

САНКТ-ПЕТЕРБУРГСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики — процессов управления  
Кафедра технологии программирования

Выпускная квалификационная работа бакалавра  
Петрова Валентина Юрьевича

**Распознавание рукописных математических выражений с  
использованием нейронных сетей**

Основная образовательная программа «Прикладная математика,  
фундаментальная информатика и программирование» СВ.5005.2015

*Научный руководитель:*  
старший преподаватель  
Малинина М. А.

*Заведующий кафедрой:*  
кандидат технических наук, доцент  
Блеканов И. С.

Санкт-Петербург  
2019 г.

## Содержание

Введение.....	3
Актуальность.....	4
Постановка задачи.....	5
Глава 1: Обзор литературы.....	7
Глава 2: Основные понятия.....	9
Глава 3: Реализация.....	14
3.1: Построение диаграммы Вороного.....	14
3.2: Обучение свёрточной нейронной сети.....	16
3.3: Распознавание.....	19
3.3.1: Первая попытка распознавания.....	19
3.3.2: Вторая попытка распознавания.....	23
Выводы.....	25
Заключение.....	25
Использованные материалы.....	27

## Введение

Человечество веками стремилось к автоматизации. Сегодня всё больше типовых задач способны взять на себя специальные устройства и грамотно написанные алгоритмы.

Одной из задач оптического распознавания символов (англ. Optical Character Recognition — OCR) является задача распознавания рукописного текста. Главное отличие и сложность рукописного текста в том, что не существует стандарта написания одних и тех же букв, символов. Каждый человек может написать их по-разному. Кроме того, в отличие от печатного текста, его рукописный аналог, зачастую, сложно разделить на строки и слова; строки могут получиться косыми, а несколько слов одного предложения слиться воедино.

Среди рукописных текстов можно встретить специализированный, направленный на определенную аудиторию текст. В данной работе рассматриваются рукописные математические тексты. Их основная сложность в наличии формул, которые, в отличие от обычных слов и букв, может быть сложно выделять из текста. Они могут быть не линейны, содержать в себе трудноотделимые символы, такие как квадратные корни и степени.

Выделяют два основных типа распознавания рукописного текста: ONLINE и OFFLINE распознавание. ONLINE распознавание чаще применяется в программному обеспечению (ПО) электронных устройств, например в сенсорных экранах телефонов и планшетов. ПО отслеживает движения пальца пользователя, когда он рисует символы. Далее эта информация участвует в алгоритме определения символа написанного пользователем.

В случае же OFFLINE распознавания, доступна лишь конечная информация о символах, к примеру, фотосканы работ учащихся 11 класса, или рукописное заявление о приеме на работу. Этот тип распознавания сложнее, так как отсутствует доступ к информации о том, как автор текста выводил символы.

Решение задачи OFFLINE распознавания рукописных математических выражений и символов и будет рассмотрено в этой работе.

## **Актуальность**

В настоящее время есть множество систем, позволяющих упростить жизнь математикам, физикам и прочим инженерам. Из самых известных примеров можно привести системы Wolfram Mathematica, Matlab и другие. Основной недостаток использования таких пакетов для конечного пользователя в том, что необходимо знать и уметь оперировать специальным синтаксисом, требующимся для решения конкретного типа задач функции.

Было бы удобнее использовать приложение, которое умеет считывать изображение с написанной от руки задачей, и решать его.

Такие системы есть. Из популярных примеров похожих приложений можно назвать приложение для смартфонов Photomath [5]. Оно имеет очень высокий рейтинг (4.7 из возможных 5 звёзд) на сервисе Google Play и большую аудиторию (1 120 000 пользователей) на момент написания данной работы. В его состав, помимо вычислительного ядра, входит часть, которая отвечает за обработку сфотографированного изображения, распознавания математического текста на нем и дальнейшей обработки.

Кроме того, данная система может иметь широкое применение в задаче автоматической проверки математических работ, например тестовых школьных заданий или ЕГЭ. Возможность автоматически определять математические уравнения позволила бы упростить этот труд.

Таким образом, обработка рукописных математических выражений является актуальной и уже сейчас успешно применяется для упрощенного решения математических задач.

## **Постановка задачи**

В представленной работе стояла задача научиться распознавать математические рукописные выражения. В качестве таких выражений рассматриваются как отдельные уравнения, так и системы линейных уравнений (СЛУ).

Из-за сложности реализации подобной задачи, было решено ввести набор допущений. Предполагаем, что на изображении не присутствует лишних символов (к примеру, принадлежность параметров системы какому-либо пространству или знака, обозначающего принадлежность нескольких уравнений к одной системе) и все присутствующие на изображении уравнения принадлежат одной системе линейных уравнений.

Необходимо разработать программу, которая:

- умеет распознавать СЛУ на изображении
- выделять отдельно каждое уравнение
- делить выделенное уравнение на символы

- распознавать каждый символ с применением нейронных сетей.

Конечная цель здесь заключается в способности перевести систему из графического вида в вид символьный, использующий специальную семантику.

