Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровой экономики

Направление подготовки «09.03.04 – Программная инженерия»

Направление профиля «Программная инженерия»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(БАКАЛАВРСКАЯ РАБОТА)

На тему: Распознавание рукописных цифр с использованием сверточной нейронной сети

Студент Кайгородов Дмитрий Константинович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия имя отчество) (личная подпись)

Руководитель Татьянкин Виталий Михайлович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия имя отчество) (личная подпись)

Нормоконтролер Вохминцев Александр Владиславович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия имя отчество) (личная подпись)

Допустить к защите

Руководитель образовательной программы

Старший преподаватель, \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ученая степень, звание, фамилия имя отчество) (личная подпись)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

г. Ханты-Мансийск

2020 год

АННОТАЦИЯ

на выполнение выпускной квалификационной бакалаврской работы

«Распознавание рукописных цифр с использованием сверточной нейронной сети»

В первой главе работы дана характеристика объекта исследования.

Во второй и третьей главе работы представлено описание обучающего и тестового наборов данных для обучения.

В третьей главе работы описаны способы обучения и проблематика обучения нейронных сетей. Обоснован выбор конкретного способа обучения.

В четвертой главе работы произведен обзор типов архитектур нейронных сетей, описан общий принцип работы сверточной нейронной сети, выполнено проектирование и описана архитектура сверточной нейронной сети.

В пятой главе работы описана разработка, тестирование прототипов.

Оценка и сравнение результатов исследования описаны в последней, шестой главе.

Выпускная квалификационная работа состоит из 63 страниц, 6 таблиц, 22 иллюстраций, 31 используемых источников и 4 приложений.

Ключевые слова: набор данных (датасет), искусственный нейрон, функция активации, сверточная нейронная сеть, глубокое обучение.

Цели работы – распознавание рукописных цифр при помощи сверточной нейронной сети.

В результате выполненной работы по разработке модуля было написано программное обеспечение, которое позволяет распознавать рукописные цифры.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровой экономики

КАЛЕНДАРНЫЙ ПЛАН

выполнения выпускной квалификационной работы

Студент Кайгородов Дмитрий Константинович

Тема «Распознавание рукописных цифр с использованием сверточной нейронной сети»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование этапа работы** | **Плановый срок выполнения этапа работы** | **Фактический срок выполнения этапа работы** | **Результат выполнения** | **Подпись руководителя** |
| Состояние вопроса. Изучение литературы и анализ аналогов. | 20/01/2020 | 24/01/2020 | Раздел ВКР, постановка задачи |  |
| Состояние вопроса. Обследование объекта темы. | 27/01/2020 | 27/01/2020 | Архитектурное решение и функциональные требования |  |
| Формирование требований | 03/02/2020 | 04/02/2020 | ТЗ (бриф) |  |
| Формирование требований | 24/02/2020 | 25/02/2020 | раздел ВКР |  |
| **Промежуточный контроль 1** | 24/02/2020 |  |  |  |
| Проектирование программного модуля | 24/03/2020 | 30/03/2020 | раздел ВКР, диаграммы |  |
| Кодирование и Тестирование | 14/04/2020 | 30/04/2020 | Первый прототип, раздел ВКР |  |
| **Промежуточный контроль 2** | 28/04/2020 |  |  |  |
| Кодирование и Тестирование | 10/05/2020 | 13/05/2020 | Второй прототип модуля, раздел ВКР, отчет о тестировании |  |

Продолжение календарного плана

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Внедрение | 20/05/2020 | 20/05/2020 | Законченные работы по внедрению, раздел ВКР |  |
| Предзащита | 14/06/2020 |  |  |  |
| Оформление |  |  | Оформление текста ВКР |  |
| Сдача готовой ВКР на кафедру | 4/07/2020 | 4/07/2020 |  |  |

Руководитель выпускной квалификационной работы

Татьянкин Виталий Михайлович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия имя отчество) (личная подпись)

Задание принял к исполнению

Кайгородов Дмитрий Константинович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия имя отчество) (личная подпись)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровой экономики

Направление подготовки «09.03.04 – Программная инженерия»

Направление профиля «Программная инженерия»

УТВЕРЖДАЮ

Руководитель образовательной программы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ученая степень, звание, фамилия имя отчество)

«\_\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

Студент(ка) Кайгородов Дмитрий Константинович

(Фамилия имя отчество)

1. Тема «Распознавание рукописных цифр с использованием сверточной нейронной сети»

2. Срок сдачи студентом законченной выпускной квалификационной работы 19.06.2020

3. Исходные работе данные к выпускной квалификационной работе примеры рукописных изображений, обучающая выборка из рукописных цифр

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов) 6 глав, 15 подразделов, 4 приложений

5. Ориентировочный перечень графического и иллюстративного материала 63 страниц, 6 таблиц, 22 иллюстраций, 31 используемых источников

6. Дата выдачи задания

«\_\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

Руководитель

выпускной квалификационной работы Татьянкин Виталий Михайлович \_\_\_\_\_\_

(Фамилия имя отчество) (личная подпись)

Задание принял к исполнению Кайгородов Дмитрий Константинович \_\_\_\_\_\_

(Фамилия имя отчество) (личная подпись)

Оглавление

[Введение 7](#_Toc43490282)

[1. Обследование предметной области 9](#_Toc43490283)

[1.1 Искусственные нейронные сети и их составляющие 9](#_Toc43490284)

[1.2 Сравнение биологического и искусственного нейрона 9](#_Toc43490285)

[1.3 Искусственные нейроны 10](#_Toc43490286)

[2. Предобработка данных 13](#_Toc43490287)

[2.1 Используемые данные для обучения нейронной сети 13](#_Toc43490288)

[2.2 Описание используемого набора данных 13](#_Toc43490289)

[3. Способы обучения сверточной нейронной сети 15](#_Toc43490290)

[3.1 Проблема исчезающего градиента 17](#_Toc43490291)

[3.2 Проблема переобучения 17](#_Toc43490292)

[3.3 Проблема сигмоидальных функций 19](#_Toc43490293)

[4. Архитектура сверточной нейронной сети 21](#_Toc43490294)

[4.1 Сверточная нейронная сеть 22](#_Toc43490295)

[4.2 Слой свёртки 24](#_Toc43490296)

[4.3 Слой объединения 27](#_Toc43490297)

[4.4 Функция активации полносвязных слоев 27](#_Toc43490298)

[5. Программная реализация 28](#_Toc43490299)

[5.1 Описание программных библиотек 28](#_Toc43490300)

[5.2 Нейронная сеть прямого распространения 30](#_Toc43490301)

[5.3 Сверточная нейронная сеть 39](#_Toc43490302)

[6. Сравнение и оценка результатов 43](#_Toc43490303)

[Заключение 45](#_Toc43490304)

[Список использованных источников 47](#_Toc43490305)

[Приложение А 50](#_Toc43490306)

[Приложение Б 54](#_Toc43490307)

[Приложение В 58](#_Toc43490308)

[Приложение Г 59](#_Toc43490309)

Введение

В последние несколько лет возрастает интерес к изучению нейронных сетей. Цифровизация нашего общества постоянно ставит новые задачи, которые функционально усложняются и расширяются, что обусловило необходимость широкого применения нейронных сетей для выполнения разнообразных функций.

Много программистов по всему миру разрабатывают разные архитектуры нейронных сетей, для решения общей задачи классификации, но для этого необходимо решать частные задачи. Одной из задач для искусственного интеллекта является распознавание объектов на изображении [25, 26, 27, 28, 30], в этой работе рассматривается еще более частный случай распознавания цифр. Сверточные нейронные сети являются инструментом для решения подобных задач. Пример реализации и использования сверточных нейронных сетей можно увидеть в современных поисковых системах от Google, Yandex и других компаний по разработке интернет-сервисов.

Актуальность темы выпускной квалификационной работы обусловлена необходимостью повышения эффективности работы сотрудников за счет уменьшения трудозатрат, сокращения количества ошибок, повышения качества и достоверности формируемых документов.

Объектом исследования выпускной квалификационной работы является процесс распознавания рукописных цифр.

Предметом исследования – процесс по распознаванию чисел.

Целью выпускной квалификационной работы является проектирование [12, 14, 17] и разработка нейронной сети для распознавания рукописного ввода цифр.

Для достижения обозначенной цели необходимо выполнить следующие задачи:

1) Провести анализ предметной области и аналогов архитектур нейронных сетей, эксплуатируемых в предметной области тематики выпускной квалификационной работы. Исследование существующих архитектур нейронных сетей для классификации изображений.

2) Проектирование и разработка нейронной сети, для решения задачи выделения и классификации рукописных символов.

3) Разработать и осуществить тестирование работы программного модуля, обеспечивающего распознавание цифр на изображениях нейронной сетью, на основе изображений, заданных пользователем системы.

4) Изучение библиотек языка программирования Python для обучения нейронных сетей.

В результате выполненной работы по разработке модуля было написано программное обеспечение, которое позволяет распознавать рукописные цифры.

Созданная НС, призвана в некоторой степени облегчить процесс распознавания цифр.

# **1. Обследование предметной области**

# **1.1 Искусственные нейронные сети и их составляющие**

# **1.2 Сравнение биологического и искусственного нейрона**

Прежде чем проводить анализ сверточной нейронной сети, необходимо пояснить как устроены искусственные нейронные сети (ИНС).

При создании ИНС мы опираемся на принцип организации нервной системы живых организмов, потому что они содержат элементы, которые выполняют схожие функции: одни – отдают различные команды-сигналы, другие – передают их, третьи воспринимают и исполняют эти команды, а четвертые – запоминают и хранят информацию. Нервная система живых организмов делает все гораздо успешнее, экономичнее так как устроена и реализована более надежно и гибко, в отличии от ИНС. Поэтому стоит задача создать ИНС, по своему совершенству приближающаяся к нервной системе. Для этого нужно понять, как устроены нервные клетки (Рисунок 1.1), и как работает мозг.

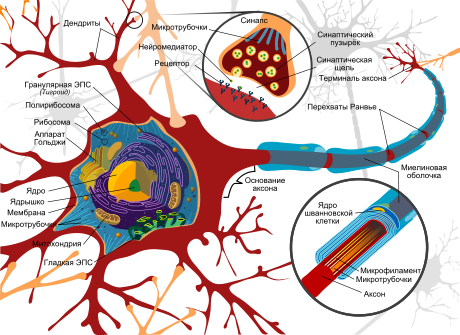


Рисунок 1.1 – Структура биологического нейрона

Нервная система для каждого живого организма может одновременно выполнять множество разных задач. ИНС тоже могут выполнять множество задач, но для них необходимо задать условие и обучить решению.

Искусственный нейрон – это очень отдаленное подобие биологического нейрона, являющийся математической нелинейной функцией (функцией активации), которая определяет входное значение нейрона и изменения весов относительно результата взвешенной суммы входов и порогового значения.

Под искусственным нейроном, для примера, будем понимать нелинейную функцию Sigmoid, а отдельно взятый нейрон будет являться полным аналогом логистической регрессии, линейного классификатора. Эта функция приведена во многих ранее предложенных учеными моделях, как пример подражания биологическим нейронам, но позднее стали применять более простую функцию активации ReLU (rectified linear unit).

Необходимо упомянуть, что существует много разных функций активации помимо ReLU, например, TANH (гиперболический тангенс), Sigmoid. Эти функции производят разные вычисления, например, функция активации ReLU – это просто замена отрицательных элементов нулем, Sigmoid приводит значения к диапазону 0 до 1.

# **1.3 Искусственные нейроны**

Самая простейшая структура нейронной сети - это искусственный нейрон, модель которого представляет из себя персептрон. Искусственная нейронная сеть [21, 22] обучается благодаря изменениям в значениях весов, которые, в свою очередь, производят изменение итогового результата функции, то есть выхода нейрона.

