МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка алгоритма распознавания текста на изображении при помощи сверточной нейронной сети»

Студент А.А. Борисов
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н, В.С. Климов
(ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко

Тольятти 2020

АННОТАЦИЯ

Темой бакалаврской работы является «Разработка алгоритма распознавания текста на изображении при помощи сверточной нейронной сети».

Объектом работы является процесс функционирования нейронных сетей.

Предметом выпускной квалификационной работы является алгоритм распознавания текста изображении.

Целью работы является разработка алгоритма распознавания текста на изображении.

Для достижения поставленной цели были выдвинуты следующие задачи:

- Изучить существующие нейронные сети.
- Выбрать нейронную сеть для создание модели и изучить ее строение.
- Реализовать выбранную нейронную сеть.
- Обучить выбранную нейронную сеть.
- Преобразовать входные данные для подачи их в модель нейронной сети.

Выпускная квалификационная работа состоит из введения, трех глав, заключения и списка литературы.

В ведении описывается актуальной данной работы.

В первой главе описываются понятия биологического и искусственного нейрона, а также теоритические сведения о существующих на данные момент нейронных сетях и их применении.

Во второй главе описывается теоритическая схема работы алгоритма, выбор средств его программной реализации и сама программная реализация системы.

В третьей главе проводится тестирование разработанной системы при различных входных данных. Также приводятся таблицы и графики о работе разработанной системы.

В заключение вынесены выводы о проделанной работе.

Данная выпускная квалификационная работа содержит в себе пояснительную записку, состоящую из 42 страниц, 28 рисунков, 3 таблиц, 1 графика, 2 формул и списка литературы из 20 источников.

ABSTRACT

The title of the graduation work is *Development of a text recognition algorithm on an image using a convolutional neural network*.

This graduation work is devoted to developing a algorithm capable of converting a text written in an image into a printed text by using neural networks.

The research dwells on details of using neural networks for converting a text presented in an image into a printed text.

The graduation work consists of an explanatory note, introduction, 28 figures, 3 tables, the list of 20 reference including 5 foreign sources and one appendix.

The key issue of the graduation work is to check the effectiveness of the developed text recognition algorithm.

The aim of the research is to develop a system for recognizing a text presented in an image.

The object of the research is the process of neural system functioning.

The subject of the graduation work is the recognition algorithm of the text presented in the image.

The graduation work may be divided into several logically connected parts that describe various types of the neural networks, the text recognition algorithm, its implementation and testing.

Overall, the results suggest that the developed algorithm is able to recognize a text in an image but it needs to be improved.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире все чаще встречается необходимость автоматического распознавания текста с фотографий, картинок или видео. Будь то перевод рукописного текста на другой язык или же просто быстрая оцифровка информации с бумажных носителей – так или иначе, данная возможность значительно облегчает жизнь человека.

Реализовать алгоритм распознавания текста возможно с помощью искусственных нейронных сетей. Нейронная сеть — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей. По сути, в конечном итоге мы получаем продукт, имитирующий человеческое мышление.

Актуальность данной работы состоит в разработке нового алгоритма распознавания текстов, который исправит несовершенства существующих систем, а также сократит объем используемых ресурсов и затраченного времени.

Новизна бакалаврской работы состоит в разработке нового алгоритма распознавания текста на изображении.

Практическая ценность бакалаврской работы заключается в необходимости совершенствования методов распознавания текста, в уменьшении количества затрачиваемых на это ресурсов. Возможность автоматизировать огромное количество ручного монотонного труда несет немалую практическую ценность.

Объектом исследования является процесс функционирования алгоритмов распознавания текстов на изображении.

Предметом исследования является алгоритм распознавания текстов на изображении.

Цель исследования заключается в разработке нового алгоритма распознавания текстов на изображении.

Для достижения цели поставлены следующие задачи:

- Изучить существующие нейронные сети.
- Разработать алгоритм распознавания текста на изображении.
- Выбрать нейронную сеть для создание модели и изучить ее строение.
- Реализовать выбранную нейронную сеть.
- Обучить выбранную нейронную сеть.
- Преобразовать входные данные для подачи их в модель нейронной сети.
- Протестировать алгоритм на примерах реальных текстовых изображений.

Бакалаврская работа состоит из введения, трех глав, заключения и списка используемой литературы.

В первой главе рассматривается биологическое понятие нейрона, принципы организации искусственных нейронных сетей, виды известных искусственных нейронных сетей и применение каждого из них в современном мире.

Во второй главе рассмотрены общие принципы организации алгоритмов распознавания текстов на изображении, осуществлен и обоснован выбор средств для реализации алгоритма и описана непосредственно его разработка.

В третьей главе осуществлено тестирование разработанного алгоритма, подведены итоги его эффективности и продуктивности.

В заключении подводятся общие итоги исследования, формируются и описываются выводы и результаты проделанной работы.

Глава 1 РАЗНОВИДНОСТИ НЕЙРОННЫХ СЕТЕЙ И ИХ ПРИМЕНЕНИЕ

1.1 Биологическое понятие нейрона и искусственные нейронные сети

В человеческом мозге находятся миллиарды нейронов – узко специализированных клеток, предназначенных для приема извне, обработки, хранения, передачи и вывода наружу информации с помощью электрических и химических сигналов.

Строение нейрона очень просто. Нейрон состоит из тела клетки, дендритов и аксона. Тело клетки состоит из протоплазмы, которая ограничена снаружи мембраной из липидного слоя. Липиды состоят из гидрофильных головок и гидрофобных хвостов. Липиды располагаются гидрофобными хвостами друг к другу, образуя гидрофобный слой. Также тело нейрона содержит отростки, называемые аксонами и дендритами. Аксонами являются длинные отростки нейрона, которые приспособлены для передачи информации от тела нейрона к нейрону, либо от нейрона к исполнительному органу. Дендритами являются короткие и сильно разветвленные отростки нейрона, главной задачей которых является образование влияющих на нейрон возбуждающих и тормозящих синапсов – мест контакта между двумя нейронами (исполнительным органом).

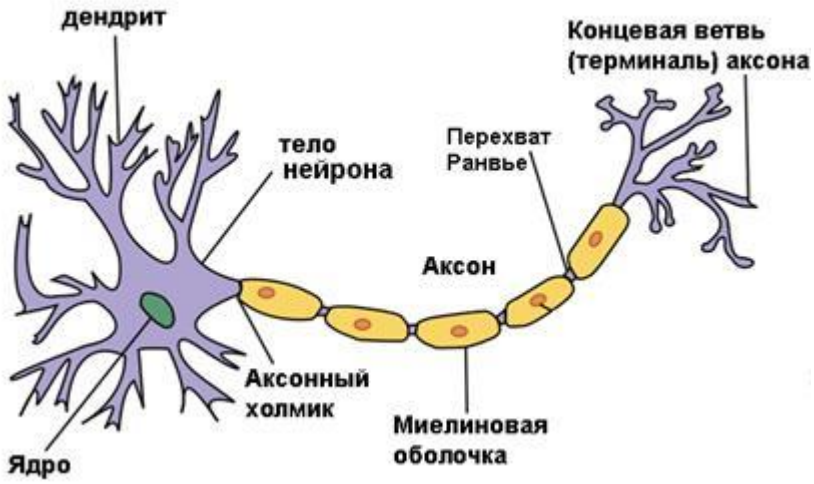


Рисунок 1.1 – Строение нейрона

На основании числа и расположения элементов нейрона учены классифицировали их на несколько видов по структурному и функциональному признаку. По структурному признаку нейроны делятся на:

- Безаксонные – небольшие клетки нейрона, сгруппированные вблизи спинного мозга и не имеющие анатомических признаков разделения отростков на аксоны и дендриты. Функциональное назначение данных нейронов слабо изучено.
- Униполярные – нейроны с одним отроском.
- Биполярные – данный вид нейронов имеет один аксон и один дендрит. Такие нейроны встречаются в специализированных сенсорных органах, например: сетчатке глаза, обонятельном эпителии и луковице, слуховом и вестибулярном гениталиях.
- Мультиполярные – нейроны с одним аксоном и несколькими дендритами. Данный вид нейронов преобладает в центральной нервной системе.
- Псевдоуниполярные – уникальный вид нейронов, у которого ведь единый тракт покрыт миелиновой оболочкой и структурно представляет аксон.

По функциональному признаку нейроны делятся на:

- Афферентные – данный тип нейронов можно отнести к сенсорным нейронам. К данному типу нейронов относятся первичные клетки органов чувств.
- Эфферентные –такой тип нейронов можно назвать двигательным. К нейронам данного типо относятся конечные нейроны.
- Ассоциативные – данный тип нейронов осуществляет связь между эфферентными нейронами и афферентными.
- Секреторные – нейроны, секретирующие высокоактивные вещества, у которых хорошо развит комплекс Гольджи.