Для решения задачи необходимо решить следующие подзадачи:

- уметь разбивать поданную на вход систему на строки, а строки — на символы
- найти датасет, содержащий рукописные буквы, цифры и символы, присутствующие в математических выражениях
- натренировать нейронную сеть распознавать отдельно каждый символ

## Глава 1: Обзор литературы

Первым делом, необходимо найти способ выделять в рассматриваемой рукописной системе линейных уравнений строки, для их дальнейшего анализа и распознавания. Так как в данной работе рассматривается OFFLINE распознавание, отсутствует информация о том, как именно была написана система. Невозможно отследить действия человека, написавшего уравнение, которые могли помочь в определении того, какой строке принадлежит тот или иной символ.

Кроме того, дополнительную сложность представляет сама по себе рукописная природа символов, которые необходимо распознавать. Символы одного выражения, в отличие от печатного случая, могут располагаться не на одной горизонтальной прямой, а быть сдвинуты по горизонтали, а также иметь разный межсимвольный и межстрочный интервалы.

В работе [1] описывается приём построения текстового скелета, который можно использовать для разделения текста на строки, а строки на слова. Автор показывает его эффективность на печатном тексте, в том числе, искаженном относительно осей абсцисс и ординат.

Наиболее подходящим для поставленной задачи, а именно, выделение выражений на изображении, содержащем СЛУ, является подход, подразумевающий построение скелета текста, в качестве которого выступает диаграмма Вороного. Она будет использоваться, как текстовый скелет, по аналогии с [1]. В статье [2] описываются алгоритмы на основе диаграммы Вороного, позволяющие разделять выражения между собой, используя различные метрики. В частности, важными метриками являются близость

символа к границе его клетки и расстояние между двумя соседними ячейками Вороного.



## Глава 2: Основные понятия

В этой главе приводятся необходимые для работы математические алгоритмы. Кроме того, будут перечислены некоторые термины, относящиеся к темам нейронных сетей и распознаванию изображений.

**Перцептрон (*perceptron*) Розенблатта** — нейронная сеть, имеющая 1 скрытый слой, позволяющий ей решить «проблему XOR».

**Многослойный перцептрон (*multilayer perceptron, MLP*)** — идейное продолжение перцептрона Розенблатта, суть которого в наличии более одного скрытого слоя. Для данной модели важной и непростой задачей является задача обучения сети, а именно настройка синапсов — весов, попарно соединяющих слои нейронов.

**Метод обратного распространения ошибки (*backpropagation method*)** — метод обучения многослойного перцептрона, основанный на алгоритме градиентного спуска. Ошибка, полученная методом градиентного спуска, распространяется дальше в сеть. Причем, сильнее корректируются веса, соединяющие нейроны, которые показывали наибольшую ошибку на тестирующем подмножестве из тестовой коллекции.

**Свёрточная нейронная сеть (*convolutional neural network*)** — особый тип нейронной сети, созданный для работы с изображениями; очень хорошо зарекомендовал себя в задачах распознавания изображений.

На вход свёрточной нейронной сети подаётся изображение. В случае, если изображение является трёх-канальным (RGB изображение), возможны два

варианта обработки: приведение к чёрно-белому или рассмотрение каждого из каналов отдельно. Во втором случае, все сказанное далее будет верно, с той оговоркой, что проход по сети будет осуществляться для каждого канала отдельно.

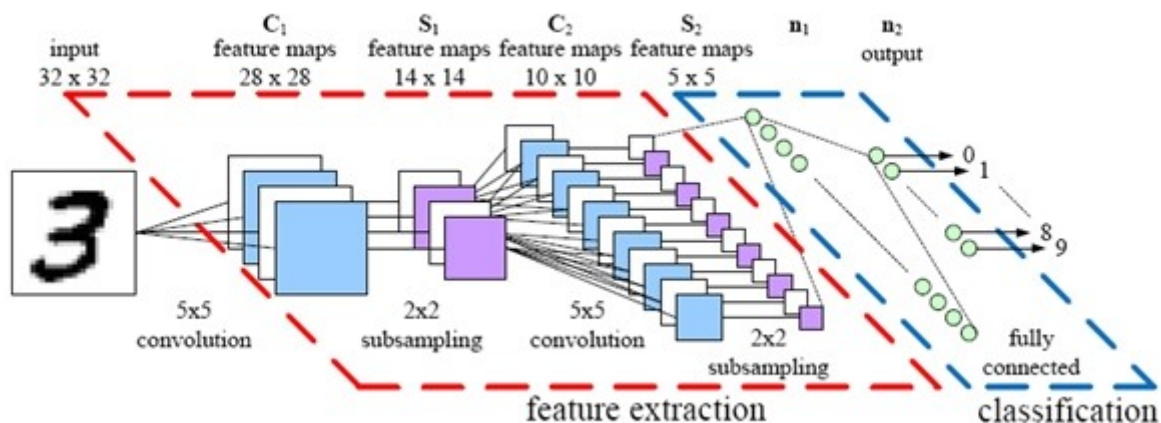


Рисунок 1: Архитектура свёрточной нейронной сети

Далее каждое изображение проходит последовательно через свёрточный (convolutional) и подвыборочный (sub-sampling / pooling) слои. Чередуюсь между собой несколько раз, они формируют входной вектор признаков для многослойного персептрона, который является завершающей частью свёрточной нейросети.

