



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

Институт математики и компьютерных технологий
Департамент программной инженерии и искусственного интеллекта

Лемеш Владислав Евгеньевич

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

бакалаврская работа

вид ВКР

РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ ДЛЯ ТРЕЙДИНГА НА ОСНОВЕ
НЕЙРОННЫХ СЕТЕЙ. ПОДСИСТЕМЫ ФОРМИРОВАНИЯ ТОРГОВЫХ СИГНАЛОВ И
ТЕСТИРОВАНИЯ

по направлению подготовки (специальности) 09.03.04 «Программная инженерия»
профиль «Программная инженерия»

Владивосток
2023

<p>В материалах данной выпускной квалификационной работы не содержатся сведения, составляющие государственную тайну, и сведения, подлежащие экспортному контролю</p> <p>Уполномоченный по экспортному контролю</p>	<p>Автор работы подпись</p> <p>группа Б9119-09.03.04прогин</p> <p>« _____ » 2023 г.</p>
<p>И.Л. Артемьева</p> <p>подпись</p> <p>И.О. Фамилия</p> <p>« _____ » 2023 г.</p>	<p>Руководитель ВКР доцент, к.ф.-м.н. должность, ученое звание</p> <p>В.Н. Лиховидов</p> <p>подпись</p> <p>И.О. Фамилия</p> <p>« _____ » 2023 г.</p>
<p>Защищена в ГЭК с оценкой</p> <p>«Допустить к защите»</p> <p>и.о. директора департамента</p>	
<p>Секретарь ГЭК</p> <p>О.А. Крестникова</p> <p>подпись</p> <p>И.О. Фамилия</p> <p>« _____ » 2023 г.</p>	<p>ученая степень, ученое звание</p> <p>О.А. Крестникова</p> <p>подпись</p> <p>И.О. Фамилия</p> <p>« _____ » 2023 г.</p>

Оглавление

Введение.....	4
1 Обзор предметной области.....	7
1.1 Искусственная нейронная сеть	7
1.1.1 Формальная модель нейрона	7
1.1.2 Топология нейронной сети	10
1.1.3 Сверточная нейронная сеть.....	14
1.1.4 Обучение нейронной сети	17
1.1.5 Задача обучения с учителем.....	18
1.2 Перенос обучения (Transfer Learning) в нейронных сетях	19
1.3 Обзор существующих решений.....	20
1.3.1 AmiBroker.....	21
1.3.2 Trade Ideas	22
1.3.3 Tickeron	22
1.3.4 NeuroShell Trader.....	23
1.3.5 Scanz	25
1.3.6 Вывод	26
2 Анализ предметной области	28
2.1 Описание выборки данных	28
2.2 Модель нейронной сети для формирования торгового сигнала	31
2.3 Обучение нейронной сети	34
2.4 Применение трансферного обучения.....	37
2.5 Модель системы формирования торговых сигналов	39
3 Разработка проекта системы	41
3.1 Контекстная диаграмма	41
3.2 Архитектурно-контекстная диаграмма	42
3.3 Диаграмма потоков данных	44
3.4 Диаграмма use-case	46
3.5 Функциональные требования	48
4 Реализация и тестирование программной системы.....	52
4.1 Тестирование системы.....	52
4.2 Исследование результатов применения трансферного обучения.....	55
Заключение	63
Список источников	64
Приложение А	67

Введение

Технологии, связанные с нейронными сетями, применяются сегодня для решения самых разных задач в науке, технике, бизнесе и медицине, где необходима обработка больших объемов данных, классификация и прогнозирование. Сложность и размеры тех сетей, которые при этом создаются, достигли огромных размеров. Например, сеть CLIP (Contrastive Language-Image Pre-training) – модель, разработанная компанией OpenAI, которая обучается ассоциировать изображения и текст. Она содержит около 400 миллионов параметров и способна классифицировать изображения по текстовому описанию и выполнять связанные задачи. Ещё одним примером может служить сеть BigGAN (Big Generative Adversarial Network) – это генеративная модель, разработанная Google Brain. Она содержит около 11 миллиардов параметров и способна генерировать высококачественные изображения различных классов, таких как животные, люди и объекты. Чтобы настроить (обучить) сеть таких масштабов, необходимо иметь наборы данных (обучающие выборки) соответствующих размеров. Статистические соображения подсказывают, что для обучения сети, имеющей N настраиваемых параметров (“весов”), надо иметь обучающую выборку известных объектов в несколько раз большей длины.

Получение таких выборок может быть очень трудоемкой задачей, поэтому решая новую задачу, для которой подходящие выборки еще не созданы, исследователь оказывается перед сложной проблемой. Для ее преодоления разработаны и применяются несколько подходов, одним из которых является перенос обучения (transfer learning, трансферное обучение).

Предположим, необходимо обучить сеть больших размеров, например, многослойный персептрон или сверточную сеть, содержащую большое число слоев из большого числа нейронов, и для ее обучения нет подходящей обучающей выборки, а есть малая выборка, явно недостаточная для сети таких

масштабов. Тогда из сети выделяют частичную сеть (первые ее слои) и эту сеть обучаю на другой выборке. После того как эта часть сети обучена, веса в ней фиксируются (замораживаются) и обучается только оставшаяся часть сети, на имеющейся малой выборке, созданной для решаемой задачи. Таким образом, сеть обучается решению некоторых других задач, а потом дополнительно дообучается для решения целевой задачи.

Такой подход показал свою практическую эффективность и теперь трансферное обучение широко применяется в различных приложениях. Например, в работе [27] описывается успешное применение трансферного обучения для задачи прогнозирования цен на нефть. В работе [28] описано применение трансферного обучения в задаче распознавания эмоциональной окраски текста.

Идея данной дипломной работы состоит в том, чтобы автоматизировать процесс построения обучающих выборок Source-области. Предполагается, что необходимо обучать сверточные нейронные сети, для которых обучающие выборки представляют собой наборы двумерных изображений. Заранее неизвестно, какими свойствами должны обладать эти картинки, поэтому желательно иметь возможность генерировать наборы картинок, подходящие для обучения различных сверточных сетей. Источником таких изображений может служить финансовый рынок: графики цен финансовых активов, как известно, обладают неограниченной статистической изменчивостью, можно попытаться использовать их для генерирования обучающих выборок.

При анализе финансовых рынков применяются одномерные наборы данных – фрагменты ценовых графиков, используемые для обучения нейронных сетей классификации и прогнозированию. Для обучения же сверточных сетей нужны двумерные изображения. В дипломной работе предложен способ превращения одномерных объектов (фрагментов графиков) в двумерные картинки. Сначала с помощью нейронной сети (многослойного персептрона) создается классификатор, выделяющий на графиках фрагменты

некоторых классов. Затем классифицированные фрагменты превращаются в двумерные изображения. Благодаря большому разнообразию финансовых рынков (графики множества финансовых активов в различных временных масштабах) можно надеяться получить наборы двумерных изображений с самыми разнообразными статистическими свойствами.

Для уверенности в том, что выделенные фрагменты графиков в достаточной мере отражают статистические свойства рынков, имеется естественный способ: применить построенный классификатор для торговых операций на финансовом рынке, то есть, построить торговый робот. Если робот достаточно успешно торгует на рынке, значит встроенный в него классификатор правильно воспринимает статистику рынка и генерируемые им двумерные выборки, можно надеяться, будут успешно работать в обучении сверточных сетей.

Цель данной работы состоит в разработке подсистем формирования торговых сигналов и тестирования в программной системе для трейдинга на основе нейронных сетей.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- 1) Провести обзор существующих решений, рассматриваемой предметной области.
- 2) Провести анализ предметной области и построить ее модель.
- 3) Разработать проект программной системы.
- 4) Реализовать и провести тестирование программной системы.
- 5) Исследовать результаты применения трансферного обучения.

1 Обзор предметной области

1.1 Искусственная нейронная сеть

1.1.1 Формальная модель нейрона

Искусственная нейронная сеть (ИНС) – упрощенная модель биологической нейронной сети, представляющая собой совокупность искусственных нейронов, взаимодействующих между собой [16]. Искусственный нейрон принимает на вход несколько различных сигналов, преобразует их и передает другим нейронам. Другими словами, искусственный нейрон – это функция $R^n \rightarrow R$, которая преобразует несколько входных параметров в один выходной.

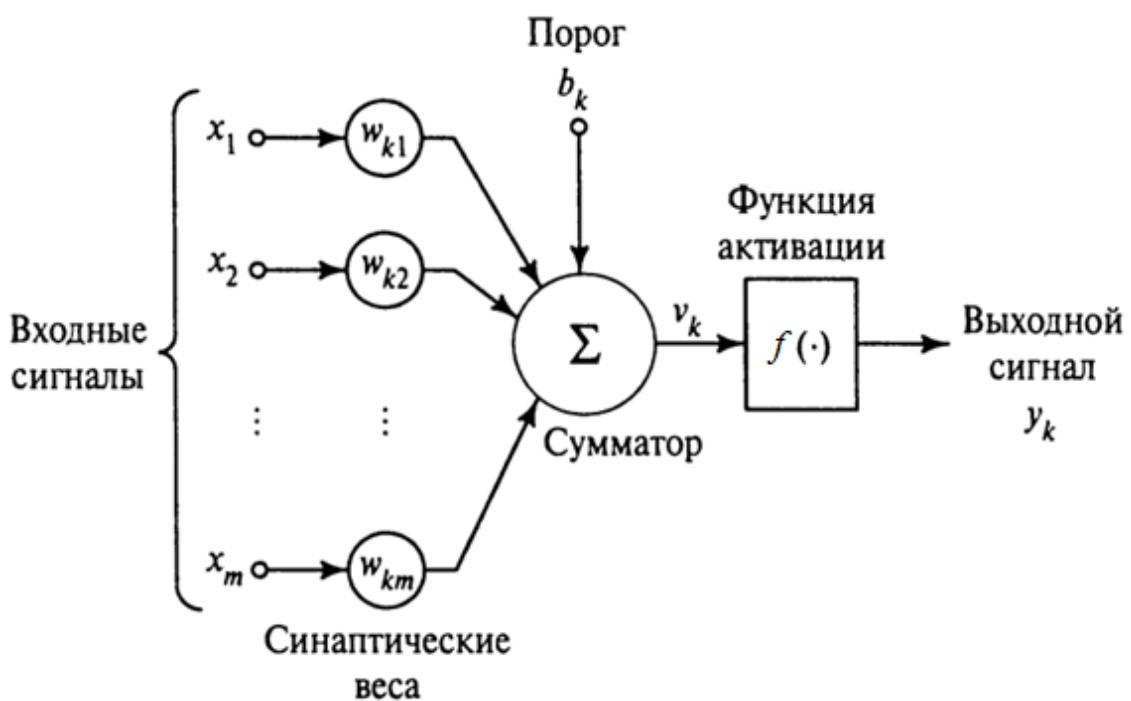


Рисунок 1 - Модель нейрона

На рисунке 1 показана модель нейрона, лежащего в основе ИНС. В этой модели можно выделить три основных элемента:

1. Набор связей, каждая из которых характеризуется своим весовым коэффициентом. В частности, входной сигнал x_j , связанный с нейроном k , умножается на весовой коэффициент w_{kj} .

2. Сумматор складывает взвешенные входные сигналы. Этую операцию можно описать как линейную комбинацию.

3. Функция активации ограничивает амплитуду выходного сигнала нейрона. Обычно нормализованный диапазон амплитуд выхода нейрона лежит в интервале $[0, 1]$ или $[-1, 1]$. Примеры наиболее часто используемых функций активации нейронов приведены на рисунке 2.

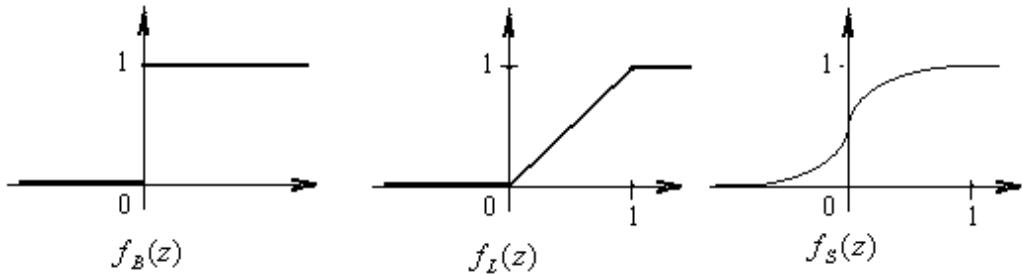


Рисунок 2 - Примеры функций активации

В модель нейрона, показанную на рисунке 1, включен пороговый элемент, который обозначен символом b_k . Эта величина отражает увеличение или уменьшение входного сигнала, подаваемого на функцию активации.

В математическом представлении функционирование нейрона k можно описать следующей парой уравнений:

$$u_k = \sum_{j=1}^m w_{kj} x_j, \quad (1)$$

$$y_k = f(u_k + b_k), \quad (2)$$

где x_1, x_2, \dots, x_m – входные сигналы; $w_{k1}, w_{k2}, \dots, w_{km}$ – весовые коэффициенты нейрона k ; u_k – линейная комбинация входных сигналов; b_k – порог; f – функция активации; y_k – выходной сигнал нейрона.

Нейрон с функцией активации, принимающей два значения,

$$f_B(z) = H(z) = \begin{cases} 0, z < 0 \\ 1, z \geq 0 \end{cases} \quad (3)$$

($H(z)$ – функция Хевисайда), называется **пороговым элементом** (бинарным нейроном); нейрон с функцией активации

$$f_L(z) = \begin{cases} 0, & z < 0 \\ z, & 0 \leq z < 1 \\ 1, & z \geq 1 \end{cases} \quad (4)$$

называется **линейным пороговым элементом**; во многих приложениях используется функция активации без верхнего ограничения **ReLU**:

$$f_p(z) = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases} \quad (5)$$

Функция

$$f_s(z) = \frac{1}{1 + e^{-az}}, \quad a > 0 \quad (6)$$

является типичным примером гладкой функции активации нейронов (**сигмоидальная функция активации**).

Еще одна часто используемая в приложениях гладкая функция активации – обратная тригонометрическая:

$$f(z) = \frac{1}{\pi} \operatorname{arctg}(az) + \frac{1}{2}, \quad (7)$$

график которой аналогичен $f_s(z)$. Возможны и другие гладкие функции активации, например функция распределения стандартного нормального закона является типичной функцией активации сигмоидального типа.

Все рассмотренные функции (за исключением ReLU) являются однополярными в том смысле, что они принимают значения в диапазоне от 0 до 1. По многим причинам в некоторых приложениях бывает удобно использовать биполярные функции активации, принимающие значения разных знаков, в диапазоне от -1 до +1, например, гиперболический арктангенс (tanh)

$$f(z) = \frac{e^{az} - e^{-az}}{e^{az} + e^{-az}}, \quad (8)$$

или

$$f(z) = \frac{1 - e^{-az}}{1 + e^{-az}}, \quad (9)$$

а также

$$f(z) = \frac{az}{\sqrt{1 + a^2 z^2}} \quad (10)$$

Графики этих функций отличаются от изображенных на Рисунке 2 удвоенным масштабом по оси ординат и центрированием значений функции f (для всех этих функций $f(0) = 0$); иначе говоря, если f_1 однополярная функция, то ей соответствует биполярная функция f_2 вида

$$f_2(z) = 2f_1(z) - 1 \quad (11)$$

1.1.2 Топология нейронной сети

Топология нейронной сети задается ориентированным графом:

$$G = \{(i, j); i, j \in \{1, 2, \dots, Nr\}\} \quad (12)$$

Множество вершин $N = \{1, 2, \dots, Nr\}$ графа G изображает нейроны сети (количество нейронов равно Nr), а существование в графе дуги (i, j) означает наличие соединения, направленного от нейрона i данной сети к нейрону j . Для того чтобы определить множество входных и выходных нейронов сети, введем формально две вершины I и O (не входящие в множество вершин графа G) и назначим некоторое подмножество **Input** вершин графа G в **качестве входных нейронов** (введя в рассмотрение дуги, ведущие от I к входным нейронам), а другое подмножество – **Output** в качестве **выходных нейронов** (выходные нейроны соединены дугами с O). Тогда нейрон i принадлежит множеству входных нейронов, если существует дуга (I, i) , а нейрон j принадлежит множеству выходных нейронов, если существует дуга (j, O) .

Веса синаптических связей, присвоенные межнейронным соединениям, составляют **матрицу весов**

$$W = \{w_{ij}, (i, j) \in G\} \quad (13)$$

В матрицу W обычно (если не оговорено иное) будут включаться и внутренние параметры (пороги) нейронов w_{0j} (w_{0j} обозначает пороговый параметр нейрона j), не имеющие отношения к межнейронным связям.

В матричном виде изменение состояния нейронной сети во времени представляется уравнением

$$Y(t+1) = f(WY(t) + w), \quad (14)$$

где w обозначает вектор, составленный из w_{01}, \dots, w_{0Nr} .

Для удобства записи таких уравнений часто будет применяться расширенная форма представления матрицы весов и вектора состояний сети: к вектору Y в этом случае добавляется компонента, всегда равная 1, а к матрице весов сети добавляется столбец w , составленный из пороговых значений нейронов сети

$$W \rightarrow [W|w_{0i}, i = 1, \dots, Nr], Y \rightarrow [Y|1]^T \quad (15)$$

В этих обозначениях уравнение динамики принимает более компактную форму

$$Y(t+1) = f(WY(t)) \quad (16)$$

которая будет использоваться в тех случаях, когда не возникает необходимости конкретизировать структуру внутренних взаимосвязей в сети.

Характер изменения состояния сети во времени (траектория сети) существенно зависит от ее топологии. Два наиболее важных в этом отношении типа топологии – сети прямого распространения и сети с обратными связями. **Сеть прямого распространения** (forward propagation net) имеет граф G без циклов: если в графе G существует путь от вершины i к вершине j , то есть последовательность дуг вида

$$R(i, j) = \{(i, j_1), (j_1, j_2), \dots, (j_{n-1}, j_n), (j_n, j)\}, \quad (17)$$

то в графе G не существует пути от вершины j к i .

В сети прямого распространения можно естественным образом установить прямое функциональное соответствие между фиксированным значением вектора входов сети $X \in R^K$ (K обозначает количество входных нейронов сети, то есть, размерность входного вектора) и состояниями всех нейронов сети. Действительно, обозначим наибольшую из длин путей,

соединяющих нейрон i с множеством входов (длина пути понимается как число входящих в него дуг):

$$L_i = \max_{j \in Input} \{R(j, i)\}, \quad (18)$$

где L_i есть расстояние нейрона i от входного множества.

Все нейроны сети естественным образом упорядочиваются по их удаленности от входного множества, при этом выходная реакция y_i нейрона i зависит только от состояний нейронов, входящих в пути следующего вида:

$$R(j, i), j \in Input, \quad (19)$$

Иначе говоря, y_i зависит только от y_k , для которых $L_k < L_i$. При фиксированном векторе входов X выход $y_i(t)$ нейрона i будет постоянным при $t \geq L_i$. Состояние сети (набор выходов всех нейронов сети), а следовательно, и выходная реакция сети будут постоянными, если входной вектор X зафиксировать на время, большее чем

$$T = \max_i \{L_i\} \quad (20)$$

После изменения значения внешнего входного вектора переходной процесс в сети прямого распространения заканчивается через время T , если вход остается фиксированным. Поэтому, если укрупнить масштаб времени, сделав T единичным шагом по оси времени, то получится полностью статическое описание динамики сети: выход сети в данный момент времени является функцией от вектора входов сети в этот же момент времени. Поэтому сети прямого распространения рассматриваются далее как функциональные преобразователи: выход сети – вектор $Y \in R^M$, представляющий набор реакций выходных нейронов сети (M – количество выходных нейронов) – является функцией от входного вектора $X \in R^K$, зависящей от матрицы весов W как от параметра:

$$Y = N(W, X) \quad (21)$$

Назовем L_i - расстояние нейрона i от входа сети – **уровнем нейрона i** .

