

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ПЕТРА ВЕЛИКОГО»  
Институт компьютерных наук и технологий

---

Отчет о прохождении производственной технологической (проектно-технологической) практики

на тему: «Применение нейронных сетей для автоматической торговли на бирже».

Кан Максим Евгеньевич

---

(Ф.И.О. обучающегося)

3 курс, 3530203/80101

---

(номер курса обучения и учебной группы)

02.03.03 Математическое обеспечение и администрирование информационных систем

---

(направление подготовки (код и наименование))

Место прохождения практики: ФГАОУ ВО «СПбПУ», ИКНТ, ВШИСиСТ.

---

(указывается наименование профильной организации или наименование структурного подразделения)

г. Санкт-Петербург, ул. Обручевых, д. 1, лит. В

---

ФГАОУ ВО «СПбПУ», фактический адрес)

Сроки практики: с 05.06.2021 по 03.07.2021.

Руководитель практической подготовки от ФГАОУ ВО «СПбПУ»: Сабинин Олег Юрьевич, к. т. н., доцент ВШИСиСТ.

---

Консультант практической подготовки от ФГАОУ ВО «СПбПУ»: Резединова Евгения Юрьевна, ст. преп. ВШИСиСТ.

---

Руководитель практической подготовки от профильной организации: нет.

---

Оценка:

---

Руководитель практической подготовки

от ФГАОУ ВО «СПбПУ»:

Сабинин О.Ю.

---

Консультант практической подготовки

от ФГАОУ ВО «СПбПУ»:

Резединова Е.Ю.

---

Обучающийся:

Кан М.Е.

---

Дата:

---

## СОДЕРЖАНИЕ

Введение .....	3
Глава 1. Применение технического анализа .....	4
Глава 2. Нейронные сети .....	5
2.1. Архитектура нейронных сетей .....	5
2.1.1. Многоуровневый персептрон (Multi-Layer Perceptron, MLP) .....	6
2.1.2. Сверточная нейронная сеть (Convolutional Neural Network, CNN) ....	6
2.1.3. Рекуррентная нейронная сеть (Recurrent Neural Network, RNN).....	7
2.2. Способы обучения .....	9
2.2.1. Генетические алгоритмы[8] .....	9
2.2.2. Генеративно-состязательные алгоритмы .....	9
2.2.3. Алгоритмы обучения с подкреплением.....	10
Глава 3. Программная реализация и тестирование.....	10
3.1. Загрузка данных.....	11
3.2. Окружение для обучения.....	12
3.3. Настройка и обучение нейросети .....	13
3.4. Тестирование.....	14
Заключение .....	16
Список сокращений и условных обозначений.....	17
Список использованных источников.....	18
Приложение 1. UML-схема некоторых классов.....	19
Приложение 2. Инициализация DataFeed из файла.....	20
Приложение 3. Методы класса SimpleBroker.....	21
Приложение 4. Метод step класса BrokerMdp.....	23
Приложение 5. Инициализация, обучение и сохранение весов нейронной сети.....	24

## ВВЕДЕНИЕ

Искусственные нейронные сети – популярный инструмент для решения всевозможных задач в самых разных предметных областях. Можно натренировать нейросеть выигрывать чемпионов по шахматам и го, находить предметы на фотографиях, создавать музыку и писать книги.

Одно из достоинств нейронной сети в том, что программисту, обучающему её, нужно знать относительно немного о предметной области. При правильном подборе параметров сети и при наличии достаточного количества тренировочных данных человеку не нужно принимать непосредственное участие в обучении. Со временем эффективность сети растет, во многих случаях превышая эффективность традиционных программ.

Использование автоматизированных средств торговли на бирже, основанных на техническом анализе, само по себе малоэффективно. Обычно такими инструментами пользуются в ручном режиме, то есть последнее слово остается за человеком. Искусственная нейронная сеть, которая по сути имитирует процесс принятия решений человека, может стать более мощным инструментом, чем анализ на основе алгоритмов. Так как она самообучаема, нейросеть может превзойти другие способы автоматической торговли.

Цель данной работы: оценить целесообразность использования нейронной сети для автоматической торговли на бирже, в частности – на длительном промежутке времени (от 1 до 10 лет). Нужно определить, какого результата может достигнуть нейросеть при использовании только текущих данных о ценах, без дополнительной информации.

В процессе выполнения необходимо решить следующие задачи:

- А. Изучить архитектуру, особенности, способы обучения нейросетей;
- В. Создать окружение, в котором нейросеть будет действовать;
- С. Найти оптимальные значения гиперпараметров нейросети.

## ГЛАВА 1. ПРИМЕНЕНИЕ ТЕХНИЧЕСКОГО АНАЛИЗА

Технический анализ – полезный инструмент, помогающий более обоснованно торговать ценными бумагами. В этой главе рассматривается технический анализ, его эффективность и применение, потенциальное его развитие и обобщение с помощью искусственных нейронных сетей.

Подходить к торговле акциями без предварительного анализа опасно. Для снижения рисков и максимизации прибыли используются разные виды анализа. Некоторые из них тяжело автоматизировать. Например, при **фундаментальном анализе** рассматривается компания, её доходы, новости, глобальные тренды; цель - найти недооцененную или переоцененную компанию, чтобы заработать на росте или падении цен на ее акции. Другие автоматизировать проще, поскольку они опираются на точную информацию – исторические цены и формализованные отчеты.

В данной работе будет использоваться так называемый **технический анализ**. Его суть в том, чтобы по изменениям цен в прошлом оценивать риски и предсказывать цены в будущем. Для этого используются технические индикаторы - математические функции над графиками, известные паттерны.