На основе нейронов человеческого мозга была составлена искусственная нейронная сеть. Искусственная нейронная сеть – это математическая модель, в том числе ее программное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей живого организма. Главным понятием искусственных нейронных сетей является искусственный нейрон, который в своем роде является сумматором всех входящих в него сигналов. Строение искусственного нейрона показано на рисунке 1.2

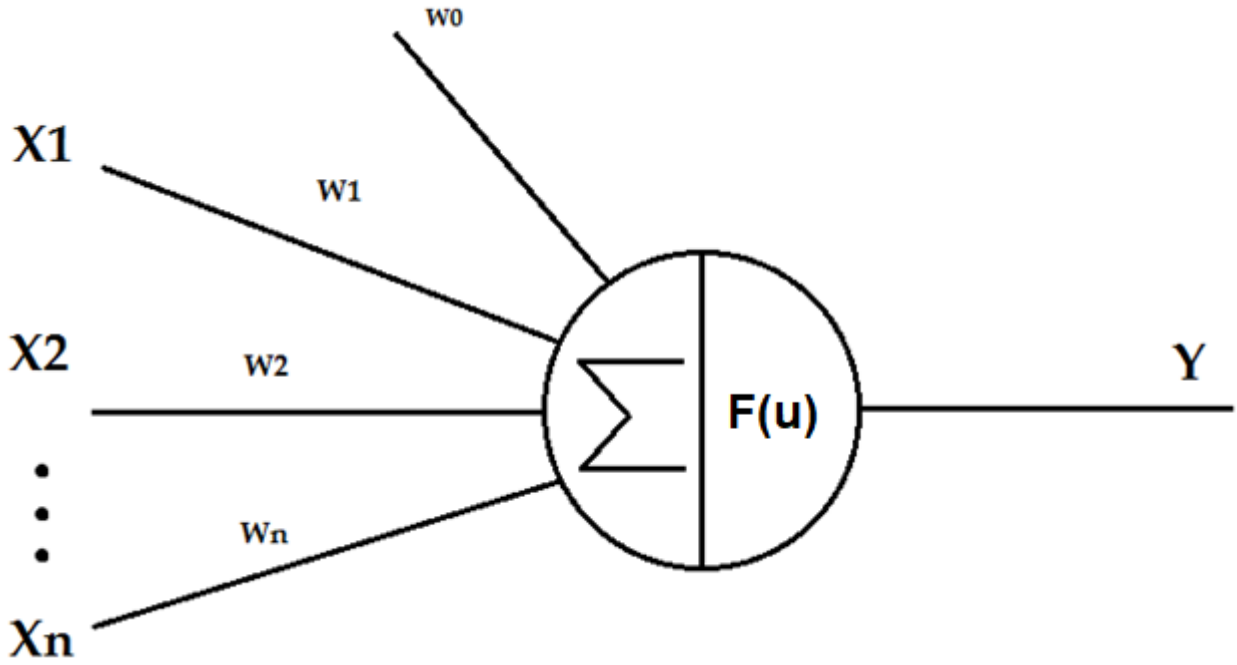


Рисунок 1.2 – Строение искусственного нейрона

На вход искусственного нейрона подается какой-либо набор данных, обозначенный на рисунке как X1, X2, … , Xn. Далее входные данные преобразуются с помощью весов, обозначенных на рисунке как w1,w2, … , wn и передаются в сумматор, обозначенную на рисунке знаком ∑. Формула сумматора представлена формулой 1:

$$u = \sum_{i=1}^{n} w_0 * x_0 + w_i * x_i, \quad (1)$$

Где,

u – значение сумматора,

w – вес,

x – входные данные.

Далее значение сумматора передается передаточной функции или по-другому функции активации F(u). Данные функции имеют различные формулы и именно от них зависит выходное значение нейрона.

1.2 Виды нейронных сетей и их применение

Базовыми архитектурами нейронных сетей являются сети прямого распространения и персептроны. Данные нейронные сети очень прямолинейны. Информация по ним передается от входа сразу к выходу. Клетки слоя данных сетей не связаны между собой, в отличии от соседних слоев, которые обычно полностью связаны. Нейронные сети прямого распространения обычно обучаются по методу обратного распространения ошибки. При таком обучении нейронная сеть получает множество данных как на вход, так и на выход. Этот процесс называется обучением с учителем.

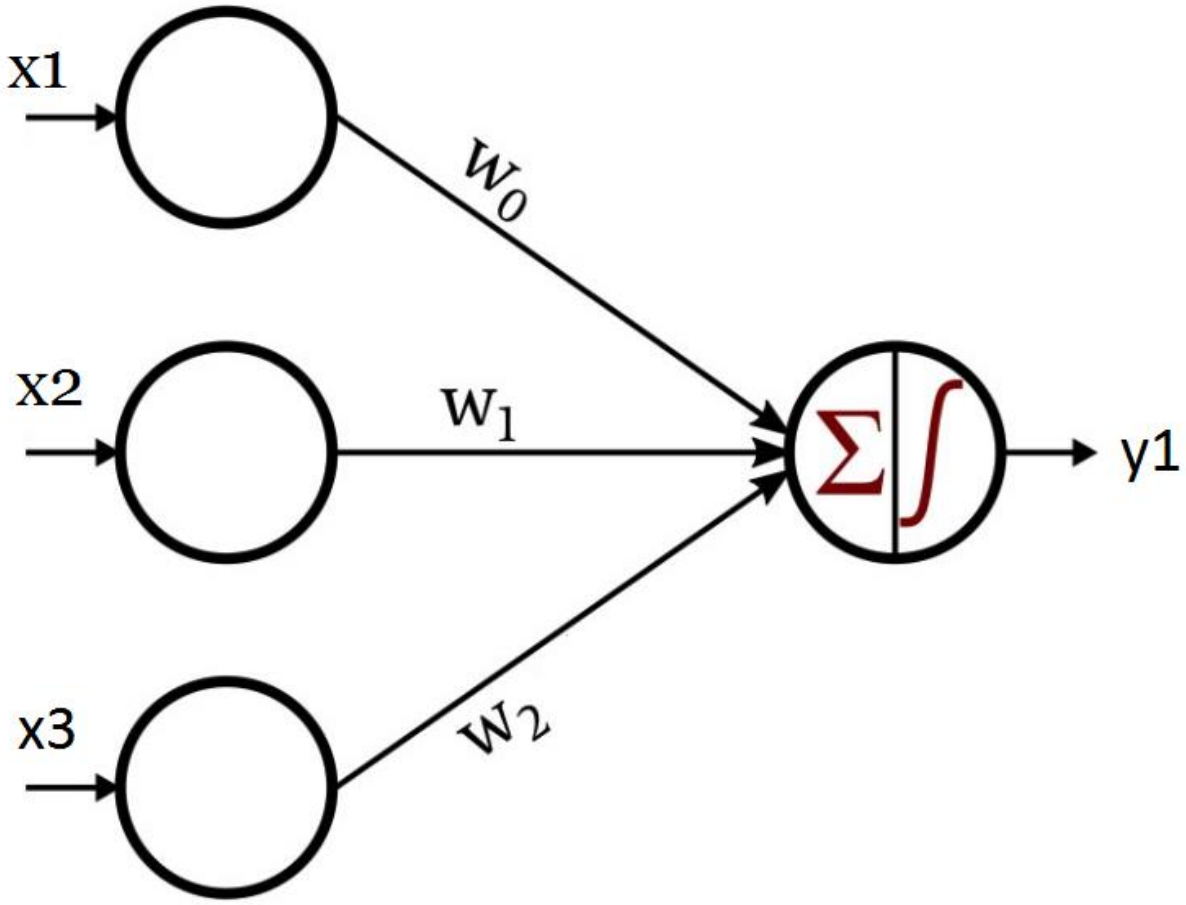


Рисунок 1.3 – Пример строения персептрона

Еще одной разновидностью базовых нейронных сетей является нейронная сеть Хопфилда. Данная нейронная сеть используется для восстановления изображений. Нейронная сеть Хопфилда является полносвязной нейронной сетью с симметричной матрицей связей. Понятие полносвязной означает, что каждый нейрон передает свой выходной сигнал остальным нейронам, включая самого себя. Количество нейронов данной сети определяется количеством входов и выходов. Также нейронная сеть Хопфилда обучается только с учителем, так как для восстановления изображения ей необходимо знать исходные значения изображения, которое необходимо восстановить.