Сеть прямого распространения называется **многоуровневой (многослойной) сетью**, если все нейроны сети могут быть упорядочены в L ($L \geq 1$) уровней таким образом, что все нейроны k -го уровня имеют входы только от нейронов уровня $k-1$, а выходы нейронов k -го уровня являются входами нейронов только $k+1$ -го уровня. Иначе говоря, в графе G многослойной сети для всякого i не существует дуг, отличных от дуг вида $(j,i), L_j = L_i - 1$ и $(i,j), L_j = L_i + 1$.

В многослойной сети все нейроны распределены на упорядоченные непересекающиеся подмножества (слои) так, что нейроны нижнего слоя могут иметь связи только к нейронам следующего слоя. Максимальное значение L среди уровней нейронов определяет количество слоев (уровней) сети. Общий вид топологии многослойной сети представлен на рисунке 3. В теории нейронных сетей такого типа структуры имеют общее название – **персептроны**.

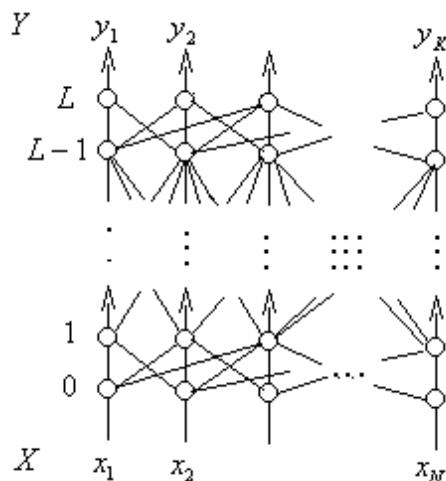


Рисунок 3 - Схема многоуровневого персептрана

Персептрон, изображенный на рисунке 3, имеет L уровней, N входных и K выходных нейронов. Матрица весов такого персептрана естественным образом распадается на L наборов $\{w_{ij}(l), l=1, \dots, L\}$, где $w_{ij}(l)$ - вес связи, ведущей от нейрона i уровня $l-1$ к нейрону j уровня l . Нейроны уровня 0 (входные) в

этих обозначениях не имеют весов, а являются просто разветвительными элементами, передающими входные сигналы X на все нейроны первого уровня. Выходом многослойной сети $Y = N(W, X)$ является вектор $Y \in R^K$, состоящий из выходных сигналов нейронов L -го слоя.

1.1.3 Сверточная нейронная сеть

В данной дипломной работе для исследований применяются сверточные сети; они наиболее широко используются в приложениях. Так, из большого числа работ, рассмотренных в обзоре [20] более 30% сетей являются сверточными.

Свёрточные нейронные сети (CNN) представляют собой класс нейронных сетей, которые используют операцию свертки в своей работе. Они широко применяются в компьютерном зрении для задач, таких как распознавание изображений и сегментация объектов [13].

Архитектура CNN состоит из трех типов слоев: сверточных слоев, субдискретизирующих слоев и полносвязных слоев. Сверточные и субдискретизирующие слои чередуются, формируя входной вектор признаков для полносвязной нейронной сети. На рисунке 4 изображен пример архитектуры сверточной нейронной сети.

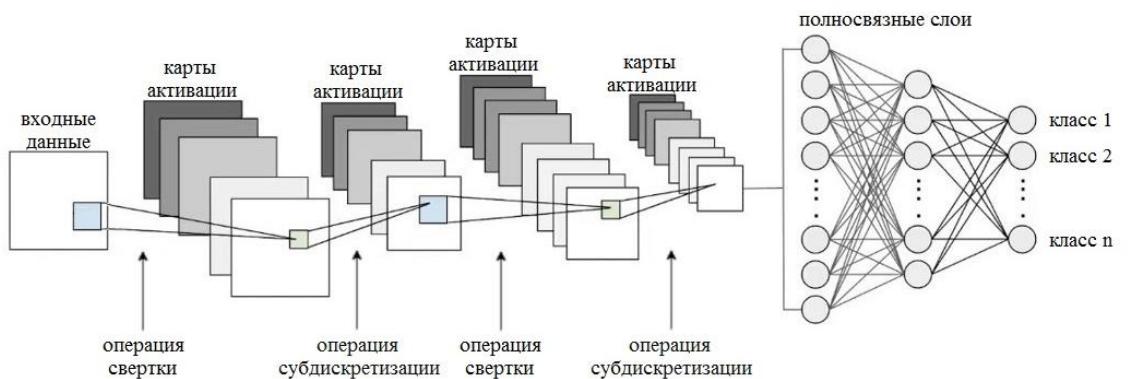


Рисунок 4 - Архитектура сверточной нейронной сети

Сверточный слой применяет набор фильтров (ядер) к входным данным, представляющим собой набор матриц признаков, применяя операцию свертки. Каждый фильтр представляет собой матрицу весовых коэффициентов, которая проходит по входным матрицам с определенным шагом и вычисляет сумму произведений весов и соответствующих признаков в каждой позиции [13]. Результатом операции свертки является набор матриц, называемых картами активации. На рисунке 5 изображена операция свертки.

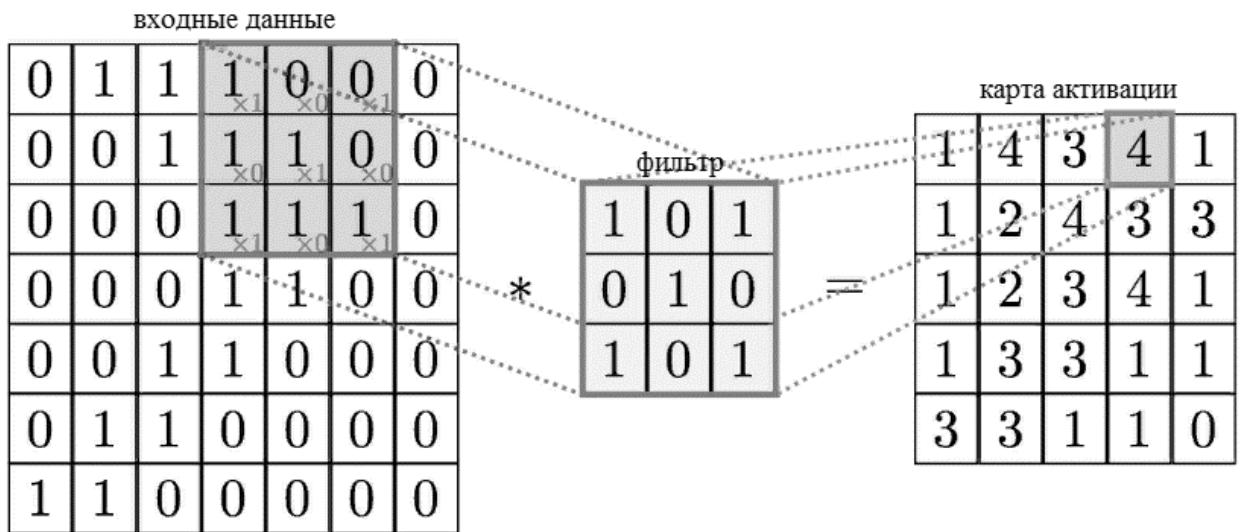


Рисунок 5 - Операция свертки

Операция субдискретизации выполняется путем разбиения входной карты активации на непересекающиеся подматрицы и выбора одного элемента, например, максимального значения в каждой подматрице. Затем из полученных элементов формируют новую карту активации меньших размеров [13]. На рисунке 6 изображена операция субдискретизации.



Рисунок 6 - Операция субдискретизации

Основное назначение сверточных сетей – распознавание двумерных изображений. Но на самом деле эти сети являются весьма универсальным инструментом, так как часто применяется такой прием, когда объекты, не являющиеся изображениями, представляют в виде двумерных изображений, а затем к полученным датасетам применяют сверточные сети.

Например, такое представление объекта в виде 2-мерного изображения для классификации использовано в работе [26]: числовое представление вредоносной программы (malware) преобразуется в двумерное изображение и датасеты таких изображений анализируются далее с помощью сверточных нейронных сетей. На рисунке 7 представлены примеры двумерных представлений различных типов вредоносных программ.

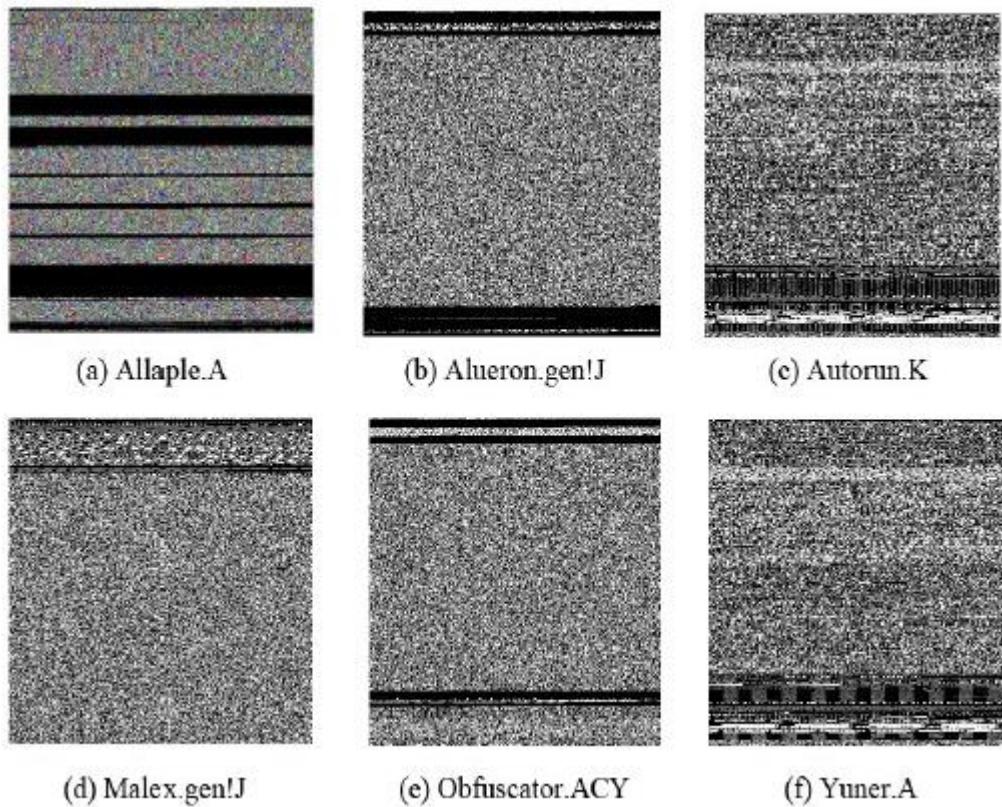


Рисунок 7 - Двумерные представления различных типов вредоносных программ

1.1.4 Обучение нейронной сети

Обучение нейронной сети – это поиск такого набора весовых коэффициентов, при котором входной сигнал после прохода по сети преобразуется в нужный нам выходной.

Обучающая выборка – конечный набор входных сигналов, по которым происходит обучение сети.

Прежде чем использовать нейронную сеть после обучения, обычно производится оценка качества ее работы на так называемой контрольной выборке.

Контрольная выборка – конечный набор входных сигналов, по которым происходит оценка качества работы сети.

Обучение нейронной сети можно разделить на два подхода: обучение с учителем и обучение без учителя. В первом случае веса меняются так, чтобы

ответы сети минимально отличались от уже готовых правильных ответов, а во втором случае сеть самостоятельно классифицирует входные сигналы.

1.1.5 Задача обучения с учителем

При обучении с учителем нейронная сеть обучается на размеченном наборе данных и предсказывает ответы, которые используются для оценки точности алгоритма на обучающих данных.

Одним из типов задач обучения с учителем, является задача классификации, которая имеет следующий вид.

Задано множество объектов X , множество допустимых ответов Y , и существует целевая функция $y^*: X \rightarrow Y$, значения которой $y_i = y^*(x_i)$ известны только на конечном множестве объектов $\{x_1, \dots, x_\ell\} \subset X$. Пары «объект-ответ» (x_i, y_i) называются прецедентами. Совокупность пар $X^\ell = (x_i, y_i)|_{i=1}^\ell$ называется обучающей выборкой.

Задача обучения по прецедентам заключается в том, чтобы по выборке X^ℓ восстановить зависимость y^* , то есть построить решающую функцию $\alpha: X \rightarrow Y$, которая приближала бы целевую функцию $y^*(x)$, причем не только на объектах обучающей выборки, но и на всем множестве X . Решающую функцию α называют алгоритмом.

Признак f объекта x – это результат измерения некоторой характеристики объекта. Формально признаком называется отображение $f: X \rightarrow D_f$, где D_f – множество допустимых значений признака. В частности, любой алгоритм $\alpha: X \rightarrow Y$ так же можно рассматривать как признак. Пусть имеется набор признаков f_1, \dots, f_n . Вектор $(f_1(x), \dots, f_n(x))$ называют признаковым описанием объекта $x \in X$. В дальнейшем мы не будем различать объекты из X и их признаковые описания, полагая $X = D_{f_1} \times \dots \times D_{f_n}$.

Если $Y = \{0, \dots, M\}$, то это задача классификации на M непересекающихся классов. В этом случае всё множество объектов X

разбивается на несколько классов. Алгоритм $\alpha(x)$ должен давать ответ на вопрос «какому классу принадлежит объект $x?$ ».

1.2 Перенос обучения (Transfer Learning) в нейронных сетях

В публикациях по трансферному обучению используются следующие понятия, сформулированные первоначально в работе [24].

Прикладной областью \mathcal{D} (**Domain**) для нейросетевых классификаторов назовем пару

$$\mathcal{D} = (\mathfrak{X}, \mathcal{P}(\mathcal{X})), \quad (22)$$

где \mathfrak{X} обозначает выборочное пространство признаков, из которого на вход нейронной сети поступает обучающая выборка $\mathcal{X} = \{X_1, X_2, \dots, X_n\}$ (training dataset), а $\mathcal{P}(\mathcal{X})$ – распределение вероятностей на пространстве \mathfrak{X} .

Задачей классификации (**Task**) назовем пару

$$\mathcal{T} = (\mathcal{Y}, f(\mathcal{X})), \quad (23)$$

где \mathcal{Y} – пространство образов (**label space**), а $f(\mathcal{X})$ – классификатор (**predictive function**), в общем виде представляющий собой условное распределение вероятностей, $f(\mathcal{X}) = Q(Y|\mathcal{X}), Y \in \mathcal{Y}$.

Перенос обучения (**Transfer learning**): пусть имеется прикладная область \mathcal{D}_s и на ней задача классификации \mathcal{T}_s , которые назовем источником знания (**Source knowledge**), а также другая пара $(\mathcal{D}_t, \mathcal{T}_t)$, назовем ее целевым знанием (**Target knowledge**); предполагается, что $\mathcal{D}_s \neq \mathcal{D}_t$ (то есть, $\mathfrak{X}_s \neq \mathfrak{X}$ либо $\mathcal{P}_s(\mathcal{X}_s) \neq \mathcal{P}_t(\mathcal{X}_t)$) или же $\mathcal{T}_s \neq \mathcal{T}_t$ (то есть, $\mathcal{Y}_s \neq \mathcal{Y}_t$ либо $Q_s(Y_s|\mathcal{X}_s) \neq Q_t(Y_t|\mathcal{X}_t)$).

Перенос обучения (**Transfer learning**) состоит в том, что при обучении классификатора $f_t(\mathcal{X})$ в прикладной области \mathcal{D}_t используется информация (знание), доступная из классификатора $f_s(\mathcal{X})$, заданного в области \mathcal{D}_s .

Перенос обучения в некоторых публикациях обозначают как перенос знания (**Knowledge transfer**).

В данной дипломной работе используется подход, который можно отнести к категории **model-based transfer approach**: параметры нейронной сети, полученные в результате обучения на датасете source-области, используются в качестве начальных приближений для сети, применяемой при классификации в target-области. При этом CNN-сеть из target-области может представлять собой сеть из source-области, достроенную дополнительными слоями, параметры которых настраиваются с использованием датасета из target-области.

1.3 Обзор существующих решений

Разрабатываемая программная система, является автоматизированной торговой системой (АТС). АТС, лежащая в основе алгоритмической торговли, представляет собой компьютерную программу, которая создает торговые сигналы (ордера), используя торговую стратегию – предварительно определенный набор правил торговли.

Одним из наиболее перспективных классов АТС являются АТС с использованием нейронных сетей. В таких системах торговые сигналы формируются на основе нейросетевых моделей. Далее проведен обзор существующих АТС на основе следующих критериев:

- автоматический сбор и обработка данных;
- генерация обучающих выборок на основе обработанных данных;
- генерация торговых сигналов на основе обученной модели;
- отображение сгенерированных торговых сигналов;
- ведение автоматической торговли с использование обученной модели;
- обучение моделей для формирования торговых сигналов на новых финансовых инструментах;
- возможность торговли на российских биржах;
- тестирование моделей на исторических данных.

1.3.1 AmiBroker

AmiBroker [20] – это программное обеспечение для технического анализа и разработки торговых стратегий на финансовых рынках. Оно позволяет трейдерам и инвесторам анализировать данные рынка, создавать и тестировать торговые системы, а также автоматизировать торговлю.

Основной функционал AmiBroker включает:

- создание различных типов графиков (свечи, бары, линии);
- применение технических индикаторов;
- использование встроенных торговых стратегий, в том числе основанных на нейронных сетях;
- отображение торговых сигналов;
- ведение автоматической торговли;
- тестирование торговых стратегий на исторических данных.

На рисунке 8 представлено главное окно программы, в котором отображен свечной график акций Microsoft, а также индикатор RSI.



Рисунок 8 - Пример работы AmiBroker

1.3.2 Trade Ideas

Trade Ideas [23] – это платформа, предоставляющая инструменты и ресурсы для анализа финансовых рынков и поиска торговых возможностей.

Основной функционал Trade Ideas включает:

- ведение автоматической торговли;
- генерация торговых сигналов на основе алгоритмов машинного обучения;
- прогнозирование ценового движения;
- фильтрация акций на основе параметров, таких как цена, объем торгов, волатильность;
- построение технических индикаторов.

