Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-ВОСТОЧНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИМЕНИ М.К. АММОСОВА»

Институт математики и информатики

Кафедра математической экономики и прикладной информатики

|  |  |
| --- | --- |
|  | ДОПУСТИТЬ К ЗАЩИТЕ:  Зав. кафедрой \_\_\_\_\_\_\_\_\_\_ С.П. Кайгородов  Протокол №\_\_\_ от «\_\_\_\_»\_\_\_\_\_20\_\_г. |

Выпускная квалификационная работа

на тему:

ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РАСПОЗНАВАНИЯ РУКОПИСНЫХ БУКВ

Направление подготовки: 09.03.03 Прикладная информатика

Профиль: Прикладная информатика в экономике

|  |  |  |
| --- | --- | --- |
| Выполнил: студент 4 курса группы БА-ПИ-19-1 ИМИ СВФУ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | А.А. Кычкин |
| Руководитель: к.ф.-м.н., доцент каф. МЭПИ ИМИ СВФУ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | М.А. Иванова |

Якутск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc136714912)

[ГЛАВА 1 ОСНОВНЫЕ ПОНЯТИЯ И ОБЛАСТИ ПРИМЕНЕНИЯ, ВИДЫ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ 6](#_Toc136714913)

[1.1 Понятие искусственных нейронных сетей 6](#_Toc136714914)

[1.2 Задачи и области применения нейронных сетей 6](#_Toc136714915)

[1.3 Виды нейронных сетей 7](#_Toc136714916)

[1.4 Преимущества и недостатки нейронных сетей 11](#_Toc136714917)

[1.5 Свойства почерка человека и проблемы его распознавания 12](#_Toc136714918)

[1.6 Выводы к главе 1 13](#_Toc136714919)

[ГЛАВА 2 РАЗРАБОТКА И ОБУЧЕНИЕ МОДЕЛИ НЕЙРОННОЙ СЕТИ 15](#_Toc136714920)

[2.1 Выбор и описание средств разработки нейронной сети 15](#_Toc136714921)

[2.2 Набор данных для обучения нейронной сети 18](#_Toc136714922)

[2.3 Создание и обучение сверточной нейронной сети распознавания рукописных букв 19](#_Toc136714923)

[2.4 Создание графического интерфейса для нейронной сети 29](#_Toc136714924)

[2.5 Выводы к главе 2 33](#_Toc136714925)

[ГЛАВА 3 ТЕСТИРОВАНИЕ МОДЕЛИ НЕЙРОННОЙ СЕТИ 34](#_Toc136714926)

[3.1 Общая оценка модели нейронной сети 34](#_Toc136714927)

[3.2 Оценка распознавания нейронной сетью при повороте буквы 36](#_Toc136714928)

[3.3 Сравнение аналогов Pen to Print, MyScript Write 37](#_Toc136714929)

[3.4 Выводы к главе 3 39](#_Toc136714930)

[ЗАКЛЮЧЕНИЕ 41](#_Toc136714931)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ 43](#_Toc136714932)

[ПРИЛОЖЕНИЕ А 45](#_Toc136714933)

[ПРИЛОЖЕНИЕ Б 50](#_Toc136714934)

[ПРИЛОЖЕНИЕ В 57](#_Toc136714935)

[ПРИЛОЖЕНИЕ Г 59](#_Toc136714936)

ВВЕДЕНИЕ

На сегодняшний день в мире активно развивается такая область искусственного интеллекта, как нейронные сети. Актуальность разработок в области нейронных сетей обуславливается их широкими возможностями применения в самых различных сферах.

Сейчас известно достаточно обширное количество областей использования искусственных нейронных сетей, где человеческий интеллект малоэффективен, а аналитические расчеты достаточно трудоемки и физически неадекватны. Наиболее распространенные из них: финансы, экономика, производство, медицина, военная промышленность, авиация, энергетика, научные исследования, информационные технологии, искусственный интеллект и т.д.

Рукопись – это отличное средство общения и документирования, которое используется на протяжении тысячелетий, но в последнее время они чаще всего заменяются набором текста на клавиатуре. Тем не менее, рукопись остается самым простым и наиболее эффективным способом выражения мыслей, который позволяет передавать информацию миллионам людей по всему миру. В эпоху компьютерных технологий использование рукописного текста и букв в вычислительных машинах было неудобно и малопрактично для использования в промышленных масштабах, однако с ускорением развития информационных технологий, а также появлением сенсорных устройств, более точных и быстрых камер, сканеров возможность преобразовывать рукописный текст и буквы в данные, понятные компьютеру, стало актуальной задачей на сегодняшний день.

