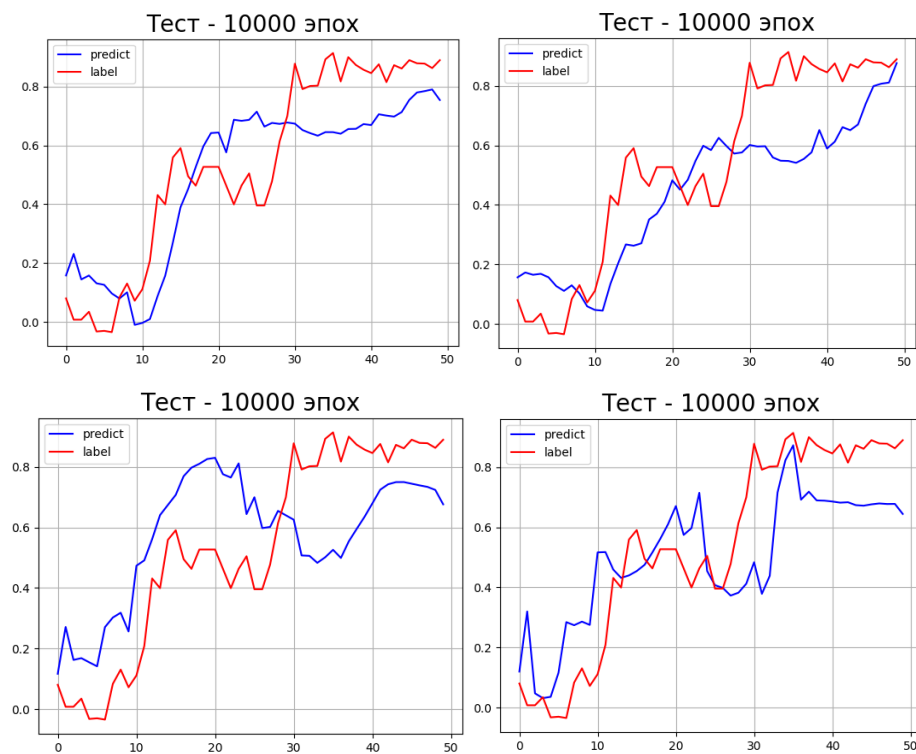


Рисунок 19. Тест с 10000 эпохами (слева – MSE, справа – MAE; сверху – MLP, Снизу – CNN).

Данные графики описывают результаты «предугадывания» НС в результате работы с тестовым датасетом (этот набор информации НС еще не видела и обучалась она не на нем).

MLP	Time	CNN	Time
50	1.00	50	3.484
100	2.00	100	4.673
200	3.586	200	8.708
400	6.720	400	13.0394
800	13.572	800	25.757
1600	27.80	1600	48.983
3200	56.043	3200	98.291
6400	195.29	6400	193.839
10000	180.47	10000	299.83

Рисунок 20. Зависимость времени обучения НС от количества эпох при использовании MSELoss.

MLP	Time	CNN	Time
50	1.00	50	3.895
100	2.00	100	4.762
200	3.455	200	8.210
400	7.208	400	13.410
800	14.6937	800	25.776
1600	28.3952	1600	51.802
3200	58.797	3200	102.44
6400	117.310	6400	202.842
10000	187.9521	10000	317.748

Рисунок 21. Зависимость времени обучения НС от количества эпох при использовании MAELoss.

По рисунку 20 и 21 можно увидеть, что время обучения НС от количества эпох зависит линейно. По остальным метрикам, которые представлены с точными значениями в файле «KP.xlsx», можно понять, что MLP обучается в 1.5 раза быстрее, чем CNN и имеет более точный результат и на этапе обучения, и на этапе тестирования. Также можно увидеть, что отсутствует сильная разница между использованием MSELoss или MAELoss. Обе функции справляются со своей задачей.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы были достигнуты следующие цели:

- 1) Рассмотрено, что такое сверточная нейронная сеть;
- 2) Чем сверточная сеть отличается от простого линейного перцептрона;
- 3) Описана работа по составлению датасета на основе данных о тикере MSFT;
- 4) Описана работа по настройке и подбору нужных функции-оптимизации, функции-потерь, функции-активации и других настроек архитектуры НС;
- 5) Были проведены анализы работы сверточной НС при разных параметрах;
- 6) Было проведено сравнение с работой линейного перцептрона при разных параметрах.

Результаты экспериментов оказались достаточно «живые» и говорят о том, что сверточные нейронные сети способны анализировать не только изображения, но и большие матрицы данных. Однако биржа является предметом с очень сложными зависимостями и закономерностями и CNN не имеет возможностей точно предугадать следующее поведение биржи.

Оптимизация параметров модели и правильный выбор архитектуры CNN существенно влияют на достижение высокой точности классификации. Также важным аспектом является качество предварительной обработки данных, включая масштабирование, аугментацию и нормализацию, что дополнительно улучшает способность модели к обобщению.

Полученные результаты предоставляют основу для дальнейших исследований и применения сверточных нейронных сетей в различных областях. Возможным продолжением для данной работы, является проведение тестовых торгов на основе предсказанных нейронной сетью результатов. Таким способом можно будет корректно проанализировать прибыльность использования такой сети и ее актуальность.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. «Сверточная нейронная сеть с нуля» [Электронный ресурс] – URL: <https://programforyou.ru/poleznoe/convolutional-network-from-scratch-part-zero-introduction> (дата обращения: 20.10.2023 - 24.10.2023).
2. Блог о сверточной нейронной сети [Электронный ресурс] – URL: <https://habr.com/ru/companies/skillfactory/articles/565232/> (дата обращения: 20.10.2023).
3. PyTorch Tutorials [Электронный ресурс] – URL: <https://pytorch.org/tutorials/> (дата обращения: 20.10.2023 - 30.10.2023).
4. PyTorch Documentation [Электронный ресурс] – URL: <https://pytorch.org/docs/stable/index.html> (дата обращения: 20.10.2023 - 30.10.2023).
5. Сайт с датасетами [Электронный ресурс] – URL: <https://www.kaggle.com/datasets/d4rklucif3r/cat-and-dogs?select=dataset> (дата обращения: 21.10.2023).