На рисунке 9 приведено главное окно Trade Ideas

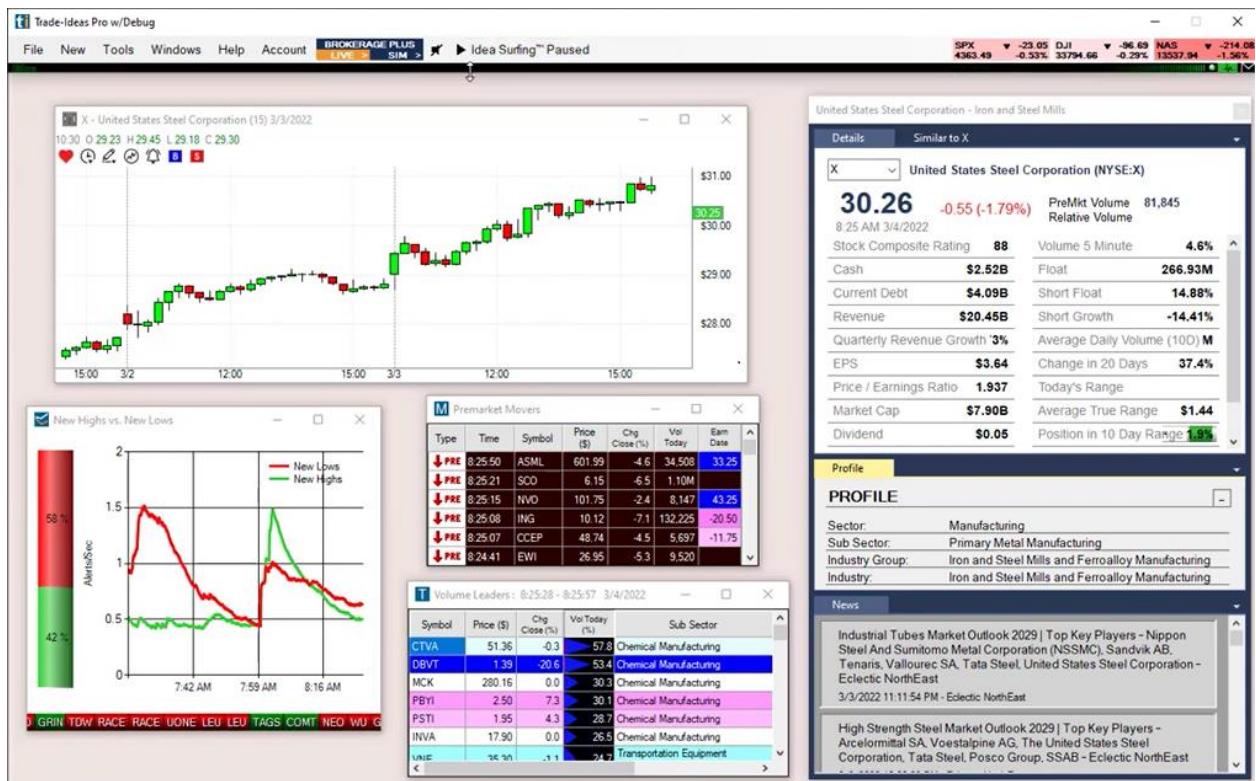


Рисунок 9 - Интерфейс Trade Ideas

1.3.3 Tickeron

Tickeron [22] – это платформа, предоставляющая инструменты и аналитические ресурсы для трейдеров и инвесторов. Она использует

искусственный интеллект для анализа данных рынка и генерации торговых сигналов.

Основной функционал Tickeron включает:

- создание различных типов графиков (свечи, бары, линии);
- применение технических индикаторов;
- генерация торговых сигналов на основе нейросетевых моделей;
- прогнозирование движений цен на основе нейросетевых моделей;
- ведение автоматической торговли;
- создание отчетов об эффективности торговых стратегий.

На рисунке 10 представлено окно программы, в котором отображен график результатов автоматической торговли за выбранный период с информацией о количестве открытых сделок и текущей прибыли на каждую дату.

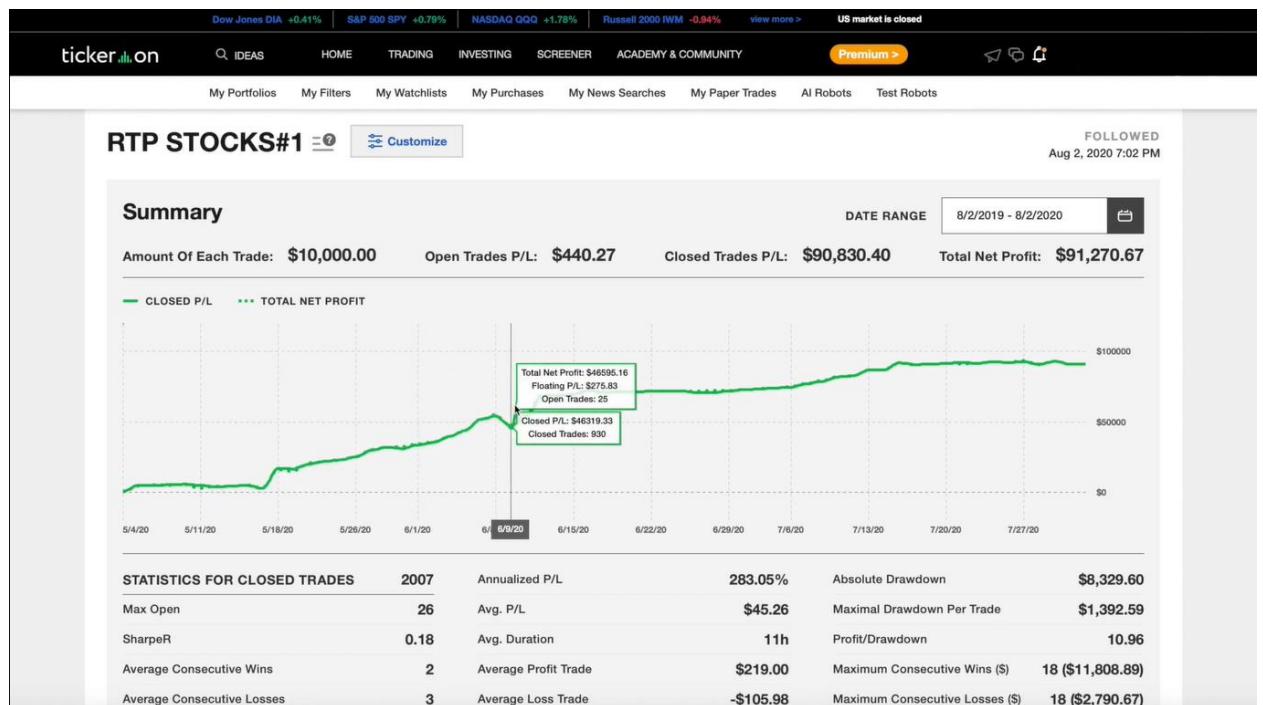


Рисунок 10 - Пример работы tickeron

1.3.4 NeuroShell Trader

NeuroShell Trader [21] – это программное обеспечение для разработки и автоматизации торговых стратегий, которое предоставляет возможности

использования нейронных сетей и других методов анализа данных для принятия торговых решений.

Основной функционал NeuroShell Trader включает:

- создание и тестирование торговых стратегий с использованием предобученных нейронных сетей;
- автоматическое выполнение торговых стратегий на основе заданных условий и сигналов;
- создание графиков, отчетов и статистики для анализа результатов торговых стратегий;
- тестирование торговых стратегий на исторических данных.

На рисунке 11 представлено окно программы, в котором отображен дневной график баров криптовалюты Hesman Shard.



Рисунок 11 - Пример работы NeuroShell Trader

1.3.5 Scanz

Scanz [24] — это платформа, объединяющая множество инструментов для анализа финансовых рынков и принятия торговых решений.

Основной функционал Scanz включает:

- применение технических индикаторов;
- анализ ценовых графиков с использованием графических инструментов;
- фильтрации акций и других финансовых инструментов на основе заданных условий;
- возможность настраивать уведомления о важных событиях на рынке, таких как новости, объявления о прибылях компаний, изменения в технических показателях и других событиях;
- ведение торговых журналов, для отслеживания сделок и результатов стратегий.

На рисунке 12 представлено окно программы, в котором отображен результат фильтрации финансовых инструментов.

Pro Scanner Result										
<input type="checkbox"/> Pause Filter	Found: 216			<input type="button" value="Open Log"/>	<input type="button" value="Filter Info"/>					
Time	Symbol	Company Name	Last	% Chg	% Chg 5 Day	20 ADV	Volume	\$ Volume	News	
14:30:35	IEMG	iShares Core MSCI Emerging Ma...	50.725	-0.75 %	-1.58 %	10,557,609	16,486,527	\$ 835,665,236		
14:30:25	HCA	HCA Healthcare Inc	132.50	-9.35 %	+5.81 %	1,264,130	5,374,492	\$ 713,784,124	6	
14:30:33	SSNC	SS&C Technologies Holdings Inc	50.04	-16.17 %	-14.08 %	1,130,056	13,283,446	\$ 650,232,248	3	
14:30:28	GRUB	GrubHub Inc	72.29	-9.38 %	+1.03 %	1,589,722	7,634,928	\$ 552,311,627	17	
14:30:34	UAA	Under Armour Inc	24.03	-12.43 %	-11.62 %	3,248,051	21,348,283	\$ 501,894,057	4	
14:30:32	MO	Altria Inc	48.20	-4.19 %	-3.68 %	5,512,749	10,206,337	\$ 492,847,638	9	
14:30:12	MDY	SPDR S&P MID-CAP 400	361.19	+0.43 %	+1.35 %	717,487	1,279,714	\$ 459,922,515		
14:30:37	EXAS	Exact Sciences Corp	110.92	-0.55 %	-0.18 %	1,645,277	3,573,917	\$ 426,507,159		
14:30:37	WP	Worldpay Inc	135.92	+0.49 %	+2.60 %	2,058,582	3,129,888	\$ 426,427,201		
14:30:20	RNG	RingCentral Inc	141.335	+12.04 %	+17.27 %	555,282	2,798,227	\$ 399,159,861	2	
14:30:33	GLW	Corning Inc	31.61	-7.39 %	-8.87 %	3,775,268	11,951,344	\$ 377,678,860	6	
14:30:37	ECL	Ecolab Inc	207.38	+3.37 %	+3.71 %	803,920	1,746,181	\$ 361,862,694	8	
14:30:37	IT	Gartner Inc	138.57	-18.84 %	-18.18 %	269,225	2,512,136	\$ 354,295,822	12	
14:30:35	LIN	Linde Plc	197.45	-1.18 %	-4.53 %	1,147,507	1,716,878	\$ 338,159,178	2	
14:30:31	WDAY	Workday Inc	208.03	-2.10 %	-1.80 %	1,216,720	1,556,119	\$ 330,389,462		
14:30:36	BERY	Berry Global Group Inc	47.31	-14.64 %	-12.88 %	1,193,487	6,803,639	\$ 316,834,591	10	
14:30:28	NBIX	Neurocrine Biosciences Inc	95.26	+9.52 %	+11.14 %	512,054	3,036,133	\$ 287,889,486	1	
14:30:01	EWJ	iShares MSCI Japan Index Fund	54.438	-0.58 %	-1.37 %	3,718,193	5,167,733	\$ 281,416,280		
14:30:33	TREX	Trex Co Inc	79.97	+17.57 %	+14.31 %	550,527	3,460,534	\$ 272,637,755	1	
14:30:22	CMI	Cummins Inc	169.309	+4.56 %	+4.65 %	977,559	1,647,257	\$ 271,580,625	11	
14:30:36	MYL	Mylan NV	21.385	+2.91 %	+16.60 %	8,042,499	12,341,591	\$ 262,625,617	4	
14:30:37	NOV	National Oilwell Varco Inc	22.98	+11.12 %	+5.22 %	3,735,955	11,860,621	\$ 261,296,570	7	
14:30:37	SO	Southern Co	55.11	-3.28 %	-0.56 %	3,290,576	4,557,056	\$ 253,147,936	5	
14:30:34	VEEV	Veeva Systems Inc	163.58	+1.13 %	-1.30 %	1,110,748	1,501,425	\$ 243,524,412		
14:30:11	AOS	A.O. Smith Corp	45.02	+2.74 %	+0.78 %	1,589,112	5,458,853	\$ 238,662,923	6	
14:30:34	PAGS	Pagseguro Digital	44.05	-0.78 %	-5.92 %	2,163,254	5,357,227	\$ 237,646,482	1	
14:30:26	DHI	DR Horton Inc	46.43	+9.52 %	+6.47 %	2,814,386	5,056,753	\$ 233,506,730	21	
14:30:34	VRTX	Vertex Pharmaceuticals Inc	170.83	+2.77 %	-2.52 %	956,713	1,369,030	\$ 232,629,076		

Рисунок 12 - Пример работы Scanz

1.3.6 Вывод

В рамках этой работы были рассмотрены пять программных средств, AmiBroker, Trade Ideas, Tickeron, NeuroShell Trader, Scanz. На основе обзора построена сравнительная таблица 1.

Таблица 1 – Сравнение функциональных особенностей

Критерии	AmiBroker	TradeIdeas	Tickeron	NeuroShell	Scanz
Автоматический сбор и обработка данных	+	+	+	+	+
Генерация обучающих выборок	–	–	–	–	–
Генерация торговых сигналов	+	+	+	+	+
Отображение сгенерированных сигналов	+	+	+	+	+
Ведение автоматической торговли	+	+	+	+	–
Обучение моделей для формирования торговых сигналов	–	–	–	–	–
Возможность торговли на российских биржах	–	–	–	–	–
Тестирование моделей на исторических данных	+	–	–	+	–

Проведенный анализ показал, что существующие программные средства реализуют только часть критериев, выдвинутых выше. В частности, ни одно из рассмотренных программных решений не реализует обучение моделей для

формирования торговых сигналов на новых финансовых инструментах. Объясняется это тем, что при разработке АТС данного класса возникает проблема ограниченности данных: нейронные сети требуют большого объема данных для обучения и эффективной работы. Однако, качественные исторические данные финансовых рынков могут быть труднодоступными. Большинство доступных данных являются неполными, что ограничивает возможности обучения.

В данной дипломной работе для решения данной проблемы предлагается использовать трансферное обучение. Таким образом, разработка системы для трейдинга на основе нейронных сетей остаётся актуальной.

2 Анализ предметной области

2.1 Описание выборки данных

Предметная область, рассматриваемая в данной работе – алгоритмическая торговля на основе нейронных сетей. Профессионалами в этой предметной области являются трейдеры, главная цель которых – заработать на разнице валютных курсов.

Трейдер в реальном времени анализирует валютную пару, представленную в виде временного ряда, и принимает решение о заключении сделки. Для этого необходимо определить, как изменится направление движения цены, то есть определить является ли текущий элемент временного ряда разворотной точкой тренда или нет. Под точкой разворота тренда понимают элемент временного ряда, в котором происходит изменение общего направления движения цены с восходящего на нисходящее или наоборот. Нейтральная точка – элемент временного ряда, случайно взятый между точками разворота. От класса элемента зависит, будет ли совершена сделка и если будет, то принимается решение о покупке или продаже.

Таким образом имеется последовательность $P_k = \{p_{t_k}\}$ точек разворота тренда, а также нейтральных точек, взятых из временного ряда $P = \{p_t\}$, состоящего из значений курса обмена одной валюты к другой в моменты времени $t \in T$ соответственно, причем для $\forall p_i, p_{i+1} \in P_k |p_i - p_{i+1}| \geq delta$, $i \in T$, где

$$delta = \frac{\max(\{p_t\}) - \min(\{p_t\})}{100} h, h \in \mathbb{N} \quad (24)$$

На рисунке 13 представлены точки разворота тренда, а также нейтральный точки для валютной пары евро-доллар.

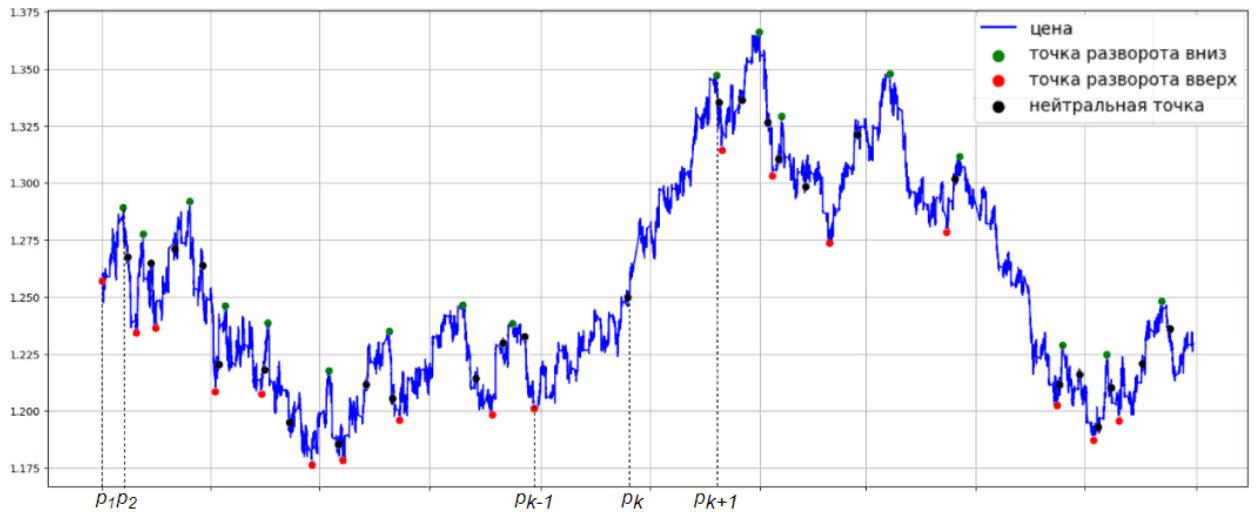


Рисунок 13 - Точки разворота тренда для валютной пары евро/доллар

Каждому элементу p_i последовательности P_k ставится в соответствие вектор, состоящий из n предыдущих значений $x_i = (p_{i-n+1} \dots p_i)$ временного ряда P .

Каждый элемент последовательности P_k кодируется следующим образом:

- если p_i является точкой разворота тренда, в которой происходит изменение направления с восходящего на нисходящее, то p_i представима в виде $(x_i, (0,0,1))$;
- если p_i является точкой разворота тренда, в которой происходит изменение направления с нисходящего на восходящее, то p_i представима в виде $(x_i, (0,1,0))$;
- если p_i не является точкой разворота тренда, то p_i представима в виде $(x_i, (1,0,0))$.

Далее на основе вектора x_i формируется матрица X_i одним из двух способов: при помощи **грамианского углового поля (gramian angular field, GAF)**, или при помощи **многослойного персептрана (MLP)**.

При преобразовании к двумерным данным при помощи грамианского углового поля значения вектора $x_i = (p_1 \dots p_n)$ нормализуются согласно формуле:

$$\hat{p}_j = \frac{(p_j - \max(x_i)) + (p_j - \min(x_i))}{\max(x_i) - \min(x_i)} \quad (25)$$

После нормализации вектор x_i преобразуется путем кодирования значения \hat{p}_j следующим образом:

$$\varphi_j = \arccos(\hat{p}_j) \quad (26)$$

Далее формируется матрица X_i :

$$X_i = \begin{pmatrix} \cos(\varphi_1 + \varphi_1) & \dots & \cos(\varphi_1 + \varphi_n) \\ \cos(\varphi_2 + \varphi_1) & \dots & \cos(\varphi_1 + \varphi_2) \\ \vdots & \ddots & \vdots \\ \cos(\varphi_n + \varphi_1) & \dots & \cos(\varphi_n + \varphi_n) \end{pmatrix} \quad (27)$$

Для наглядности полученную матрицу можно представить как изображение в виде тепловой карты. На рисунке 14 представлен пример полученного изображения.

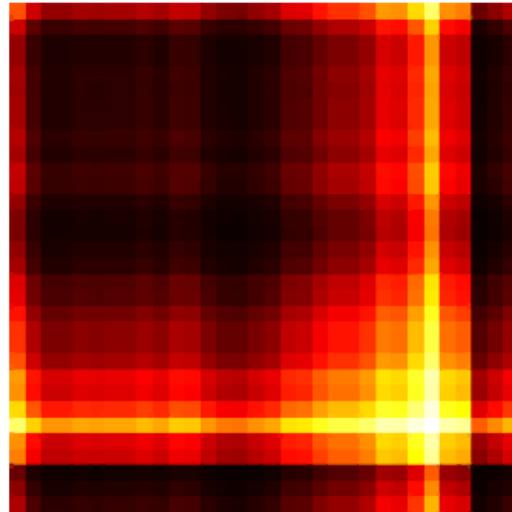


Рисунок 14 - Пример изображения, полученного при помощи GAF

При преобразовании к двумерным данным при помощи многослойного персептрана (MLP) имеется набор векторов $\mathcal{X} = (x_1, x_2, \dots, x_N)$, где $x_i = (p_{i-n+1} \dots p_i)$ служит размеченной обучающей выборкой, содержащей фрагменты из трех классов: точка разворота вверх, точка разворота вниз, нейтральная точка. На этой обучающей выборке обучается сеть типа многослойный персептран (MLP), выходы которого представлены в виде вектора вероятностей принадлежности к трем классам.