На данном стадии развития системы распознавания рукописных букв существуют непрерывно расширяющиеся возможности для улучшения машинного перевода, почтовых пересылок, проверки подписей, ведение повседневных записей и других областей применения. Главной целью всех систем распознавания рукописных букв заключается в создании машин, способных читать любые буквы с такой же точностью распознавания, что и люди, но при обеспечивающих гораздо большую скорость выполнения задач.

Целью выпускной квалификационной работы является разработка модели нейронной сети, позволяющей распознавать рукописные буквы русского алфавита.

В соответствии с целью были поставлены следующие задачи:

1. Изучение основных принципов функционирования искусственных нейронных сетей, их видов, преимуществ и недостатков;
2. Изучение свойств почерка человека и проблем его распознавания;
3. Разработка и обучение модели нейронной сети;
4. Тестирование модели нейронной сети на точность при распознавании рукописных букв и сравнение с аналогами.

Объектом исследования является разработка модели нейронной сети и её обучение. Предметом исследования является процесс распознавания рукописных букв с помощью, разработанной и обученной нейронной сети.

В теоретической части были рассмотрены основные понятия и области применения искусственных нейронных сетей, а также были выделены виды нейронных сетей и проведён сравнительный анализ с целью выявления их преимуществ и недостатков.

В практической части работы разработана и обучена сверточная нейронная сеть для распознавания рукописных букв. Кроме того, в рамках исследования было проведено общее тестирование нейронной сети, а также ее тестирование при повороте букв, что позволило оценить ее эффективность и точность в различных условиях. Было также проведено сравнение созданной нейронной сети с аналогами разработанными другими командами из разных стран.

Ранее была опубликована научная статья в мультидисциплинарном научном интернет-журнале «Наукосфера», входящий в Российский индекс научного цитирования (РИНЦ) – Создание веб-приложения для ОС Android (на примере сайта ЭИОС СВФУ).

Также данная работа (Применение нейронных сетей для распознавания рукописных букв) опубликована в журнале «Научно - технический вестник Поволжья» (№7 2023, С. 161-163), который включен ВАК РФ в перечень научных журналов, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученых степеней доктора и кандидата наук.

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

ГЛАВА 1 ОСНОВНЫЕ ПОНЯТИЯ И ОБЛАСТИ ПРИМЕНЕНИЯ, ВИДЫ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

1.1 Понятие искусственных нейронных сетей

Искусственные нейронные сети (ИНС) являются математическими моделями, а также их программные или аппаратные реализации, построенные по принципу нейронной сети живого организма. Мозг живого организма – это крайне сложный, нелинейный, параллельный компьютер. Он способен организовывать своим структурные компоненты, называемые нейронами, так чтобы они могли выполнять конкретные задачи (распознавание объектов, обработку сигналов органов чувств, моторные функции) во много раз быстрее, чем могут позволить самые быстродействующие современные компьютеры [13]. Человеческий мозг при рождении имеет совершенную структуру, позволяющую строить собственные правила на основании того, что называется «опытом». Опыт накапливается с течением времени, и особенно масштабные изменения происходят в первые два года жизни. В этот период формируется остов общей структуры, но при этом развитие продолжается до конца жизни.

Принцип работы нейронных сетей отличается от стандартных методов машинного обучения – способность обучаться.

Поэтому нейронная сеть является методом искусственного интеллекта, который обучает компьютеры обрабатывать данные так же, как мозг человека. Он создает адаптивную систему, в котором компьютеры учатся на ошибках и постоянно совершенствуются. То есть искусственные нейронные сети пытаются решать сложные задачи, например, резюмирование документов или распознавание лиц, с более высокой точностью.

1.2 Задачи и области применения нейронных сетей

Область применения нейронных сетей в основном – это решение аналитических задач, сравнимо с теми, которые решает человеческий мозг. Чаще всего нейронные сети помогают быстро получать результаты в следующих задачах:

* Классификация. Нейронные сети определяют, соответствуют ли анализируемые объекты с заданными параметрами и относят ли их к той или иной группе. Возможности нейронных сетей используют различные крупные компании для предварительных оценок возможностей;
* Прогнозирование. На основе полученных данных нейронные сети предсказывают, как поведут себя в ближайшее время, например, акции конкретной компании на бирже, то есть вырастут, либо снизятся в цене;
* Распознавание. По моему мнению на данный момент распознавание является одним из популярных функций. Поиск по фотографии в поисковых системах Яндекс или Google, либо идентификация предметов или сцен (человек, пейзаж, животное) и выставление оптимальных настроек цветокоррекции, яркости и контрастности в реальном времени в смартфонах.