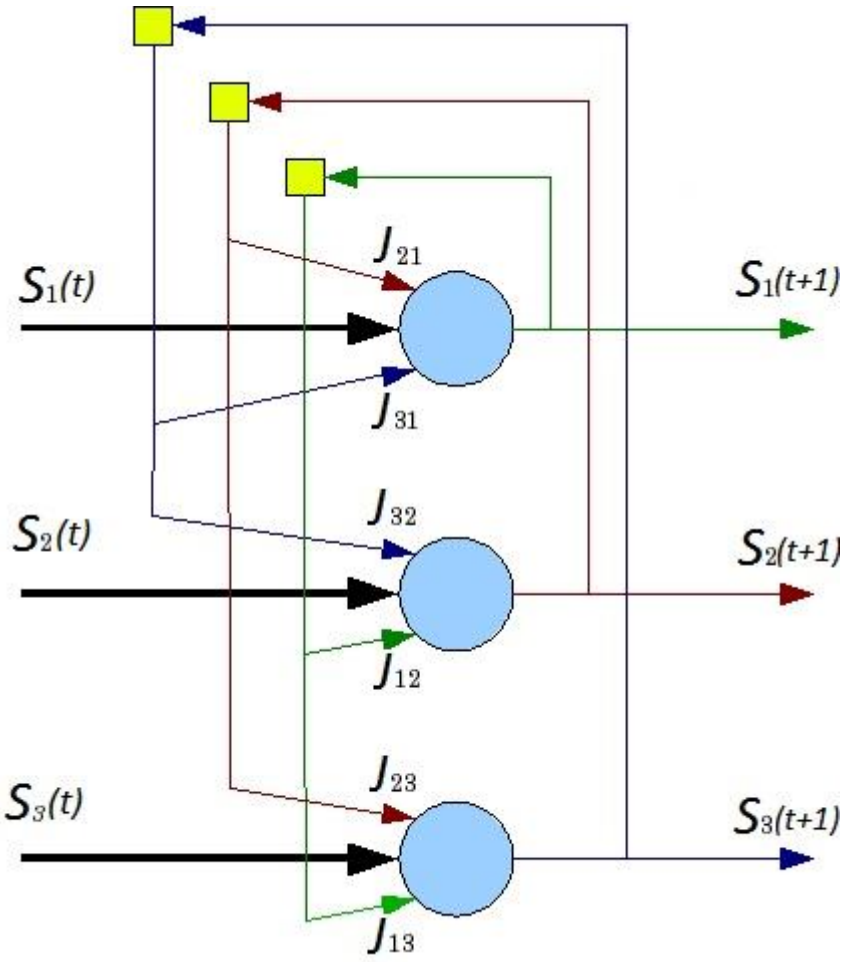


Рисунок 1.4 – Пример строения нейронной сети Хопфилда

Еще одним видом нейронной сети является Машина Больцмана. Чаще всего данную нейронную сеть рассматривают как стохастический

генеративный вариант нейронной сети Хопфилда. Машина Больцмана используется для обучения алгоритм имитации отжига. Также данная нейронная сеть оказалась первой нейронной сетью, способной обучаться внутренним представлениям и решать сложные комбинаторные задачи.

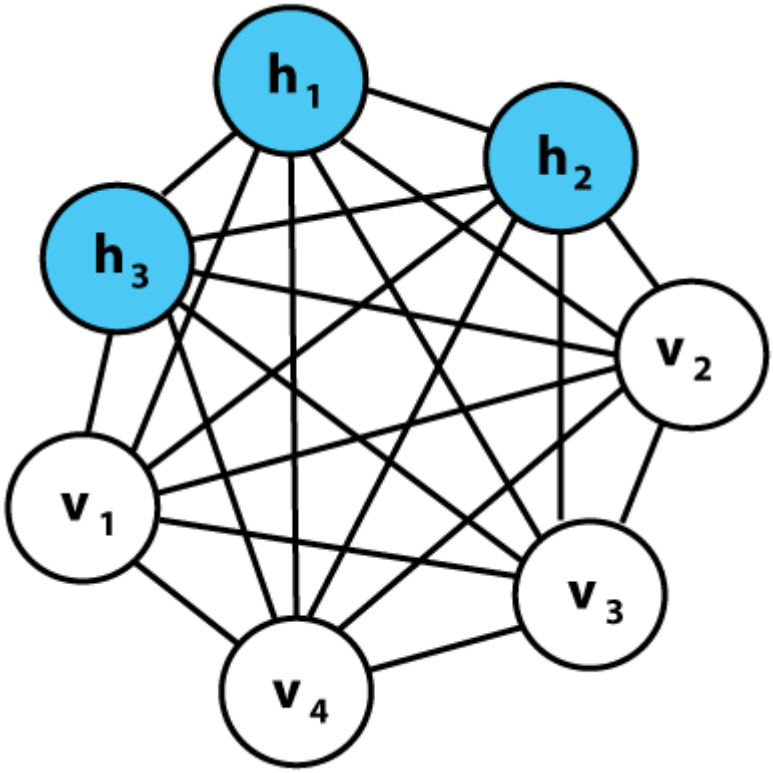


Рисунок 1.5 – Пример строения Машины Больцмана

Также существуют свёрточные нейронные сети. Данный вид нейронных сетей сильно отличается от остальных. Свёрточные нейронные сети обычно используют для классификации изображений. Главной особенностью данной нейронной сети является «сканер», который считывает изображение кусками, и процесс «свертки», который уменьшает размер матрицы признака изображения.

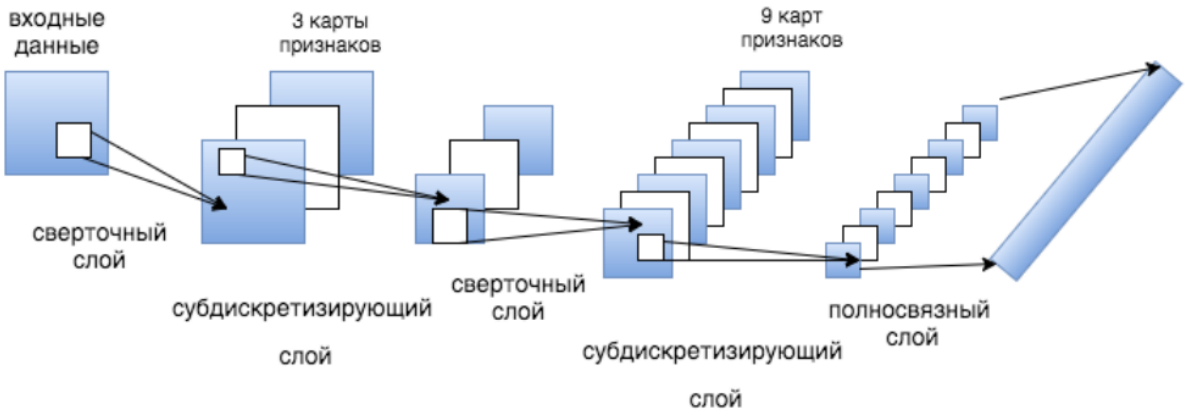


Рисунок 1.6 – Пример строения свёрточной нейронной сети

К более продвинутым конфигурациям нейронных сетей относятся рекуррентные нейронные сети. Особенностью этих нейронных сетей является то, что они получают информацию не только с предыдущего слоя, но также и от самих себя с предыдущего прохода. Из-за этой особенности становится важным порядок подачи данных для обучения нейронной сети. Сложностью в использовании нейронных сетей такого типа является проблема исчезающего градиента, которая определяется быстрой потерей информации с течением времени. Данная проблема влияет только на веса при обучении модели. Чаще всего нейронные сети такого типа используют для автоматического дополнения информации.

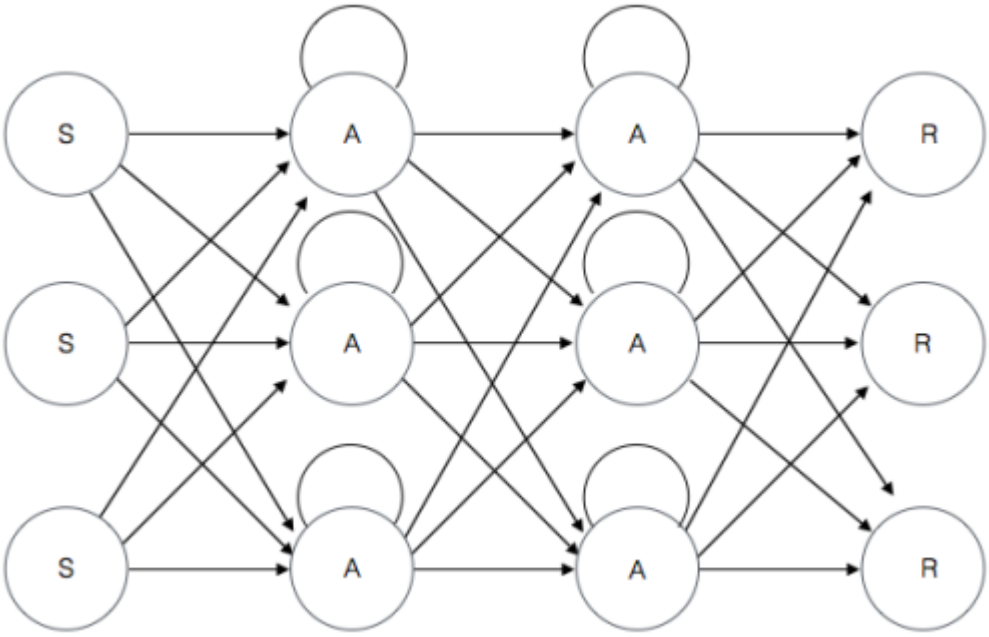


Рисунок 1.7 – Пример рекуррентной нейронной сети

Нейронные сети Кохонена, также именуемые как самоорганизующиеся карты, применяют соревновательное обучение для классификации данных без учителя. Для данной нейронной сети подаются входные данные, которые нейронная сеть сама распределяет по нейронам, которые совпадают с ними в большей степени. После чего нейроны меняются для большей точности совпадения, двигая своих соседей.