Обученный таким образом классификатор используется для формирования обучающей выборки, состоящей из двумерных изображений. Для каждой точки разворота формируется по три матрицы; элемент матрицы представляет собой ответ обученного MLP на фрагментах предыстории ($p_{i-n+1-k} \dots p_{i-k}$), где $k = 0, \dots, n^2$. На рисунке 15 представлен пример полученных изображений

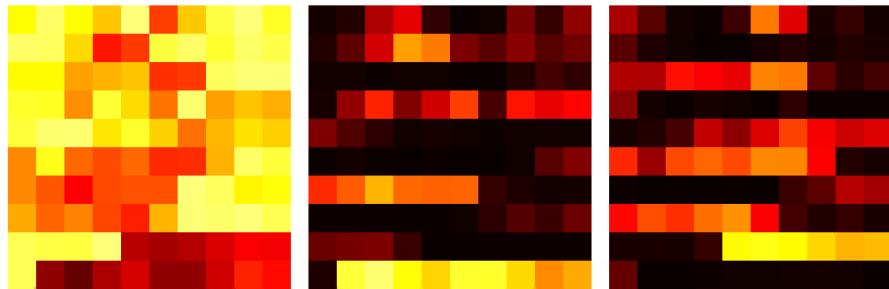


Рисунок 15 - Пример изображений, полученных при помощи MLP

Таким образом, имеется множество пар (прецедентов) $\{(X_i, y_i)\}$, где $X_i \in \mathbb{R}^{3 \times n \times n}$ – тройка матриц признаков элемента при преобразовании при помощи MLP, либо $X_i \in \mathbb{R}^{n \times n}$ – матрица признаков элемента при преобразовании при помощи GAF $y_i \in \{(1,0,0), (0,1,0), (0,0,1)\}$ – класс элемента соответственно.

2.2 Модель нейронной сети для формирования торгового сигнала

В качестве исходной модели используется сверточная нейронная сеть, имеющая два сверточных слоя, два скрытых полносвязных слоя, выходной полносвязный слой и функцию Softmax. На рисунке 16 изображена схема архитектуры исходной модели.

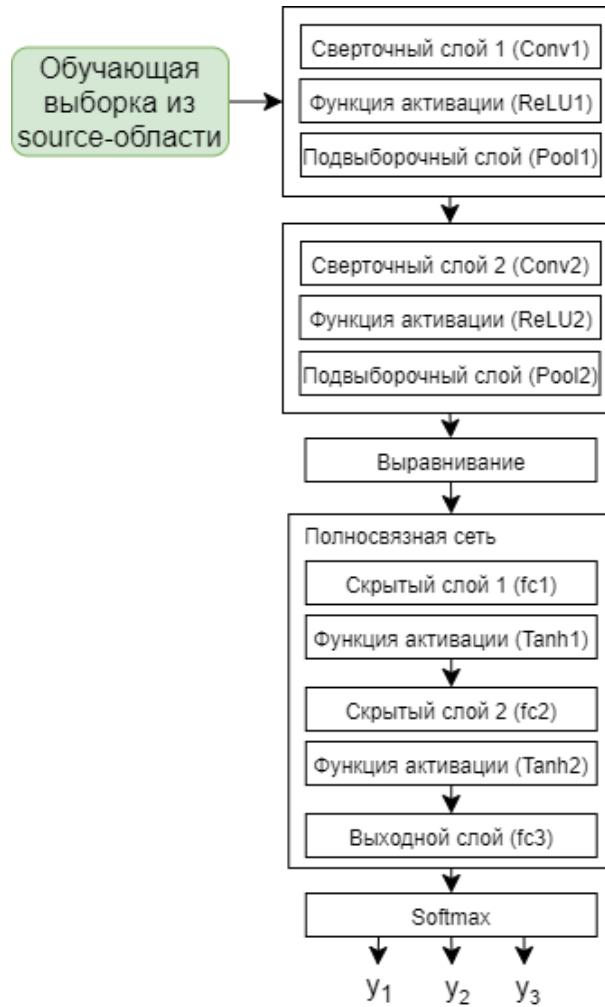


Рисунок 16 - Схема сверточной нейронной сети

На вход сверточного слоя 1 подаётся матрица X размерности (N, N) . Пусть W_1^f – фильтр(ядро) первого сверточного слоя, размерности (K_1, K_1) , $f = 1 \dots F_1$, где F_1 – количество фильтров, S_1 – размер шага. К матрице X применяется операция свертки Conv1:

$$Z_1^f[i, j] = \sum_{p=1}^{K_1} \sum_{q=1}^{K_1} X[i + p, j + q] \cdot W_1^f[p, q] \quad (28)$$

Таким образом формируется карта активации Z_1^f фильтра W_1^f первого слоя, размерности (N_1, N_1) , где:

$$N_1 = \frac{N - K_1}{S_1} + 1 \quad (29)$$

Функция активации ReLU1 применяется поэлементно к карте активации Z_1^f и определяется следующим образом:

$$A_1^f[i, j] = \max(0, Z_1^f[i, j]), \quad (30)$$

где A_1^f – карта активации после применения функции ReLU.

Далее A_1^f поступает на подвыборочный слой Pool1, где A_1^f разбивается на непересекающиеся подматрицы размерности (H, H) . Результатом подвыборочного слоя является матрица P_1^f :

$$P_1^f[i, j] = \max_{p=1 \dots K_2} \max_{q=1 \dots K_2} A_1^f[H \cdot (i - 1) + p, H \cdot (j - 1) + q] \quad (31)$$

Аналогично применяется слой свертки Conv1, функция активации ReLU1 и подвыборочный слой Pool1 для каждого фильтра W_1^f . Полученные карты активации P_1^f становятся входами для следующего сверточного слоя, который определяется аналогично.

Результатом второго сверточного слоя являются матрицы P_2^g , $g = 1 \dots F_2$, где F_2 – количество фильтров. Для перехода к скрытому полносвязному слою fc1, матрицы P_2^g преобразуются в вектор x , путем конкатенации строк матриц P_2^g .

Пусть D – размерность вектора $x = (x_1, \dots, x_D)$, $w_i^1 = (w_{i1}^1, \dots, w_{ij}^1)$ – вектор весовых коэффициентов нейрона i , где $i = 1 \dots H_1$, H_1 – количество нейронов первого полносвязного слоя fc1, b_i^1 – смещение нейрона i . Для каждого нейрона i , вычисляется выходное значение z_{1i} :

$$z_{1i} = \sum_{j=1}^D w_{ij}^1 \cdot x_j + b_i^1 \quad (32)$$

К каждому выходному значению z_{1i} применяется функция активации гиперболический тангенс:

$$a_{1i} = \frac{e^{z_{1i}} - e^{-z_{1i}}}{e^{z_{1i}} + e^{-z_{1i}}} \quad (33)$$

Далее на вход второму полносвязному слою fc2 подаётся вектор $a_1 = (a_{11}, \dots, a_{1H_1})$:

$$z_{2i} = \sum_{j=1}^{H_1} w_{ij}^2 \cdot a_{1j} + b_i^2, \quad (34)$$

$$a_{2i} = \frac{e^{z_{2i}} - e^{-z_{2i}}}{e^{z_{2i}} + e^{-z_{2i}}}, \quad (35)$$

где $w_i^2 = (w_{i1}^2, \dots, w_{iH_1}^2)$ – вектор весовых коэффициентов второго полносвязного слоя fc2, b_i^2 – смещение нейрона i . Результат применения слоя fc2 – вектор $a_2 = (a_{21}, \dots, a_{2H_2})$, подаётся на вход третьему полносвязному слою fc3:

$$z_{3i} = \sum_{j=1}^{H_2} w_{ij}^3 \cdot a_{2j} + b_i^3, \quad (36)$$

где H_2 – количество нейронов слоя fc2.

Результатом применения слоя fc3 является вектор $z_3 = (z_{31}, z_{32}, z_{33})$, к которому применяется функция Softmax:

$$\hat{y} = (\hat{y}_1, \hat{y}_2, \hat{y}_3) = \left(\frac{e^{z_{31}}}{\sum_{i=1}^3 e^{z_i}}, \frac{e^{z_{32}}}{\sum_{i=1}^3 e^{z_i}}, \frac{e^{z_{33}}}{\sum_{i=1}^3 e^{z_i}} \right) \quad (37)$$

2.3 Обучение нейронной сети

В качестве функционала качества ℓ выбрана кросс-энтропия, имеющая следующий вид:

$$\ell(\hat{y}, y) = \{l_1, \dots, l_N\}^T, \quad (38)$$

$$l_n = - \sum_{c=1}^C \log(\hat{y}_{n,c}) \cdot y_{n,c}, \quad (39)$$

$$\ell(\hat{y}, y) = \frac{\sum_{n=1}^N l_n}{N}, \quad (40)$$

где C – количество классов, N – размер выборки \hat{y} – выходные сигналы нейронной сети, y – реальное значение класса. Данный функционал качества минимизируется при приближении выходных сигналов нейронной сети к реальным значениям классов. Таким образом, оптимизация сети направлена

на уменьшение значения функционала, что соответствует улучшению точности предсказания классов.

В качестве алгоритма обучения выбрана модификация стохастического градиентного спуска – метод Adam:

В рамках обратного распространения ошибки для обновления весовых коэффициентов w_i^3 выходного нейрона i необходимо вычислить градиент функционала качества l по каждому из весовых коэффициентов w_{ij}^3 :

$$\frac{\partial l}{\partial w_{ij}^3} = \frac{\partial l}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_{3i}} \cdot \frac{\partial z_{3i}}{\partial w_{ij}^3}, \quad (41)$$

$$\frac{\partial l}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i} (\log(\hat{y}_i) \cdot y_i) = -\frac{y_i}{\hat{y}_i}, \quad (42)$$

$$\begin{aligned} \frac{\partial \hat{y}_i}{\partial z_{3i}} &= \frac{\partial}{\partial z_{3i}} \left(\frac{e^{z_{3i}}}{\sum_{j=1}^3 e^{z_j}} \right) = \frac{e^{z_{3i}}}{e^{z_{31}} + e^{z_{32}} + e^{z_{33}}} - \\ &- \left(\frac{e^{z_{3i}}}{e^{z_{31}} + e^{z_{32}} + e^{z_{33}}} \right)^2 = \hat{y}_i(1 - \hat{y}_i), \end{aligned} \quad (43)$$

$$\frac{\partial z_{3i}}{\partial w_{ij}^3} = a_{2j}, \quad (44)$$

$$\frac{\partial l}{\partial w_{ij}^3} = -\frac{y_i}{\hat{y}_i} \cdot (\hat{y}_i(1 - \hat{y}_i)) \cdot a_{2j} = -y_i(1 - \hat{y}_i) \cdot a_{2j} \quad (45)$$

Аналогично вычисляется градиент функционала качества l по смещению b_i^3 :

$$\frac{\partial l}{\partial b_i^3} = -y_i(1 - \hat{y}_i) \quad (46)$$

Далее согласно алгоритму adam для w_{ij}^3 производится инициализация экспоненциального скользящего среднего градиента $m_{ij}^{3(0)} = 0$, экспоненциального скользящего среднего квадрата градиента $v_{ij}^{3(0)} = 0$ и шага алгоритма $t = 0$. Для обновления весовых коэффициентов вычисляется экспоненциальное скользящее среднее градиента и экспоненциальное

скользящее среднее квадрата градиента по формулам (47) и (48) соответственно:

$$m_{ij}^{3(t)} = \beta_1 \cdot m_{ij}^{3(t-1)} + (1 - \beta_1) \cdot \frac{\partial l}{\partial w_{ij}^{3(t)}}, \quad (47)$$

$$v_{ij}^{3(t)} = \beta_2 \cdot v_{ij}^{3(t-1)} + (1 - \beta_2) \cdot \left(\frac{\partial l}{\partial w_{ij}^{3(t)}} \right)^2 \quad (48)$$

Далее производится коррекция смещения для экспоненциальных скользящих средних градиента и квадрата градиента:

$$\hat{m}_{ij}^{3(t)} = \frac{m_{ij}^{3(t)}}{1 - \beta_1}, \quad (49)$$

$$\hat{v}_{ij}^{3(t)} = \frac{v_{ij}^{3(t)}}{1 - \beta_2} \quad (50)$$

Затем обновляются весовые коэффициенты:

$$w_{ij}^{3(t)} = w_{ij}^{3(t-1)} - \eta \cdot \frac{\hat{m}_{ij}^{3(t)}}{\sqrt{\hat{v}_{ij}^{3(t)}} + \epsilon}, \quad (51)$$

где β_1 и β_2 являются параметрами, контролирующими скорость экспоненциального сглаживания градиента и квадрата градиента соответственно, η - скорость обучения.

Далее ошибка распространяется на полносвязный слой fc2. Градиент для обновления весовых коэффициентов w_{ij}^2 нейрона i вычисляется следующим образом:

$$\frac{\partial l}{\partial w_{ij}^2} = \frac{\partial l}{\partial a_{2i}} \cdot \frac{\partial a_{2i}}{\partial z_{2i}} \cdot \frac{\partial z_{2i}}{\partial w_{ij}^2}, \quad (52)$$

$$\frac{\partial l}{\partial a_{2i}} = \sum_{c=1}^C \frac{\partial l}{\partial \hat{y}_c} \cdot \frac{\partial \hat{y}_c}{\partial z_{3c}} \cdot \frac{\partial z_{3c}}{\partial a_{2i}}, \quad (53)$$

$$\frac{\partial a_{2i}}{\partial z_{2i}} = \frac{\partial}{\partial z_{2i}} \left(\frac{e^{z_{2i}} - e^{-z_{2i}}}{e^{z_{2i}} + e^{-z_{2i}}} \right) = 1 - \left(\frac{e^{z_{2i}} - e^{-z_{2i}}}{e^{z_{2i}} + e^{-z_{2i}}} \right)^2 = 1 - (a_{2i})^2, \quad (54)$$

$$\frac{\partial z_{2i}}{\partial w_{ij}^2} = a_{1j}, \quad (55)$$

$$\frac{\partial l}{\partial w_{ij}^2} = \sum_{c=1}^C (-y_c(1 - \hat{y}_c) \cdot w_{ci}^3) \cdot (1 - (a_{2j})^2) \cdot a_{1j} \quad (56)$$

Аналогично ошибка распространяется на полносвязный слой fc1:

$$\frac{\partial l}{\partial w_{ij}^1} = \frac{\partial l}{\partial a_{1i}} \cdot \frac{\partial a_{1i}}{\partial z_{1i}} \cdot \frac{\partial z_{1i}}{\partial w_{ij}^1}, \quad (57)$$

$$\frac{\partial l}{\partial w_{ij}^1} = \sum_{k=1}^{H_2} \left(\sum_{c=1}^C (-y_c \cdot (1 - \hat{y}_c) \cdot w_{ci}^3) \cdot (1 - (a_{2k})^2) \cdot w_{ki}^2 \right) \cdot (1 - (a_{1j})^2) \cdot x_j \quad (58)$$

Градиент функционала качества l по каждому из весовых коэффициентов $W_2^g[m, n]$ второго сверточного слоя вычисляется следующим образом:

$$\frac{\partial l}{\partial W_2^g[m, n]} = \sum_{i=1}^{N_2} \sum_{j=1}^{N_2} \frac{\partial l}{\partial Z_2^g[i, j]} \cdot \frac{\partial Z_2^g[i, j]}{\partial W_2^g[m, n]}, \quad (59)$$

$$\frac{\partial Z_2^g[i, j]}{\partial W_2^g[m, n]} = A_1^f[i + m, j + n], \quad (60)$$

$$\frac{\partial l}{\partial Z_2^g[i, j]} = \sum_{k=1}^D \frac{\partial l}{\partial x_k} \cdot \frac{\partial x_k}{\partial Z_2^g[i, j]} \quad (61)$$

Аналогично вычисляется градиент для обновления весовых коэффициентов $W_1^f[m, n]$.

2.4 Применение трансферного обучения

Целевая нейронная сеть строится на основе исходной, в несколько этапов:

1. Фиксация весовых коэффициентов исходной модели. При обучении фиксированные веса не обновляются.
2. Удаление выходного полносвязного слоя.
3. Добавление нового скрытого и нового выходного слоя.

На рисунке 17 приведена схема построения целевой нейронной сети.

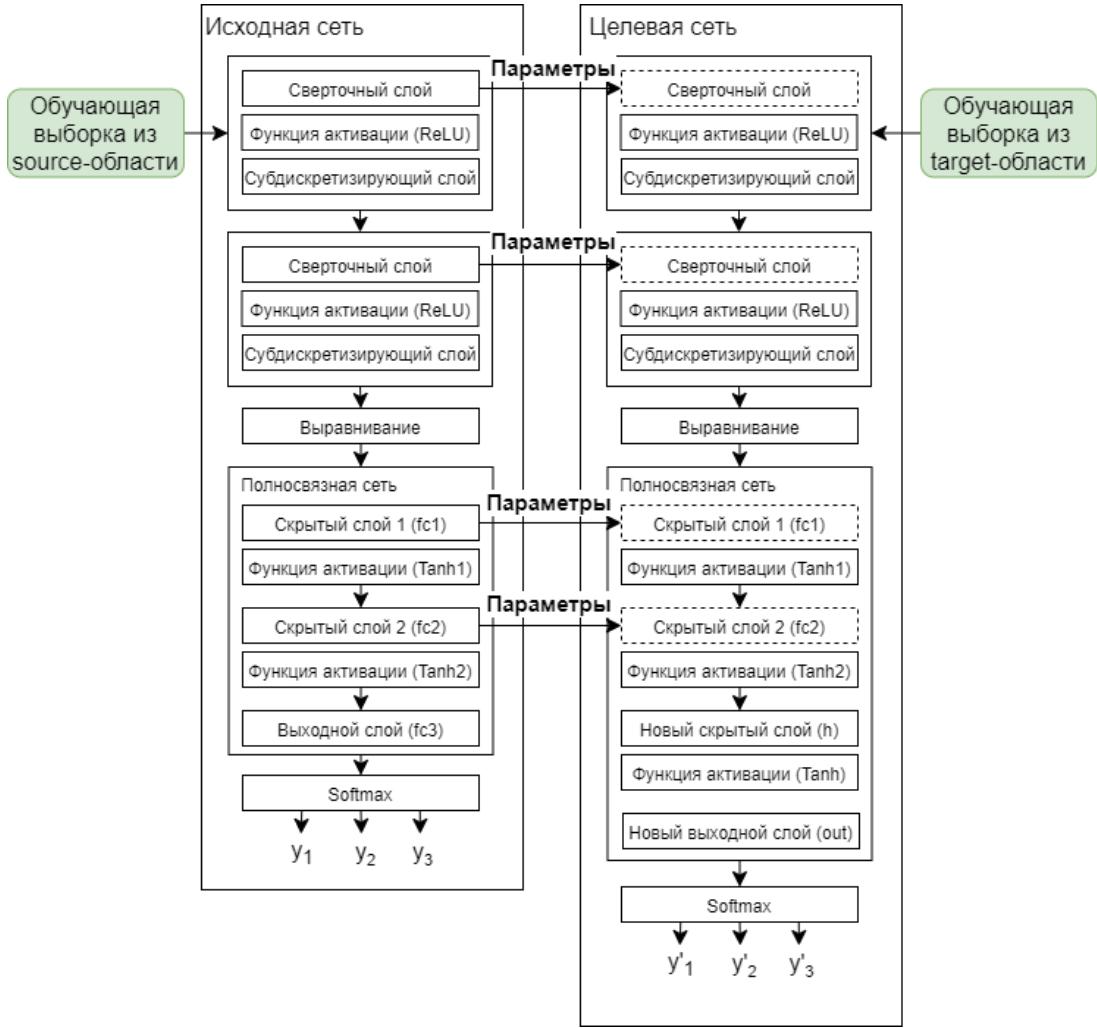


Рисунок 17 - схема построения целевой нейронной сети

Пусть (X', y') – прецедент из новой обучающей выборки. Прямое распространение в целевой нейронной сети выполняется аналогично исходной, тогда как обучение происходит только в рамках обновления весовых коэффициентов $w'^3_i = (w'^3_{i1}, \dots, w'^3_{iH_2})$ и смещений b'^3_i нового скрытого слоя h , а также $w'^4_i = (w'^4_{i1}, \dots, w'^4_{iH_3})$ и b'^4_i нового выходного слоя out :

$$\frac{\partial l}{\partial w'^4_{ij}} = -\frac{y'_i}{\hat{y}'_i} \cdot (\hat{y}'_i(1 - \hat{y}'_i)) \cdot a'^3_j = -y'_i(1 - \hat{y}'_i) \cdot a'^3_j, \quad (62)$$

$$\frac{\partial l}{\partial b'^4_i} = -\frac{y'_i}{\hat{y}'_i} \cdot (\hat{y}'_i(1 - \hat{y}'_i)) = -y'_i(1 - \hat{y}'_i), \quad (63)$$

$$\frac{\partial l}{\partial w'^3_{ij}} = \sum_{c=1}^C (-y'_c (1 - \hat{y}'_c) \cdot w'^4_{ci}) \cdot (1 - (a''_{3j})^2) \cdot a_{2j}, \quad (64)$$

$$\frac{\partial l}{\partial b'^3_i} = \sum_{c=1}^C (-y'_c (1 - \hat{y}'_c) \cdot w'^4_{ci}) \cdot (1 - (a''_{3j})^2), \quad (65)$$

где вектор $a'_3 = (a'_{31}, \dots, a'_{3H_3})$ – результат применения функции активации Tanh к нейронам слоя h, где H_3 – количество нейронов, \hat{y}' – ответ целевой нейронной сети.