Главное достоинство технического анализа в том, что он опирается на строгие математические конструкции. Для обычного трейдера он предоставляет точный набор инструкций, который не позволит обанкротиться из-за неправильной оценки ситуации. Для программиста же появляется возможность полностью автоматизировать процесс торговли[9].

Недостаток в том, что для технического анализа не имеют значения субъективные признаки. Большое влияние на цену акций оказывает психология. Например, распространены проявления *эффекта толпы*[7], которые могут начинаться не только из-за колебаний цен, но из-за совокупности случайных факторов, что может привести к нестабильности.

Для того, чтобы обойти это ограничение, можно попробовать применить нейронные сети[1]. Имея набор данных о ценах большого количества акций на достаточно длинных временных отрезках, можно научить нейросеть анализировать цены и индикаторы, чтобы вовремя принимать решение о покупке или продаже. С другой стороны, хорошо обученная нейросеть сможет заметить нерациональное поведение инвесторов и воспользоваться им для увеличения прибыли. Таким образом, в случае успеха, можно пользоваться не только точностью машины, но

и "интуицией", исходящей из опыта, накопленного нейросетью после анализа огромного объема исторических данных.

Нейронная сеть способна, в какой-то мере, и на фундаментальный анализ. Для этой цели НС обучают на новостных статьях извлекать самые важные факты, которые могут влиять на стоимость компании. В этой работе рассматривается только технический анализ с помощью НС.

## **ГЛАВА 2. НЕЙРОННЫЕ СЕТИ**

Одним термином "нейронная сеть" невозможно описать разнообразие архитектур и способов обучения, созданных для разных задач. Чтобы добиться высокой эффективности, необходимо правильно подобрать все компоненты. В этой главе рассматриваются виды искусственных нейронных сетей, их компоненты, подходы к обучению. Также будет обоснован выбор компонентов, с которым будет проводиться работа в рамках проекта.

### **2.1. Архитектура нейронных сетей**

Искусственная нейронная сеть – математическая модель, имитирующая работу мозга. Модель основана на искусственных нейронах, соединенных между собой подобно настоящим нейронам, соединенным синапсами. У каждого нейрона есть вход и выход, каждый из которых соединен с одним или несколькими нейронами, функция активации, преобразующая набор входных данных в вывод, и вес, определяющий значимость вывода нейрона. В совокупности, НС – "черный ящик", принимающий набор входных значений и возвращающий набор выходных значений или сигналов.

Необученная НС – "пустое полотно". Чаще всего веса нейронов распределены случайно, поэтому в самом начале вывод НС не соответствует ожидаемому. Чтобы подстроить модель под предметную область, проводят ее обучение. Для этого проводят большое количество итераций, в каждой из которых на входы НС передаются данные, а полученный вывод сравнивают с ожидаемым. В зависимости от того, насколько результат близок к идеальному, веса нейронной сети пересчитываются.

Для разных задач применяется разная архитектура. Наиболее часто используемые из них - MLP, CNN, RNN[3]. В рамках этой работы будем рассматривать только сети типа feed-forward, предназначенные для преобразования набора вводов в набор выводов.

### ***2.1.1. Многоуровневый персептрон (Multi-Layer Perceptron, MLP)***

Базовый вид нейронной сети. Хотя он выделен отдельно, он входит в состав других видов нейронных сетей, обеспечивая их вычислительную мощность.

Состоит из отдельных слоев из нейронов, идущих друг за другом. Ввод каждого нейрона связан с выводами всех нейронов предыдущего слоя, вывод - со входами всех нейронов следующего. Такая нейронная сеть с одним скрытым слоем (т. е. одним входным, одним выходным слоем и одним слоем между ними) способна вычислять результат любой нелинейной функции [6]. Применяются для задач с малым количеством входных параметров.

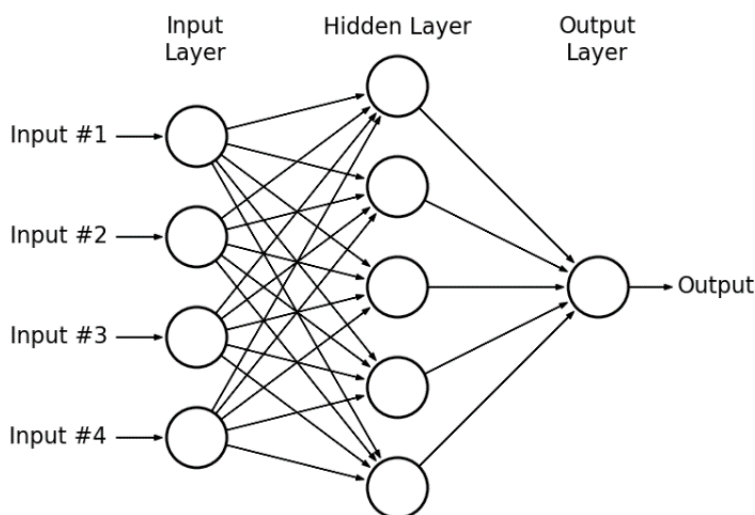


Рис.2.1. Схема многоуровневого персептрона

### ***2.1.2. Сверточная нейронная сеть (Convolutional Neural Network, CNN)***

Если входные данные расположены в пространстве (напр., фотография), необходимо использовать такую сеть. Если сместить изображение объекта на фотографии из одного угла в другой, MLP примет ее за новую. К тому же, для кодирования изображения нужно большое количество вводов (три ввода на каждый пиксель – для трех цветовых каналов), отчего сложность обработки для MLP многократно увеличивается.