В результате рассмотрения понятия нейрона было изучено строение и функционал биологического нейрона, который состоит из тела клетки, дендритов и аксонов и служит для приема, обработки и передачи информации. Также было рассмотрено строение искусственного нейрона, который является математической моделью биологического нейрона и выполняет схожие функции.

Кроме рассмотрение искусственного нейрона в отдельности, были рассмотрены виды нейронных сетей – совокупности искусственных нейронов. Каждая искусственная нейронная сеть, описанная в данной главе отличается как строением, так и сферой применения.

Глава 2 ПРОЕКТИРОВАНИЕ АЛГОРИТМА РАСПОЗНАВАНИЯ ТЕКСТА НА ИЗОБРАЖЕНИИ

2.1 Описание работы алгоритма распознавания

Основными этапами для реализации программы были выделены следующие:

1) Обучение нейронной сети для распознавания букв английского алфавита.

2) Определение на изображении букв.

3) Разбить изображение на части с буквами.

4) Распознать на отдельных изображениях находящиеся на них буквы.

5) Составить из полученных с изображений букв слово.

6) Расставить пробелы между разными словами.

Для обучения модели была выбрана свёрточная нейронная сеть, которая была обучена различать рукописные английские буквы и цифры. Выбор свёрточной нейронной сети обусловлено тем, что на данный момент у данной разновидности нейронных сетей один из лучших алгоритмов распознавания и классификации изображений. По сравнению с полносвязной нейронной сетью у нее гораздо меньше количество настраиваемых весов. Главной особенностью свёрточной нейронной сети является «свертка». Процесс «свертки» представляет собой уменьшение размера матрицы признаков входного изображения. Для получения ячейки матрицы уменьшенного размера элементы исходной матрицы в определенной области умножают на вес с последующим суммированием всех элементов в этой области. Чтобы получить следующую ячейку уменьшенной матрицы происходит сдвиг области и выполнение тех же действий.

Описанную выше последовательность действий можно записать формулой:

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} \times I_{x+i-1,y+k-1}\ , \qquad (2)$$

где I – исходная матрица признака,

K – матрица веса,

x и y – индексы выбранного блока

h и w – высота и ширина.

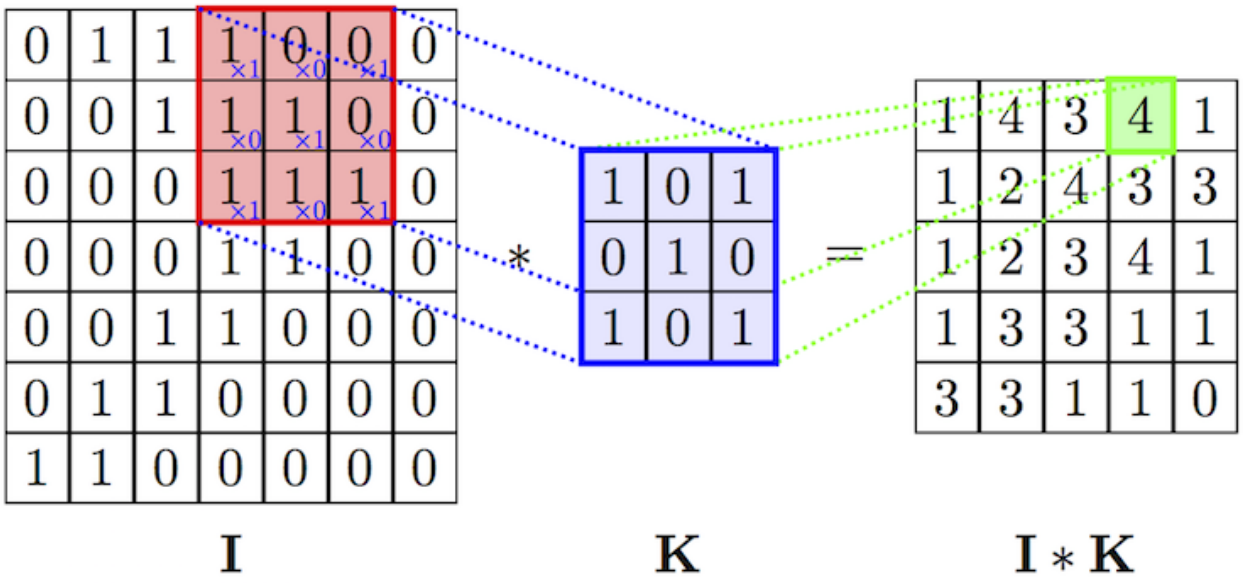


Рисунок 2.1 – Процесс «свертки»

На примере, представленном выше показан процесс свертки матрицы признака размерностью 7*7 =49 признаков, которая обозначена буквой I. «Сетка» входной матрицы I размером 3 на 3 «клетки» умножается поэлементно на элементы матрицы весов K, после чего полученные значения матрицы суммируются, а полученное значение заносится в клетку выходной матрицы. Затем происходит сдвиг «сетки» и повторение вышеописанных действий.

Обучение данной модели происходило на датасете Extended Modified National Institute of Standards and Technology (EMNIST), который содержит 70000 рукописных вариаций всех английских букв и цифр (A – Z, 0 – 9). Данный датасет был выбран исходя из доступности и вариативности написания необходимых для проекта символов. 60000 изображений из датасета EMNIST используются для обучения модели нейронной сети, а оставшиеся 10000 для тестирования. Также данная база данных является

стандартом, предложенным Национальным институтом стандартов и технологий США. Образцы изображений, находящихся в данной базе были нормализованы и приведены к серому полутоновому изображению размером 28 на 28 пикселей. Средний уровень ошибки при обучении нейронной сети на основе данных из датасета EMNIST равен 0,23%, что является хорошим результатом. Минимальный процент ошибки при обучении с использованием датасета EMNIST был достигнут 0,18% при случайном мультимодульном глубоком обучении (RMDL), когда обучались 30 моделей: 10 Convolutional Neural Network (CNN), 10 Recurrent Neural Network (RNN) и 10 Deep Neural Network (DNN)).

После получения обученной модели нейронной сети, необходимо подать ей на вход данные, а для того чтобы их подать сначала нужно их добыть. Так как полученная модель нейронной сети обучена определять буквы на входном изображении, то передавать на вход нейронной сети нужно тоже быквы.

Для того, чтобы выделить на изображении буквы, необходимо перевести изображение в черно-белые цвета. Для этого сначала изображение переводится в оттенки серого, а затем в черно-белые цвета. После выполнения этих несложных операций можно найти контуры букв для определения границ будущих входных данных. Контуры букв находятся с помощью алгоритма Suzuki85, разработанный японским программистом Сатоси Сузуки.

Алгоритм поиска контуров имеет следующую последовательность действий:

1. находятся две самые удаленные друг от друга точки контура,
2. находится самая удаленная точка контура от отрезка, образованного на предыдущем шаге,
3. находится самая удаленная точка контура от контура, образованного на предыдущем шаге отрезками,

4. повторяется предыдущий шаг, пока не выполнится условие по длине порога.