2.5 Модель системы формирования торговых сигналов

Система формирования торговых сигналов – модуль ответственный за принятие решения о заключении сделки купли-продажи. Пусть в момент времени t имеется класс y_t элемента временного ряда p_t , где y_t имеет вид (y_{t1}, y_{t2}, y_{t3}) – сигналы выходных нейронов сети.

y_t интерпретируется следующим образом:

- если $y_{t1} > y_{t2}$ и $y_{t1} > y_{t3}$, то p_t относится к нейтральному классу, в этом случае положим $y_t = 0$;
- если $y_{t2} > y_{t1}$ и $y_{t2} > y_{t3}$, то p_t является точкой разворота тренда, в которой происходит изменение направления с нисходящего на восходящее, в этом случае $y_t = 1$;
- если $y_{t3} > y_{t1}$ и $y_{t3} > y_{t2}$, то p_t является точкой разворота тренда, в которой происходит изменение направления с восходящего на нисходящее, в этом случае $y_t = 2$.

Введем следующие переменные: buy_size – объем сделки, $base_currency$ – начальный капитал, выраженный в базовой валюте, $quote_currency$ – начальный капитал, выраженный в целевой валюте.

На основании класса торговая система принимает решение по открытию позиции в размере buy_size базовой валюты.

Таким образом если:

- класс элемента $y_t = 0$, то позиция не открывается;
- класс элемента $y_t = 1$, то торговая система совершают покупку, т. е. открывает короткую позицию;
- класс элемента $y_t = 2$, то торговая система совершает продажу, т. е. открывает длинную позицию.

После совершения сделок имеется последовательность открытых позиций $\{(c_t, t)\}$, где c_t – цена валюты при открытии соответствующей позиции, t – момент времени открытия позиции.

Для описания принятия решения о закрытии позиции введём следующие константы: *take_profit*, *stop_loss*.

Каждый элемент c_t сравнивается с элементом p_i в настоящий момент времени. В зависимости от характера позиции принимаются следующие решения:

- если позиция длинная и $p_i - c_i > take_profit$, то позиция закрывается, $base_currency \leftarrow base_currency - buy_size$, $quote_currency \leftarrow quote_currency + buy_size \cdot p_i$;
- если позиция короткая $c_i - p_i > take_profit$, то позиция закрывается, $base_currency \leftarrow base_currency + buy_size$, $quote_currency \leftarrow quote_currency - buy_size \cdot p_i$;
- если позиция длинная и $c_i - p_i > stop_loss$, то позиция закрывается, $base_currency \leftarrow base_currency - buy_size$, $quote_currency \leftarrow quote_currency + buy_size \cdot p_i$;
- если позиция короткая и $p_i - c_i > stop_loss$, то позиция закрывается, $base_currency \leftarrow base_currency + buy_size$, $quote_currency \leftarrow quote_currency - buy_size \cdot p_i$.

3 Разработка проекта системы

3.1 Контекстная диаграмма

Контекстная диаграмма – это графическое представление программной системы, которое используется для обозначения внешнего контекста, исходящих и входящих потоков информации, а также для идентификации основных объектов, которые могут влиять на систему или с ней взаимодействовать. Контекстная диаграмма является высокоуровневой диаграммой, которая охватывает только общий контекст системы и не углубляется в ее внутреннюю структуру или детали функций. На рисунке 18 представлена контекстная диаграмма разрабатываемой программной системы для трейдинга на основе нейронных сетей.



Рисунок 18 - контекстная диаграмма программной системы для трейдинга на основе нейронных сетей

3.2 Архитектурно-контекстная диаграмма

Архитектурно-контекстная диаграмма – это диаграмма, которая определяет основные подсистемы, обеспечивающие все функции разрабатываемой программной системы, а также данные и интерфейсы управления между ними на высоком уровне абстракции. На рисунке 19 представлена архитектурно-контекстная диаграмма разрабатываемой программной системы для трейдинга на основе нейронных сетей.

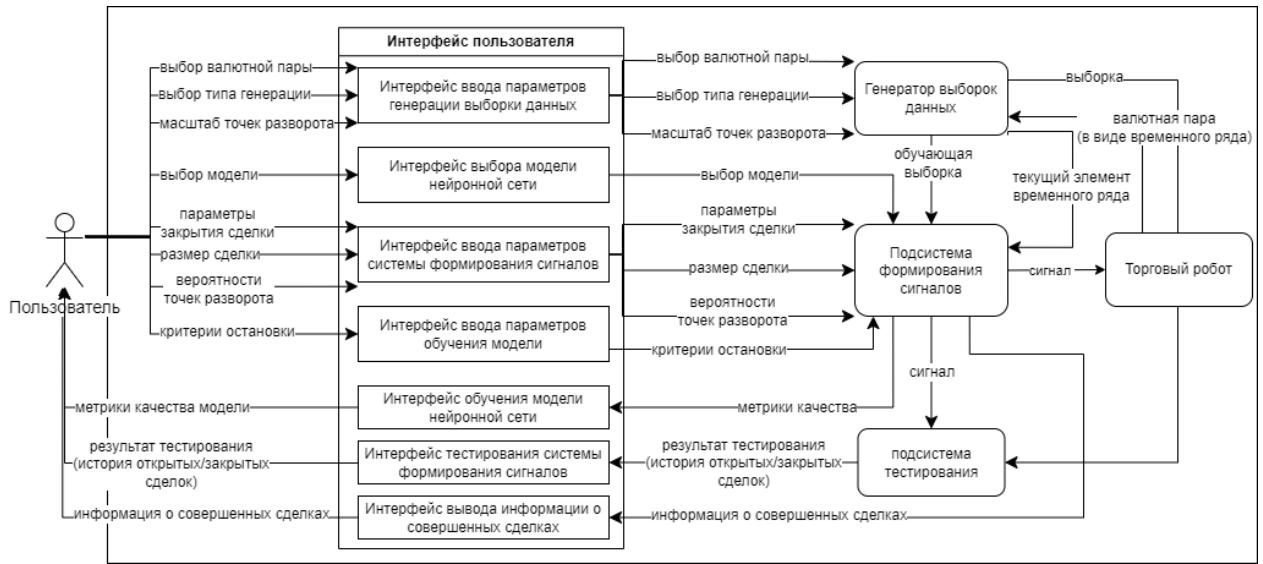


Рисунок 19 - архитектурно-контекстная диаграмма программной системы для трейдинга на основе нейронных сетей

Программная система для трейдинга на основе нейронных сетей включает следующие компоненты:

- генератор выборок данных;
- база моделей;
- подсистема тестирования;
- подсистема формирования сигналов;
- интерфейс пользователя.

Данная выпускная квалификационная работа заключается в разработке подсистем формирования торговых сигналов и тестирования. На рисунке 20 представлена архитектурно-контекстная диаграмма подсистемы формирования сигналов.

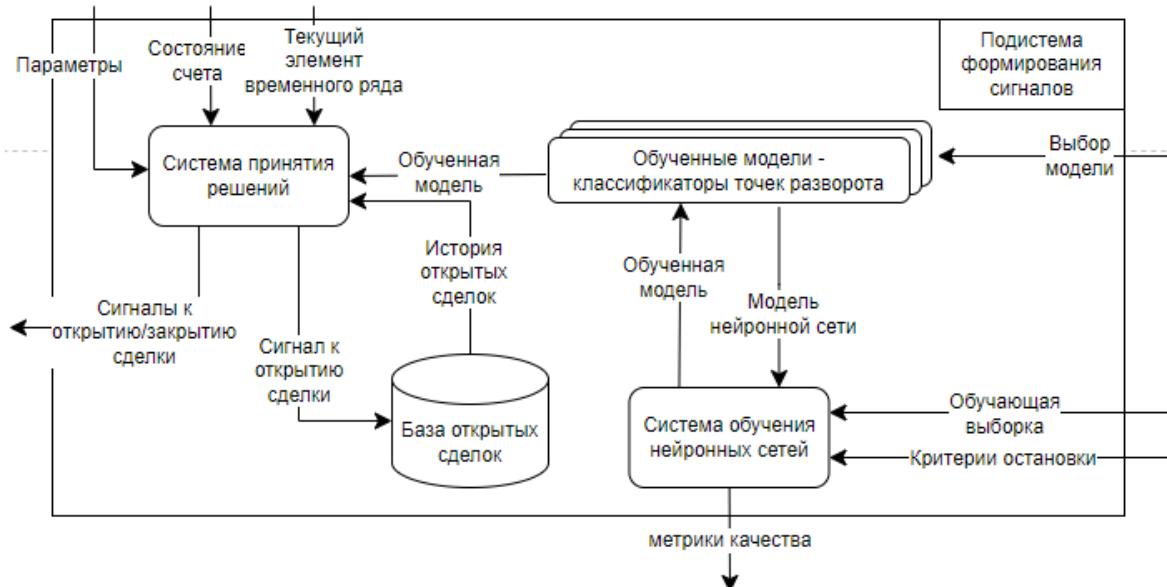


Рисунок 20 - архитектурно-контекстная диаграмма подсистемы формирования торговых сигналов

3.3 Диаграмма потоков данных

Диаграмма потоков данных — это графическое представление процесса или системы, в котором представлены различные компоненты и потоки данных между ними. DFD используется для моделирования и анализа информационных систем, чтобы понять, как данные перемещаются через систему, какие процессы и функции выполняются над этими данными, и какие внешние сущности взаимодействуют с системой. Диаграммы потоков данных могут представлять системы на самых разных уровнях абстракции: диаграммы высокого уровня предоставляют целостный вид компонентов данных и обработки в многоэтапном процессе, дополняющем точное, детальное представление, приведенное в функциональных требованиях.

На рисунке 21 представлен уровень 0 диаграммы потоков данных программной системы для трейдинга на основе нейронных сетей.

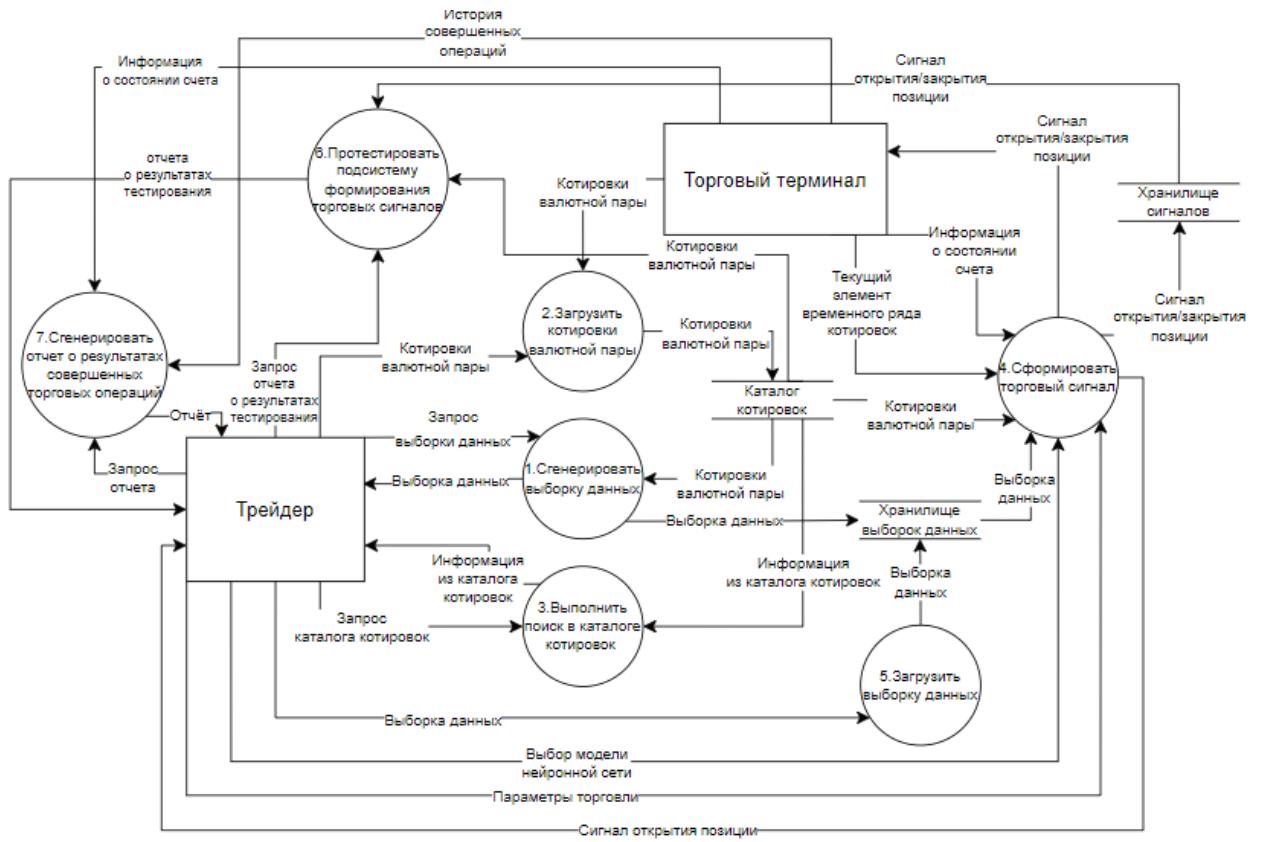


Рисунок 21 - уровень 0 диаграммы потоков данных программной системы для трейдинга на основе нейронных сетей

На рисунке 22 представлен уровень 1 диаграммы потоков данных подсистемы формирования торговых сигналов программной системы для трейдинга на основе нейронных сетей.

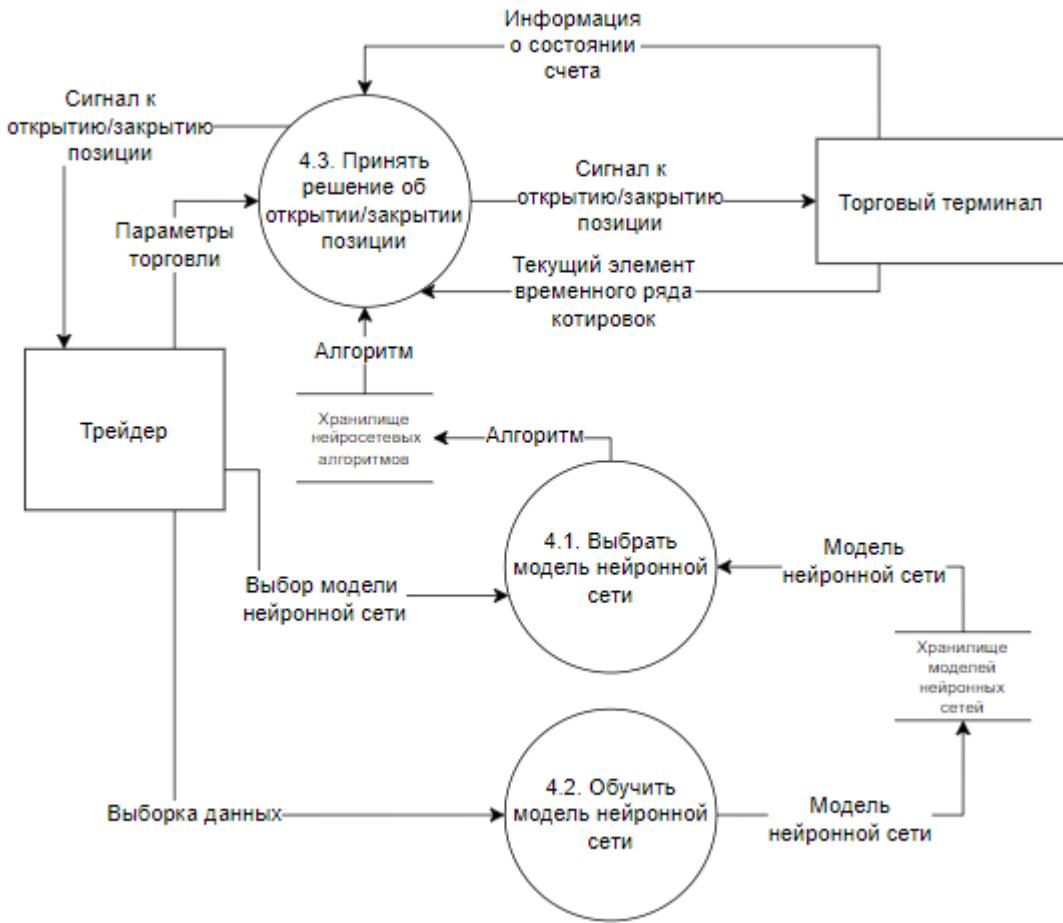


Рисунок 22 - уровень 1 диаграммы потоков данных подсистемы формирования торговых сигналов программной системы для трейдинга на основе нейронных сетей

3.4 Диаграмма use-case

Use-case диаграмма позволяет моделировать функциональные требования системы с точки зрения ее взаимодействия с действующим лицом(пользователями, другими системами или внешними компонентами).

На рисунке 23 представлена use-case диаграмма программной системы для трейдинга на основе нейронных сетей.