Для таких задач используют сверточные нейронные сети. Входные данные в виде тензора передаются в сверточный слой, где для каждого пикселя вычисляется значение некоторой функции в пределах окна  $N \times N$  пикселей[10]. Значение функции определяет признак (feature), найденный в этом окне (например, линия или точка). Значение функции вычисляется с помощью фильтра, который изменяется в процессе обучения нейронной сети.

Вывод сверточного слоя идет на следующий сверточный слой, который будет работать уже с найденными признаками, а не исходной картинкой (например, второй слой может собрать несколько линий в букву).

Используя большое количество слоев, можно натренировать нейронную сеть находить на изображении объекты с большой точностью.

На рис.2.2 изображена сверточная нейронная сеть, где помимо свертки показана операция пулинга (pooling). С ее помощью выделяются самые ярко выраженные на изображении признаки (например, если на участке виден край стола с фактурой дерева, после пулинга может остаться только более четко выраженный край стола, но не остальные линии).

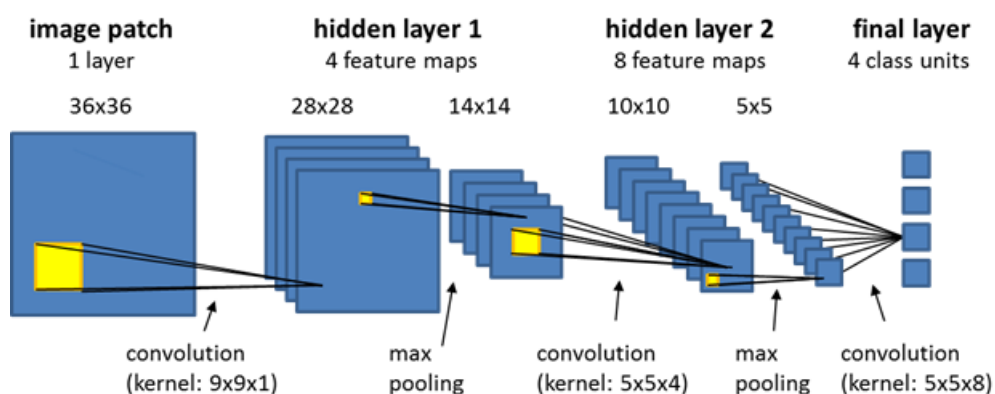


Рис.2.2. Схема сверточной сети

### 2.1.3. Рекуррентная нейронная сеть (Recurrent Neural Network, RNN)

Для данных, расположенных во времени, необходимо использовать рекуррентные сети. В этом случае каждый набор вводных данных будет описывать одну точку во времени, но нейронная сеть также должна принимать во внимание предыдущие значения.

В таких сетях применяются рекуррентные слои, в которых каждый нейрон отправляет свой вывод нейронам следующего слоя и на свой собственный вход. Значение на выводе определяется по предыдущему и новому значению.

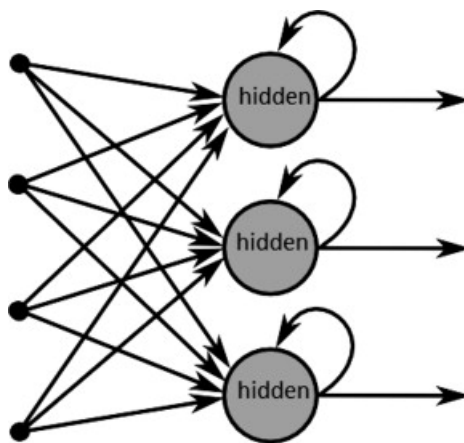


Рис.2.3. Схема рекуррентных нейронов

В теории это значит, что нейросеть "помнит" все состояния до текущего, но на практике это не так. Каждый раз, когда вычисляется значение на входе рекуррентного нейрона, предыдущее значение только частично влияет на результат. С каждым шагом старые данные постепенно вытесняются новыми, поэтому такие сети на самом деле обращают внимание только на несколько последних шагов.

Более продвинутая версия рекуррентного нейрона - ячейка LSTM (Long Short-Term Memory, долгая краткосрочная память). У нее есть специальные переключатели, которые нужно использовать, чтобы очистить ячейку, сохранить или вывести значение. С помощью LSTM нейросеть может выбирать, какую информацию нужно сохранить, а от какой нужно избавиться.

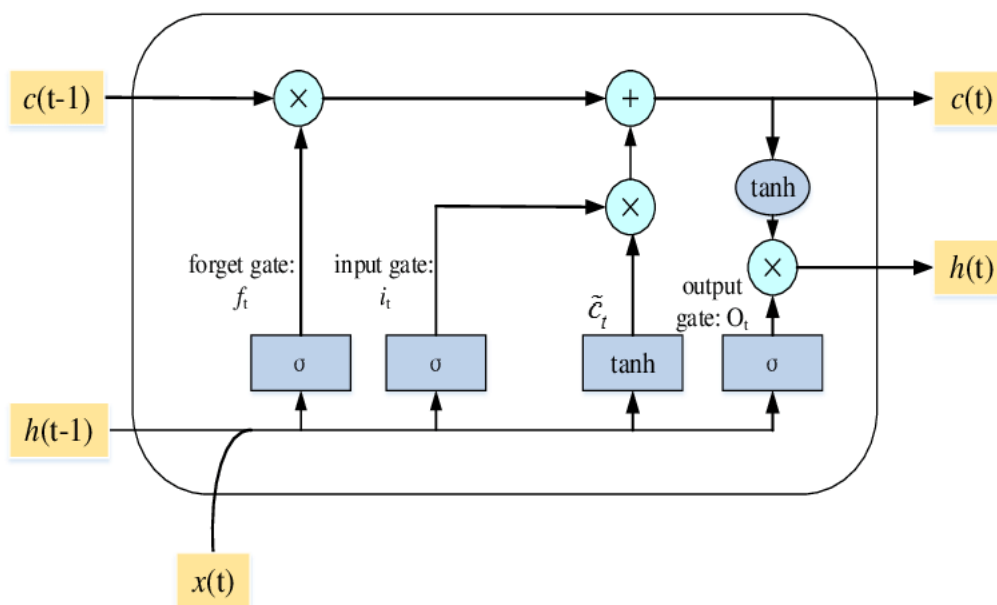


Рис.2.4. Схема ячейки LSTM

Для автоматической торговли на бирже нужно использовать рекуррентную нейронную сеть, так как присутствует движение во времени. Используется слой с