Схема персептрона, математической модели восприятия информации мозгом, который используют для описания искусственного нейрона представлена на рисунках 1.2 и 1.3.

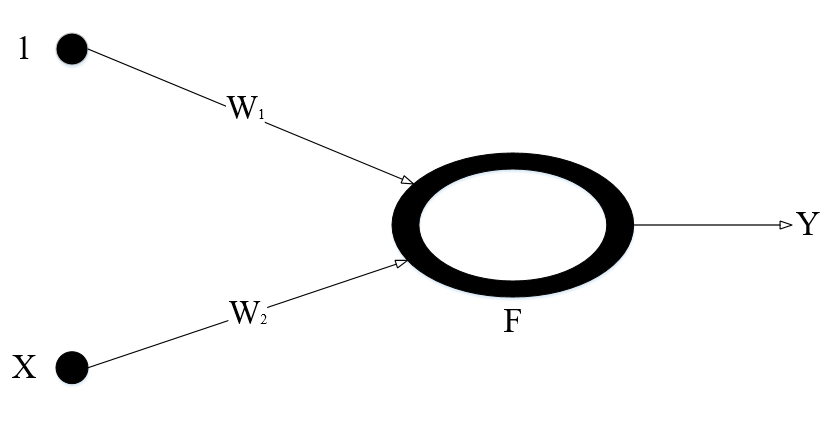


Рисунок 1.2 – Схема персептрона.

На рисунке 1.2 латинской буквой «X» обозначено значение входного сигнала, «W» обозначено значение веса с индексом нейрона, к которому он приходит, «Y» это выходной сигнал, который определяется зависимостью от функции активации «F».

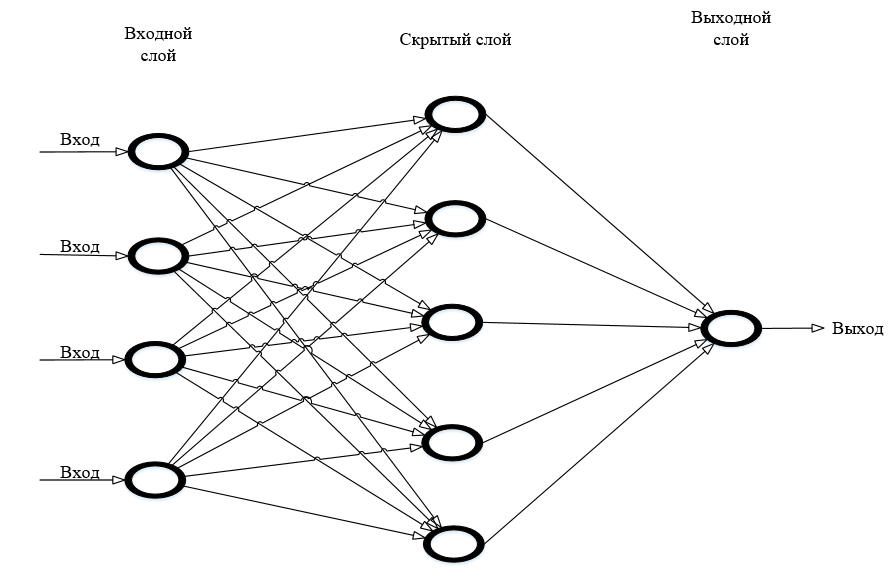


Рисунок 1.3 – Схема многослойного персептрона

К примеру, функция активации Sigmoid нелинейная и непрерывная, входные значения, так же, как и на выходе или результат функции, будут соответствовать диапазону между 0 и 1 (что такое 0 и 1?).

При использовании непрерывной функции активации, становится возможным использование при обучении градиентных методов. В зависимости от положения нейронов в сети их можно разделить на группы:

1) Нейроны входного слоя – принимающие исходный вектор данных.

2) «Скрытые» нейроны, в которых осуществляются основные вычислительные операции.

3) Выходные нейроны, выдающие результат работы сети.

# **2. Предобработка данных**

# **2.1 Используемые данные для обучения нейронной сети**

При обучении нейронных сетей выделяют два основных подхода: обучение с учителем и без учителя. Подразумевая обучение нейронной сети без учителя, следует отметить, что нейронная сеть не получает никаких пояснений к тому, что подается ей на вход, выделяет те признаки объектов, что присутствуют во входном потоке данных. Обучая нейронную сеть с учителем предполагается, что для части входных образов, называемых обучающей выборкой, известно, какой выходной результат необходимо получить.

Данные, на которых мы обучаем модель – это объекты реального мира. Изначально мы не располагаем векторами. Всё что у нас есть – это сложное описание каждого объекта в выборке.

Задача состоит в том, чтобы извлечь всю цифровую информацию, которая может как-то охарактеризовать каждый признак объекта.

Данные используемые для обучения так же называются датасетом (dataset). Для классификации изображения используют подход обучения с учителем, который описан выше. Это подразумевает, что можно полагаться только на размеченные данные для обучения. Зачастую помеченных данных бывает недостаточно, и, следовательно, для многих задач трудно получить необходимое количество примеров для соответствия параметров сложной модели. Учитывая высокую степень выразительности глубоких нейронных сетей, обучение на небольшом наборе данных приведет к переобучению.

**2.2 Описание используемого набора данных**

Рассматриваемая в этой работе задача очень распространена, и в результате анализа, было принято решение использовать максимально подходящую базу данных MNIST [11], которая состоит из 60 тыс. размеченных данных, которые включают в себя 50 тыс. изображений для обучения и 10 тыс. тестовых изображений.

Используемый формат изображений 28\*28 пикселей с оттенками серого. Для удобства обучения весь набор данных MNIST упакован в 1 бинарный файл. Благодаря этому, за один раз ИНС может прочитать все необходимые данные для обучения.

MNIST содержит образы рукописных цифр (Рисунок 2.1)

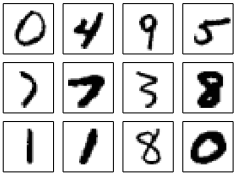


Рисунок 2.1 – Примеры рукописных цифр из набора данных

# **3. Способы обучения сверточной нейронной сети**

Существуют несколько способов обучения нейронных сетей [7, 31], но все эти способы объединяет принцип: найти или подобрать веса так, чтобы сеть приближала необходимую функцию с заданной точностью.

В машинном обучении существует 2 важных принципа обучения: обучение с учителем, когда есть какие-то конкретные примеры и обучение без учителя, то есть когда мы хотим, чтобы сеть сама изъяла какой-то «смысл». Нас больше интересует обучение с учителем, так как мы учим нашу сеть отличать на фото или на картинке определенное рукописное число, то есть объект из реального мира.

При обучении с учителем, на вход подается набор объектов, в котором уже заранее известно, какие ответы правильны. При обучении без учителя на вход подается набор объектов, в котором неизвестно, какие ответы правильны или нет.

Для всех способов обучения искусственных нейронных сетей глубокого обучения применяют разные алгоритмы, например:

1) Алгоритм обратного распространения ошибки

2) Алгоритм упругого распространения

3) Генетический алгоритм [24]

4) Стохастический градиентный спуск

При использовании первого алгоритма, в ходе поступления новых данных нейронная сеть поочередно передаёт её от одного нейрона к следующему с помощью синапсов, до тех пор, пока информация не окажется на выходном слое и не будет получена как результат. Подобный метод называется передачей вперёд. После получения результата, рассчитывается ошибка, и на ее основании производится обратная передача, что означает поочередно поменять веса нейронов, начиная с выходного и продвигаясь к входному слою. При этом значение веса изменяется в сторону наилучшего результата. Для использования подобного способа обучения подходят только лишь те функции активации, которые возможно дифференцировать. Обратное распространение вычисляется с помощью высчитывания разницы результатов и умножения ее на производную функции от входного значения.

При использовании второго алгоритма в качестве основы применяется обучение по эпохам и использует только знаки производных частного случая с целью корректировки весовых коэффициентов. Принцип работы данного алгоритма следующий: инициализируется величина коррекции, рассчитываются частные производные, подсчитывается новое значение корректировки значений весов, изменяются веса, в случае, если условие остановки алгоритма никак не выполняются, процесс переходит к вычислению производных и вновь повторяет цикл. Подобный подход позволяет достичь сходимости нейронной сети быстрее, в отличие от других вариантов обучения.

Третий тип, генетический алгоритм представляет собой простую интерпретацию природного метода, основанного на скрещивании итогов. То есть, по сути, происходит перекрещивание результатов, подбор оптимальных, и формирование на их основе нового поколения.

Последний, рассмотренный способ обучения – стохастический градиентный спуск, который является одним из базовых способов обучения сверточных нейронных сетей, реализован во всех основных программных библиотеках для обучения искусственного интеллекта. Одним из важных достоинств этого способа является возможность обучения на больших наборах данных. Этот способ действует следующим образом: случайно выбирается один объект из обучающей выборки, и дальше вычисляется градиент функционала только на этом объекте, то есть градиент только одного слагаемого в функционале ошибки, и вычитается именно этот градиент из текущего приближения вектора весов.

При обучении ИНС возникают различные проблемы [29], которые необходимо учесть при построении ИНС для решения рассматриваемой задачи классификации.

# **3.1 Проблема исчезающего градиента**

Трудность, возникающая при обучении искусственных нейронных сетей с использованием алгоритмов обучения на основе градиента и обратного распространения ошибки, называют проблемой исчезающего градиента. В таких алгоритмах каждый вес нейронной сети обновляется пропорционально градиенту функции ошибки относительно текущего веса на каждой итерации обучения. Стандартные функции активации, такие как гиперболический тангенс, имеют градиенты в диапазоне (-1, 1), а метод обратного распространения ошибки вычисляет их по цепному правилу. После умножения этих чисел для вычисления градиентов входных слоев в многослойной сети градиент (сигнал ошибки) экспоненциально уменьшается вместе с количеством пройденных слоев, а входные слои обучаются очень медленно.

# **3.2 Проблема переобучения**

При обучении ИНС часто возникает трудность, называемая проблемой переобучения.

Переобучение можно описать как чрезмерно близкую подгонку, которая выражается в слишком точном соответствии искусственной нейронной сети к набору обучающих примеров. Из-за этого сеть теряет способность к обобщению признаков.

Эта проблема может возникнуть по причине слишком долгого обучения или недостаточной размерности набора обучающих примеров или при переусложненности структуры самой искусственной нейронной сети.

Переобучение связано с тем, что выбор обучающего (тренировочного) множества является случайным. С первых шагов обучения происходит уменьшение ошибки. На последующих шагах с целью уменьшения ошибки (целевой функции) параметры подстраиваются под особенности обучающего множества. Однако при этом происходит "подстройка" не под общие закономерности ряда, а под особенности его части - обучающего подмножества. При этом точность прогноза уменьшается.

Один из вариантов решения проблемы переобучения искусственной нейронной сети – это деление обучающей выборки на множества обучающих и тестовых примеров. На обучающем множестве происходит обучение нейронной сети. На тестовом множестве осуществляется проверка построенной модели. Эти множества не должны пересекаться.

С каждым шагом параметры модели изменяются, однако постоянное уменьшение значения целевой функции происходит именно на обучающем множестве. При разбиении множества на два мы можем наблюдать изменение ошибки прогноза на тестовом множестве параллельно с наблюдениями над обучающим. Какое-то количество шагов ошибки прогноза уменьшается на обоих множествах. Однако на определенном шаге ошибка на тестовом множестве начинает возрастать, при этом ошибка на обучающем продолжает уменьшаться. Этот момент считается концом настоящего обучения, с него и начинается переобучение. На рисунке 3.1 представлены графики, наглядно показывающие возникновение данной проблемы.