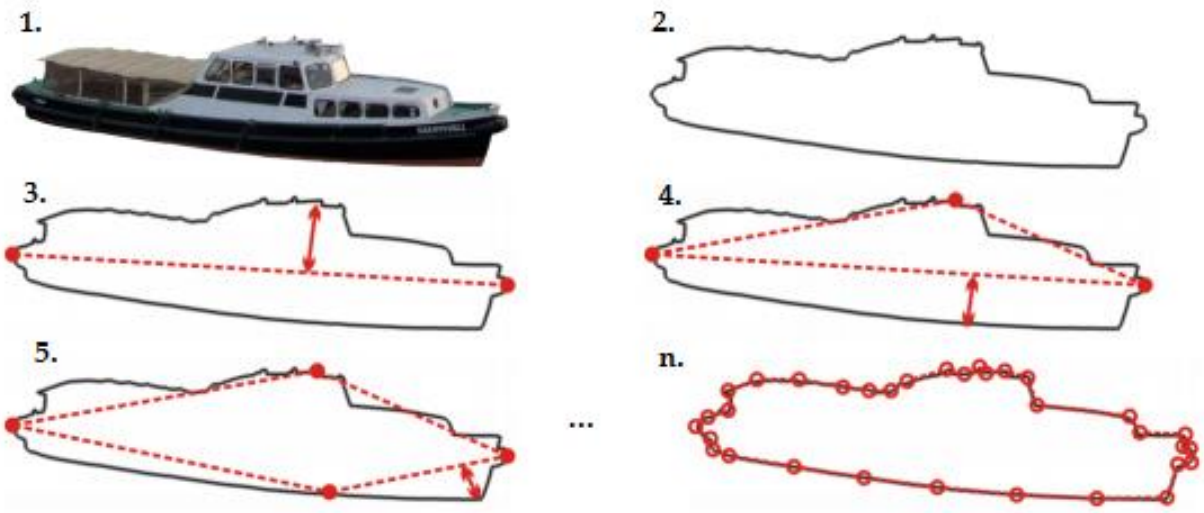
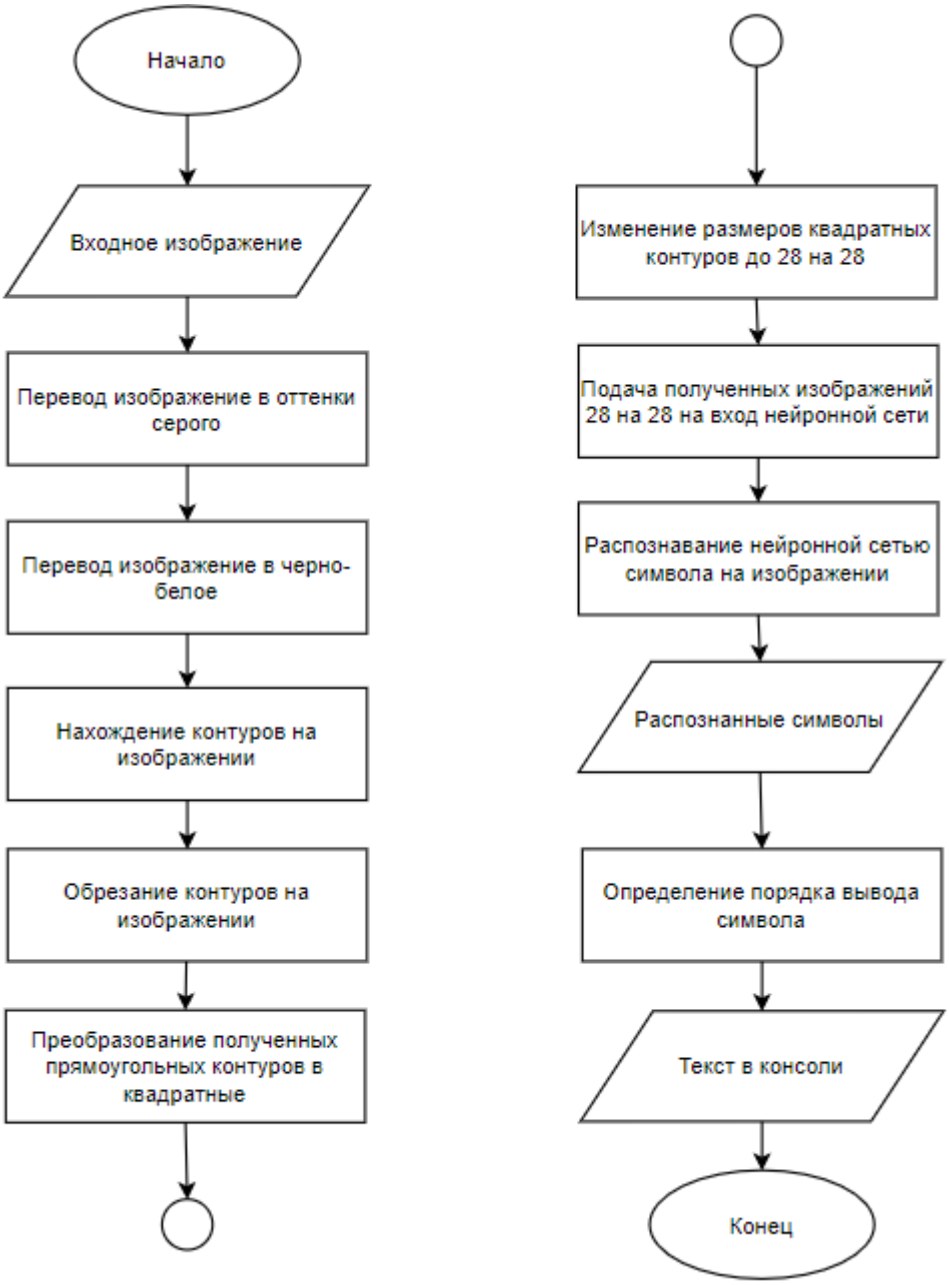


Рисунок 2.2 - Процесс нахождения контура фигуры алгоритмом Suzuki85

После определения контура буквы, данная область вырезается так, чтобы весь контур входил в прямоугольную область. Следующим шагом является изменение размеров изображения до 28 на 28 пикселей для того, чтобы обученная нейронная сеть смогла использовать полученные входные данные.

После изменения размера полученных букв до 28 на 28 пикселей, они передаются на вход нейронной сети, которая определяет принадлежность находящей на входном изображении буквы к буквам английского алфавита. Затем происходит сравнение процентной принадлежности буквы, находящейся на изображении, подающемся на вход, ко всем буквам английского алфавита и выбирается наилучший вариант. Буква, выбранная нейронной сетью, запоминается. После нахождения всех определенных на изображении букв, в консоль выводятся буквы последовательно слева направо, исходя из их координат по оси X. Пробел между буквами ставится в том случае, если длина между последней координатой одного изображения и первой координаты другого изображения по оси X больше или равна трети размера изображения буквы по оси X.

Полученный алгоритм можно представить в виде следующей блок-



схемы:

Рисунок 2.3 – Блок-схема разработанного алгоритма

После разработки алгоритма можно преступить к его программной реализации. Для этого сначала нужно выбрать средства, с помощью которых будет реализовывать разрабатываемый алгоритм.

2.2 Выбор средств для реализации программы

Для того, чтобы разработать любую программу, сначала необходимо выбрать язык программирования, на котором будет она написана.

Существует множество языков программирования и на каждом можно написать эквивалентную программу. Отличиями между эквивалентными программами будут являться структура написания кода и средства, используемые для получения нужного результата. Основными языками программирования, на которых решаются задачи, связанные с нейронными сетями являются:

- Python;
- Java;
- C++;
- Matlab.

Для реализации системы распознавания текста был выбран язык Python, так как он имеет ряд плюсов, выделяющих его среди остальных языков программирования:

- Быстрая разработка;
- Простота в освоении;
- Большое количество библиотек;
- Поддержка на любой платформе;
- И ряд других плюсов.

Быстрой разработке на данном языке способствуют простой синтаксис языка, не содержащий сложных конструкций, а также большое количество библиотек, созданных сообществом, встроенные функции которых уменьшают количество написанного кода.

Программы, написанные на языке Python выполняются на большинстве современных операционных систем, что позволяет использовать программу на различных устройствах, не совершая глобальных изменений в коде.

Еще одним несомненным плюсом является то, что программы, написанные на Python, имеют высокую скорость выполнения. Это связано с тем, что основные библиотеки Python написаны на языке C++ и выполнение задач занимает меньше времени, чем на других языках высокого уровня.

Определившись с языком программирование, можно начинать думать над тем, какие библиотеки будут использоваться в разработке.

Так как программа будет работать с изображениями, то необходимо обязательно подключить библиотеку, содержащую основные функции по работе с ними. Самыми популярными библиотеками, поддерживающие язык программирования Python, для работы с изображениями являются:

- OpenCV;
- NumPy;
- SciPy;
- Pillow.

Для работы с изображениями были выбраны библиотеки OpenCV и NumPy, так как совместно они выполняют все необходимые функции для реализации проекта, а именно:

- Перевод изображения в оттенки серого;
- Перевод изображения в черно-белое;
- Определение контуров на изображении.

Также необходимо создать нейронную сеть, для распознавания изображенных на картинке букв. Для этого было решено использовать библиотеку Tensorflow и надстройку над ней - Keras. С помощью этого сочетания можно создать и обучить нейронную сеть. Функции, встроенные в Keras, позволяют выполнять все необходимые действия для построения нужной модели нейронной сети.

2.3 Реализации программы

После определения последовательности выполняемых действий и средств реализации, можно приступать к написанию программного кода.

Первым делом необходимо обучить модель нейронной сети распознавать на изображении буквы и определять принадлежность к одной из букв алфавита. Для этого необходимо создать функцию и с помощью встроенных в библиотеку Keras методов создать и обучить модель.

```
def TextWriter_model():
    model = Sequential()
    model.add(Convolution2D(filters=32, kernel_size=(3, 3), padding='same', input_shape=(28, 28, 1), activation='relu'))
    model.add(Convolution2D(filters=32, kernel_size=(3, 3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu'))
    model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(512, activation="relu"))
    model.add(Dropout(0.5))
    model.add(Dense(len(emnist_labels), activation="softmax"))
    model.compile(loss='categorical_crossentropy', optimizer=RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0), metrics=['accuracy'])
    return model
```

Рисунок 2.4 – Листинг кода создания модели нейронной сети

Class Sequential позволяет создавать последовательную группировку линейного стека слоев.

Следующими действиями в созданную модель добавляются следующие параметры:

- Convolution2D – позволяет создать слой ядра свертки, который позволяет получить тензор выходных данных. Тензор можно представить, как многомерную таблицу определенной размерности, заполненную числами. Если тензор имеет второй ранг, то его представляют в виде матрицы. В данную функцию передается множество параметров, давайте разберем их по порядку. Filters – определяет размер выходного пространства, другими словами – количество выходных фильтров в свертке. kersel_size – определяет размер ядра, первый параметр определяет высоту, а второй ширину окна свертки. Input_shape – данная функция преобразует входное двумерное изображение размером 28 на 28 пикселей в одномерный массив, состоящий из 784 пикселей. Activation = ‘relu’ – функция активации. Функция активации необходима для вычисления выходного сигнала нейрона.