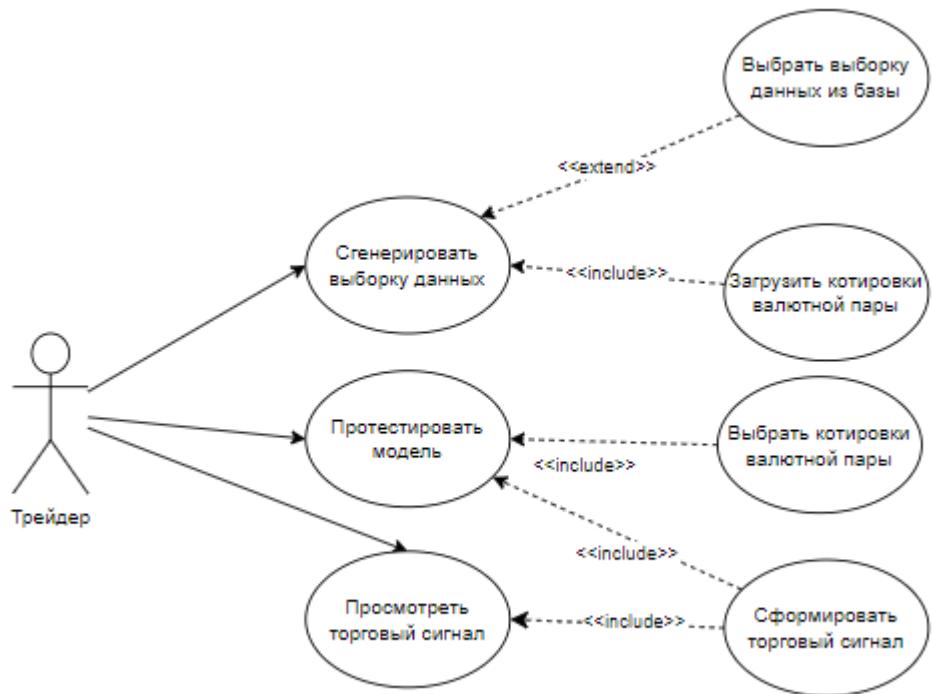


Рисунок 23 - use-case диаграмма программной системы для трейдинга на основе нейронных сетей

На данной use case отображены пользовательские требования первого уровня, такие как:

- сгенерировать выборку данных;
- протестировать модель;
- просмотреть торговый сигнал.

На рисунке 24 представлена use-case диаграмма подсистемы формирования торговых сигналов

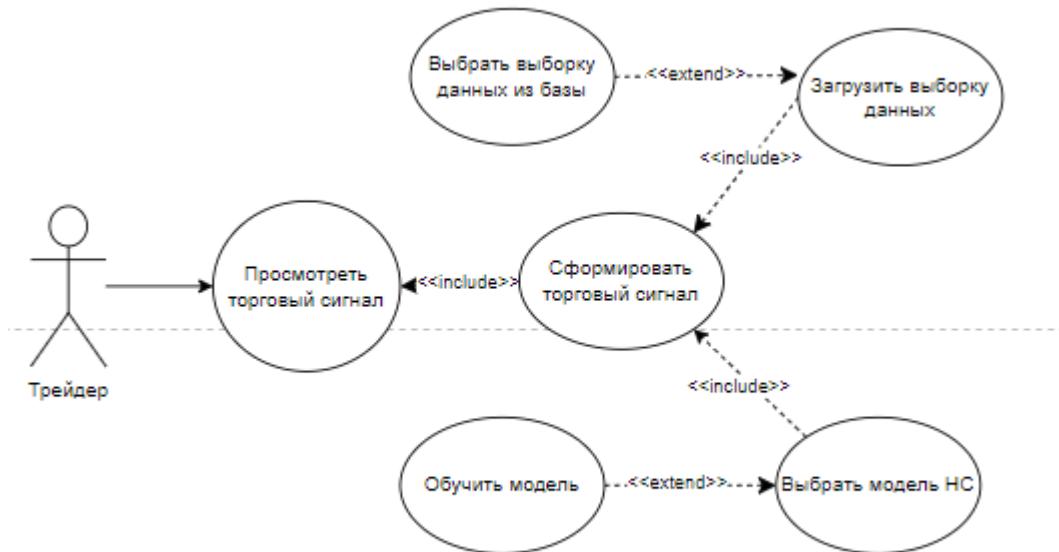


Рисунок 24 - use-case диаграмма подсистемы формирования торговых сигналов

Данная use-case диаграмма включает пользовательские требования третьего уровня, связанные с подсистемой формирования торговых сигналов, и поясняющие требование второго уровня «сформировать торговый сигнал» такие как:

- выбрать модель нейронной сети;
- обучить модель;
- загрузить выборку данных;
- выбрать выборку данных из базы.

3.5 Функциональные требования

Ниже представлена таблица 2, содержащая идентификатор требования, название функций и их описание, для подсистем формирования торговых сигналов и тестирования.

Таблица 2 - Описание функциональных требований

Идентификатор требования	Функция	Описание функциональных требований
REQ_MO_01	Выбор модели.	При нажатии на кнопку «выбрать модель» в окне «информация о модели», запускается торговый робот, формирующий торговые сигналы на основе выбранной модели.
REQ_MO_02	Загрузка моделей.	Система должна иметь возможность загружать модели из базы данных и отображать их на графическом интерфейсе.
REQ_MO_03	Отображение информации о моделях.	Система должна отображать окно с информацией о каждой загруженной модели при нажатии на соответствующую область графического интерфейса главного окна, такую как название модели, точность, значение кросс-энтропийной ошибки, значения f1-метрик для каждого класса.
REQ_MO_04	Удаление модели.	В системе должна быть реализована возможность удаления загруженных моделей. При нажатии на кнопку «удалить модель» в окне «информация о модели» соответствующая модель удаляется из базы данных.
REQ_MO_05	Ввод критериев остановки обучения.	В системе должна быть реализована возможность вводить критерии остановки обучения модели, такие как F1-мера для каждого класса, значение кросс-энтропийной ошибки и общая точность. Соответствующее окно с вводом критериев открывается при нажатии на кнопку «обучить новую модель» в главном окне.
REQ_MO_06	Обучение новой модели.	Система должна иметь функцию для обучения новой модели. Обучение модели запускается при нажатии на кнопку «сохранить» в окне с вводом критериев остановки.
REQ_MO_07	Выбор валютной пары.	В системе должна быть реализована возможность выбора валютной пары для обучения модели
REQ_MO_08	Выбор типа преобразования к двумерным данным.	В системе должна быть реализована возможность выбора типа преобразования к двумерным данным, для данных на которых обучается модель.
REQ_MO_09	Ввод дельты для формирования обучающей выборки.	В системе должна быть реализована возможность ввода дельты, задающей масштаб точек разворота в обучающей выборке.

Продолжение таблицы 2

Идентификатор требования	Функция	Описание функциональных требований
REQ_MO_10	Отображение информации об обучении.	При обучении модели должны отображаться метрики качества, а также график зависимости кросс-энтропийной ошибки от итерации обучения.
REQ_MO_11	Остановка обучения.	В системе должна быть реализована возможность остановки обучения, при нажатии на кнопку «остановить обучение» в окне «обучение модели». После остановки обучения в главном окне выводится информация о новой модели.
REQ_MO_12	Дообучение модели.	В системе должна быть реализована возможность продолжить обучение загруженной модели. При нажатии на кнопку «дообучить модель» в окне «информация о модели» открывается окно, в котором отображаются метрики качества, а также график зависимости кросс-энтропийной ошибки от итерации обучения.
REQ_MT_01	Тестирование подсистемы формирования торговых сигналов.	В системе должна быть реализована возможность тестирования на исторических данных подсистемы формирования торговых сигналов, основанной на выбранной модели при нажатии на кнопку «протестировать модель» в окне «информация о модели».
REQ_MT_02	Выбор параметров тестирования.	В системе должна быть реализована возможность выбора параметров тестирования, таких как линии stop_loss и take_profit, минимальные вероятности классов при которых, совершаются соответствующие сделки, размер покупки, выраженный в единицах целевой валюты, начальный счет.
REQ_MT_03	Отображение графика котировок.	Система должна отображать график котировок, на которых будет производиться тестирование.
REQ_MT_04	Отображение графика изменения счета.	Система должна отображать график изменения счета во время тестирования.
REQ_MT_05	Отображение истории совершенных сделок.	Система должна отображать историю сделок, совершенных во время тестирования

Окончание таблицы 2

Идентификатор требования	Функция	Описание функциональных требований
REQ_MT_06	Остановка тестирования.	В системе должна быть реализована возможность остановки тестирования при нажатии на кнопку «остановить тестирование» в окне «тестирование модели».
REQ_MT_07	Сохранение истории совершенных сделок при тестировании.	В системе должна быть реализована возможность сохранения истории совершенных сделок в формате csv при нажатии на кнопку «сохранить в формате csv» в окне «тестирование модели».

4 Реализация и тестирование программной системы

4.1 Тестирование системы

На основе функциональных требований были разработаны тестовые ситуации и проведено тестирования подсистем формирования торговых сигналов и тестирования.

Далее представлена таблица 3, содержащая идентификатор тестовой ситуации, описание тестовой ситуации, покрываемые функциональные требования, ожидаемый результат, результат.

Таблица 3 - Описание тестовых ситуаций и результатов тестирования

Идентификатор тестовой ситуации.	Описание тестовой ситуации.	Покрываемые функциональные требования.	Ожидаемый результат.	Результат.
TEST_MO_01	Получение развернутой информации о модели.	REQ_MO_02 REQ_MO_03	Открывается окно «информация о модели» в котором отображена информация о модели, а именно: имя модели, общая точность, значение кросс-энтропийной ошибки, f1-метрики.	Тест пройден
TEST_MO_02	Некорректный ввод в поле ввода дельты	REQ_MO_09	Выведена ошибка «Введено некорректное значение».	Тест пройден
TEST_MO_03	Нажатие на кнопку «обучить новую модель»	REQ_MO_05	Открывается окно с полями ввода критериев остановки: f1-меры, кросс-энтропийная ошибка, общая точность.	Тест пройден

Продолжение таблицы 3

Идентификатор тестовой ситуации.	Описание тестовой ситуации.	Покрываемые функциональные требования.	Ожидаемый результат.	Результат.
TEST_MO_04	Некорректный ввод в поля ввода критериев остановки	REQ_MO_05	Выведена ошибка «Введено некорректное значение».	Тест пройден
TEST_MO_05	Введено пустое значение в поля ввода критериев остановки	REQ_MO_05	Открывается окно «обучение модели». Обучение продолжается пока не будет нажата кнопка «остановить обучение».	Тест пройден
TEST_MO_06	Нажатие на кнопку «остановить обучение» в окне «обучение модели»	REQ_MO_11	Закрывается окно «обучение модели», появляется соответствующая информация о новой модели в главном окне.	Тест пройден
TEST_MO_07	Нажатие на кнопку «удалить модель»	REQ_MO_04 REQ_MO_02	Закрывается окно «информация о модели». Соответствующая модель удаляется с главного окна.	Тест пройден
TEST_MO_08	Нажатие на кнопку «выбрать модель»	REQ_MO_01	Запускается торговый робот, открывается окно с выводом информации о совершаемых сделках.	Тест пройден
TEST_MO_09	Нажатие на кнопку «дообучить модель»	REQ_MO_12	Открывается окно «дообучение модели».	Тест пройден

Продолжение таблицы 3

Идентификатор тестовой ситуации.	Описание тестовой ситуации.	Покрываемые функциональные требования.	Ожидаемый результат.	Результат.
TEST_MO_10	Нажатие на кнопку «остановить обучение» в окне «дообучение модели»	REQ_MO_11 REQ_MO_02	Закрывается окно «дообучение модели», соответствующая информация о модели обновляется в главном окне.	Тест пройден
TEST_MT_01	Нажатие на кнопку «протестировать модель»	REQ_MT_01 REQ_MT_03	Открывается окно «параметры тестирования».	Тест пройден
TEST_MT_02	Ввод некорректных значений в поля ввода параметров тестирования	REQ_MT_02	Выведена ошибка «Введено некорректное значение».	Тест пройден
TEST_MT_03	Нажатие на кнопку «начать тест»	REQ_MT_04 REQ_MT_05	Открывается окно «тестирование модели», выводится информация о совершаемых сделках на исторических данных	Тест пройден
TEST_MT_04	Нажатие на кнопку «остановить тестирование»	REQ_MT_06	Тестирование модели останавливается	Тест пройден

Окончание таблицы 3

Идентификатор тестовой ситуации.	Описание тестовой ситуации.	Покрываемые функциональные требования.	Ожидаемый результат.	Результат.
TEST_MT_05	Нажатие на кнопку «сохранить в формате csv»	REQ_MT_07	Открывается диалоговое окно с выбором директории для сохранения файла, история совершенных сделок сохраняется в формате csv	Тест пройден

4.2 Исследование результатов применения трансферного обучения

В данном разделе проведен сравнительный анализ моделей сверточных нейронных сетей, обученных с применением трансферного обучения и моделей, обученных без применения трансферного обучения. Для сравнения представлены модели, обученные на основе валютных пар доллар/рубль (USD/RUB), евро/рубль (EUR/RUB), юань/рубль (CNY/RUB) с использованием преобразования к двумерным данным при помощи грамианского углового поля, а также преобразования при помощи многослойного персептрона. В анализе рассматриваются показатели метрик качества, такие как: кросс-энтропийная ошибка, f1-мера для нейтральных точек, f1-мера для точек разворота вверх, f1-мера для точек разворота вниз, общая точность.

F1-мера – это метрика оценки качества моделей, используемых в задачах классификации, которая объединяет в себе показатели точности (precision) и полноты (recall):

$$f1 = \frac{2(precision * recall)}{(precision + recall)} \quad (66)$$

Показатель precision для каждого класса определяется как отношение числа верно классифицированных объектов этого класса к общему числу объектов, которые модель отнесла к данному классу:

$$\text{precision} = \frac{tp}{(tp + fp)}, \quad (67)$$

где tp – количество объектов класса, которые были верно классифицированы, fp – количество объектов, которые были неверно классифицированы как данный класс.

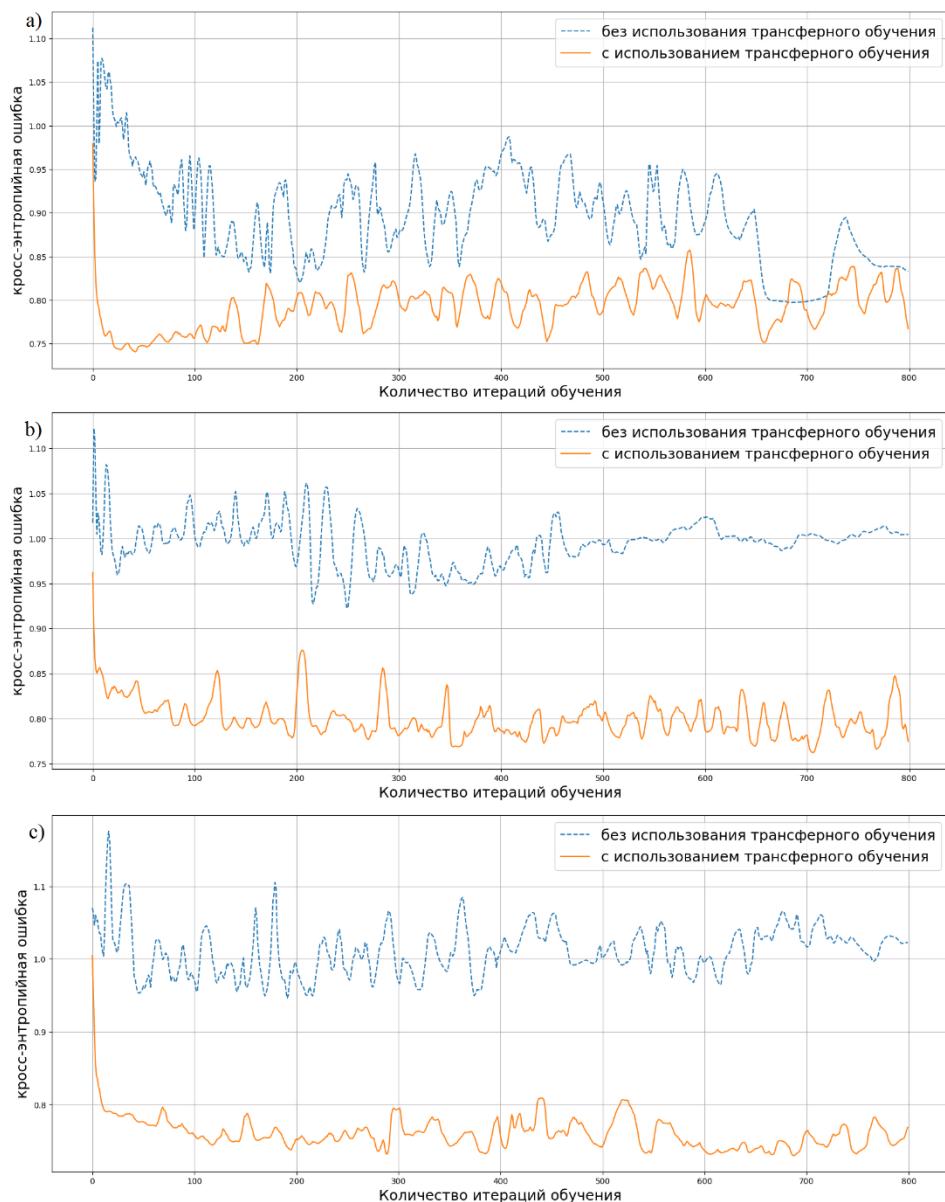
Показатель recall для каждого класса определяется как отношение числа верно классифицированных объектов этого класса к общему числу объектов данного класса:

$$\text{recall} = \frac{tp}{(tp + fn)}, \quad (68)$$

где fn – количество объектов класса, которые были неверно классифицированы как другие классы.

Ниже приведены графики зависимости, рассматриваемых метрик качества от количества итераций обучения на данных преобразованных к двумерным при помощи грамианского углового поля (GAF).

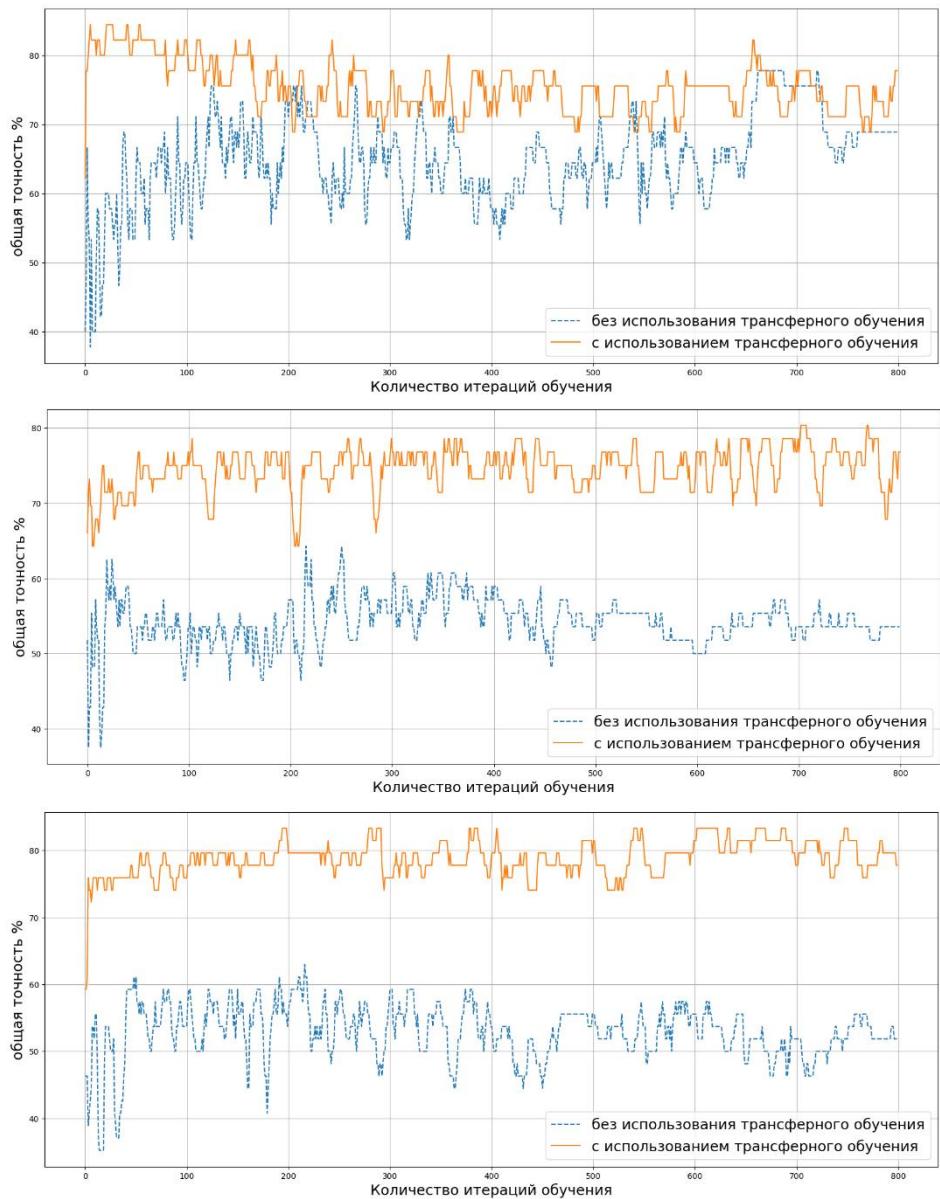
На рисунке 25 изображены графики зависимости кросс-энтропийной ошибки от количества итераций обучения.



a) – USD/RUB; b) – EUR/RUB; c) – CNY/RUB

Рисунок 25 - Графики зависимости кросс-энтропийной ошибки от количества итераций

На рисунке 26 изображены графики зависимости общей точности от количества итераций обучения на различных валютных парах.