**Свёрточный слой** представляет собой массив так называемых карт признаков. Каждой карте соответствует своё сканирующее ядро (или, как иногда говорят, фильтр). Пришедшее с предыдущего слоя изображение подвергается свёртке - сканирующее ядро проходит по всему изображению, начиная с левого верхнего угла, и на каждой итерации записывает сумму поэлементного произведения своих элементов и элементов, которые подвергаются свёртке.

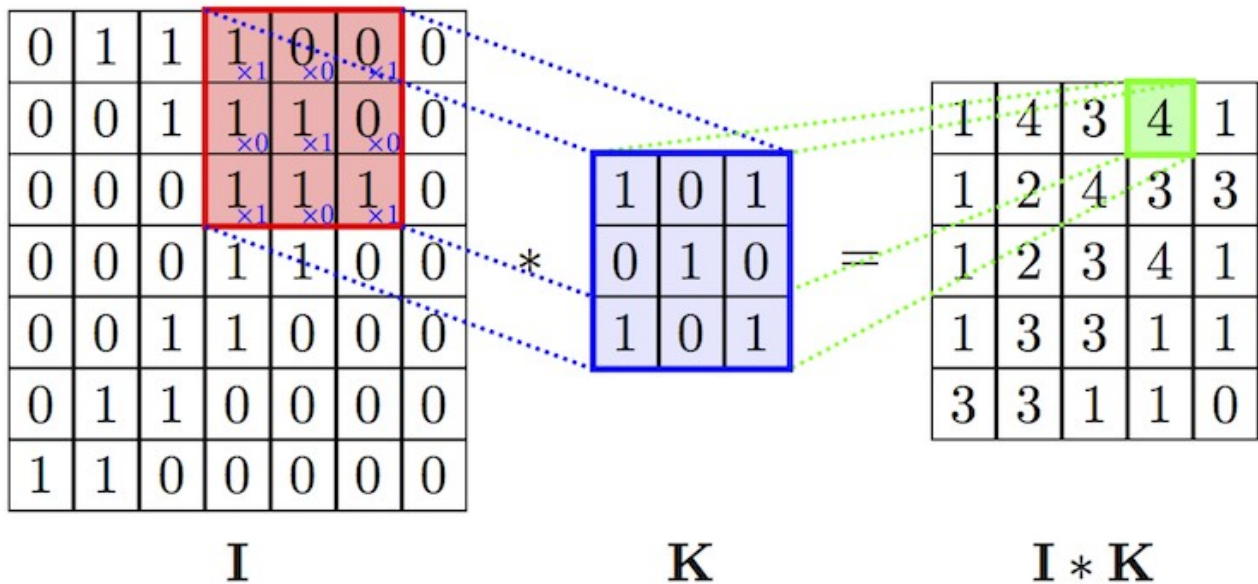


Рисунок 2: Свёртка

Существует несколько **гиперпараметров**, то есть констант, которые необходимо задать вручную до начала тренировки сети. К их числу относятся:

**1) Размер сканирующего ядра**

Обычно выбирают квадратную размерность, между 3x3 и 7x7, так как слишком маленькое или большое ядро может не выявить каких-либо специфических черт данного класса изображений

**2) Количество карт в одном слое свёртки**

Для каждой карты применяется свой фильтр, таким образом можно выявлять несколько признаков в одном свёрточном слое

**3) Поведение на границе**

Очевидно, что нужно как-то разрешать случаи, когда ядро проходит по изображению, выходя за его границы. Если несколько способов решения этой

ситуации: замещение граничными элементами, зеркальное отражение элементами, попавшими в слой

#### 4) Шаг ядра

Количество пикселей, на которые фильтр перемещается за одну итерацию

После слоя свёртки к получившейся карте признаков поэлементно применяется **активационная функция**. Очень популярной на этом этапе является функция ReLU, которая определяется, как:

$$f(x) = \max(0, x)$$

Основное преимущество такого выбора в отсутствии ресурсоёмких операций, и, как следствие, более быстрого обучения сети. Минусы же в том, что нейроны с такой активационной функцией будут быстро "умирать". Во избежание таких ситуаций, используют различные модификации ReLU. К примеру, вместо нуля задается какое-либо малое значение. Исходя из [6], использование одной из модифицированных версий функции может быть полезным в некоторых случаях.

По аналогии со свёрточным, в **подвыборочном слое** также присутствуют карты, причём количество этих карт совпадает с количеством карт предыдущего слоя. Цель данного слоя - уменьшение размерности изображения (после выделения некоторых особенностей в свёрточном слое, необходимо "уплотнить" картинку).