ячейками LSTM, позволяющий нейросети запоминать только значимые признаки на любой период времени.

## 2.2. Способы обучения

### 2.2.1. Генетические алгоритмы[8]

В генетических алгоритмах каждый экземпляр нейронной сети (особь) рассматривается как набор генов, которые определяют его поведение. Алгоритмы основываются на биологической эволюции, а точнее на трех основных принципах:

1. **Естественный отбор.** Оставить потомство могут только те особи, которые смогли приспособиться к окружению и выжить. Так обеспечивается постоянное совершенствование популяции при переходе между поколениями.
2. **Репродукция и скрещивание.** Из текущего поколения выбираются пары особей, гены которых смешиваются для создания новой особи в следующем поколении. Так обеспечивается генетическое разнообразие, что помогает более быстрому развитию.
3. **Мутация.** Часть генов должна пройти случайную мутацию. Благодаря этому новые особи не только комбинируют поведение своих родителей, но и приобретают новые, потенциально полезные гены.

### 2.2.2. Генеративно-состязательные алгоритмы

Используют две нейронных сети – генератор и дискриминатор. Генератор создает входные данные, похожие на реальные. Задача дискриминатора – имея на вводе один настоящий набор данных и один сгенерированный, определить, какой из них настоящий.

Две нейронные сети стремятся достичь противоположные цели: генератор – создать реалистичный набор входных данных, дискриминатор – определять, настоящий перед ним набор данных или сгенерированный.

Такие алгоритмы можно использовать для улучшения компьютерной графики или увеличения изображений.

### **2.2.3. Алгоритмы обучения с подкреплением**

Используются для обучения агента (т.е. нейросети, которая выполняет какие-то действия). Для этого после каждого действия вычисляется награда, исходя от достигнутых результатов. Например, если агент управляет автомобилем, за каждый квант времени агент будет получать положительное количество очков, а за въезд на тротуар - отрицательное.

В этом случае обучается только одна нейросеть, но благодаря системе наград она учится делать правильное действие в определенной ситуации.

Этот алгоритм можно применять без нейронной сети, если количество возможных состояний конечно (например, прохождение лабиринта). Нужно создать таблицу, где для каждого состояния (положения агента в лабиринте) будут обозначены оптимальные действия (шаг в одну из четырех сторон).

Тем не менее, в случае с неограниченным количеством состояний нужно использовать нейросеть, которая сможет выбирать действие не только каждой конкретной выученной ситуации, но и в похожих на нее.

Обучение с подкреплением используется в данном проекте. Этот алгоритм имеет важное свойство: нейросеть получает обратную связь после каждого шага. В отличие от генетического алгоритма, агент после каждого действия будет понимать, насколько оно было эффективно. В случае, когда входные данные – это серия размером до 4 тыс. временных точек, это необходимо для эффективного обучения.

## **ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ**

В этой главе рассматривается разработанная программа.

Программа написана на языке Java с использованием фреймворка Java FX для графического интерфейса. Источник исторических данных о стоимости акций – Alpha Vantage API. Обучение нейронной сети производится с помощью фреймворка deeplearning4j, в частности пакета RL4J[5], предназначенного для обучения с подкреплением.

Далее более подробно описывается программная реализация, процесс обучения и тестирования.

### 3.1. Загрузка данных

Для обучения нейросети нужен большой объем данных. Существует много источников, но в общедоступных хранятся данные только о тех компаниях, которые работают сейчас. Отсутствие в наборе компаний, ушедших с биржи из-за банкротства или по другим причинам, может повлиять на качество обучения и оценки.

Один из источников данных - Alpha Vantage[2], который предоставил академический доступ к своей базе данных. В базе данных есть информация о колебаниях цен внутри дня (intraday) за последние два года и ежедневные записи с 1999 года (daily). В проекте используется датасет daily-adjusted, который содержит стоимость акций, информацию о выплачиваемых дивидендах и дроблении акций.

Всего удалось получить данные по примерно 8000 активным компаниям и 3700 снятых с торгов.

Из загруженных данных случайно выбран тестовый набор (500 компаний). Оставшиеся компании разделены на 10 наборов для обучения по возрастанию волатильности (здесь вычислялась как максимальное изменение цены в процентах между двумя измерениями).