Перечисленными областями использование нейронных сетей не ограничивается, существуют и другие перспективные способы их применения.

1.3 Виды нейронных сетей

Нейронные сети можно разделить на 2 структуры:

1. Однослойная структура;
2. Многослойная структура.

Рассмотрим их поподробнее, однослойная структура представляет собой структуру, где нейронная сеть выдаёт результат сразу после загрузки некоторого массива данных, назовем их слоями входа и выхода. На слое входе нейронная сеть только принимает и распределяет данные, а нужные вычисления происходят уже на слое выхода. Многослойная структура помимо входного и выходного слоя имеет несколько промежуточных (скрытых) слоёв. Их число зависит от сложности самой нейронной сети. Она отдаленное напоминает структуру биологической нейронной сети. Такая структура была разработана относительно недавно, до этого все процессы выполнялись однослойной структурой нейронных сетей. Благодаря многослойной структуре повысились их возможности, если сравнивать с однослойными, так как в процессе обработки данных каждый промежуточный слой – это промежуточный этап, на котором осуществляется обработка и распределение информации.

Ниже описаны наиболее распространенные типы нейронных сетей:

1. Нейронные сети прямого распространения (Feed Forward Neural Networks – FFNN, перцептрон) – это самый стандартный тип нейронной сети, где соединения между нейронами не формируют замкнутых циклов (рисунок 1.1). Для проектирования классических нейронных сетей они являются первыми и простейшими типами. в нейросетях прямого распространения связи движутся только в прямом направлении, от входов через скрытые слои (если присутствуют) к выходам. Обычно используются для задач классификации или регрессии, где требуется прогнозирование конкретных выходных значений на основе определенных входных параметров;

Входной слой

Скрытый слой

Выходной слой

Рисунок 1.1 – Структура перцептрона.

1. Рекуррентные нейронные сети (с обратными связями) (Recurrent Neural Networks – RNN) – это тип нейронных сетей, которые предназначены для задач обработки последовательности входных данных (рисунок 1.2). Они особенно хорошо подходят для задач, связанных с естественным языком, таких как распознавание речи, машинный перевод или извлечение информации. Рекуррентные нейронные сети способны запоминать информацию о предыдущих входных данных, и использовать ее при обработке последующих входов;

Входной слой

Выходной слой

Скрытые слои

Рисунок 1.2 – Структура рекуррентной нейронной сети.

1. Сверточные нейронные сети (Convolutional Neural Networks – CNN) – это тип нейронных сетей, используемый для обработки изображений и видео. Они могут обнаруживать различные структуры на входных данных, такие как границы и текстуры (рисунок 1.3).

Структура сверточных нейронных сетей состоит из [16]:

* Сверточного слоя: этот слой выполняет операцию свертки между фильтрами и исходными данными. Фильтры – это матрицы весов, которые применяются к каждому куску исходных данных, чтобы извлечь признаки. Результатом свертки свяляется карта признаков, которая представляет собой двухмерную матрицу значений;

Формула функции сверточного слоя выглядит следующим образом [16]:

где, – входные данные в позиции и -м канале, – выходные данные в позиции , и -м канале, – вес свертки, относящийся к -й строке, -м столбцу, -м каналу и -м фильтру, – смещение для -го фильтра, – функция активации.

* Подвыборочного слоя: этот слой применяет нелинейную функцию активации к результату предыдущего слоя. Это позволяет модели выучить сложные зависимости между входными и выходными данными;

Формула функции подвыборочного слоя выглядит следующим образом [16]:

где, – входные данные, – выходные данные, – веса сверточного ядра, – смещение, – функция активации, и – размеры сверточного ядра, – количество каналов входных данных, а – шаг свертки.

* Полносвязного многослойного перцептрона: этот слой связывает все нейроны предыдущего слоя со всеми нейронами следующего слоя. Он позволяет модели вычислять более сложные функции.

Формула полносвязного многослойного перцептрона выглядит следущим образом [16]:

где, – входные данные, – первый скрытый слой, – -й скрытый слой с функцией активации , – выходные данные, и – матрица весов и вектор смещений между -м и -м слоем, соответственно.

Функция активации обычно является нелинейной функцией, такой как ReLU или сигмоид.

512

нейронов

1024

нейрона

Ядра

3х3

Ядра

3х3

Ядра

3х3

Ядра

3х3

Слой №4

32х32

Слой №3

64х64

Слой №2

128х128

Слой №1

256х256

33

нейрона

Слой №0

Изображение 278х278

Входной

слой

Выходной

слой

Подвыборочный

слой

Подвыборочный

слой

Сверточный

слой

Сверточный

слой

Полносвязный

слой

Рисунок 1.3 – Структура сверточной нейронной сети.