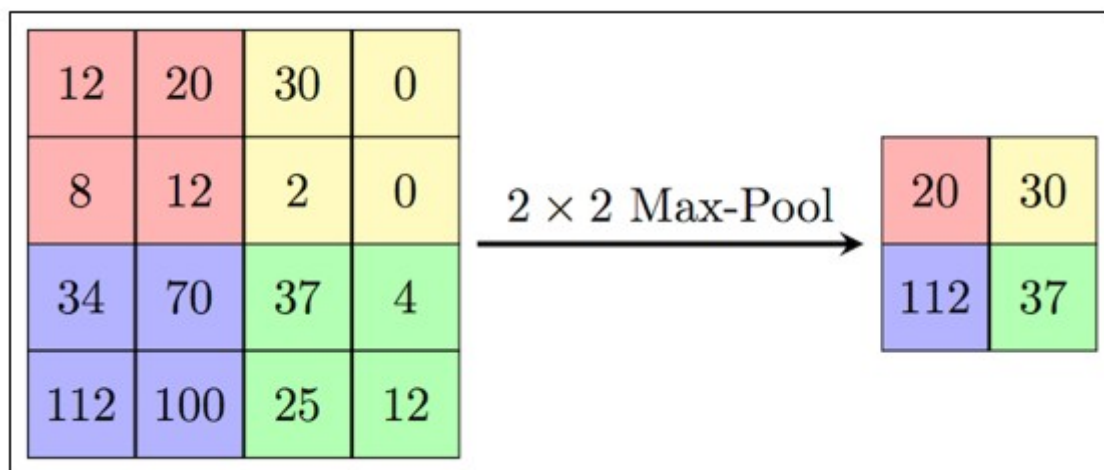


Рисунок 3: MaxPooling

В качестве **ядра**, уменьшающего размерность изображения, часто используется MaxPooling размерности 2x2. Этот алгоритм работает следующим образом: фильтр проходит по всей карте (никогда не используя один и тот же пиксель дважды), выбирая максимальное значение из захваченных им элементов, и отражает его на соответствующий пиксель выходной карты.

После нескольких чередующихся свёрточных и подвыборочных слоёв, завершающей частью структуры свёрточной нейросети является полносвязный слой, представляющий из себя обычный многослойный перцептрон. Из карт последнего подвыборочного слоя формируется входной вектор для многослойного перцептрона, и выполняется обычный проход по сети.

## Глава 3: Реализация

В качестве языка для реализации было решено выбрать Python 3.6 ввиду очень простого синтаксиса и наличия большого количества библиотек для машинного обучения.

### 3.1: Построение диаграммы Вороного

Подаваемое на вход программе изображение должно быть предварительно обработано: а именно, по нему должна быть построена диаграмма Вороного. Реализация самой диаграммы с использованием алгоритма Форчуна на Python представлена в [4].

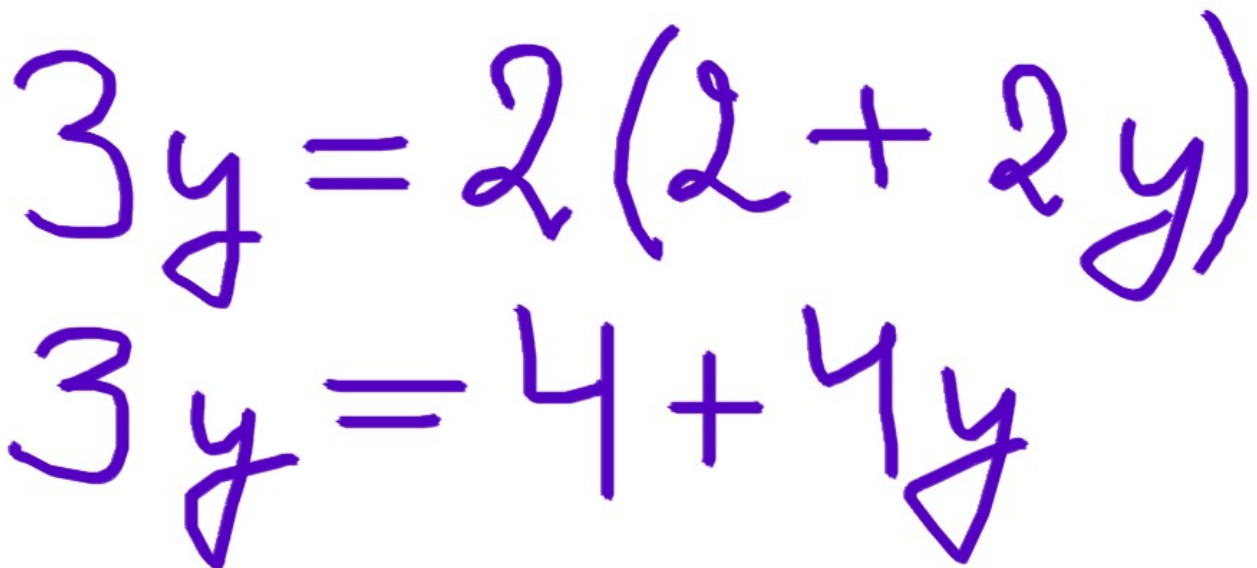

$$3y = 2(2 + 2y)$$
$$3y = 4 + 4y$$

Рисунок 4: Пример входного изображения (СЛУ)

Однако же, данный алгоритм рассчитан на работу с конечным числом точек, которые будут являться точками-генераторами, или, по другому, центрами ячеек Вороного. Для получения этого конечного числа точек используется метод островов: изображение последовательно обходится

попиксельно. При встрече «острова» (пикселя изображения, интенсивность черного цвета которого выше заранее определенного порога), осуществляется рекурсивный обход этого пикселя, а он удаляется с изображения.