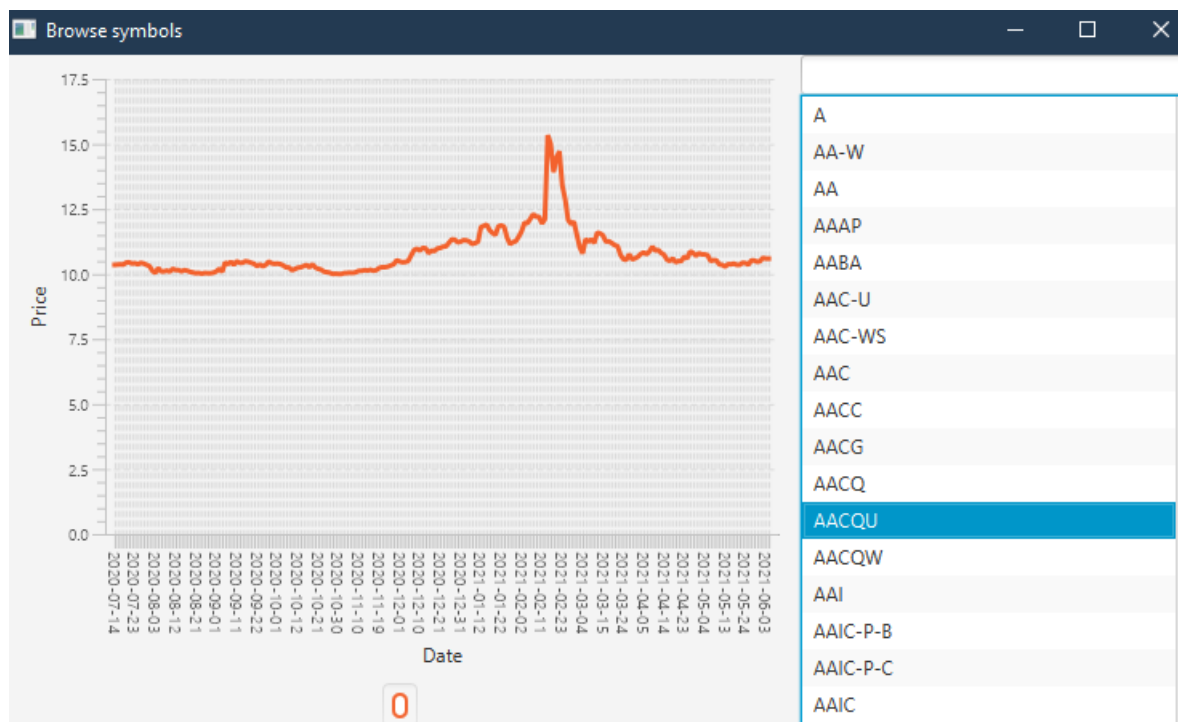


Рис.3.1. Просмотр загруженных данных

### 3.2. Окружение для обучения

Для того, чтобы обучать нейронную сеть в RL4J, нужно создать окружение, с которым она будет взаимодействовать. С этой целью были используются классы: **SimpleBroker**.

Хранит текущую цену акций, баланс и портфолио. Три основные функции: купить N акций, продать N акций, обновить данные.

Данные читаются из объекта класса DataFeed. При инициализации брокера DataFeed выбирает случайный файл из набора и случайные временные рамки (до 7 лет). При обновлении считываются данные для следующей временной точки.

RL4J требует, чтобы SimpleBroker использовал интерфейс Encodable, то есть, чтобы он мог кодировать свое состояние в виде массива, который потом будет использоваться как набор входных данных нейронной сети.

#### **BrokerMdp.**

С этим классом непосредственно взаимодействует нейронная сеть. Он должен использовать интерфейс MDP (Markov Decision Process), который определяет необходимые методы для обучения с подкреплением.

Среди них методы для получения пространства наблюдения и пространства действий (сопоставляются с вводами и выводами нейросети); метод для отправки в нейросеть очередного набора входных данных и начисления "награды".

После многочисленных попыток обучения и тестирования система начисления очков оказалась выстроена следующим образом:

- За каждую покупку или продажу брокер забирает 1% стоимости акций в виде комиссии;
- При каждой операции покупки или продажи награда будет составлять отношение прибыли (или убытка) к начальному балансу, то есть прибыль или убыток в процентах;
- На свободные деньги действует "инфляция" в 10%, которая влияет на очки, но не на количество денег на счете;
- Дополнительные очки начисляются, если нейросеть не совершает покупку или продажу.

Так как RL4J поддерживает только дискретное пространство действий, выбраны 25 действий:

1. Не покупать и не продавать;
2. Купить  $2^n$  акций, n от 1 до 12;

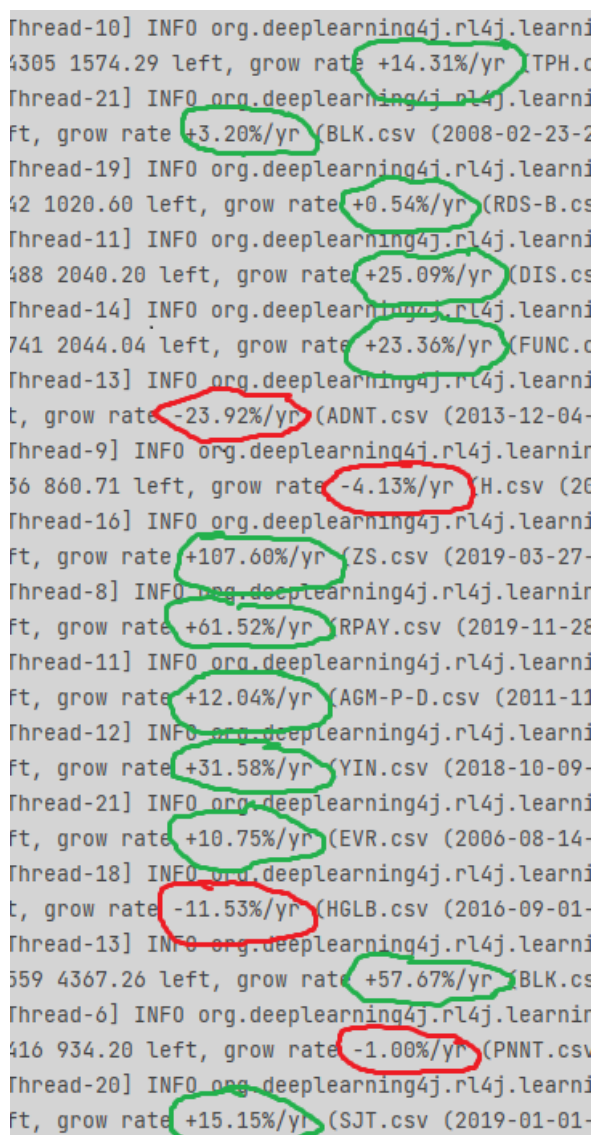
3. Продать  $2^n$  акций,  $n$  от 1 до 12.