Relu – яыляется выпрямленной линейной функцией активации. Ее формула выглядит так: $f(s) = \max(0, s)$.

- MaxPooling2D – устанавливает размер сканирующего окна. Данное сканирующее окно проходит по массиву Numpy, позволяя проводить точечные операции с исходным массивом. Одной из операций является операция сравнения, которая позволяет выделять схожие признаки между изображениями.

- Dropout – исключает нейрон с вероятностью, указанной в параметрах. Данная методика помогает тем, что вместо обучения одной нейронной сети получается ансамбль из нескольких нейронных сетей. Данные, полученные из обученных нейронных сетей, потом усредняют, что в среднем дает лучший результат, чем при обучении без исключения нейронов.

- Flatten – данный метод объединяет все массивы данных модели в один массив.

- Dense – добавляет в нейронную сеть еще один «плотный» слой. Плотным слоем называют слой начального уровня, предоставляемый надстройкой Keras, который принимает в число нейроном или единиц в качестве своего требуемого параметра.

- Compile – данная функция позволяет настроить функции потерь, оптимизации и метрики нейронной сети.

После создания модели, она была обучена распознаванию букв английского алфавита при помощи датасета EMNIST, который содержит 60000 изображений рукописных букв данного алфавита размером 28 на 28 пикселей.

Теперь необходимо подготовить изображение для подачи его на вход нейронной сети. Для этого после записи в переменную изображения переводим изображение в оттенки серого при помощи встроенной в библиотеку OpenCV функцию cv2.cvtColor() и передаем в нее такие параметры:

- Переменную, с записанным с нее изображением.

- Параметр преобразования изображения. Для того, чтобы изображение перешло в оттенки серого необходимо указать: cv2.COLOR_BGR2GRAY. OpenCV изначально работает с цветовой палитрой blue, green, red (BGR), поэтому параметр преобразования начинается с BRG, а не RGB.

Теперь, имея изображение в серых тонах, можно преобразовать его в черно-белое изображение. Для этого используется функция cv2.threshold(). Параметры для данной функции используются следующие:

- Переменная, с записанным в нее изображением в оттенках серого.
- Численное значение яркости цвета, в которое обращаются все пиксели, у которых яркость цвета меньше 127
- Численное значение яркости цвета, в которое обращаются все пиксели, у которых яркость цвета больше или равна 127.
- Функция, отвечающая за вычисление преобразования цвета.

После выполнения предыдущей операции мы получили черно-белое изображение. Чтобы его контуры стали более четкими, воспользуемся функцией cv2.erode(). Параметры, передаваемые функции следующие:

- Переменная, с записанным в нее изображением.
- Структурирующий элемент, который является квадратной матрицей какого-либо размера.
- Количество итераций эрозии.

Выполнив это действие мы закончили подготовку изображения с текстом к разбитию написанных на нем слов на буквы. Сделав из входного изображения черно-белое, мы облегчили нахождение контуров букв слова. Для нахождения контуров букв было решено воспользоваться функцией cv2.findContours(), которая пользуется алгоритмом нахождения контуров Suzuki85, описанным в начале этой главы. Параметры, передаваемые в функцию следующие:

- Переменная, с записанным в нее изображением.
- Режим поиска контуров. Используется cv2.RETR_TREE. Данный режим извлекает все контуры на изображении и восстанавливает полную иерархию вложенных контуров.
- Метод аппроксимации контуров. Используется cv2.CHAIN_APPROX_NONE. Данный метод хранит все точки контура.

```
def get_letters(img_file: str, out_size=28):
    img = cv2.imread(img_file)
    # Shade of gray
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Black-white
    ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY)
    # Erode
    img_erode = cv2.erode(thresh, np.ones((3, 3), np.uint8), iterations=1)
    # Get contours
    contours, hierarchy = cv2.findContours(img_erode, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
```

Рисунок 2.5 – Листинг кода преобразований входного изображения

После нахождения контуров нужно разбить изображение на части, содержащие буквы слов. Для этого можно воспользоваться функцией cv2.boundingRect(), которая находит значения приблизительного прямоугольника вокруг найденных контуров изображения. Данная функция имеет только один параметр – переменную, с записанным в нее изображением, на котором были выделены контуры букв.

Так как созданная ранее нейронная сеть была обучена распознавать буквы английского алфавита размером 28 на 28 пикселей, то необходимо изменить размеры букв до размера 28 на 28 пикселей для того, чтобы была возможность подать данные изображения на вход сети. Для этого проводится сравнение длины и высоты изображения, если они различаются, то приводятся к одному значению. Затем с помощью функции cv2.resize(), которая имеет следующие параметры:

- Переменная, содержащая изображение буквы.
- Размер выходного изображения.

- Метод интерполяции. Используется cv2.INTER_AREA, который использует метод передискретизации и использованием отношения площади пикселя.

```
for idx, contour in enumerate(contours):
    (x, y, w, h) = cv2.boundingRect(contour)
    if hierarchy[0][idx][3] == 0:
        cv2.rectangle(output, (x, y), (x + w, y + h), (70, 0, 0), 1)
        letter_crop = gray[y:y + h, x:x + w]
        # Resize letter canvas to square
        size_max = max(w, h)
        letter_square = 255 * np.ones(shape=[size_max, size_max], dtype=np.uint8)
        if w > h:
            y_pos = size_max//2 - h//2
            letter_square[y_pos:y_pos + h, 0:w] = letter_crop
        elif w < h:
            x_pos = size_max//2 - w//2
            letter_square[0:h, x_pos:x_pos + w] = letter_crop
        else:
            letter_square = letter_crop
        # Resize letter to 28x28 and add letter and its X-coordinate
        letters.append((x, w, cv2.resize(letter_square, (out_size, out_size), interpolation=cv2.INTER_AREA)))
```

Рисунок 2.6 – Листинг кода изменения масштаба вырезанных частей букв

Далее для удобства все буквы сортируются по координате x.

Остается только подать подготовленные изображения на вход нейронной сети. После того как нейронная сеть определит какие буквы находится на изображениях, остается только по очереди вывести полученные значения в консоль и расставить пробелы.

```
def text_to_terminal(model: Any, img_file: str):
    letters = get_letters(img_file)
    s_out = ""
    for i in range(len(letters)):
        whitespace = letters[i+1][0] - letters[i][0] - letters[i][1] if i < len(letters) - 1 else 0
        s_out += emnist_predict_img(model, letters[i][2])
        if (whitespace > letters[i][1]/4):
            s_out += ' '
    return s_out
```

Рисунок 2.6 – Листинг кода перевода символов на изображениях в символы в терминале

На данном этапе функционал программы подходит к концу.

В данной главе был описан процесс создание алгоритма, способного распознать текст на изображении, выбор средств для реализации данного алгоритма и сама реализации. В результате выполнения всех действий была разработана программная реализация алгоритма распознавания текста, который способен получать на вход изображение, обработать и подать его на

вход нейронной сети, которая распознает символы на полученном изображении, и вывести их на экран в порядке очереди.

Теперь можно переходить к тестированию разработанной программы.

Глава 3 ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО АЛГОРИТМА

3.1 Тестирование разработанного алгоритма с использованием различных изображений

В ходе выполнения выпускной квалификационной работы была спроектирована, разработана и протестирована система распознавания текста на изображении.

Данная система выполняет следующие действия:

- Создает модель свёрточной нейронной сети.
- Обучает модель свёрточной нейронной сети.
- Получает изображение и преобразует его в черное-белое.
- Находит контуры на черно-белом изображении.
- Выделяет контуры на изображении в прямоугольные формы.
- Изменяет изображения, выделенные прямоугольниками, в квадратные изображения размером 28 на 28 пикселей.
- Отправляет на вход нейронной сети измененные изображения букв.
- Получает данные о принадлежности изображения к определенной букве.
- Выводит на экран полученную строчку из букв на изображении в порядке слева направо, расставляя пробелы.

Создание и обучение модели нейронной сети не требует действий пользователя, так как все параметры изначально записаны в коде.

Единственным действием, которое должен совершить пользователь, является загрузка фотографии в систему. Это действие выполняется перемещением изображения в папку, указанную в коде и изменением названия изображения, чтобы путь в коде совпадал с путем исходного изображения.

Остальные действия выполняет система и пользователю нужно лишь увидеть результат выполненной программы в консоли.

Для проверки работоспособности программы, ей на вход было отправлено изображение с черным текстом на белом фоне, представленное на рисунке 3.1.

Рисунок 3.1 – Изображение для тестирования работы алгоритма.

После того, как изображения было записано в переменную, оно было переведено в оттенки серого, а потом в черно-белое. Так как изначально изображение являлось черно-белым, то никаких изменений в нем не произошло.

Следующим действием происходит выделение контуров элементов на изображении и создание из них отдельных изображений размеров 28 на 28 пикселей.

Рисунок 3.2 – Выделение контуров букв на изображении

Полученные изображения подаются на вход нейронной сети. Распознанные нейронной сетью буквы выводятся в консоль.