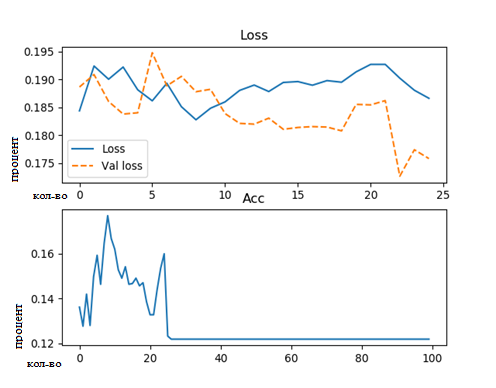


Рисунок 3.1 – Графики переобучения нейронной сети

Данные графики показывают изменение значений функции потерь и долю корректных прогнозов нейронной сети. По горизонтальной оси обозначены эпохи (epochs), итерации обучения, а по вертикальной точность (accurancy).

# **3.3 Проблема сигмоидальных функций**

Использование сигмоидальных активационных функций может вызвать замедление при обучении ИНС, а именно значения активаций в конечном слое будут близки к нулю на ранних этапах обучения. Были предложены альтернативные активационные функции (таблица 3.1), которые не так страдают от ограничения.

Линейные модели не могли работать с коррелирующими параметрами, так как это приводило к плохой обусловленности матриц и неограниченному росту весов. Сегодня проблема мультиколлинеарности векторов практически исчерпана за счёт использования продвинутых алгоритмов и методов предотвращения переобучения.

Таблица 3.1 – Функции активации нейронов

|  |  |  |
| --- | --- | --- |
| Название | Формула функции активации | Область значений |
| Пороговая |  | (0, 1) |
| Знаковая |  | (-1, 1) |
| Сигмоидальная |  | (0, 1) |
| Полулинейная |  | (0, ) |
| Линейная |  | (-) |
| Гиперболический тангенс |  | (-1, 1) |
| Треугольная |  | (0, 1) |

(читается «пси») – условное обозначение функции активации в таблице.

# **4. Архитектура сверточной нейронной сети**

Нейронные сети имеют множество архитектур [1, 2, 3, 4] (Рисунок 4.1, приложение В), но процесс их работы схож, представляет из себя движение потока внешних данных по структуре нейронной сети и преобразование этих данных. Не зависимо от типа сети, поток данных может формировать и перекрёстные, и обратные связи.

Нейроны группируются в последовательность слоёв: входные сигналы поступают на первый слой и последовательно проходят через все слои до выходного слоя сети. Следует упомянуть, что бывают и рекуррентные структуры, обеспечивающие циркуляцию некоторого набора внутренних сигналов.

Структура нейронной сети может быть адаптирована к задаче. В Нейронной сети могут быть включены или исключены дополнительные нейроны, связи между ними и даже слои нейронов, если исходно она была не способна обеспечить нужную точность решения. Нейронная сеть может сама выделить наиболее информативные для задачи входные сигналы, отбросить неинформативные, шумовые сигналы и в итоге повысить точность решения. При этом коррекция размеров нейронной сети не приводит к полному «забыванию» ранее сформированных при обучении навыков, что ускоряет процесс дальнейшего обучения нейронной сети.

В зависимости от того, какая архитектура у нейронной сети, будут меняться ее свойства и в следствии этого, будет меняться назначение создаваемой сети.

Для решения задачи распознавания и классификации объектов на изображении используют сочетание сверточной нейронной сети и сети прямого распространения.

# **4.1 Сверточная нейронная сеть**

Сверточные нейронные сети (СНС) похожи на другие типы нейронных сетей, потому что они построены на основе нейронов, которые обладают изменяющимся весом и смещениями. Каждый нейрон получает некоторые входные данные и выполняет скалярное произведение информации. Как и в случае с другими типами нейронных сетей, вся СНС выражает одну дифференцируемую функцию взноса (эффективный взнос): с одной стороны, это необработанные пиксели изображения, с другой - вывод класса или группы вероятных классов, характеризующих картинку. В структуре используемой нейронной сети также присутствует функция потери на последнем (полностью подключенном) слое.

Особенность СНС в том, что все необходимые ядра свертки, нужные для выделения необходимых для классификации объектов на изображении определяются автоматически в процессе обучения. Благодаря этому факту, для решения задачи распознавания, лучше всего будет подходить сверточная нейронная сеть.

СНС используют несколько принципов для решения вышеописанных недостатков, а именно: локальное восприятие, разделяемые веса и уменьшение размерности. Если в полносвязной нейронной сети каждый нейрон подключается к каждому пикселю входного изображения, то в СНС нейрон получает на вход только ограниченное количество пикселей, обрабатывая только часть изображения, которая может пересекаться с предыдущим обработанным участком изображения. Операция, выполняемая этими нейронами, называется «свертка».

В архитектуру создаваемой сети заложены основополагающие знания из предметной области компьютерного зрения: пиксели изображения сильнее связаны с соседними и объект на изображении может встретиться в любой части изображения. Описание архитектуры сверточной нейронной сети представлено в таблице 4.1.

Таблица 4.1 Архитектура СНС.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип слоя | Форма ядра свертки | Шаг | Форма выхода | Число обучаемых весов |
| Conv2D | (3, 3) | (1, 1) | (None, 28, 28, 32) | 816 |
| MaxPooling2D | (2, 2) | (2, 2) | (None, 14, 14, 32) | 0 |
| Conv2D | (3, 3) | (1, 1) | (None, 10, 10, 64) | 52 224 |
| MaxPooling2D | (2, 2) | (2, 2) | (None, 5, 5, 64) | 0 |
| Conv2D | (3, 3) | (1, 1) | (None, 3, 3, 64) | 1 671 168 |
| MaxPooling2D | (2, 2) | (2, 2) | (None, 1, 1, 64) | 0 |
| Flatten |  |  |  |  |
| Dense (ReLU) |  |  | (None, 128) | 16 600 |
| Dense (SoftMax) |  |  | (None, 10) | 160 |

Рассмотрим выбранные слои для модели нейронной сети. Всего используется 4 разных типа слоев, а именно:

1) Conv2D – это слой двумерной свертки, который создает сверточное ядро, которое связано с входным слоем для получения вектора выходных данных. Если активация для входов нейронов этого слоя не задана, то она применяется к выходам. При использовании этого слоя в качестве первого слоя в модели необходимо указывать ключевое слово-аргумент input\_shape, которое обозначает количество входов в нейрон.

2) Dense – описание параметров этого слоя изложено в предыдущем пункте.

3) Flatten – Сглаживает ввод. Не влияет на размер партии обучения.

4) MaxPooling2D – уменьшает выборку входного слоя, взяв максимальное значение по окну, заданному параметром pool\_size. Окно сдвигается шагами в каждом измерении.

# **4.2 Слой свёртки**

Операция свертки подразумевает, что рассматривается определенная область изображения, в которой выделятся ядро этой области, которое имеет тот же размер, что и рассматриваемая область. Операция свертки выполняется следующим образом: берутся значения интенсивности пикселей в рассматриваемом участке и умножается на соответствующий элемент ядра свертки, после этого полученные элементы складываются, и мы получаем итоговое значение, которое можно применить в какой-либо нелинейной функцию активации.

Свертка используется для обработки сетей независимо от изображений. Было придумано много разных ядер свертки которые позволяют выполнить различные операции с изображениями, например, размыть, выделить границы, повысить четкость изображения, но при таком подходе к анализу изображения, выявляется проблема того, что необходимо самим придумать ядра свертки, для нужных нам операций.

В данном случае СНС сама определяет какие именно ядра свертки нужны, в зависимости то того, какие данные она обрабатывает.

Все нейроны СНС используют одинаковые ядра свертки для обработки разных участков изображения. Происходит поиск одного определенного выбранного элемента в разных участках изображения и за счет использования одних и тех же весов в ядрах свертки существенно уменьшается количество весов, которые нужно вычислить в процессе обучения.

Уменьшение размерности решает задачу нахождения объекта независимо от того, в каком они масштабе. В нашем случае, цифра на изображении может быть большой и занимать большую часть изображения или малую, но так или иначе необходимо понять, что изображена именно она, а не что-либо другое. Для уменьшения размерности в СНС используются слои подвыборки, в которых один нейрон подключен к нескольким нейронам предыдущего слоя и его значение определятся на основе значений предыдущих подключенных к нему нейронов, для этого используется выбор максимального значения.

Проектируемая СНС будет состоять из нескольких поочередно идущих слоев свертки и подвыборки соединенных между собой. В качестве примера рассмотрим сеть LeNet-5 [5, 6].

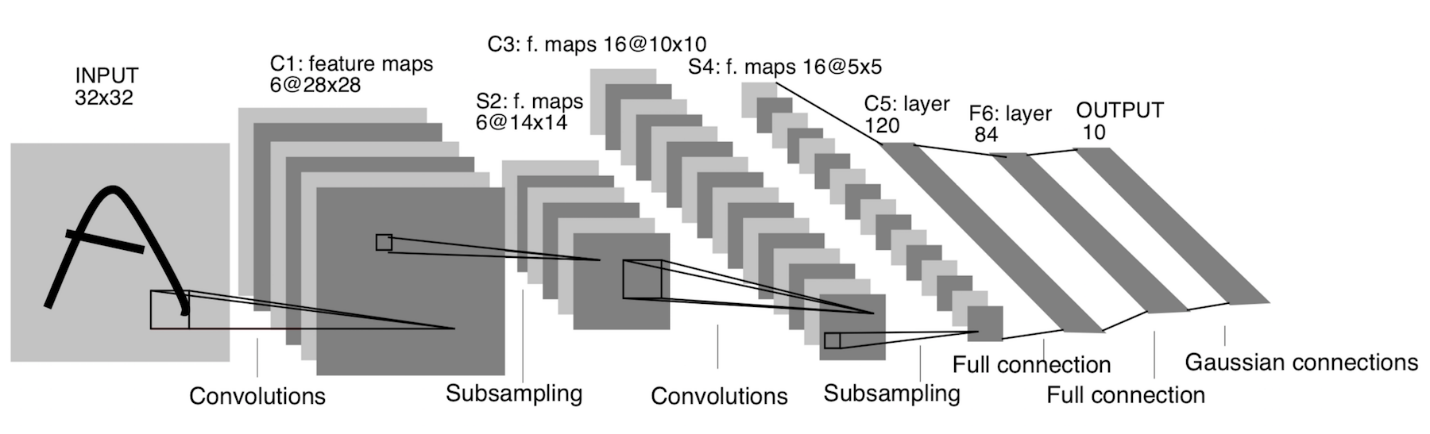


Рисунок 4.2 – Архитектура сверточной нейронной сети LeNet-5

Представленная на рисунке 4.2 для рассмотрения СНС состоит из пяти слоев, которая работает с изображениями размером 32\*32 пикселя. Первый слой свертки использует ядра 5\*5, при чем используется не одно ядро, а шесть разных ядер для поиска разных признаков на изображении. Набор слоев нейронов, каждый из которых использует разное ядро свертки называют картой признаков. Второй слой подвыборки использует усреднение, размер окна которого 2\*2. Третий слой свертки, на котором используется 16 ядер свертки и на следующем слое подвыборки, вновь происходит усреднение с размером окна 2\*2, стоит отметить что на последнем слое содержатся необходимые признаки для распознавания изображений. Для решения задачи классификации используется 2 полносвязных слоя, выходной слой содержит 10 нейронов, каждый из которых говорит о наличии одной из цифр, как и в многослойном персептроне.

Для обучения СНС используется метод обратного распространения ошибки, но с ограничением на веса, которые должны быть одинаковы.

Основным компонентом свёрточной нейронной сети является слой свёртки, включающий в себя свой фильтр для каждого канала. Ядро свёртки фильтра обрабатывает предыдущий слой по фрагментам смещаясь на один пиксель. Весовые коэффициенты ядра свёртки определяются в процессе обучения сети.