### 3.3. Настройка и обучение нейросети

Используемая архитектура нейронной сети:

- Входной слой (4 нейрона);
- 32 скрытых слоя прямого распространения (по 32 нейрона);
- Слой LSTM (32 нейрона);
- Выходной слой (25 нейронов).

Функция обновления Adam(0.01)[4], l2-регуляризация с параметром 0.01.  $\gamma = 0.995$ .



```

Thread-10] INFO org.deeplearning4j.rl4j.learnir
4305 1574.29 left, grow rate +14.31%/yr (TPH.c
Thread-21] INFO org.deeplearning4j.rl4j.learnir
ft, grow rate +3.20%/yr (BLK.csv (2008-02-23-2
Thread-19] INFO org.deeplearning4j.rl4j.learnir
42 1020.60 left, grow rate +0.54%/yr (RDS-B.cs
Thread-11] INFO org.deeplearning4j.rl4j.learnir
488 2040.20 left, grow rate +25.09%/yr (DIS.cs
Thread-14] INFO org.deeplearning4j.rl4j.learnir
741 2044.04 left, grow rate +23.36%/yr (FUNC.c
Thread-13] INFO org.deeplearning4j.rl4j.learnir
t, grow rate -23.92%/yr (ADNT.csv (2013-12-04-
Thread-9] INFO org.deeplearning4j.rl4j.learnir
56 860.71 left, grow rate -4.13%/yr (H.csv (20
Thread-16] INFO org.deeplearning4j.rl4j.learnir
ft, grow rate +107.60%/yr (ZS.csv (2019-03-27-
Thread-8] INFO org.deeplearning4j.rl4j.learnir
ft, grow rate +61.52%/yr (RPAY.csv (2019-11-28
Thread-11] INFO org.deeplearning4j.rl4j.learnir
ft, grow rate +12.04%/yr (AGM-P-D.csv (2011-11
Thread-12] INFO org.deeplearning4j.rl4j.learnir
ft, grow rate +31.58%/yr (YIN.csv (2018-10-09-
Thread-21] INFO org.deeplearning4j.rl4j.learnir
ft, grow rate +10.75%/yr (EVR.csv (2006-08-14-
Thread-18] INFO org.deeplearning4j.rl4j.learnir
t, grow rate -11.53%/yr (HGLB.csv (2016-09-01-
Thread-13] INFO org.deeplearning4j.rl4j.learnir
559 4367.26 left, grow rate +57.67%/yr (BLK.cs
Thread-6] INFO org.deeplearning4j.rl4j.learnir
416 934.20 left, grow rate -1.00%/yr (PNNT.csv
Thread-20] INFO org.deeplearning4j.rl4j.learnir
ft, grow rate +15.15%/yr (SJT.csv (2019-01-01-

```

Рис.3.2. Прибыль нейросети в процессе обучения (процентов за год)

### 3.4. Тестирование

Несмотря на хороший результат во время обучения, нейросеть плохо проявляет себя при тестировании.

Результат тестирования одного из вариантов нейросети на рис.3.3.



Рис.3.3. Результат тестирования(1)

На верхнем графике указана цена акции; на среднем - общая стоимость акций в портфолио нейронной сети и свободных денег; на нижнем - количество акций в портфолио.

На нижнем графике сразу видна самая большая проблема: нейросеть не научилась ждать. На каждом шаге она либо продает, либо покупает большие объемы акций. Из-за этого она терпит убытки от комиссии даже там, где цена не меняется.

Помимо избыточной активности, нейросети в процессе обучения очень часто входили в две крайности: либо вообще ничего не покупали, либо с самого начала покупали максимум, который могли себе позволить.

Первый случай означает, что нейросеть считает риск слишком большим и решает не вкладывать деньги, даже несмотря на снятие баллов "инфляции".

Второй случай возникал, если большая часть акций, с которыми работала нейросеть, в конце временного отрезка сами дорожали. Нейросеть становится "ленивой" и просто продолжает использовать проверенную стратегию.



Рис.3.4. Результат тестирования(2)

При этом второй вариант технически оказался наиболее прибыльным. При тестировании на отдельном датасете средняя годовая прибыль во всех случаях составляла не менее 8% (рис.3.4).

## ЗАКЛЮЧЕНИЕ

Проведено исследование нейронных сетей. Запрограммировано окружение и графический интерфейс для работы с нейронной сетью. Построена и обучена рекуррентная нейронная сеть (RNN) с помощью алгоритма АЗС.

Нейросеть не достигла приемлемых результатов. Причинами этого могут быть:

- Использование большого количества дискретных выводов, что увеличивает сложность обучения по сравнению с одним непрерывным выводом;
- Неудачный выбор входных данных нейросети и системы наград;
- Неправильный выбор гиперпараметров;
- Недостаточное обучение.