Далее, по полученным N множествам точек необходимо построить «центры масс» точек, то есть усредненное значение всех значений осей абсцисс и ординат. Таким образом, модель полученного на вход изображения представляет собой N точек.

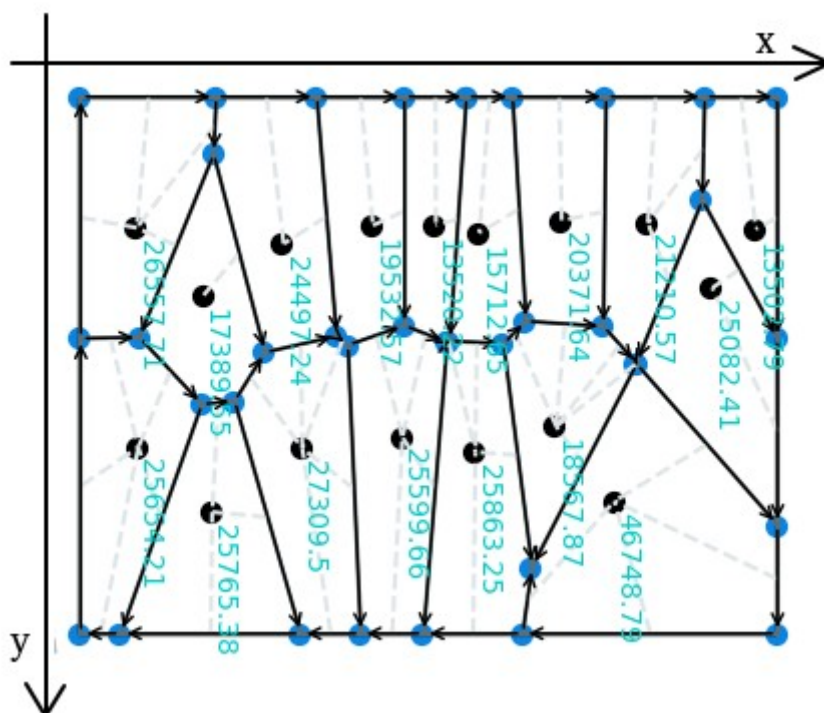


Рисунок 5: Построенная диаграмма Вороного

После этого, используются алгоритмы, описанные в статье [2]. Для каждой ячейки Вороного определяются соседи. Рекурсивно обходя эти ячейки, можно выделять отдельные выражения с помощью описанных в вышеуказанной статье метрик.

У границ каждой ячейки Вороного можно выделить свои точки- вершины. Эти точки ограничивают области изображения, содержащие отдельный символ. Таким образом, для каждого выражения известен набор областей изображения, на которых присутствует отдельный символ.

Основная сложность здесь заключается в том, что нейронная сеть принимает на вход изображение фиксированного размера, тогда как полученное на данном этапе изображение может иметь любой размер и пропорции, равно как и его элементы. Было решено дополнять изображение «пустыми» (интенсивность черного равна 0) пикселями до квадратной размерности, и далее программно сжимать до необходимого размера.

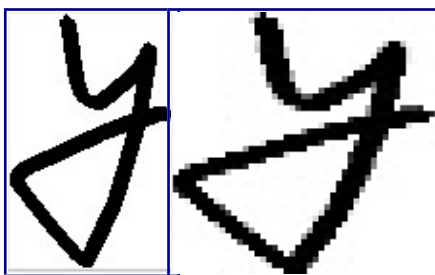


Рисунок 6: Буква «у» до и после сжатия соответственно

Остается лишь последовательно передать символы каждого выражения в заранее натренированную сеть и получить в ответ набор классов — программное представление распознанных символов.

### 3.2: Обучение свёрточной нейронной сети

**Keras** — фреймворк для машинного обучения. Его ключевыми особенностями является простота синтаксиса (очень многое делается самим фреймворком и доступно «из коробки», то есть, в заранее готовом и настроенном виде, что очень хорошо для быстрого старта) и то, что в качестве back-end'a предлагается на выбор одна из нескольких популярных библиотек. То



есть, другими словами, Keras — удобная обертка над настоящими фреймворками, и предоставляет пользователю упрощенное API. В работе используется API на Python. Среди предоставляемых back-end реализаций будет использоваться TensorFlow.

В качестве тестовой коллекции было решено взять датасет [7], включающий в себя цифры, буквы латинского алфавита, а так же наиболее часто встречающиеся математические символы, буквы греческого алфавита и функции (логарифм, синус и так далее).

Для классификации были выбраны следующие 47 математических символа (далее «классы»): числа от 0 до 9, буквы латинского алфавита от «a» до «z», а также знаки «+», «-», «=», «>», «>=», «<», «<=», «(«, «)», «\*», «/».

В качестве программной реализации диаграммы Вороного использован проект [4]. Данная реализация строит необходимую диаграмму по набору заданных точек-генераторов, позволяя посмотреть получаемый на каждом шаге результат. Это может быть очень удобно на раннем этапе построения модели.