Сверточные нейронные сети работают на основе фильтров, которые занимаются распознаванием определенных характеристик изображения. Фильтр - это коллекция ядер. Иногда в фильтре используется одно ядро. Ядро - это обычная матрица чисел, называемых весами, которые «обучаются» с целью поиска на изображениях определенных характеристик. Фильтр перемещается вдоль изображения и определяет, присутствие необходимой характеристики объекта в конкретной его части. Для того, чтобы найти свойственную объекту характеристику совершается операция свертки, которая является суммой произведений элементов фильтра и матрицы входных сигналов.

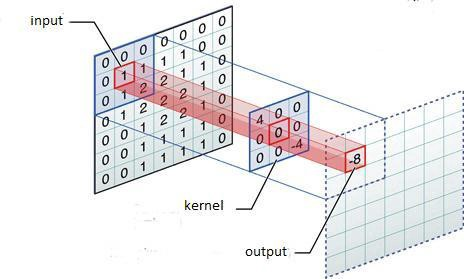


Рисунок 4.3 – Визуализация процесса свертки

Если некоторая необходимая для классификации характеристика присутствует во фрагменте изображения, операция свертки на выходе будет выдавать число с относительно большим значением. Если же характеристика отсутствует, выходное число будет иметь меньшее значение.

# **4.3 Слой объединения**

Нелинейным уплотнением карты признаков, называют слой объединения (пуллинга). Использование слоя объединения позволяет существенно уменьшить пространственный объём изображения. Операция объединения представляется следующим образом: если на предыдущей операции свёртки уже были выявлены некоторые признаки, то для дальнейшей обработки изображение уплотняется до менее подробного. Фильтрация ненужных деталей помогает сети не переобучаться.

Обычно слой объединения ставят между слоями свертки: для реализации обычно используется функция максимума.

# **4.4 Функция активации полносвязных слоев**

Функция ReLU соответствует биологической аналогии того, что нейроны могут находиться в активном состоянии либо нет. Если пройден определенный порог, то срабатывает функция, а если не пройден, то нейрон просто остается неактивным, с активацией равной нулю.

Слой сверточной нейронной сети с функцией активации ReLU можно поставить после всех поочередно идущиъ свёрточных и объединяющих слоев. Функция ReLU (1) выбирается вместо сигмоиды или гиперболического тангенса, поскольку показывает хорошие результаты при обучении нейронных сетей и «отсекает» ненужные детали в канале. Ректификатор, приближение ректификтора, также называемый softplus, и производная softplus – логистическая функция [20, 23].

(1)

,

,

,

# **5. Программная реализация**

Языком разработки был выбран Python по нескольким причинам

Писать на Python просто и быстро, в сравнении с другими языками программирования, например, чтобы описать ИНС на C нужно гораздо большее количество строк кода. Python хорошо подходит для проведения экспериментов, так как код программы легко модифицировать, при этом он остается хорошо читаемым и поддерживаемым.

Программы на Python работают медленнее в сравнении с C [16], однако библиотеки глубокого обучения языка программирования Python используют различные методы для ускорения программ, например, используя математические библиотеки, написанные на C, а некоторые библиотеки динамически генерируют код и выполняют его на другом языке программирования (C) гораздо быстрее. Например, для повышения производительности, можно использовать GPU графические ускорители. В качестве редактора исходного кода используется Visual Studio Code [19].

# **5.1 Описание программных библиотек**

Обработка изображений направлена на преобразование пиксельных, визуальных изображений в формальное представление для облегчения обработки информации компьютером.

Современные сверточные нейронные сети умеют осуществлять решение задач различного рода, такие как: извлечение информации, машинный перевод, обобщение, поиск и «человеко-компьютерное» взаимодействие. Они варьируются от синтактических задач, например, частичная речевая маркировка, разборов текста (parsing), до смысловых, например, смысловое значение слова, тональность текста.

Для решения рассматриваемой задачи распознавания рукописных цифр в результате анализа для реализации нейронной сети были отобраны и используются следующие программные библиотеки:

1) TensorFlow (TF) - открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия. Применяется как для исследований, так и для разработки собственных продуктов Google. Основной API для работы с библиотекой реализован для Python, также существуют реализации для C++, Java, Go и Swift [10].

2) NumPy - это библиотека для научных вычислений на Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для операций с этими массивами [8].

Помимо очевидного научного использования, NumPy использован в качестве эффективного многомерного контейнера общих данных. Произвольные типы данных могут быть определены. Это позволяет NumPy легко и быстро интегрироваться с широким спектром баз данных.

3) Keras – открытая библиотека, написанная на языке Python. Она представляет собой надстройку над фреймворками Deeplearning4j, TF и Theano. Использует TF или Theano для выполнения вычислений. Нацелена на оперативную работу с сетями глубокого обучения, при этом спроектирована так, чтобы быть компактной, модульной и расширяемой. Keras был разработан с целью обеспечения быстрого экспериментирования. Быть способным перейти от идеи к результату как можно быстрее - это ключ к проведению хороших исследований. Моя задача описать модель, выставить параметры, а Keras уже построит необходимую сеть, для проведения вычислений будет вызывать высокоэффективные методы из TF [9].

# **5.2 Нейронная сеть прямого распространения**

Персептроны при объединении в сеть образуют многослойный персептрон, который принято называть НС прямого распространения (FFNN, feed forward neural network).

Для проведения обучения НС прямого распространения необходимо указать данные на которых обучается сеть, изображения с рукописными цифрами. В исходном коде программы они содержатся в переменной X\_train. В переменной y\_train содержатся правильные ответы для изображений из тестового набора данных.

Перед тем как обучить нейронную сеть, необходимо выполнить предобработку входных данных, неразмеченных изображений. Для этого нужно изменить размерность входного массива в виде одномерного массива из 784 пикселей. Далее данные нормализуются для получения наилучшего качества обучения. Признаками распознаваемых образов можно выделить тип и класс распознаваемого образа. Интенсивность каждого пикселя будет располагаться в диапазоне от 0 до 1.

Следующим этапом преобразуются правильные ответы для представления по классам, так как это связано с тем, что на выходе НС выдает не одно число или метку, а десять выходных сигналов, соответствующие десяти классам изображений, поэтому необходимо преобразовать метки в категориальное представление – массив из 10 элементов, в котором все элементы будут равны 0, за исключением одного, который является правильным ответом.

После подготовки входных данных, можно описать НС. Для описания НС в Keras использует модели. Мы используем последовательную модель Sequential, в которой слои идут друг за другом, образуя полносвязную нейронную сеть. Данная сеть имеет архитектуру (таблица 5.1), состоящую из двух слоев: входной и выходной.

Таблица 5.1 Архитектура первого прототипа используемой НС прямого распространения.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип слоя | Функция активации | Количество нейронов | Число обучаемых весов |
| Dense | ReLU | 800 | 627 200 |
| Dense | SoftMax | 10 | 8 000 |

Первый, входной слой содержит 800 нейронов, у каждого из нейронов 784 входа. Входов так много, потому что качестве входных сигналов используются значения интенсивности пикселей изображения, всего их 784, это количество определено форматом изображения – 28\*28 пикселей. Веса входных значений инициализируются случайно, с помощью нормального распределения, в качестве функции активации используется ReLU.

Второй слой использует 10 нейронов, по количеству распознаваемых цифр от 0 до 9. Значение на выходе соответствует диапазону вероятности от 0 до 1, следовательно, вероятность на выходе будет стремиться к 1 в случае если на изображении показана определенная цифра. Веса инициализируются с помощью нормального распределения, но используется функция активации SoftMax (нормализованная экспоненциальная функция).

Оба слоя имеют тип Dense, что означает, что выполняется соединение всех нейронов одного слоя, со всеми нейронами следующего слоя.

Описав модель, ее необходимо скомпилировать, а также задать параметры, используемые при обучении, а именно: метод обучения – стохастический градиентный спуск, меру ошибки, и обозначить метрику, параметры по которым будем выполнять последующую оптимизацию и сравнение прототипов.

Оптимизация производится, используя метод стохастического градиентного спуска, размер выборки составляет 200 элементов, это означает, что, анализируя 200 изображений определяется направление градиента, и в соответствии с этим направлением, выполняется изменение весов. Количество эпох варьируется в разных экспериментах, их количество указывает сколько раз выполняется обучение при помощи выбранного набора данных, verbose – печатает диагностическую информацию в процессе обучения.

Например, разные варианты построения НС прямого распространения могут отличаться количеством слоев, в частном случае, добавлением слоя Dropout, который представляет из себя метод регуляризации, в таблице 5.2 представлена архитектура второго прототипа нейронной сети.

Таблица 5.2. Архитектура второго прототипа используемой НС прямого распространения.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип слоя | Функция активации | Количество нейронов | Коэффициент | Число обучаемых весов |
| Dense | ReLU | 1024 | - | 802 816 |
| Dropout | - | - | 0.2 | 160 563 |
| Dense | ReLU | 1024 | - | 164 416 512 |
| Dropout | - | - | 0.2 | 32 883 302 |
| Dense | SoftMax | 10 | - | 328 833 020 |

Дропаут (dropout) - простой и эффективный метод регуляризации, дополняющий представленные ранее методы. Основной принцип состоит в том, что в процессе обучения из структуры нейронной сети случайным образом многократно выделяется подсеть, и обновление весов выполняется только в рамках этой подсети.

Нейроны попадают в подсеть с определенной вероятностью, которая называется коэффициентом дропаута. Во время проходов по тестовому набору данных дропаут не применяется, вместо этого веса умножаются на коэффициент дропаута, в результате чего можно получить усредненную оценку для объединенных подсетей. На практике коэффициент дропаута p обычно выбирают равным 0,5, но его можно подобрать с помощью валидационного набора данных. Во время тестирования прототипов выяснилось, что при обучении на выбранном наборе данных, оптимальным значением коэффициента дропаута будет 0,2.

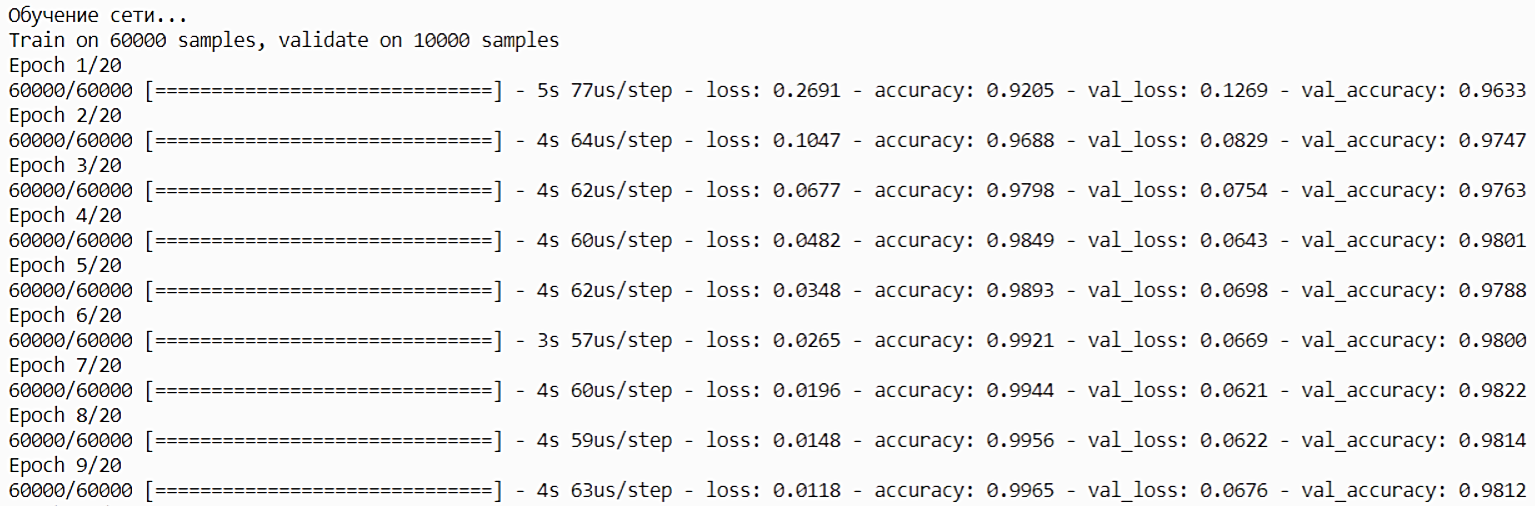


Рисунок 5.1 – Диагностическая информация об обучении сети прямого распространения

На данном снимке экрана (Рисунок 5.1) продемонстрированы значения метрики, по которым можно определить на сколько хорошо обучается нейронная сеть. Для более удобного представления этого процесса, по итогу обучения составляется графики (Рисунок 5.2, 5.3).