Полученная в процессе выполнения работы нейросеть не способна соревноваться с другими инструментами технического анализа. После тщательной настройки и обучения возможно улучшение производительности.



## СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

- НС**    Нейронная сеть.
- MLP**   Multi-Layer Perceptron.
- RNN**   Recurrent Neural Network.
- LSTM**   Long Short-Term Memory.
- CNN**   Convolutional Neural Network.
- A3C**   Asynchronous Advantage Actor-Critic.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A graph-based CNN-LSTM stock price prediction algorithm with leading indicators / Jimmy Ming-Tai Wu Zhongcui Li. — URL: <https://link.springer.com/article/10.1007/s00530-021-00758-w#Sec22> (visited on 25.06.2021).
2. Alpha Vantage. — URL: <https://www.alphavantage.co/> (visited on 25.06.2021).
3. *Brownlee J.* When to Use MLP, CNN, and RNN Neural Networks. — 2018. — URL: <https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/> (visited on 25.06.2021).
4. *Bushaev V. Adam* — latest trends in deep learning optimization. — 2018. — URL: <https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c> (visited on 25.06.2021).
5. *Fiszel R.* Reinforcement Learning and DQN, learning to play from pixels. — 2016. — URL: <https://rubenfiszel.github.io/posts/rl4j/2016-08-24-Reinforcement-Learning-and-DQN.html> (visited on 25.06.2021).
6. *Fortuner B.* Can neural networks solve any problem? — 2017. — URL: <https://towardsdatascience.com/can-neural-networks-really-learn-any-function-65e106617fc6> (visited on 25.06.2021).
7. Herd Instinct. — URL: <https://www.investopedia.com/terms/h/herdinstinct.asp> (visited on 25.06.2021).
8. *Nicholson C.* A Beginner's Guide to Genetic and Evolutionary Algorithms. — 2020. — URL: <https://wiki.pathmind.com/evolutionary-genetic-algorithm> (visited on 25.06.2021).
9. *Smigel L.* Algorithmic Trading: Is It Worth It? — 2018. — URL: <https://analyzingalpha.com/algorithmic-trading-is-it-worth-it> (visited on 25.06.2021).
10. *Stewart M.* Simple Introduction to Convolutional Neural Networks. — 2019. — URL: <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac> (visited on 25.06.2021).