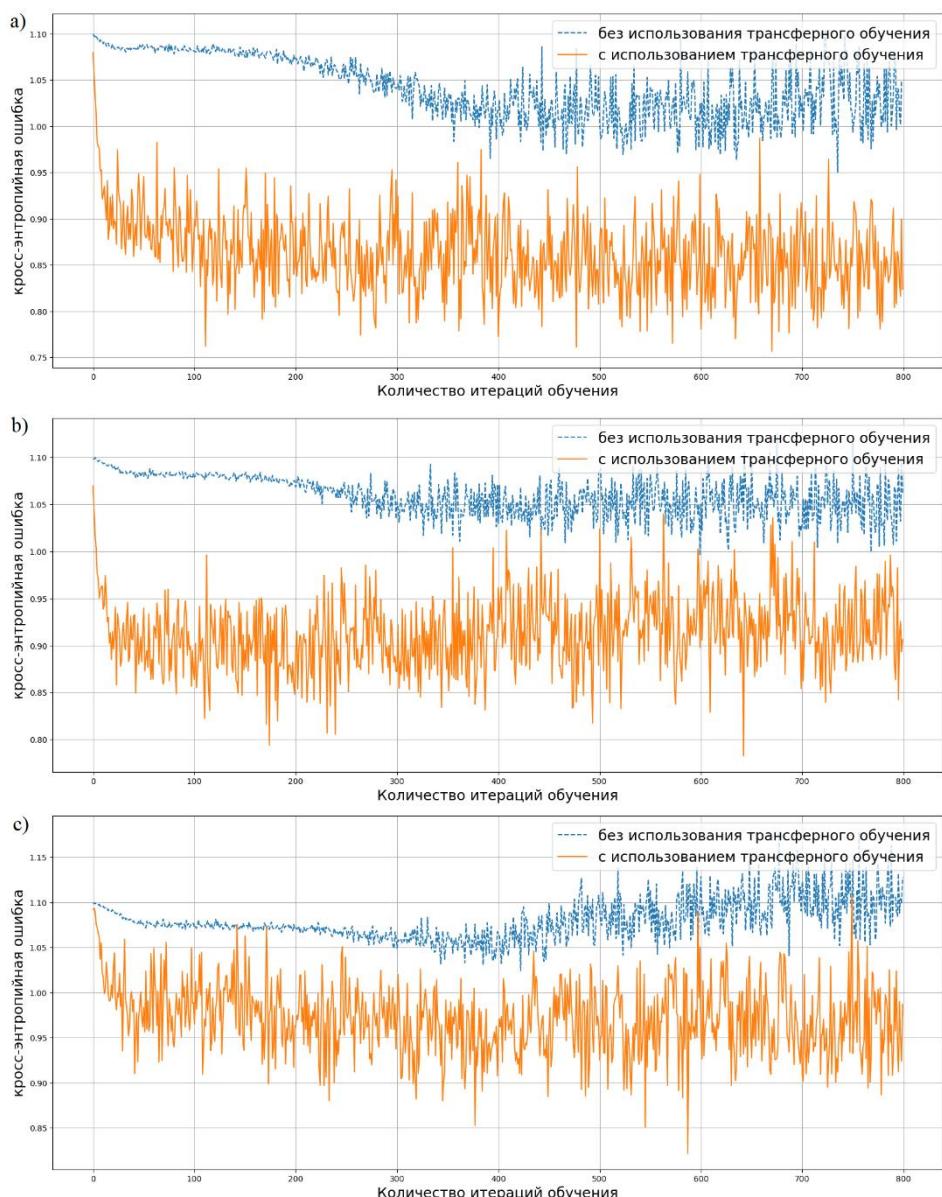
a) – USD/RUB; b) – EUR/RUB; c) – CNY/RUB

Рисунок 26 - Графики зависимости общей точности от количества итераций

В приложении А на рисунках А.1, А.2, А.3 представлены графики зависимости f1-мер от количества итераций обучения, для каждого класса точек.

Далее приведены графики зависимости, рассматриваемых метрик качества от количества итераций обучения на данных преобразованных к двумерным при помощи многослойного персептрона (MLP).

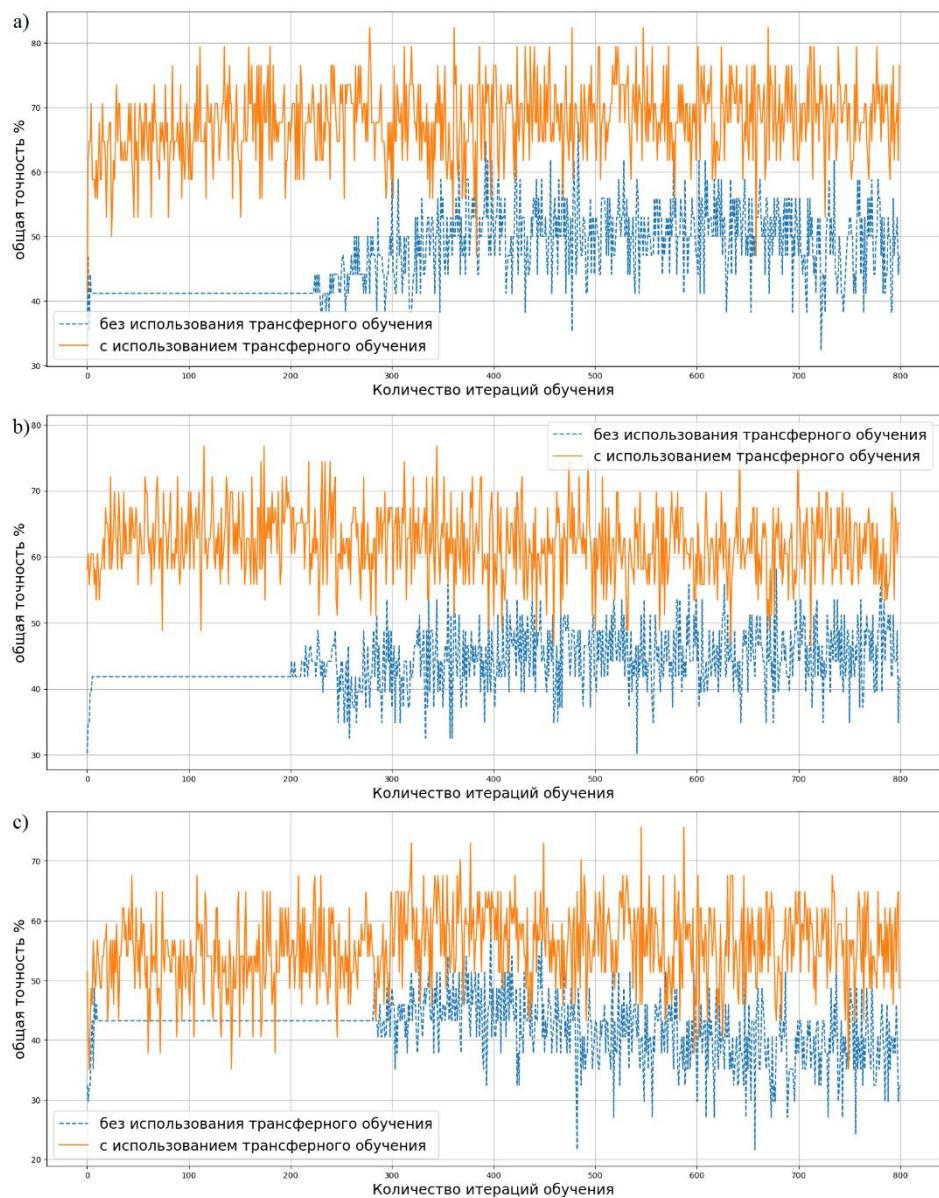
На рисунке 27 изображены графики зависимости кросс-энтропийной ошибки от количества итераций обучения на различных валютных парах.



a) – USD/RUB; b) – EUR/RUB; c) – CNY/RUB

Рисунок 27 - Графики зависимости кросс-энтропийной ошибки от количества итераций

На рисунке 28 изображены графики зависимости общей точности от количества итераций обучения на различных валютных парах.



a) – USD/RUB; b) – EUR/RUB; c) – CNY/RUB

Рисунок 28 - Графики зависимости общей точности от количества итераций

В приложении А на рисунках А.4, А.5, А.6 представлены графики зависимости f1-мер от количества итераций обучения, для каждого класса точек.

Из графиков видно, что при использовании трансферного обучения наблюдается более быстрая сходимость к оптимальным значениям метрик качества, а также более высокие показатели точности и более низкие показатели кросс-энтропийной ошибки.

Ниже приведена сравнительная таблица 4, построенная на основе рассматриваемых метрик качества.

Таблица 4 - Сравнительная таблица метрик качества обучения

Метрики	Тип преобразования к двумерными данным	Трансферное обучение			Без трансферного обучения		
		USD/ RUB	EUR/ RUB	CNY/ RUB	USD/ RUB	EUR/ RUB	CNY/ RUB
F1-мера для нейтральных точек	GAF	80.0%	79.3%	81.6%	74.9%	70.5%	71.1%
	MLP	82.3%	78.9%	80.0%	75.0%	63.4%	75.1%
F1-мера для точек разворота вверх	GAF	85.7%	76.9%	82.7%	80.0%	58.1%	62.8%
	MLP	90.0%	92.3%	84.2%	63.6%	60.8%	55.6%
F1-мера для точек разворота вниз	GAF	89.6%	87.5%	89.7%	87.9%	66.7%	66.7%
	MLP	95.2%	78.2%	80.0%	70.5%	60.7%	60.9%
Общая точность	GAF	84.4%	80.3%	83.3%	77.8%	64.2%	62.9%
	MLP	82.3%	79.0%	75.6%	67.6%	58.1%	62.1%
Кросс-энтропийная ошибка	GAF	0.74	0.76	0.73	0.79	0.92	0.95
	MLP	0.75	0.78	0.82	0.95	0.99	1.02

Таким образом при использовании трансферного обучения результаты метрик выше, чем без трансферного обучения. Для всех рассматриваемых метрик (f1-мера для нейтральных точек, точек разворота вверх и вниз, общая точность и кросс-энтропийная ошибка) GAF и MLP показывают различные уровни качества в зависимости от валютных пар. В целом, MLP достигает

более высоких результатов f1-меры для точек разворота вверх, чем GAF. Однако, для f1-меры нейтральных точек, а также точек разворота вниз GAF показывает лучшие результаты. Общая точность и кросс-энтропийная ошибка также варьируются в зависимости от метода преобразования и валютной пары.

В среднем улучшение метрик при использовании метода преобразования GAF составило:

- для f1-меры нейтральных точек – 8.1%;
- для f1-меры точек разворота вверх – 14.8%;
- для f1-меры точек разворота вниз – 15.1%;
- для общей точности – 14.4%;
- для кросс-энтропийной ошибки – 0.14.

При использовании MLP среднее улучшение метрик составило:

- для f1-меры нейтральных точек – 9.2%;
- для f1-меры точек разворота вверх – 28.8%;
- для f1-меры точек разворота вниз – 20.4%;
- для общей точности – 16.3%;
- для кросс-энтропийной ошибки – 0.2.

Заключение

Целью выпускной квалификационной работы являлась разработка подсистем формирования торговых сигналов и тестирования, программной системы для трейдинга на основе нейронных сетей.

Для достижения данной цели были выполнены следующие задачи:

- 1) Проведен обзор существующих решений, рассматриваемой предметной области.
- 2) Проведен анализ предметной области и построена её модель.
- 3) Разработан проект программной системы.
- 4) Реализована и протестирована программная система.
- 5) Исследованы результаты применения трансферного обучения.

Таким образом, цель выпускной квалификационной работы на тему “Разработка программной системы для трейдинга на основе нейронных сетей. Подсистемы формирования торговых сигналов и тестирования” была достигнута.

Список источников

1. Ананченко И. В. Классификация и общая оценка торговых роботов для валютного рынка Forex // European research. 2020. – С. 24–26.
2. Ананченко И. В., Чагина П. А. Перспектива машинного обучения нейросетей для разработки и оптимизации торговых роботов // Евразийское Научное Объединение. 2021. № 1–2. – С. 78–82.
3. Ананченко И. В., Чагина П. А. Практическое применение нейронных сетей в трейдинге: проблемы и решения // European research. 2020. – С. 27–30.
4. Букунов С. В., Климин П. Ю. Автоматизированная торговая система для работы на финансовых рынках // Инженерный вестник. 2019. № 4. – С. 32.
5. Варламов М. С., Тесля Н. Б. Информационные технологии в финансовой сфере с использованием нейронных сетей // Экономика XXI века. 2020. – С. 455–458.
6. Васяева Т. А., Мартыненко Т. В., Суббота Н. С. Прогнозирование финансовых временных рядов с помощью нейронных сетей с использованием библиотеки Keras в Python // Информатика и кибернетика. 2019. № 2. – С. 41–50.
7. Галиуллина А. Ш. Нейронные сети и технический анализ в трейдинге // Научно-практические исследования. 2018. № 2. – С. 21–24.
8. Гасанов И. И., Ерешко А. Ф. Программный комплекс для высокочастотной биржевой торговли // Управление развитием крупномасштабных систем. 2021. – С. 405–413.
9. Гончаров А. Б., Исаев А. Л. О методах технического анализа, применяемых в торговых роботах частных экспертных систем поддержки

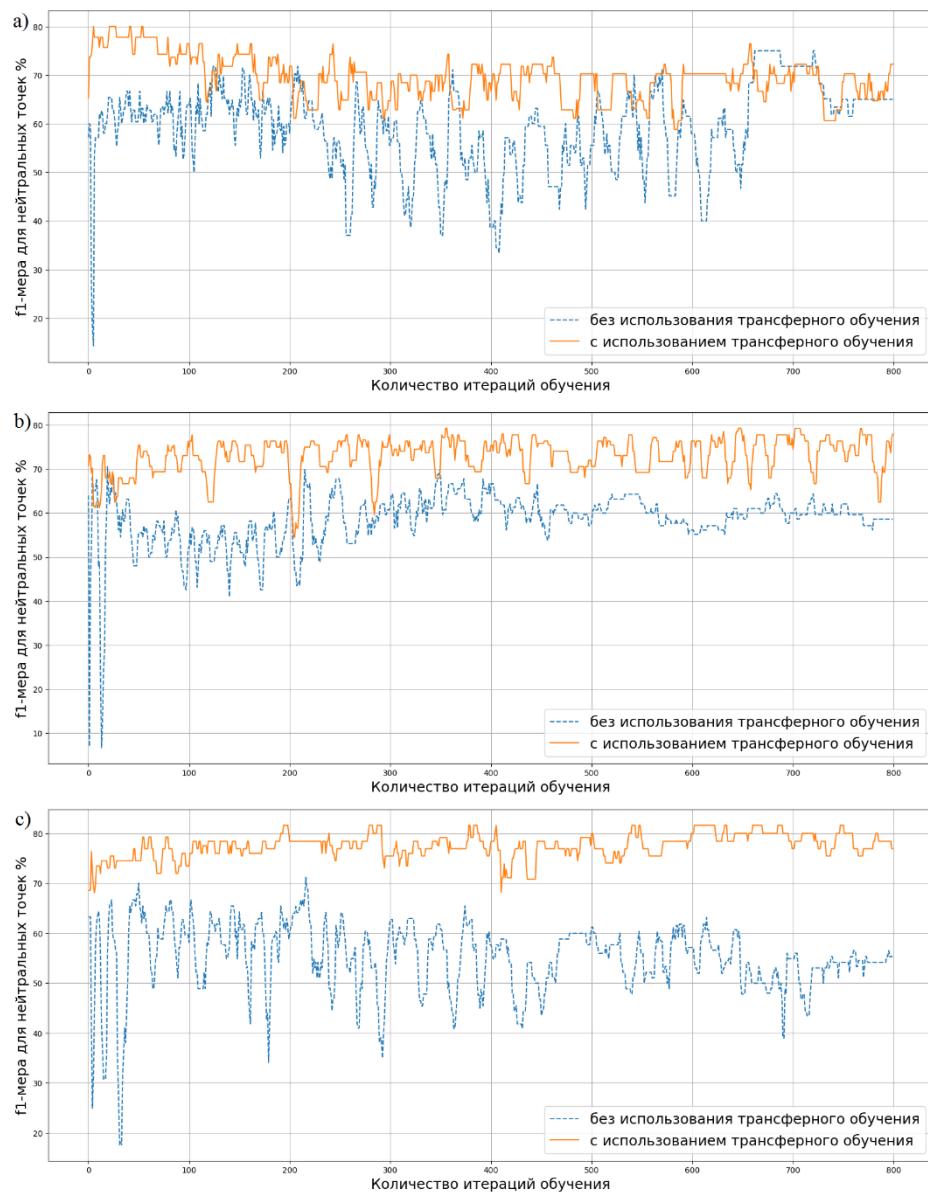
принятия решений // Тенденции развития науки и образования. 2020. № 62–7. – С. 53–55.

10. Гюлумян А. Ю., Нишатов Н. П. Искусственный интеллект: эволюция, виды нейронных сетей, использование на финансовом рынке // Финансовые рынки и банки. 2018. № 1. – С. 74–78.
11. Железко Б. А., Стадник А. О., Синявская О. А. Использование технического анализа и индикаторов в алгоритмическом трейдинге // Экономическая наука сегодня. 2022. № 15. – С. 119–130.
12. Иванов М. А. Использование торговых роботов на рынке Форекс: преимущества и недостатки // European science. 2018. № 6. – С. 23–27.
13. Иэн Гудфеллоу. Глубокое обучение : учеб. пособие / Иэн Гудфеллоу, Йошуа Бенджио. – Изд. 2-е: MIT Press. 2016. 654 с.
14. Коноплева Ю. А., Пакова О. Н., Гаврилов К. К. Валютные рынки и валютные операции в условиях цифровизации // Вестник Северо-Кавказского федерального университета. 2020. № 6. – С. 104–111.
15. Плеханов Л., Плеханов С. Нейронные сети как инструмент распознавания фигур. // Рынок ценных бумаг. №3–2005. – С. 68–72.
16. Саймон Хайкин. Нейронные сети: полный курс – 2006 г. – С. 219–241.
17. Чагина П. А. Анализ возможности применения машинного обучения и нейронных сетей для проектирования и оптимизации торговых роботов // Альманах научных работ молодых ученых Университета ИТМО. 2022. – С. 388–390.
18. Чагина П. А. Использование возможностей нейронных сетей для оптимизации алгоритмов торговых роботов // Молодежный исследовательский потенциал. 2021. – С. 15–28.
19. Шангаряев Т. М., Кусков В. М. Торговые роботы как инструмент совершения торговых операций на финансовых рынках // Colloquium-journal. 2019. № 13–11. – С. 191–193.