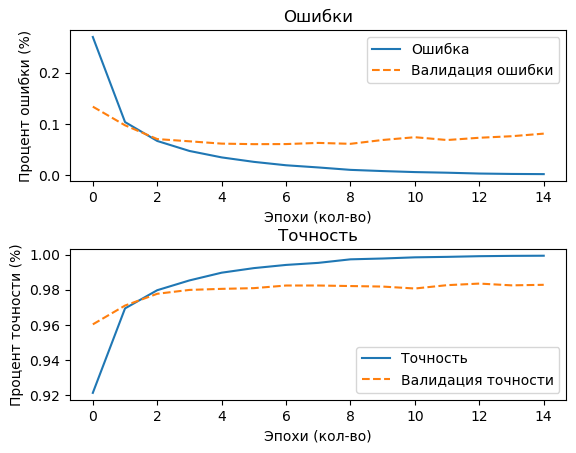


Рисунок 5.2 – Графики обучаемости первого прототипа нейронной сети прямого распространения, обучение на 15 эпохах

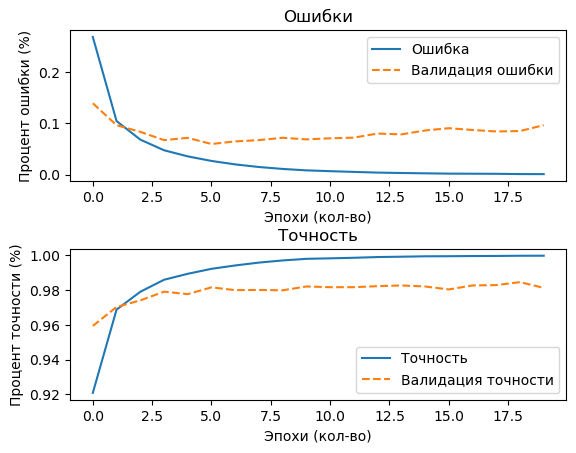


Рисунок 5.3 – Графики обучаемости первого прототипа нейронной сети прямого распространения, обучение на 20 эпохах

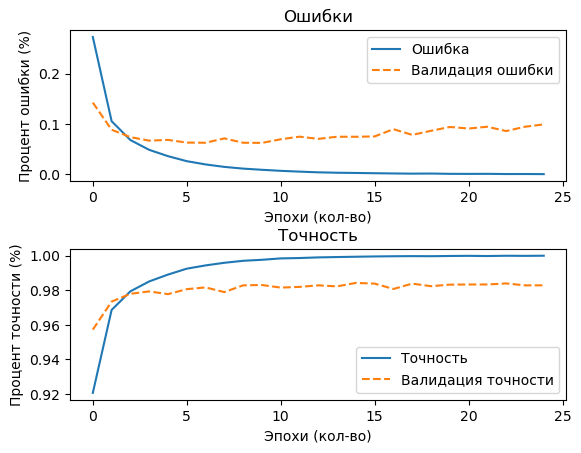


Рисунок 5.4 – Графики обучаемости первого прототипа нейронной сети прямого распространения, обучение на 25 эпохах

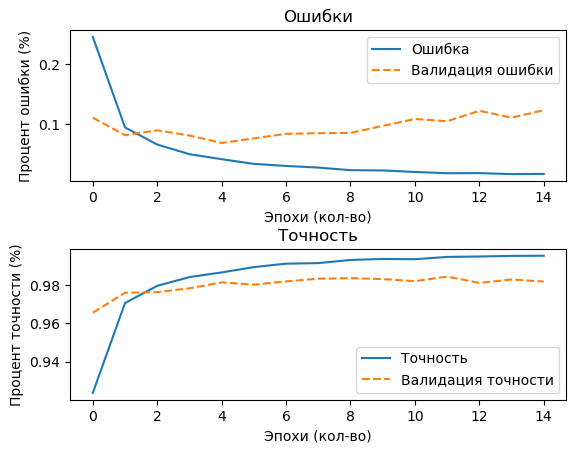


Рисунок 5.5 – Графики обучаемости второго прототипа нейронной сети прямого распространения, обучение на 15 эпохах

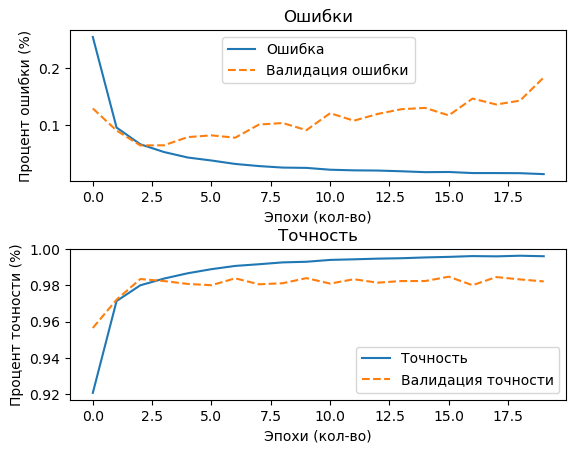


Рисунок 5.6 – Графики обучаемости второго прототипа нейронной сети прямого распространения, обучение на 20 эпохах

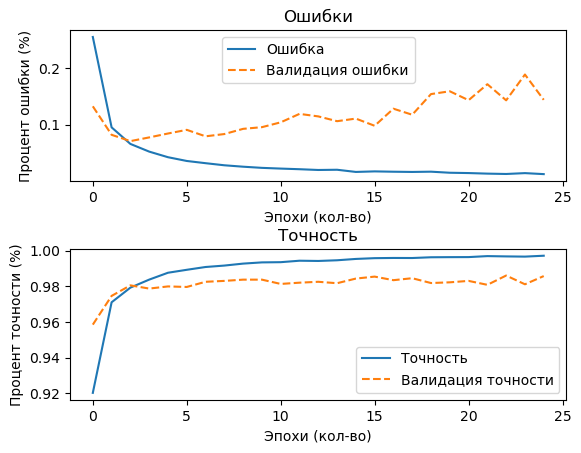


Рисунок 5.7 – Графики обучаемости второго прототипа нейронной сети прямого распространения, обучение на 25 эпохах

Созданные графики (Рисунки 5.2 – 5.7) показывают изменение значений функции потерь и долю корректных прогнозов нейронной сети. По горизонтальной оси обозначено количество итерации обучения, а по вертикальной точность обучения, измеряемой в процентах. Проанализировав графики, можно сделать вывод о хорошей обученности каждого из прототипов сети. Главное отличие обучения прототипов состоит в том, что с каждым разом изменяется количество итераций обучения, в зависимости от этого, у обучаемых прототипов сетей меняются их значения точности и ошибки. Эти эксперименты необходимы для того, чтобы выбрать наилучшее количество эпох обучения, для созданной архитектуры, а также, убедиться в том, что сеть не переобучается, имеет необходимую точность. Отслеживаемые параметры влияют на итоговое распознавание и классификацию рукописных цифр, в этом можно убедиться наблюдая ответы сети на рисунках 5.8, 5.9.

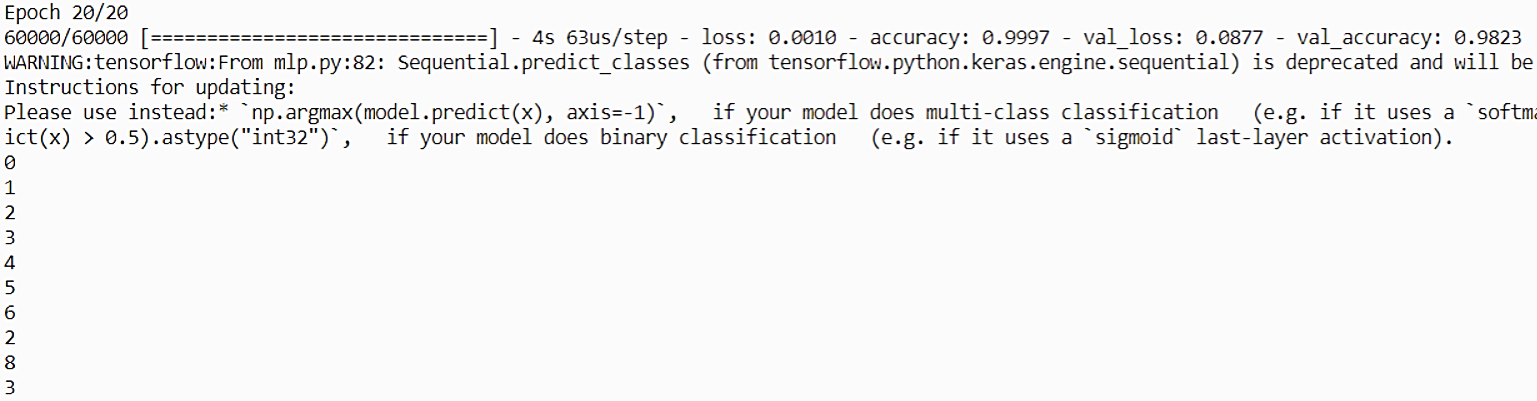


Рисунок 5.8 – Результаты распознавания изображений первым прототипом нейронной сети прямого распространения

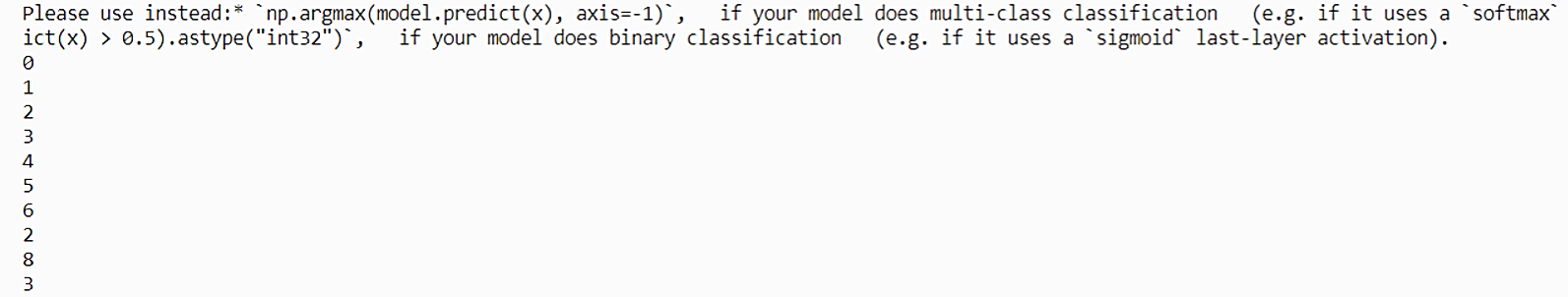


Рисунок 5.9 – Результаты распознавания изображений вторым прототипом нейронной сети прямого распространения

После прохождения этапа обучения, для сети предоставляются изображения цифр. Эти изображения рукописные, созданы в редакторе Paint в соответствии с изображениями из набора данных. Из ответов (Рисунок 5.4, 5.5) обоих прототипов сети можно сделать вывод о том, что хоть сеть и хорошо показала себя на тестовой выборке, однако если ей показать более реальную картину, то точность выполнения задачи снижается.

# **5.3 Тестирование прототипа сверточной нейронной сети**

Сети прямого распространения имеют два основных недостатка для решаемой задачи распознавания рукописного ввода цифр на изображениях.