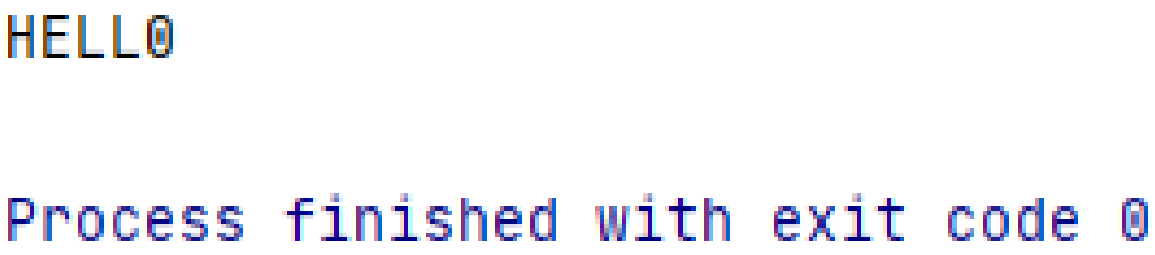


Рисунок 3.3 – Результат работы нейронной сети

Как видно из рисунка, представленного выше, обученная нейронная сеть правильно распознала все буквы, кроме буквы «О», которую сеть перепутало с цифрой «0», так как нейронная сеть обучалась распознавать как английские буквы, так и цифры.

Теперь подадим на вход программы изображение, на котором буквы будут окрашены в различные цвета. Изображение, которое было подано на вход, представлено ниже.

Рисунок 3.4 – Исходное изображение с цветными буквами

Буквы на изображении яркие и программа без труда перекрасила поданное на вход изображение в черно-белое, что представлено на рисунке 3.5.

Рисунок 3.5 – Перевод изображение с цветными буквами в черно-белое

Так как все буквы на изображении были переведены в черно-белое пространство, то система может определить на них контуры. Результат определения контуров на цветном изображении представлен ниже.

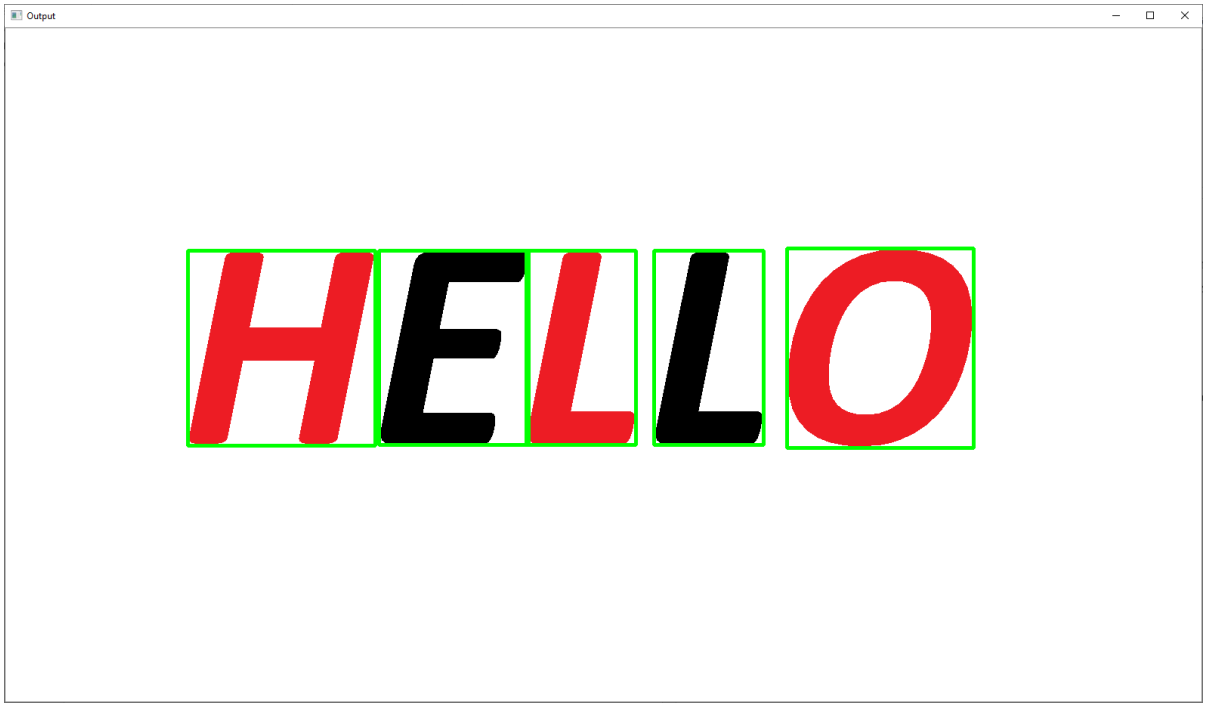


Рисунок 3.6 – Определение контуров на изображении с цветными буквами

После определения контуров букв, система может определить принадлежность полученных изображений к буквам английского алфавита.

```
HELLC

Process finished with exit code 0
```

Рисунок 3.7 – Результат работы нейронной сети

Как видно из представленного выше рисунка, нейронная сеть не смогла точно определить некоторые буквы на изображении.

На следующей попытке тестирования было решено уменьшить яркость цвета буквы путем выбора светлого цвета. Изображение, которое будет отправлено на вход программы представлено ниже.

Рисунок 3.8 – Исходное изображение с буквами разной яркости

Запустив программу и увидев результат преобразования изображения, стало ясно, что функция перевода изображения в черно-белое превращает все

светлые буквы в белые, что убирает их из слова и не дает возможности их идентифицировать. Оставшиеся буквы слова, которые было окрашены в черный цвет, успешно прошли идентификацию и были отображены в консоли.

Рисунок 3.9 – Перевод изображения с буквами разной яркости в черно-белое

Рисунок 3.10 - Определение контуров на изображении с буквами разной яркости

Результат работы программы приведен ниже.

```
4 L

Process finished with exit code 0
```

Рисунок 3.11 – Результат работы программы

Как видно из представленного выше рисунка, программа определила лишь два символа на изображении, один из которых определен неверно.

Также был проведен эксперимент, где буквы были написаны белым на темном фоне. Изображение, которое было подано на вход показано на рисунке 3.12.

Рисунок 3.13 – Исходное изображение с белыми буквами на черном фоне.

Так как изображение уже является черно-белым, то можно перейти сразу к выделению контуров.

Рисунок 3.14 - Определение контуров на изображении с белыми буквами на черном фоне

Библиотека OpenCV не смогла определить на представленном выше изображении контуров из-за чего в нейронную сеть не поступило никаких данных, а в командную строку не было выведено ни одного символа.

Из проведенных выше тестов можно сказать, что система работает хорошо лишь на черно-белых изображениях, где текст является черным, а фон светлым.

3.2 Подведение итогов тестирования разработанного алгоритма

Во время выполнения бакалаврской работы было проведено тестирование алгоритма распознавания текста с изображения. Во время тестирования на вход программы подавались разные изображения, имеющие различия между собой. Основными параметрами тестирования являлись следующие:

- Количество символов в тексте равно 5.

- Использованные шрифты: Arial, Times New Roman, Calibri.
- Текст на изображении написан на светлом фоне.
- Текст на изображении может быть либо черным, либо цветным.
- Цветной текст на изображении должен быть ярким.
- Текст написан с использованием английского алфавита.
- Размер шрифта одинаковый

Было проведено 10 тестов для каждого шрифта. 5 тестов проводились, когда текста на изображении был черного цвета, еще 5 тестов, где текст на изображении был цветным. По итогам всех тестов была построена таблица:

Таблица 1 – Результаты тестирования алгоритма для черно-белых изображений

Шрифт	Количество правильно определенных символов					Всего
Arial	5	5	5	4	5	24
Times New Roman	2	4	3	2	4	15
Calibri	5	5	5	3	4	22
Всего	12	14	13	9	13	

Из данных таблицы 1 можно сделать следующий вывод. Алгоритм хорошо определяет символы на черно-белых изображениях. Однако при распознавании символов шрифта Times New Roman алгоритм допускает ряд ошибок.

На следующей таблице представлены результаты работы алгоритма при определении цветных символов на изображении.

Таблица 2 - Результаты тестирования алгоритма для цветных изображений

Шрифт	Количество правильно определенных символов					Всего
Arial	3	2	2	3	4	14
Times New Roman	1	1	1	1	2	6
Calibri	1	1	2	1	3	8
Всего	5	4	5	5	10	

Как видно из таблицы 2 система справляется с цветными изображениями намного хуже. Одной из причин является то, что цветные изображения имею дополнительную информацию о цвете, что сбивает нейронную сеть.

Еще одной особенностью текста является размер символа. Поэтому были проведены тесты, при которых системе подавались на вход изображения с различными размерами символов. Для проведения тестов использовался шрифт Arial, который показал лучший результат среди предыдущих тестов. Размер исходного изображения – 350 на 200 пикселей. Текст на изображении содержал три слова, состоящих суммарно из 11 различных символов. Каждая новая попытка содержала другой набор символов.

Таблица 3 – Результаты тестирования алгоритма при различных размерах шрифта.