20. Программа «Amibroker» - Режим доступа: <https://amibroker.com/>
21. Программа "Neuroshell Trader" - Режим доступа: <https://try.neuroshell.com/features/>
22. Программа «Tickeron» - Режим доступа: <https://tickeron.com/>
23. Программа «Trade-ideas» - Режим доступа: <https://www.trade-ideas.com/download/#beta>
24. Программа «Scanz» - Режим доступа: <https://scanz.com/>
25. Weber E.A. Transfer Learning with Time Series Data: A Systematic Mapping Study. // IEEE Access, Dec. 2021. 25 c.
26. Aslan O., Yilmaz A.A. A New Malware Classification Framework Based on Deep Learning Algorithms. // IEEE Access, Jun. 2021. 17 c.
27. Jin Xiao, Yi Hu. A hybrid transfer learning model for crude oil price forecasting. // Statistics and its Interface, Jan. 2017. 13 c.
28. Ruijun Liu, Yuqian Shi. A Survey of Sentiment Analysis Based on Transfer Learning. // IEEE Access, Jun. 2019. 12 c.
29. Pan S.J., Yang Q. A survey on transfer learning. // IEEE Access, Oct. 2009. 14 c.
30. Jordan J. B. Cross-Domain MLP and CNN Transfer Learning for Biological Signal Processing: EEG and EMG. // IEEE Access, Jan. 2020. 13 c.

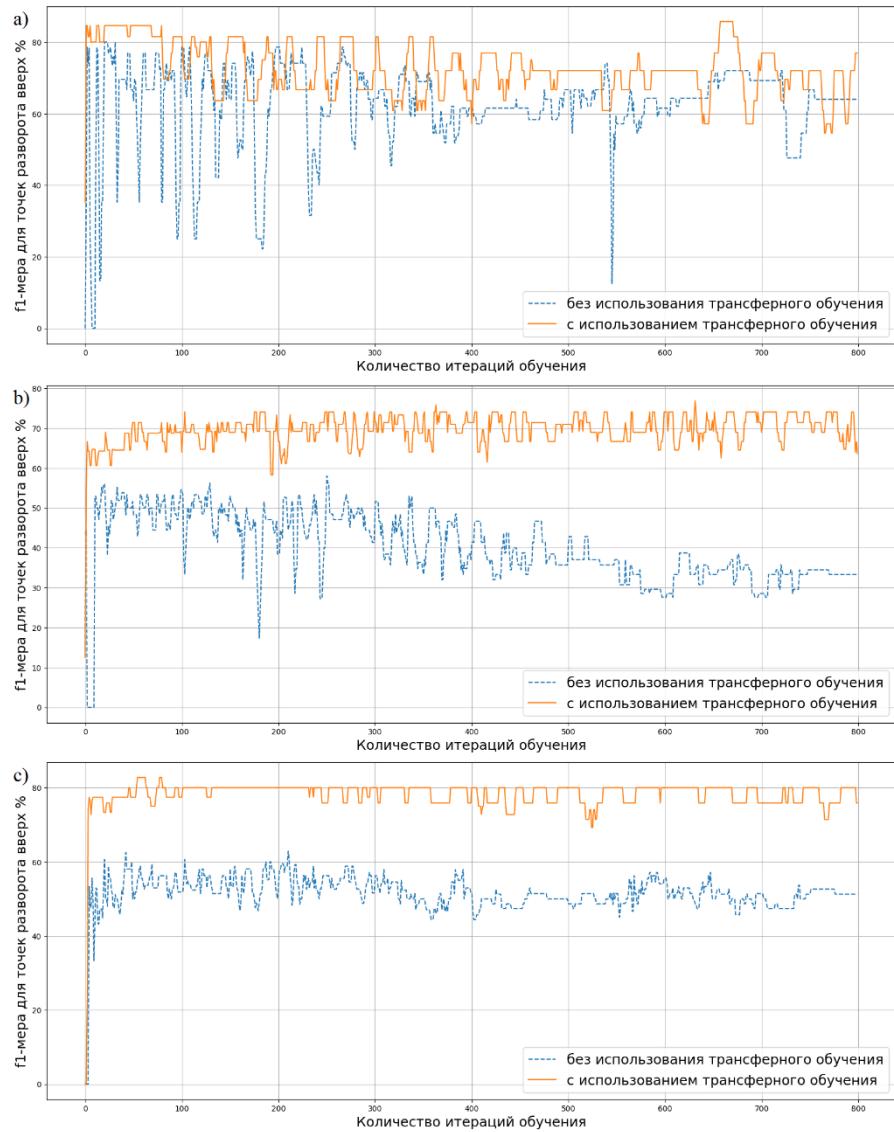
Приложение А

Графики зависимости f1-мер от количества итераций обучения модели



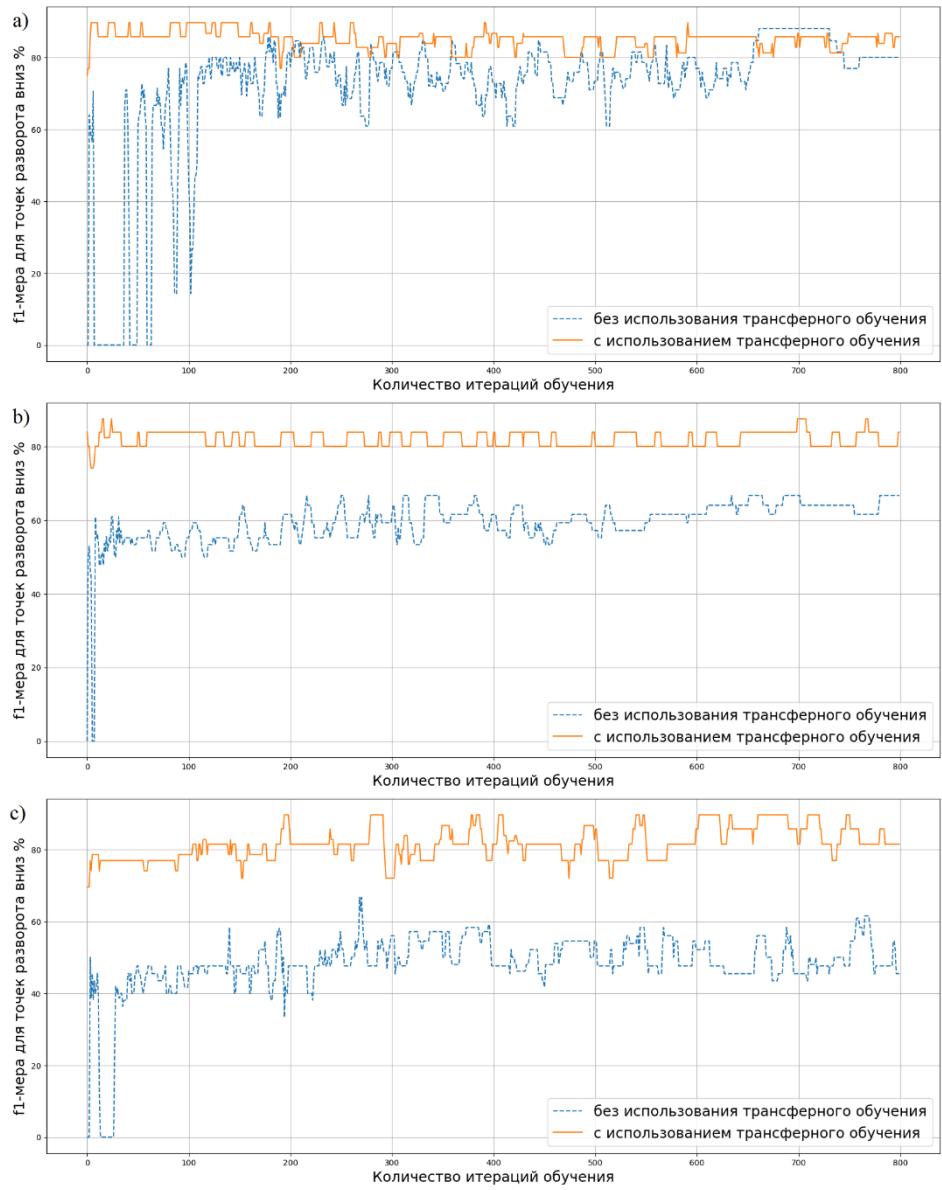
a) – USD/RUB; b) – EUR/RUB; c) – CNY/RUB

Рисунок А.1 – Графики зависимости f1-меры для нейтральных точек от количества итераций



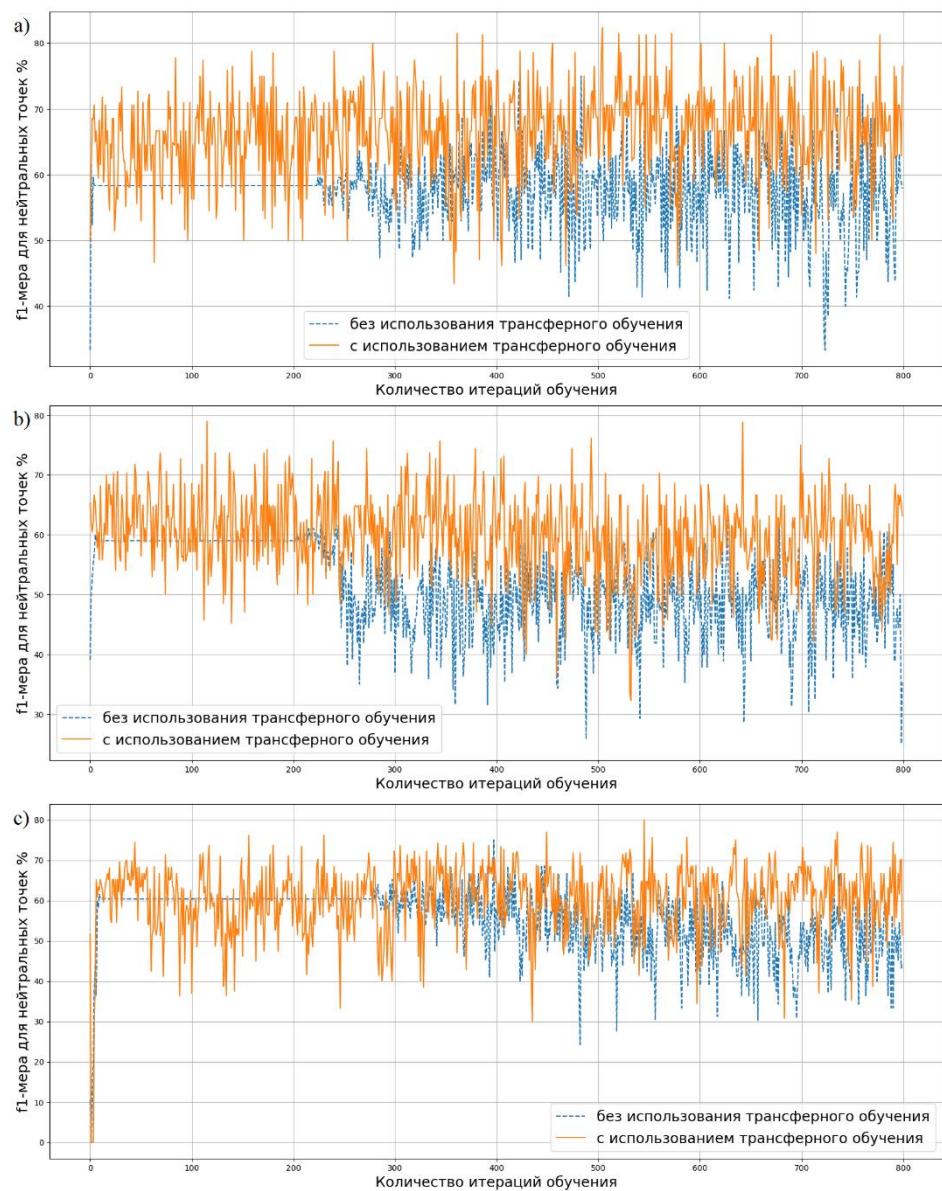
a) – USD/RUB; b) – EUR/RUB; c) – CNY/RUB

Рисунок А.2 – Графики зависимости f1-меры для точек разворота вверх от количества итераций



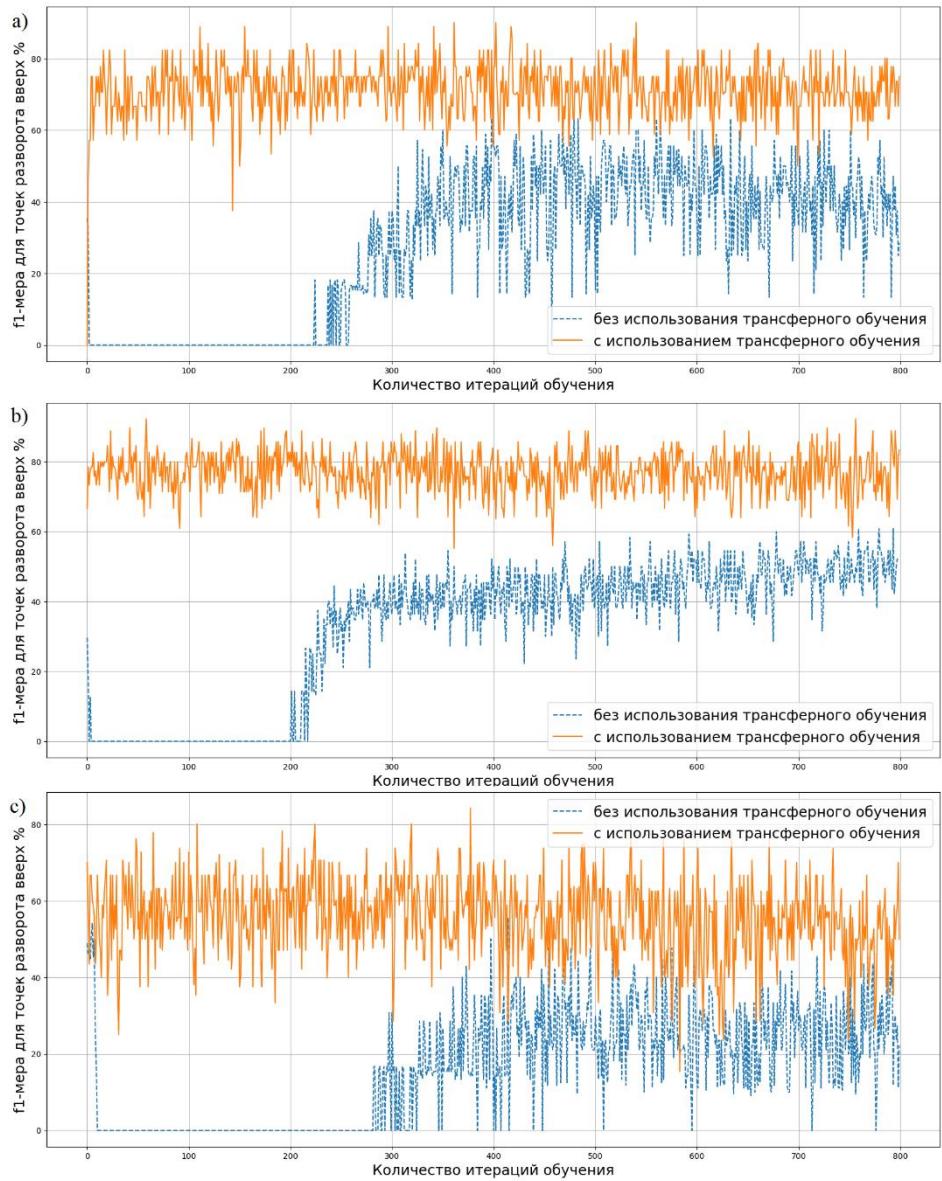
a) – USD/RUB; b) – EUR/RUB; c) – CNY/RUB

Рисунок А.3 – Графики зависимости f1-меры для точек разворота вниз от количества итераций



a) – USD/RUB; b) – EUR/RUB; c) – CNY/RUB

Рисунок А.4 – Графики зависимости f1-меры для нейтральных точек от количества итераций



a) – USD/RUB; b) – EUR/RUB; c) – CNY/RUB

Рисунок А.5 – Графики зависимости f_1 -меры для точек разворота вверх от количества итераций

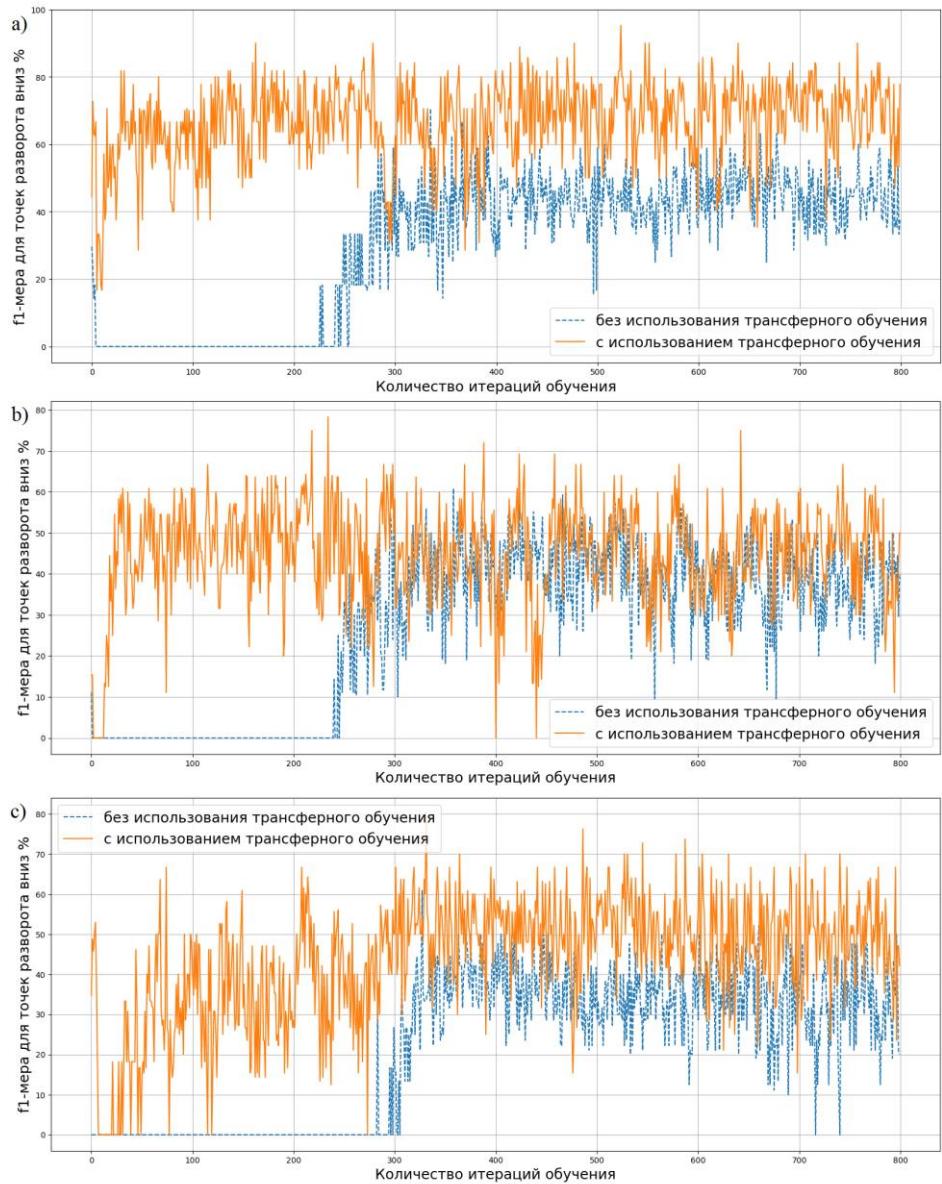


Рисунок А.6 – Графики зависимости f_1 -меры для точек разворота вниз от количества итераций