Рассмотрев первый и второй прототип, можно выделить первый недостаток, который заключается в том, что сети такого типа имеют большое количество весов, которые необходимо обучить, это определяется количеством входов на каждый нейрон входного слоя – 784 (28\*28 пикселей, размер изображения), количеством связей используемых нейронов на первом, и втором слое – 800 и 10 соответственно. В конечном итоге будет 635200 весов (784\*800 + 800\*10 = 635200), в первом прототипе, во втором больше в несколько раз, так как нейронов еще больше. При таком соотношении входных данных и итогового количества весов, с которыми работает система, снижается надежность и увеличивается вероятность ошибки. Если работать с изображениями больших размеров, чем рассмотренные для примера, то весов для обучения станет в несколько раз больше.

Второй недостаток состоит в том, что изображение представляется в виде одномерного вектора или плоского массива, в следствии чего теряется важная информация для обработки изображения и топологии НС. Производится анализ пикселей только в вертикальной плоскости, не в горизонтальной, при работе с изображениями важно учитывать обе плоскости.

Для устранения этих недостатков был создан и протестирован прототип СНС, такие сети являются альтернативой, преимущественно использующиеся на практике для анализа изображений и видео.

Для того, чтобы доказать преимущество таких сетей и судить об обучаемости сети на используемом наборе данных, необходимо рассмотреть выдаваемою сетью метрику (Рисунок 5.10). Для более удобного анализа данных сети, создаются графики (Рисунки 5.11 – 5.13) сразу после обучения сети.

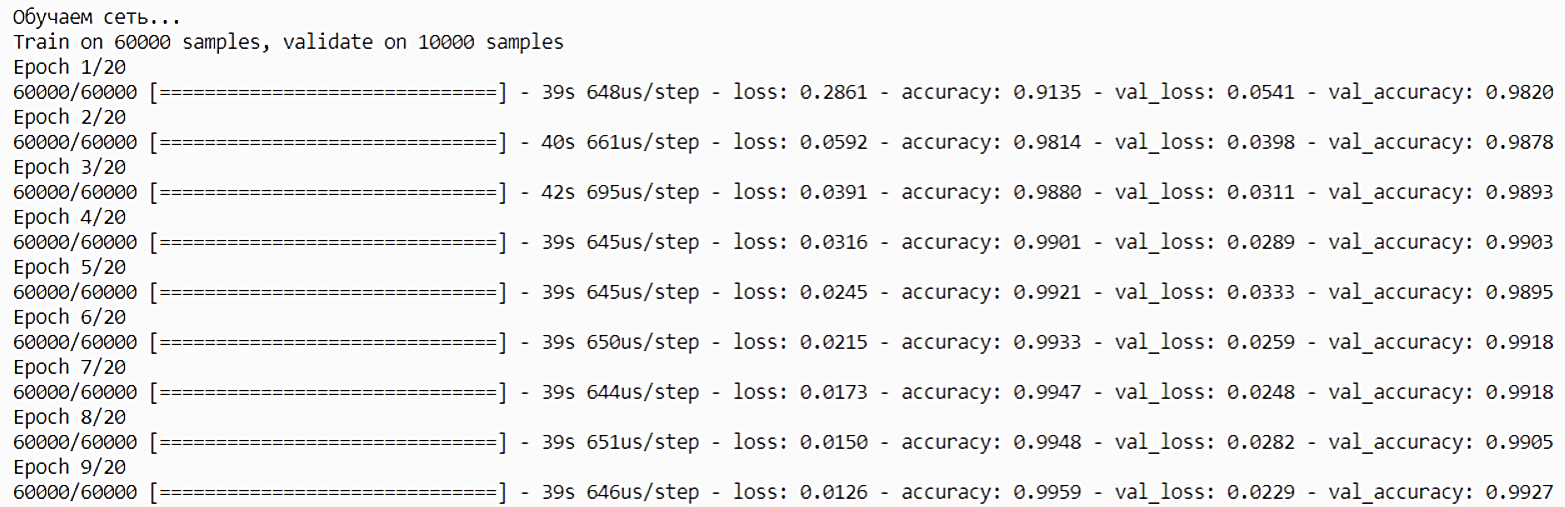


Рисунок 5.10 – Диагностическая информация об обучении сверточной нейронной сети

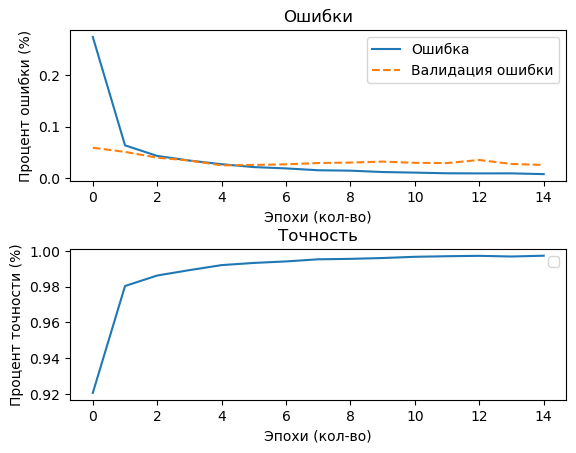


Рисунок 5.11 – Графики обучаемости прототипа сверточной нейронной сети, на 15 эпохах обучения

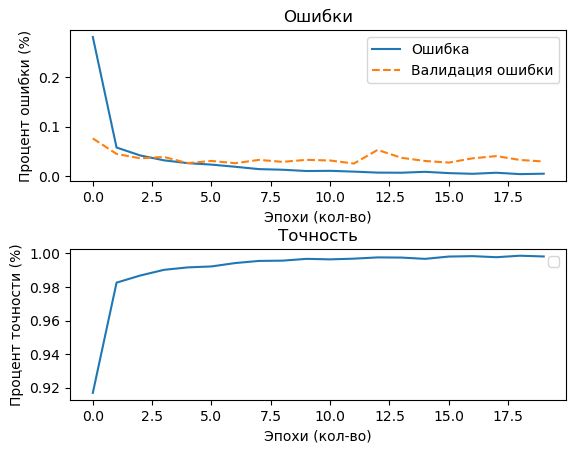


Рисунок 5.12 – Графики обучаемости прототипа сверточной нейронной сети, на 20 эпохах обучения

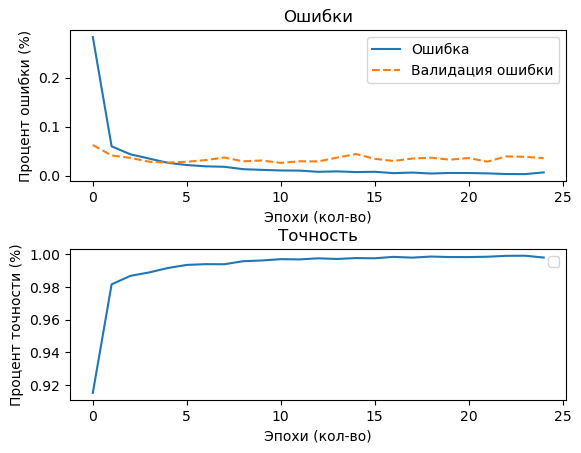


Рисунок 5.13 – Графики обучаемости прототипа сверточной нейронной сети, на 25 эпохах обучения

По горизонтальной и вертикальной осям обозначено количество итерации обучения и точность обучения в процентах соответственно. По данным графикам (Рисунки 5.11 – 5.13) можно свидетельствовать о хорошей обучаемости нейронной сети.

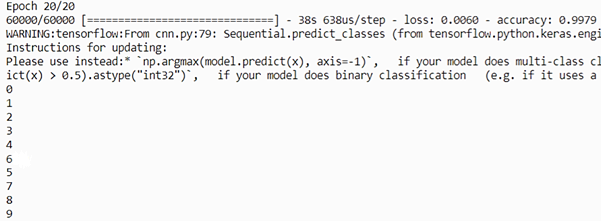


Рисунок 5.14 – Результаты распознавания цифр на изображении прототипом сверточной нейронной сети

Проанализировав графики по графикам (Рисунки 5.11 – 5.13) и результаты распознавания нейронной сети (Рисунок 5.14), не сложно определить, что сверточная нейронная сеть в отличии от приведенной в пример НС прямого распространения, лучше справляется с распознаванием рукописных цифр. Так же можно определить явную зависимость точности ответов прототипа сверточной нейронной сети от количества итераций обучения, однако слишком малое количество не сможет обеспечить сеть достаточной точностью, а слишком большое приведет к переобучению.

# **6.** **Сравнение и оценка результатов**

В результате проделанной работы, были обучены и протестированы две различные структуры нейронных сетей, а также несколько прототипов для достижения лучшего результата распознавания. В нескольких проведенных экспериментах изменялись разновидности слоев, количество нейронов, функции активации, размеры ядер свертки, количество эпох обучения.

Для итогового сравнения выбраны наиболее хорошо обученные модели ИНС и для их сравнения возьмем часто используемые показатели качества обучения:

1) Доля корректных прогнозов (ошибка) – процент ошибок, допускаемых классификатором.

2) Мера точности (точность) – характеризует количество правильных ответов, полученных от классификатора из всего используемого набора данных.

3) Мера полноты (валидация) – определяет способность классификатора определять, как можно большее число положительных ответов из ожидаемых.

4) Мера несоответсвий (ошибки валидации) – выявленные при проверке ошибки, несоответствия, которые влияют на корректность результатов валидации.

Таблица 6.1 – Эксперименты над НС прямого распространения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Эпоха | Ошибка  (%, процент) | Точность  (%, процент) | Ошибки валидации  (%, процент) | Валидация  (%, процент) |
| 1 | 26.91 | 92.05 | 12.69 | 96.33 |
| 2 | 10.47 | 96.88 | 8.77 | 97.47 |
| 3 | 6.77 | 97.98 | 8.29 | 97.63 |
| 4 | 4.82 | 98.49 | 7.54 | 98.01 |
| 5 | 3.48 | 98.93 | 6.98 | 98.00 |
| Последняя | 0.10 | 99.97 | 6.43 | 98.22 |

Таблица 6.2 – Эксперименты над СНС

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Эпоха | Ошибка  (%, процент) | Точность  (%, процент) | Ошибки валидации  (%, процент) | Валидация  (%, процент) |
| 1 | 28.61 | 91.35 | 5.41 | 98.20 |
| 2 | 5.92 | 98.14 | 4.07 | 98.78 |
| 3 | 3.91 | 98.80 | 3.98 | 98.93 |
| 4 | 2.45 | 99.01 | 3.77 | 98.95 |
| 5 | 1.73 | 99.21 | 2.85 | 99.06 |
| Последняя | 0.60 | 99.79 | 2.76 | 99.11 |

В результате экспериментов, основанных на показателях качества, был выбран наиболее совершенный прототип ИНС для решения поставленной задачи. На таблицах 6.1 и 6.2 показано как ведут себя сети при прохождении итераций обучения, для удобства представлены первые пять и последняя итерация эталонных прототипов обоих типов сетей.

# **Заключение**

Целью выпускной квалификационной работы является реализация сверточной нейронной сети, позволяющей распознавать рукописные цифры в соответствии с поставленной задачей «разработка сверточной нейронной сети, обеспечивающей распознавание цифр на изображениях».

В работе представлено описание предметной области. В процессе исследования было определено, какие входные данные нужны для обучения сети.

На основе проведенного исследования обоснована необходимость проведенной работы. Выполнен анализ других архитектур нейронных сетей, который показал, что для решения поставленной задачи подходит только сверточная нейронная сеть.

В заключительной части работы описан процесс разработки созданной архитектуры, проведено тестирование разработанных прототипов нейронных сетей.

Данная работа снижает влияние человеческого фактора при обработке данных, способствует ускорению процесса обработки, а также предоставляет возможность оптимизировать рабочий процесс.