В качестве нейросети, распознающей элементы математического выражения, по описанным во 2 главе причинам, выбрана свёрточная нейронная сеть.

Экспериментальным путём была выбрана следующая архитектура этой сети:

1. Входной слой размерности 45x45x1 (чёрно-белые изображения)
2. Свёрточный слой с 32 фильтрами, каждый размерности 7x7, с шагом 2 по горизонтали и вертикали. Активационная функция — сигмоида
3. MaxPooling слой с ядром 2x2 и шагом 2 по горизонтали и вертикали

4. Свёрточный слой с 64 фильтрами, каждый размерности 5x5, с шагом 1 по горизонтали и вертикали. Активационная функция — LeakyReLU со значением параметра 0.1
5. MaxPooling слой с ядром 2x2 и шагом 1 по горизонтали и вертикали
6. Свёрточный слой со 128 фильтрами, каждый размерности 3x3, с шагом 1 по горизонтали и вертикали. Активационная функция — тангенс
7. MaxPooling слой с ядром 2x2 и шагом 1 по горизонтали и вертикали
8. Полносвязный слой, равный по размеру числу оставшихся пикселей у изображения после трёх свёрток. Является входным для последующей полносвязной сети
9. Полносвязный слой с 2000 нейронами. Активационная функция — тангенс
10. Полносвязный слой с 1500 нейронами. Активационная функция — LeakyReLU с параметром 0.1
11. Полносвязный слой с 1000 нейронами. Активационная функция — LeakyReLU с параметром 0.1
12. Выходной слой, равный по размерности числу классов, распознаванию которых будет обучаться нейронная сеть

Преимущества использования LeakyReLU перед обычным ReLU в том, что она решает проблемы «мёртвых» нейронов. При значении  $x < 0$  такая функция выдает не 0, а  $\alpha \cdot x$ , где  $\alpha$  — параметр функции.

Была произведена равномерная выборка данных тестовой коллекции, в результате которой получилось множество из 47242 файлов-изображений. Далее изображения были переведены в csv формат: первое значение в строке — класс изображения, оставшиеся  $n \cdot m$  — интенсивность черного цвета, от 0 до 255, где  $n$  — высота картинка,  $m$  — ширина (для данной коллекции присущи изображения с  $n=m=45$  пикселей).

Далее, данные были перемешаны между собой, чтобы на вход нейросети последовательно подавались примеры разных классов.

Обучение сети проходило с использованием метода «стохастического градиентного спуска», в качестве функции потерь выбрана «categorical cross-entropy»; количество эпох обучения — 20.

В результате обучения удалось добиться точности в 93,79%.

### **3.3: Распознавание**

#### **3.3.1: Первая попытка распознавания**

На примерах изображений СЛУ построение диаграммы Вороного работает со 100% точностью, при условиях, что никакие пиксели соседних символов не соприкасаются друг с другом. Система линейных уравнений успешно делится на строки, а строки делятся на отдельные символы.

Обученная же на представленном тестовом множестве [7] свёрточная нейросеть, несмотря на высокую точность, полученную в процессе обучения, смогла распознать не более чем 70% символов. Она хорошо справляется с распознаванием цифр, но плоха в случае, если ей подадут на вход математический знак или латинскую букву.

```
[[ '3', 'y', 'y', '4', 'd', '4', 'y'], [ '3', 'y', 'y', '2', 'less', 'x', 'plus', '2', 'y', '2' ]]
```

Рисунок 7: Вывод программы для примера, изображенного на рисунке 4 (правильно распознаны лишь 11 символов из 17)

Скорее всего, данное поведение обусловлено различными почерками, представленным в тестовой коллекции и при написании СЛУ. Для достижения более высокого результата требуется коллекция с более разнообразным составом символов.

Разберем на различных примерах работу программы.