1.4 Преимущества и недостатки нейронных сетей

Преимущества нейронных сетей обуславливается их повышенной точностью по сравнению с аналогичными методами анализа данных, это связано с их способностью распознавать сложные закономерности, которые не могут обнаружить программные алгоритмы; способность решения сложных проблем, возможность моделирования сложных взаимосвязей между входными и выходными данными делает нейронные сети отлично подходящими для таких задач, как распознавание изображений и обработка естественного языка; одним из главных преимуществ и особенностей нейронных сетей является их способность обучаться. Они способны запоминать и усиливать определенные ассоциации и не распознавать менее важные. Это позволяет лучше распознавать закономерности, не требуя вмешательство со стороны человека.

Недостатками нейронных сетей является их дороговизна и продолжительность их обучения, обучение сети может быть длительным процессом, который может быть дорогостоящим как по времени, так и по ресурсам в особенности больших нейронных сетей; сложность их интерпретации – это обусловлено тем, что используемые ими шаблоны данных чаще всего трудно интерпретировать. Однако выходные данные могут быть очень точными, но бывает трудно понять почему нейронная сеть приходит к таким результатам; большим недостатком является их ограниченное понимание данных, которые им предоставляются. Нейронные сети могут распознавать только те данные, которые уже присутствуют в базе, и не могут извлекать новую информацию данных.

Хоть у нейронных сетей есть недостатки, но они перекрываются их возможностями и преимуществами над традиционными методами обучения искусственного интеллекта, что делает их мощными инструментами.

1.5 Свойства почерка человека и проблемы его распознавания

Почерк – это индивидуальный образ письма каждого человека. Он характеризуется различными свойствами, такими как форма букв, их размер, степень наклона, расстояние между буквами и так далее.

Свойства почерка могут быть очень полезны для идентификации личности при аутентификации документов или распознавания символов на изображениях рукописных букв. Несмотря на то, что технологии распознавания рукописных букв на нейронных сетях существуют уже давно, они имеют свои проблемы.

Одной из главных проблем является невозможность точного определения каждого символа. Дело в том, что у каждого человека почерк имеет свои особенности, так что два одинаковых символа могут выглядеть по-разному. Это касается не только формы букв, но и других свойств почерка, таких как наклон, размер, расстояние между буквами и так далее.

На рисунке 1.4 приведен пример возможных стилей почерка русского языка.

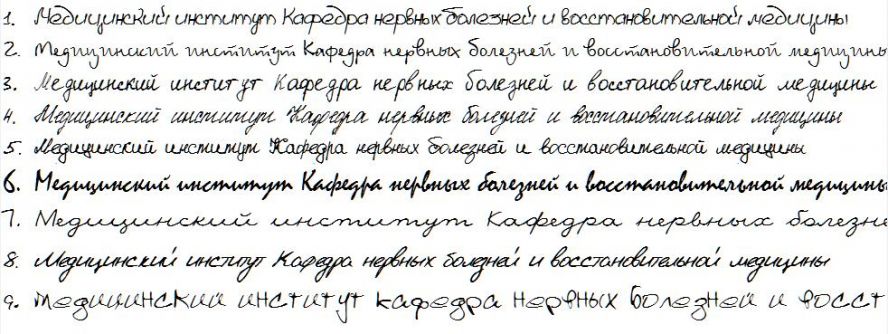


Рисунок 1.4 – Пример возможных стилей почерка на русском языке.

Кроме того, почерк может быть неоднородным и нестабильным, например, если письмо совершается не в статических условиях, например, на ходу, в движении транспорта или в других нефиксированных условиях. В таких случаях, почерк может содержать множество различных шумов, ошибок, пропущенных символов или даже совершенно незнакомых символов.

Таким образом, распознавание почерка нейронными сетями становится сложной задачей, и точность определения символа может быть ограничена. Однако, современные методы машинного обучения и нейронных сетей могут постепенно снижать ошибки в распознавании рукописных букв, благодаря большим объемам данных и эффективным методам обучения сетей, которые повышают точность определения символа.

1.6 Выводы к главе 1

Таким образов в главе 1 было рассмотрено понятие искусственных нейронных сетей, они способны моделировать работу мозга живых организмов и позволяет компьютеру обучаться также, как и человеческому мозгу, помогает быстро получать результаты в классификации, прогнозировании, и распознавании, а также находят широкое применение в других областях и задачах.