Разработанное программное обеспечение может найти широкое применение в различных сферах, структурах, организациях, занимающихся обработкой документов, заполняемых ручным способом, например, в МФЦ, почте, правоохранительных органах и так далее.

Помимо распознавания рукописных цифр в документации, эта работа может найти применение в системах универсального доступа для инвалидов по зрению, пожилых людей или цифровизации архивных рукописей имеющих историческую ценность и тому подобных.

Созданная архитектура нейронной сети предусматривает дальнейшее совершенствование. Возможности усовершенствования архитектуры, на мой взгляд, таковы – данная сеть могла бы распознавать не только цифры, но и числа, буквы алфавита, специальные знаки. Однако для этого необходимо изменить набор данных для обучения, дополнить структуру необходимыми модулями. Таким модулем может быть обработчик изображений, который сможет унифицировать входные данные.

Все задачи выпускной квалификационной работы выполнены. Цель работы достигнута.

# **Список использованных источников**

1. Van Veen, F., Leijnen, S. (2019). The Neural Network Zoo. [Электронный ресурс]. Режим доступа: <http://www.asimovinstitute.org/neural-network-zoo/> (дата обращения 13.02.20)
2. LeCun, Yann, et al. «Gradient-based learning applied to document recognition» [Электронный ресурс]. Режим доступа: <http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf> (дата обращения 13.02.20)
3. LeCun, «Back - propagation applied to handwritten zip code recognition» et al. 1989
4. Deep Neural Networks Applications in Handwriting Recognition Théodore Bluche São Paulo Meetup - 9 Mar. 2017 стр. 44
5. Krizhevsky, Alex, et al. «ImageNet Classification with Deep Convolutional Neural Networks» Communications of the ACM, vol. 60, no. 6, 2017, стр. 84–90., doi:10.1145/3065386
6. He, Kaiming, et al. «Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification» 2015 IEEE International Conference on Computer Vision (ICCV)*,* 2015, doi:10.1109/iccv.2015.123
7. Созыкин А.В. Обзор методов обучения глубоких нейронных сетей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 3. С. 32 - 40. DOI: 10.14529/cmse170303
8. Библиотека с открытым исходным кодом для языка программирования Python NumPy [Электронный ресурс]. Режим доступа: <https://numpy.org/index.html> (дата обращения 16.06.2020)
9. Библиотека с открытым исходным кодом для языка программирования Python API Keras [Электронный ресурс]. Режим доступа: <https://keras.io/> (дата обращения 16.06.2020)
10. Открытая программная библиотека для машинного обучения TensorFlow [Электронный ресурс]. Режим доступа: <https://www.tensorflow.org/> (дата обращения 16.06.2020)
11. База данных MNIST [Электронный ресурс]. Режим доступа: <http://yann.lecun.com/exdb/mnist/> (дата обращения 16.06.2020)
12. ГОСТ 19.201-78 «Техническое задание. Требования к содержанию»
13. Федеральный закон от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации»
14. Приказ Министерства образования и науки Российской Федерации от 05.04.2017 № 301 «Об утверждении Порядка организации и осуществления образовательной деятельности по образовательным программам высшего образования — программам бакалавриата, программам специалитета, программам магистратуры»
15. Методические рекомендация по разработке основных профессиональных образовательных программ и дополнительных профессиональных программ с учетом соответствующих профессиональных стандартов, утвержденных Министром образования и науки Российской Федерации Д.В. Ливановым от 22.01.2015 № ДЛ-1/05 вн.
16. Дьяченко К.Е. Персональный компьютер. – М.: ИнтерПресс, 2012. – 240 с.
17. Орлов, С.А. Технологии разработки программного обеспечения: учеб. / С.А. Орлов. – СПб. : Питер, 2014. – 464 с.
18. Официальный сайт ФГБОУ ВО «Югорский государственный университет» [Электронный ресурс]. Режим доступа: <https://www.ugrasu.ru/> (дата обращения 16.06.2020)
19. Microsoft Visual Studio Code [Электронный ресурс]. Режим доступа: <http://microsoft.ru/> (дата обращения :) (дата обращения 17.06.2020)
20. Головинский, П. А. Математические модели. Теоретическая физика и анализ сложных систем. Книга 2. От нелинейных колебаний до искусственных нейронов и сложных систем / П.А. Головинский. - М.: Либроком, 2012. - 234 c.
21. Круглов, В.В. Искусственные нейронные сети. Теория и практика: моногр. / В.В. Круглов, В.В. Борисов. - М.: Горячая линия - Телеком; Издание 2-е, стер., 2002. - 382 c.
22. Нейронные сети. Statistica Neural Networks. Методология и технологии современного анализа данных. - М.: Горячая линия - Телеком, 2008. - 392 c.
23. Гелиг, А. Х. Введение в математическую теорию обучаемых распознающих систем и нейронных сетей. Учебное пособие: моногр. / А.Х. Гелиг, А.С. Матвеев. - М.: Издательство СПбГУ, 2014. - 224 c
24. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. - М.: Горячая линия - Телеком, 2013. - 384 c.
25. Дэвид Форсайт, Жан Понс. Компьютерное зрение. Современный подход. М.: Издательский дом "Вильямс", 2004. 928 с. Англ вар.: David A. Forsyth, Jean Ponce. Computer Vision: A Modern Approach Б. Яне. Цифровая обработка изображений. /М.: Техносфера, 2007. с. 41-163
26. Эдвард А. Патрик. Основы теории распознавания образов. /М.: Советское радио, 1980. 408 с.
27. Р. Гонсалес, Р. Вудс. Цифровая обработка изображений. М.: Техносфера, 2006. с. 1104
28. Л. Шапиро, Дж. Стокман. Компьютерное зрение. М.: Бином. Лаборатория знаний, 2006 . 725 с. Англ вар.: Shapiro L., Stockman G. Computer Vision
29. Вапник В.Н., Червоненкис А.Я. – Теория распознавания образов и стохастические проблемы обучения. – M.: Наука, 1974 г. – 416 с.
30. А. С. Потапов - Распознавание образов и машинное восприятие. – М.: Политехника, 2007 г. – 552 с.
31. А. Б. Мерков - Распознавание образов. Введение в методы статистического обучения. – М.: Едиториал УРСС, 2011 г. – 256 с.

# **Приложение А**

Исходный код файла mlp.py

from tensorflow import keras #импорт библиотеки keras, основной инструмент для создания нейронных сетей

from keras import layers

from keras.models import Sequential

from keras.layers import Conv2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Activation

from keras.layers import BatchNormalization

from keras.layers import AveragePooling2D

from keras.optimizers import SGD

from keras.optimizers import RMSprop

from keras.optimizers import Adam

import tensorflow as tf

import numpy as np

import keras

import matplotlib.pyplot as plt

import tensorflow\_datasets as tfds

def mnist\_make\_model(image\_w: int, image\_h: int): #первый прототип сети прямого распространения

model = Sequential()

model.add(Dense(800, activation='relu', input\_shape=(image\_w\*image\_h,)))#входной слой, 28\*28=784 размер изображения подаваемого на вход

model.add(Dense(10, activation='softmax'))#выходной слой, 10 нейронов тк надо определять диапозон цифр от 0 до 9

model.compile(loss='categorical\_crossentropy', optimizer=RMSprop(), metrics=['accuracy'])

return model

def mnist\_make\_model2(image\_w: int, image\_h: int): #второй прототипа сети прямого распространения

model = Sequential()

model.add(Dense(1024, activation='relu', input\_shape=(image\_w\*image\_h,))) #нейронов больше, чем в первом прототипе

model.add(Dropout(0.2)) #дополнительный слой в сравнении с первым прототипом

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer=RMSprop(), metrics=['accuracy'])

return model

def mnist\_mlp\_train(model): #обучение mlp сети

(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.mnist.load\_data()# x\_train: 60000x28x28 массив, x\_test: 10000x28x28 массив

image\_size = x\_train.shape[1]

train\_data = x\_train.reshape(x\_train.shape[0], image\_size\*image\_size)

test\_data = x\_test.reshape(x\_test.shape[0], image\_size\*image\_size)

train\_data = train\_data.astype('float32')

test\_data = test\_data.astype('float32')

train\_data /= 255.0

test\_data /= 255.0

num\_classes = 10# кодируем классы - всего на выходе 10 классов

# для 3 класса -> [0 0 0 1 0 0 0 0 0 0], а для 5 -> [0 0 0 0 0 1 0 0 0 0]

train\_labels\_cat = keras.utils.to\_categorical(y\_train, num\_classes)

test\_labels\_cat = keras.utils.to\_categorical(y\_test, num\_classes)

print("Обучение сети...")

#t\_start = time.time()

history = model.fit(train\_data, train\_labels\_cat, epochs=20, batch\_size=200, verbose=1, validation\_data=(test\_data, test\_labels\_cat))

# начинается обучение сети

plt.figure(0) #создание графика обучения

plt.subplot(2, 1, 1)

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'], '--')

plt.title('Loss')

plt.legend(['Loss', 'Val loss'])

plt.subplot(2, 1, 2)

plt.title('Acc')

plt.plot(history.history['accuracy'])

plt.savefig('mlp.png', bbox\_inches='tight')

#print("Done, dT:", time.time() - t\_start)

model = mnist\_make\_model(image\_w=28, image\_h=28)#запись созданной и обученной модели в файл

mnist\_mlp\_train(model)

model.save('mlp\_784.h5')

def mlp\_digits\_predict(model, image\_file): #распознавания изображения из файла для сети

image\_size = 28

img = keras.preprocessing.image.load\_img(image\_file, target\_size=(image\_size, image\_size), color\_mode='grayscale')

img\_arr = np.expand\_dims(img, axis=0)

img\_arr = 1 - img\_arr/255.0

img\_arr = img\_arr.reshape((1, image\_size\*image\_size))

result = model.predict\_classes([img\_arr])

return result[0]

model = tf.keras.models.load\_model('mlp\_784.h5') #загрузка модели и изображений для распознавания первой обученной сетью

print(mlp\_digits\_predict(model, '0.png'))

print(mlp\_digits\_predict(model, '1.png'))

print(mlp\_digits\_predict(model, '2.png'))

print(mlp\_digits\_predict(model, '3.png'))

print(mlp\_digits\_predict(model, '4.png'))

print(mlp\_digits\_predict(model, '5.png'))

print(mlp\_digits\_predict(model, '6.png'))

print(mlp\_digits\_predict(model, '7.png'))

print(mlp\_digits\_predict(model, '8.png'))

print(mlp\_digits\_predict(model, '9.png'))

# **Приложение Б**

Исходный код файла cnn.py

from tensorflow import keras

from keras import layers

from keras.models import Sequential

from keras.layers import Conv2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Activation

from keras.layers import BatchNormalization

from keras.layers import AveragePooling2D

from keras.optimizers import SGD, RMSprop, Adam

import tensorflow\_datasets as tfds # pip install tensorflow-datasets

import tensorflow as tf

import matplotlib.pyplot as plt

import numpy as np

import keras

def mnist\_cnn\_model():#создание сверточной сети

image\_size = 28

num\_channels = 1 #1 for grayscale images

num\_classes = 10 # количество классов, цифр для распознавания

model = Sequential()

model.add(Conv2D(filters=32, kernel\_size=(3,3), activation='relu', padding='same', input\_shape=(image\_size, image\_size, num\_channels)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu', padding='same'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu', padding='same'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten()) #полносвязный слой

model.add(Dense(128, activation='relu'))#выходной слой

model.add(Dense(num\_classes, activation='softmax'))

model.compile(optimizer=Adam(), loss='categorical\_crossentropy', metrics=['accuracy'])

return model

def mnist\_cnn\_train(model): #обучение сверточной сети

(train\_digits, train\_labels), (test\_digits, test\_labels) = keras.datasets.mnist.load\_data()

image\_size = 28 #размер изображения

num\_channels = 1 # 1 для серых изображений

train\_data = np.reshape(train\_digits, (train\_digits.shape[0], image\_size, image\_size, num\_channels)) #изменение масштаба и формы данных изображений

train\_data = train\_data.astype('float32') / 255.0 #кодируем классы - всего на выходе 10 классов

#для 3 класса 3 -> [0 0 0 1 0 0 0 0 0 0], 5 -> [0 0 0 0 0 1 0 0 0 0]

num\_classes = 10

train\_labels\_cat = keras.utils.to\_categorical(train\_labels, num\_classes)

val\_data = np.reshape(test\_digits, (test\_digits.shape[0], image\_size, image\_size, num\_channels))#изменение масштаба и формы валидационных данных изображений

val\_data = val\_data.astype('float32') / 255.0

val\_labels\_cat = keras.utils.to\_categorical(test\_labels, num\_classes)#кодируем классы - всего на выходе 10 классов

print("Обучаем сеть...")