### Пример 1.

$$2 + x = 5$$

Рисунок 7: Пример 1

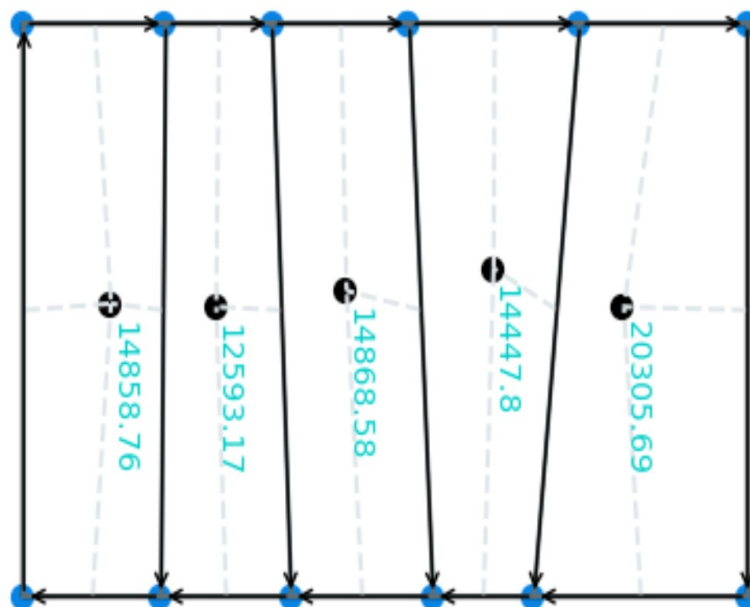
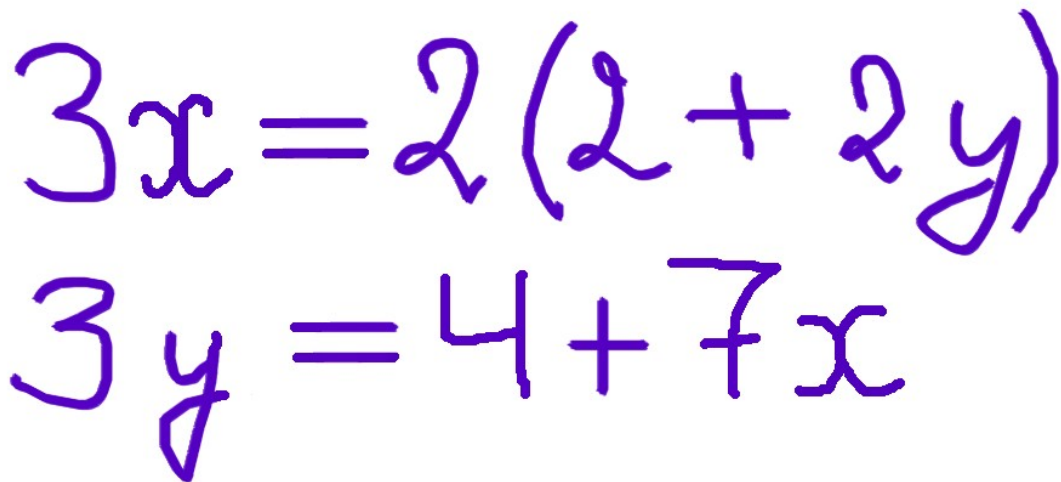


Рисунок 8: Диаграмма Вороного для примера 1

Дано черно-белое изображение, на котором нарисовано простое линейное уравнение. Вывод программы - [['2', 'plus', 'x', 'div', '5']].

Изображение было распознано с точностью в 80%. Единственная сложность, с которой столкнулась нейронная сеть — знак «=», вместо которого сеть предсказала знак деления. Это объясняется тем, что в тестовой коллекции [7] знак деления представлен двумя образами: привычным знаком «/» и мало используемым «÷» (обелюс), с которым нейронная сеть и может путать знак равенства.

**Пример 2.**



The image shows two handwritten mathematical equations in purple ink. The first equation is  $3x = 2(2 + 2y)$  and the second equation is  $3y = 4 + 7x$ . The handwriting is cursive and somewhat informal.

Рисунок 9: Пример 2

Дано цветное (имитация синей шариковой ручки для символов) изображение, на котором нарисована система линейных уравнений. Цвет не играет роли в данной задаче, так как в ходе работы рисунок приводится к черно-белому, важна лишь интенсивность цвета.

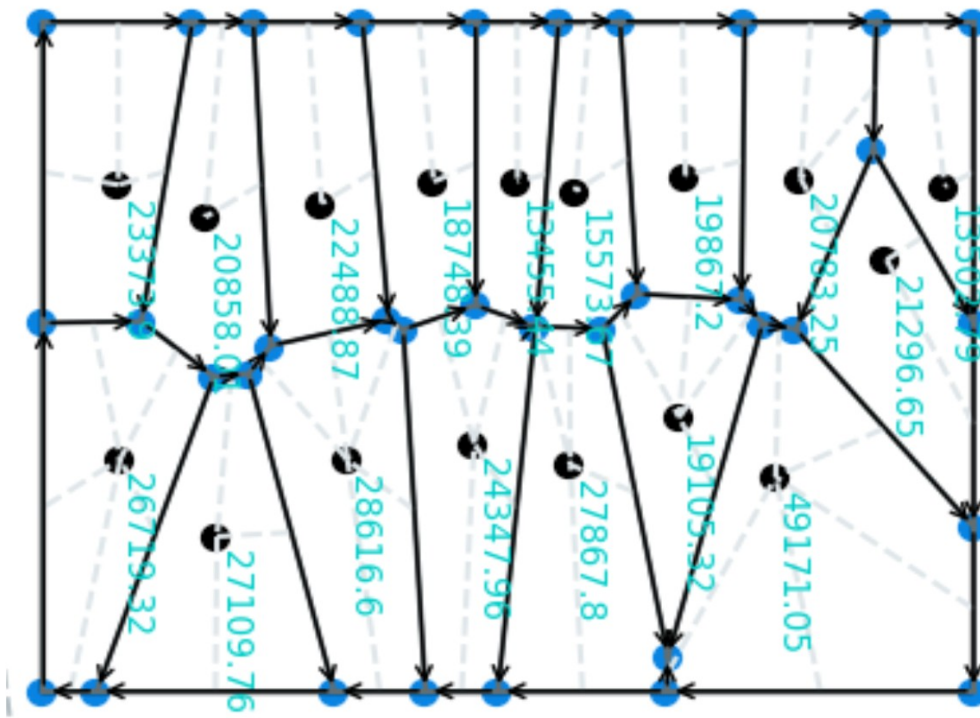


Рисунок 10: Диаграмма Вороного для примера 2

Вывод программы - [['3', 'x', 'div', '2', 'l', '2', 'p', '2', 'y', 'l'], ['3', 'q', 'div', '4', 'k', '7', 'x']].