UML-схема некоторых классов

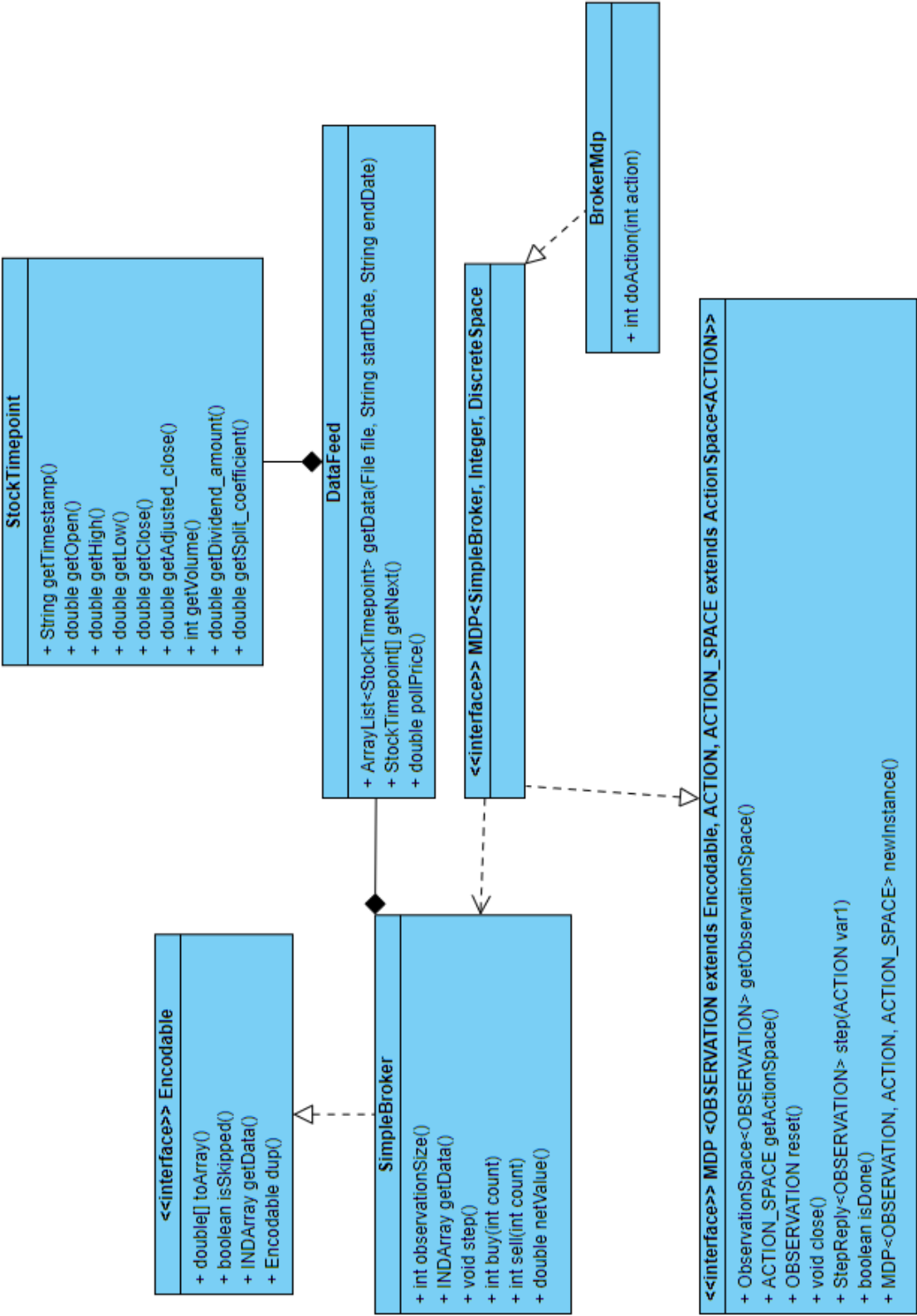


Рис.П1.1. Диаграмма некоторых классов

**Инициализация DataFeed из файла**

```
public DataFeed(Random r, int level) {
    ArrayList<StockTimepoint> fromFile;
    // Random file from the training set or the test set.
    List<File> filesList;
    if (level >= 0) {
        filesList = Utils.getTrainFiles(level);
    } else {
        filesList = Utils.getTestFiles();
    }

    do {
        // Starting date (within 1999-01-01 to 2019-12-12).
        int year = r.nextInt(21) + 1999;
        int month = r.nextInt(12) + 1;
        int day = r.nextInt(31) + 1;
        String startDate = year + "-" + String.format("%02d", month)
            + "-" + String.format("%02d", day);

        // End day is 3 to 10 years after start day.
        month = r.nextInt(12) + 1;
        day = r.nextInt(31) + 1;
        String endDate = (year + r.nextInt(8) + 3) + "-" +
            String.format("%02d", month) + "-" + String.format("%02d", day);

        File file = filesList.get(r.nextInt(filesList.size()));

        fromFile = getData(file, startDate, endDate);
        info = file.getName() + " (" + startDate + "-" + endDate + ")";
    } while (fromFile == null || fromFile.size() < 200);
    data = fromFile;
}
```

## Методы класса SimpleBroker

```
public void step() {
    if (seriesEnd) {
        throw new IllegalStateException("No more datapoints");
    }
    StockTimepoint[] res = feed.getNext();
    if (res == null) {
        seriesEnd = true;
    }
    else {
        currentPrice = res[0].getOpen();
        if (res[0].getSplit_coefficient() != 1.0 && stocksCount != 0) {
            int oldCount = stocksCount;
            stocksCount = (int) Math.round(stocksCount * res[0]
                .getSplit_coefficient());
            avgBuyPrice = avgBuyPrice * oldCount / stocksCount;
        }
        if (res[0].getDividend_amount() != 0.0 && stocksCount != 0) {
            cash += stocksCount * res[0].getDividend_amount();
        }
    }
}

public int buy(int count) {
    if (count == 0) { throw new IllegalArgumentException("Attempt to
        buy 0 stocks, which is not nice"); }
    int canBuy;
    if (feed.pollPrice() == 0) {
        return 0;
    } else {
        canBuy = (int) Math.floor(cash / (feed.pollPrice()
            * (1 + 0.01 * commissionPercent)));
    }
}
```

```

if (canBuy == 0) { return -count; };
}
int numberToBuy = Math.min(count, canBuy);
double totalPrice = numberToBuy * feed.pollPrice();
cash -= totalPrice * (1 + 0.01 * commissionPercent);

avgBuyPrice = avgBuyPrice * stocksCount + totalPrice;
stocksCount += numberToBuy;
avgBuyPrice /= stocksCount;
return (numberToBuy < count) ? numberToBuy - count : 0;
}

public int sell(int count) {
if (count == 0) { throw new IllegalArgumentException("Attempt to
sell 0 stocks, which is not nice"); }
if (stocksCount == 0) {
return -count;
}
if (feed.pollPrice() == 0) {
return 0;
}
int numberToSell = Math.min(count, stocksCount);
double total = feed.pollPrice() * numberToSell;
cash += total * (1 - 0.01 * commissionPercent);
stocksCount -= numberToSell;
return numberToSell < count ? numberToSell - count : 0;
}

```

**Метод step класса BrokerMdp**

```
public StepReply<SimpleBroker> step(Integer action) {
    double oldNetPrice = broker.netValue();
    doAction(action);
    step++;
    double newNetPrice = broker.netValue();

    double reward = -inflationRate * broker.getCash() /
        365 / startingCash;
    if (action == 12) {
        reward += 0.1;
    }
    double balance = newNetPrice - oldNetPrice;
    if (balance > 0) {
        reward += balance / startingCash;
    } else {
        reward += balance / startingCash * lossWeight;
    }

    cumulativeReward += reward;

    return new StepReply<>(broker, reward / 10, isDone(), null);
}
```

## Приложение 5

**Инициализация, обучение и сохранение весов нейронной сети**

```
A3C = A3CDiscrete.A3CConfiguration.builder()
    .seed(1)
    .maxEpochStep(365 * 11)
    .maxStep(100000)
    .numThread(16)
    .nstep(10)
    .updateStart(0)
    .rewardFactor(0.1)
    .gamma(0.995)
    .errorClamp(1.0)
    .build();

configuration = ActorCriticFactorySeparateStdDense.Configuration
    .builder()
    .updater(new Adam(0.01))
    .useLSTM(true)
    .l2(0.01)
    .numHiddenNodes(32)
    .numLayer(32)
    .build();

mdp = new BrokerMdp(1000, new Random(66), 1);
a3c.train();

mdp.close();
try {
    a3c.getPolicy().save(Configuration.getConfig("network-folder") +
        "value-zero-bias-lv1.bin", Configuration.getConfig("network-folder")
        + "policy-zero-bias-lv1.bin");
} catch (IOException e) { throw new RuntimeException(e); }
```