#t\_start = time.time()# начинается обучение сети

history = model.fit(train\_data, train\_labels\_cat, epochs=100, batch\_size=200, validation\_data=(val\_data, val\_labels\_cat))

plt.figure(0) #создание графиков после обучения

plt.subplot(2, 1, 1)

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'], '--')

plt.title('Loss')

plt.legend(['Loss', 'Val loss'])

plt.subplot(2, 1, 2)

plt.title('Acc')

plt.plot(history.history['accuracy'])

plt.savefig('cnn.png', bbox\_inches='tight')

#print("Done, dT:", time.time() - t\_start)

return model

model = mnist\_cnn\_model()#запись созданной и обученной модели в файл

mnist\_cnn\_train(model)

model.save('cnn.h5')

def cnn\_digits\_predict(model, image\_file):#для распознавания приводим изображение к формату чб изображение 28x28 пикселей

image\_size = 28

img = keras.preprocessing.image.load\_img(image\_file, target\_size=(image\_size, image\_size), color\_mode='grayscale')

img\_arr = np.expand\_dims(img, axis=0)

img\_arr = 1 - img\_arr/255.0

img\_arr = img\_arr.reshape((1, 28, 28, 1))

result = model.predict\_classes([img\_arr])

return result[0]

model = tf.keras.models.load\_model('cnn.h5') #загрузка модели и изображений для распознавания третей обученной сетью

print(cnn\_digits\_predict(model, '0.png'))

print(cnn\_digits\_predict(model, '1.png'))

print(cnn\_digits\_predict(model, '2.png'))

print(cnn\_digits\_predict(model, '3.png'))

print(cnn\_digits\_predict(model, '4.png'))

print(cnn\_digits\_predict(model, '5.png'))

print(cnn\_digits\_predict(model, '6.png'))

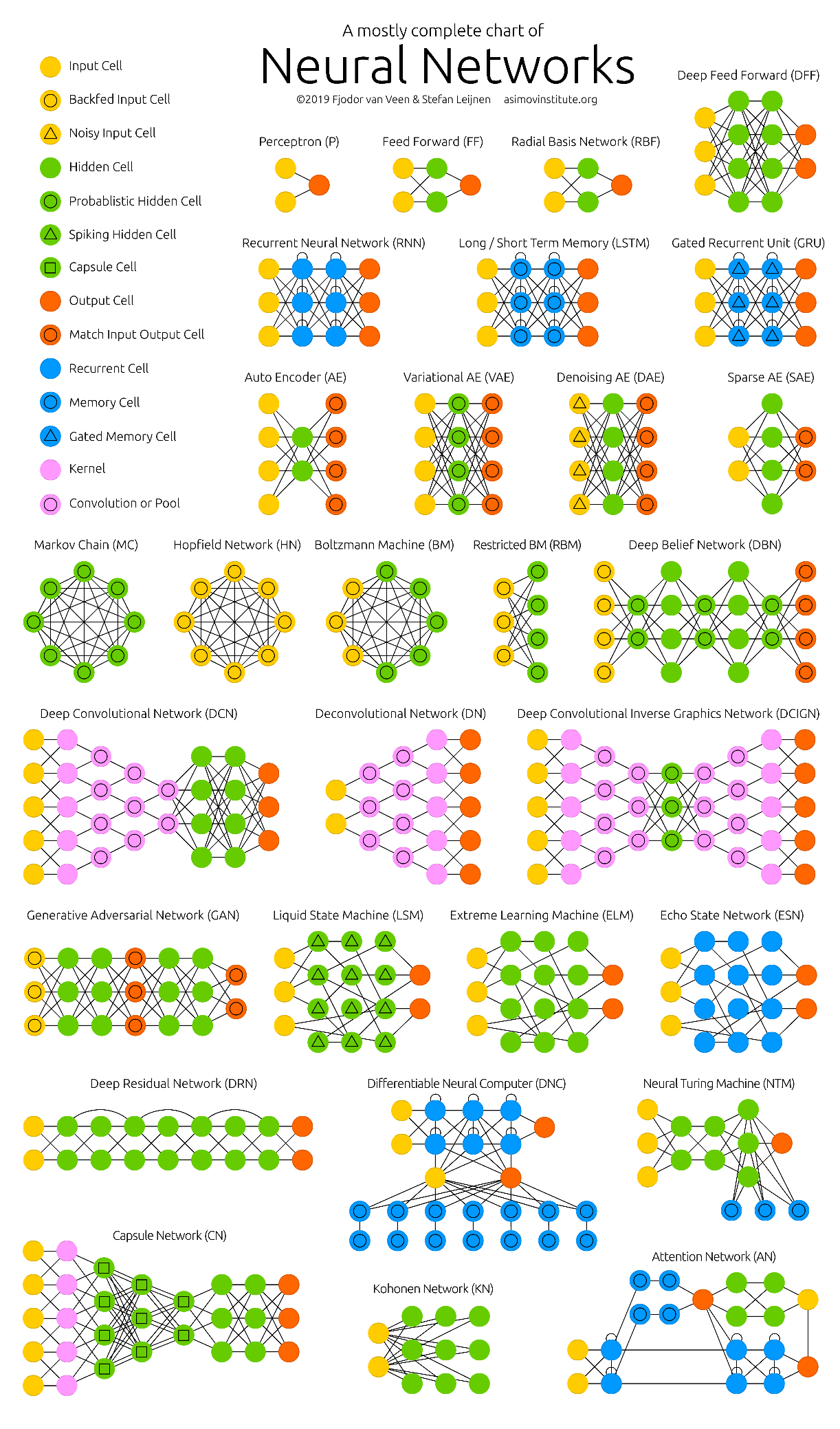
print(cnn\_digits\_predict(model, '7.png'))

print(cnn\_digits\_predict(model, '8.png'))

print(cnn\_digits\_predict(model, '9.png'))

**Приложение В**

Рисунок 4.1 – Архитектуры нейронных сетей



**Приложение Г**

Техническое задание [13, 15] на выполнение работ по созданию и внедрению модуля, выполняющего «Распознавание рукописных цифр с помощью сверточной нейронной сети»

УТВЕРЖДАЮ

Научный руководитель

\_\_\_\_\_\_\_\_\_\_/В.М. Татьянкин/

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**1 Общие сведения**

**1.1 Наименование системы**

Наименование системы: «Распознавание рукописных цифр с помощью сверточной нейронной сети».

Условное обозначение: СНС.

**1.2 Заказчик**

Заказчиком разрабатываемой нейронной сети является научный руководитель.

**1.3 Исполнитель**

Автор ВКР.

**1.4 Основание для проведения работ**

Сверточная нейронная сеть разрабатывается на основании:

Приказа ФГБОУ ЮГУ №\_\_\_ от «\_\_\_» \_\_\_\_\_\_\_\_ 2020 г. о назначении тем выпускных квалификационных работ, обучающихся по направлению 09.03.04 «Программная инженерия».

**1.5 Плановые сроки начала и окончания работы по созданию сверточной нейронной сети**

Начало работ: 20.01.2020

Окончание работ: 05.06.2020

**1.6 Сведения об источниках и порядке финансирования работ**

Работа выполняется полностью за счет исполнителя.

**1.7 Порядок оформления и предъявления заказчику результатов работ**

Сдача работ по созданию системы происходит поэтапно в соответствии с календарным планом проекта.

**2. Назначение и цели создания системы**

**2.1 Назначение**

Назначение нейронной сети состоит в автоматическом распознавании рукописных цифр. Сведения об изображении, введенное пользователем, подаются на вход, обрабатываются и сверточная нейронная сеть распознает цифру.

**2.2 Цели создания**

Распознавание рукописного ввода цифр искусственным интеллектом.

**3. Характеристика объекта исследования**

Разработка алгоритма и программного обеспечения распознавания цифр на основе сверточной нейронной сети.

**4. Требования к нейронной сети. Требования к структуре и функционированию системы**

Сверточная нейронная сеть должна обеспечить следующие функциональные возможности:

1) Распознавание рукописных цифр

2) Классификация рукописных цифр

**5. Состав и содержание работ по созданию сверточной нейронной сети**

Создание сверточной нейронной сети подразумевает выполнения этапов, описанных в таблице 5.1.

Таблица 5.1 – План-график выполнения работ

|  |  |  |
| --- | --- | --- |
| Название задачи | Начало | Окончание |
| 1. Разработка технического задания | 20.01.20 | 26.01.20 |
| 1. Проектирование архитектуры нейронной сети | 01.03.20 | 18.03.20 |
| 1. Написание модуля обработки изображений | 01.04.20 | 16.04.20 |
| 1. Разработка сверточной нейронной сети | 20.04.20 | 05.05.20 |
| 1. Тестирование сверточной нейронной сети | 05.05.20 | 20.05.20 |

Продолжение таблицы 5.1. – План-график выполнения работ

|  |  |  |
| --- | --- | --- |
| 1. Подготовка промежуточного отчета о результатах проекта | 26.02.20 | 27.02.20 |
| 1. Подготовка промежуточного отчета о результатах проекта | 25.03.20 | 28.03.20 |
| 1. Подготовка промежуточного отчета о результатах проекта | 28.04.20 | 30.04.20 |
| 1. Подготовка отчета о результатах проекта | 22.05.20 | 24.06.20 |
| Завершение разработки – 05.06.20 | | |

**5.1 Подготовка промежуточного отчета о результатах проекта**

Подготовка промежуточного отчета о результатах проекта происходит в конце каждого месяца вплоть до окончания работ над разработкой нейронной сети.

**5.2 Проектирование архитектуры нейронной сети**

На данном этапе происходит проектирование концептуальной модели предметной среды, логическое и физическое проектирование архитектуры нейронной сети.

**5.2 Разработка сверточной нейронной сети**

При создании сверточной нейронной сети должны быть выполнены следующие требования:

**6. Порядок контроля и приемки системы**

В рамках выполнения данного проекта – не предусмотрено. В перспективе выполнение этапа возможно в рамках нового проекта.

**7. Требования к документации**

**7.1 Состав программной документации**

Основными документами, регламентирующими разработку будущей программы, являются документы Единой системы программной документации (ЕСПД):

1) Техническое задание (ГОСТ 19.201-78);

2) Пояснительная записка (ГОСТ 19.404-79)

**7.2** **Специальные требования к программной документации**

Документы к программе должны быть выполнены в соответствии с ГОСТ 19.106-78 и ГОСТами к каждому виду документа (см. п. 8.1.); Пояснительная записка должна быть загружена в систему Антиплагиат.ВУЗ. Лист, подтверждающий загрузку пояснительной записки, сдается в учебную часть вместе со всеми материалами не позже, чем за день до защиты выпускной квалификационной работы (ВКР).

Техническое задание и пояснительная записка, титульные листы других документов должны быть напечатаны, подписаны академическим руководителем образовательной программы 09.03.04 «Программная инженерия» [18], руководителем разработки и исполнителем перед сдачей ВКР в учебную часть не позже одного дня до защиты.

Документация и программа также сдается в электронном виде в формате .pdf или .docx. в архиве формата .zip или .rar.