Для первого уравнения «проблемными» оказались символы «(» и «)». Они очень похожи на латинскую букву «l», поэтому нейросеть сделала ложное предсказание. Во втором уравнении рукописные «q» и «u» имеют много общего, и система спутала их.

Кроме того, для обоих уравнений, ровно как и для первого примера, неверно был распознан символ равенства.

Суммарный результат по примеру — 10/17 символов.

### 3.3.2: Вторая попытка распознавания

После анализа ошибок решено было доработать программу:

- символы должны располагаться по центру результирующих изображений, которые подаются в нейросеть, так же, как на изображениях тестовой коллекции
- в тестовой коллекции ширина линии каждого символа была очень мала, а то время как на приведенных выше примерах использовалась толстая линия

Первая проблема решилась следующим образом -- изображение символа попиксельно переносится на чистое изображение квадратной размерности по следующим правилам:

1. Пусть  $n$  — ширина исходного изображения с символом,  $m$  — высота
2. Считаём расстояние отступа символа от края изображения, как:  
$$l = (m+n)/2$$
3. Создаём пустое изображение размерности  $d = \max(x_{diff}, y_{diff})$ , где  $x_{diff}, y_{diff}$  - разница расположения минимальных и максимальных значений координат пикселей символа на исходном изображении
4. Переносим символ на пустое изображение:
  - a. Если  $y_{diff} > x_{diff}$ , то переносим изображение со сдвигом « $l$ » по оси « $x$ »
  - b. Если  $y_{diff} \leq x_{diff}$ , то переносим изображение со сдвигом « $l$ » по оси « $y$ »



Рисунок 11: Сравнение финальных изображений символа «(» до и после изменения

### Пример 3

$$3x + 5y = 0$$
$$6x = 4(1 - 8y)$$

Рисунок 12: Пример 3

Дано изображение СЛУ, аналогичное тому, что было приведено во втором примере. При написании системы была использована меньшая ширина кисти, так что линии получились тоньше.

Вывод программы после описанных выше доработок:

```
[[ '3', 'x', 'plus', '5', 'y', 'equal', '0'],  
 [ '6', 'x', 'equal', '4', 'left_bracket', '1', 'minus', '8', 'y', 'right_bracket'],]
```

Как видно из вывода, точность значительно возросла и равняется 100%.



## Выводы

Описанный в работе подход имеет место быть. Построение скелета в виде диаграммы Вороного — это хороший способ разделять на выражения систему линейных уравнений, написанных от руки. Также этот подход отлично справляется с разделением каждого выражения на символы.

В процессе распознавания полученных изображений с отдельными символами столкнулись с несколькими зависимостями: от ширины линии шрифта и от положения символа на результирующем изображении. Впоследствии, успешно решили эти проблемы и привели систему к работоспособному виду.

Распознав результирующие изображения символов, убеждаемся в эффективности свёрточной нейронной сети для распознавания изображений.

## Заключение

В данной работе был рассмотрен подход к распознаванию математических уравнений и их приведению к удобочитаемому виду. В процессе были решены следующие задачи:

- найдена и обработанная тестовая коллекция, содержащая математические символы, цифры и буквы латинского алфавита
- на тестовых данных обучена свёрточная нейронная сеть
- найден и применён алгоритм построения диаграммы Вороного алгоритмом Форчуна
- алгоритмически произведено разделение изображения на математические выражения, и каждое из них — на набор символов

- каждый из символов распознан обученной нейросетью

## Использованные материалы

- [1] Masalovitch A., Mestetskiy L. Usage of continuous skeletal image representation for document images de-warping //Proceedings of International Workshop on Camera-Based Document Analysis and Recognition, Curitiba. – 2007. – С. 45-53.
- [2] Запрягаев С. А., Сорокин А. И. Сегментация рукописных и машинописных текстов методом диаграмм Вороного //Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2010. – №. 1. – С. 160-165.
- [3] Ondrej M., Frantisek V. Z., Martin D. Algorithmic and mathematical principles of automatic number plate recognition systems //Brno University of technology. – 2007. – Т. 10.
- [4] <https://github.com/Yatoom/voronoi> - An implementation of Fortune's algorithm in python
- [5] <https://www.photomath.net/en/> - Photomath
- [6] Pedamonti D. Comparison of non-linear activation functions for deep neural networks on MNIST classification task //arXiv preprint arXiv:1804.02763. – 2018.
- [7] <https://www.kaggle.com/xainano/handwrittenmathsymbols> — Handwritten math symbols dataset
- [8] <https://github.com/Ielay/diploma> — Код проекта на Github