	Размер шрифта							
Попытка №	72	48	36	28	24	18	14	10

1	9	9	8	10	10	5	5	0
2	9	9	8	9	8	6	2	0
3	11	11	10	11	11	8	5	0

Из данных, занесенных в таблицу 3, можно сделать следующий вывод: Точность определения символа зависит от размера текста и самого символа. Чем больше текст, тем больше вероятность верного распознавания текста.

По представленной выше таблице был построен график.

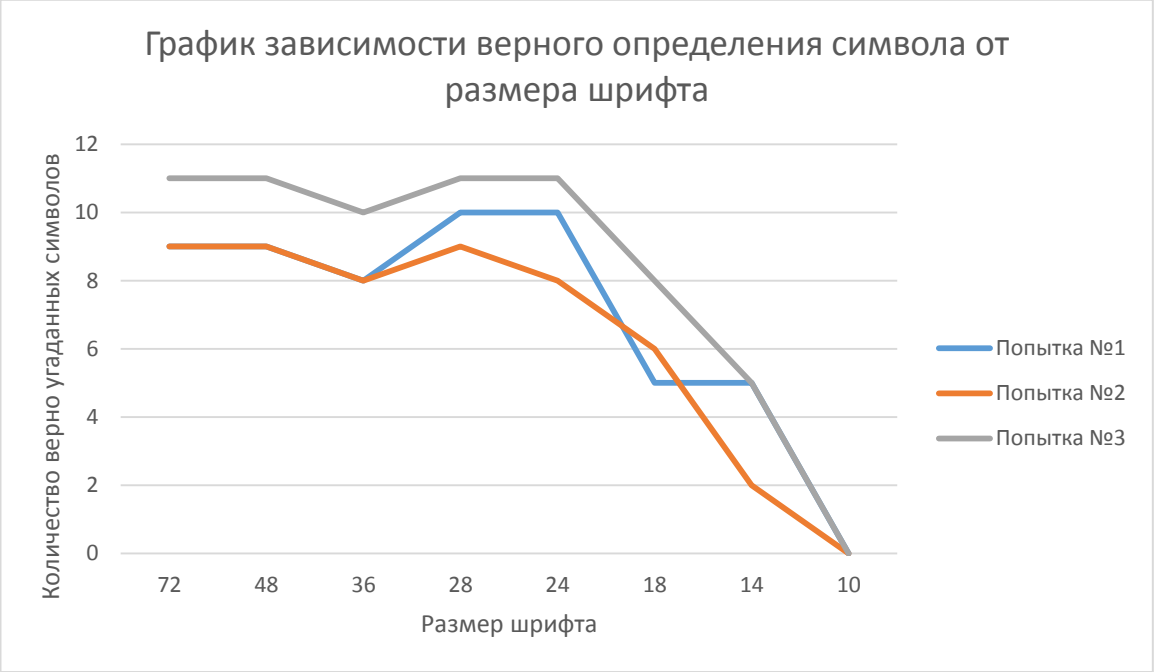


Рисунок 3.15 – График зависимости верного определения символа от размера шрифта

По данному графику можно точно сказать, что вероятность верного распознавания символов текста уменьшается при уменьшении размера шрифта и становится равным 0 при размере шрифта меньше и равном 10.

Для того, чтобы алгоритм точно распознавал текст, расстояния между буквами должно быть таким, чтобы было контуры буквы не соприкасались.

Также для улучшения распознавания символов слова необходимо переобучить нейронную сеть распознавать только символы букв, так как нейронная сеть часто заменяла изображенную букву «О» на символ цифры «0», что некорректно в условиях задачи.

Еще одним недостатком алгоритма является то, что при увеличении количества символов на изображении, значительно увеличивается время определения наличия символа на изображении, что в свою очередь увеличивает время работы программы. Это связано с тем, что найденные контуры на изображении обрабатываются последовательно. Хорошей доработкой данного алгоритма является создание его распараллеленной версии, что уменьшит время обработки изображения за счет увеличения количества нагруженных потоков.

Таким образом, в результате тестирования разработанного алгоритма были получены данные о его работе при различных условиях. Входные изображения при тестировании отличались видом шрифта, цветом символов и размером символов на изображении. После проведения всех тестов данные были занесены в таблицы и был построен график зависимости верного определения символа от размера шрифта. Также были выявлены недостатки алгоритма и описаны возможные варианты улучшения алгоритма.

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа посвящена разработке алгоритма распознавания текста на изображении при помощи сверточной нейронной сети. В результате ее выполнения был спроектирован, реализован и протестирован алгоритм, позволяющий распознать на изображении буквы слова и перевести распознанные на изображении буквы в печатный текст.

Во время выполнения данной работы были успешно завершены следующие задачи:

- Изучены виды нейронных сетей и их применение.
- Изучены строение свёрточной нейронной сети и алгоритм свертки
- Реализована свёрточная нейронная сеть с использованием библиотек машинного обучения Tensorflow и Keras.
- Свёрточная нейронная сеть была обучена с использованием датасета рукописных английских букв и цифр EMNIST.
- Были реализованы функции преобразования исходного изображения для нахождения контуров символов с использованием библиотеки OpenCV.

Для тестирования работоспособности алгоритма на вход системы подавались изображения, имеющие различные особенности, такие как: наименование символов английского алфавита, цвет символов, яркость цвета символов, размер символов, цвет фона изображения. По окончанию тестирования были построены таблицы и графики, показывающие эффективность алгоритма.

По результатам тестирования полученной программы были выявлены ее недостатки, а также предложены варианты ее улучшения.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Аггавал Чару. Нейронные сети и глубокое обучение. – Висьямс, 2020 – ISBN: - 978-5-907203-01-3
2. Антонова П. Введение в искусственный интеллект: Теоретические основы СИИ. - LAMBERT Academic Publishing, 2019 – ISBN: 978-6200279170
3. Головко В.А. От многослойных персептронов к нейронным сетям глубокого доверия: парадигмы обучения и применение // XVII Всероссийская научно-техническая конференция с международным участием, 2015
4. Документация по Python [Электронный ресурс] / URL: https://www.python.org/doc/ (дата обращения 10.02.20)
5. Классификация и типы нейронных сетей [Электронный ресурс] Режим доступа: http://datascientist.one/class-type-nn/ (дата обращения: 12.02.20)
6. Машинное обучение на практике с Python и Keras [Электронный ресурс]. – Режим доступа: https://pythonru.com/primery/mashinnoe-obuchenie-na-praktike-s-python-i-keras (дата обращения 10.02.20)
7. Нейроны головного мозга – строение, классификация и проводящие пути [Электронный ресурс]. – Режим доступа: https://sortmozg.com/structure/nejrony-golovnogo-mozga (дата обращения 28.02.20)
8. Николенко С. И., Кадурин А. А., Архангельская Е. О. Глубокое обучение. Погружение в мир нейронных сетей / Питер, 2020
9. Себастьян Рашка «Python и машинное обучение: машинное и глубокое обучение с использованием python, scikit-learn и tensorflow». - Москва, 2019 - ISBN: 978-5-907114-52-4
10. Тарик Рашид «Создаем нейронную сеть». – Вильямс, 2017 – ISBN: 978-5-9909445-7-2.

11. Франсуа Шолле «Глубокое обучение на Python». – Питер, 2018 – ISBN: 978-5-4461-0770-4

12. Что такое свёрточная нейронная сеть [Электронный ресурс] Режим доступа: https://habr.com/ru/post/309508/ (дата обращения 15.05.20)

13. Harrold Ellis, Bari M Logan, Adrian K Dixon. David j Bowden. HUMAN SECTIONAL ANATOMY. Atlas of body sections, CT and MRI images. FOURTH EDITION // CRC Press, 2015 – ISBN: 978-1-4987-0361-1

14. Keras Documentation [Электронный ресурс]. – Режим доступа: https://keras.io/api/ (дата обращения: 15.02.20)

15. Mohammad Abdur Razzaque PhD, Md. Rezaul Karim. Hands-On Deep Learning for IoT: Train neural network models to develop intelligent IoT applications // Packt Publishing, 2019

16. OpenCV: API Documentation [Электронный ресурс]. – Режим доступа: https://docs.opencv.org/2.4/modules/refman.html (дата обращения: 15.02.20)

17. Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5) [Электронный ресурс]. – Режим доступа: https://arxiv.org/pdf/1311.2524.pdf (дата обращения: 10.02.20)

18. Sudharsan Ravichandiran, Sean, Saito, Rajalingappaa Shanmugamani, Yang Wenzhuo. Python Reinforcement Learning: Solve complex real-world problems by mastering reinforcement learning algorithms using OpenAI Gym and Tensorflow. // Paclt Publishing, 2019

19. Suzuki S, Abe K. Topological Structural Analysis of Digitized Binary Images by Border Following // CVGIP. 1985. Vol. 1. P. 32-46.

20. Tensorflow API Documentation [Электронный ресурс]. – Режим доступа: https://www.tensorflow.org/api_docs/python/tf (дата обращения: 15.02.20)

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

Исходный код программы доступен для скачивания в электронном в виде.

Режим доступа:

https://yadi.sk/d/fJDMMYdqr7uwEw