Виды нейронных сетей зависят от их структуры и функциональности, которые определяются по задачам, для которых они используются. Наиболее распространёнными типами ИНС являются, нейронные сети прямого распространения, рекуррентные нейронные сети, сверточные нейронные сети.

В результате данного анализа были сделаны выводы, что распознавание почерка нейронными сетями имеет свои сложности из-за индивидуальных особенностей каждого человека и неоднородности почерка, особенно при письме в движении или в других нефиксированных условиях. К преимуществам нейронных сетей для распознавания почерка можно отнести их способность распознавать сложные закономерности, которые не могут быть обнаружены программными алгоритмами, а также возможность моделирования сложных взаимосвязей между входными и выходными данными. Недостатком же является дороговизна и продолжительность обучения нейронных сетей, а также ограниченное понимание данных, которые им предоставляются.

ГЛАВА 2 РАЗРАБОТКА И ОБУЧЕНИЕ МОДЕЛИ НЕЙРОННОЙ СЕТИ

Алгоритм разработки и обучения нейронной сети для распознавания рукописных букв состоит из 6 шагов:

1. Выбор средств разработки;
2. Сбор данных;
3. Подготовка данных;
4. Создание модели;
5. Обучение модели: научить распознавать рукописные буквы, выявлять результат;
6. Тестирование модели.

Для реализации была выбрана сверточная нейронная сеть, так как они используются для обработки изображений и визуальной информации.

2.1 Выбор и описание средств разработки нейронной сети

Нейронные сети можно программировать на многих языках программирования, основные из них:

* Python;
* Java;
* C++;
* MATLAB;
* R.

Для реализации нейронной сети был выбран язык программирования Python, так как он имеет ряд плюсов, по сравнению с другими языками программирования:

* Простота в освоении;
* Легкий и понятный синтаксис;
* Большое количество разнообразных библиотек;
* Большое количество технической документации и учебных материалов;
* Поддержка на любой платформе.

Язык программирования Python отличается простым и понятным синтаксисом, его программный код легко читается и понимается, не содержит сложных конструкций, операторы завершаются концом строки, а структура блока определяется отступом, которые разграничивают блоки кода. Правильное форматирование обеспечивается интерпретатором, что мешает создавать нечитаемый код.

После выбора языка программирования необходимо выбрать интегрированную среду разработки (IDE) – это программное обеспечение, которое интегрирует в себе различные инструменты для разработки. IDE позволяет программистам создавать, отлаживать и тестировать программы в одном окружении. В состав IDE могут входить следующие инструменты:

* Редактор исходного кода – это текстовый редактор, в котором можно писать код программы;
* Компилятор или интерпретатор – это программное обеспечение, которое используется для компиляции или интерпретации кода программы;
* Отладчик – это инструмент, который позволяет отслеживать выполнение кода программы и выявлять ошибки;
* Средства управления версиями – это инструменты, которые позволяют управлять версиями исходного кода программы;
* Инструменты автоматической сборки – это программы, которые позволяют компилировать программы автоматически;
* Инструменты анализа кода – это программы, которые анализируют код программы с целью выявления ошибок или улучшения качества кода.

IDE облегчает и ускоряет процесс разработки программного обеспечения, поскольку позволяет использовать различные инструменты в едином окружении. Однако, многие IDE требуют некоторого времени для изучения, чтобы настроить их на нужные параметры и интегрировать все инструменты в единую рабочую среду.

Существует большое множество интегрированных сред разработки языка программирования Python, выделим популярные из них:

* PyCharm – это популярная среда разработки, которая предоставляет обширные инструменты для написания и отладки кода Python. Она включает в себя функциональность автозавершения кода, инструменты отладки и анализаторы кода;
* IDLE – это интегрированная среда разработки поставляемая вместе с Python. Она предоставляет удобную интерактивную консоль для написания и отладки кода Python;
* Jupyter Notebook – это среда разработки, которая предоставляет документацию и отображение результатов вычислений на странице веб-браузера. Она может быть использована для создания и отладки интерактивных приложений и ноутбуков Python;
* Anaconda – это бесплатный дистрибутив Python с открытым исходным кодом, ориентированный для научных вычислений на языке Python и R. Она включает в себя множество пакетов и библиотек, необходимых для работы с данными и научных расчетов.

Для реализации был выбран IDE PyCharm, так как удобен, высокая производительность, наличие интегрированных средств разработки, кроссплатформенность, автоматизация рутинных задач.

После выбора интегрированной среды разработки необходимо выбрать библиотеки, которые будут работать с изображениями. Самые популярные библиотеки для работы с изображениями:

* Pillow;
* OpenCV;
* NumPy;
* MatplotLib.

Для работы с изображениями были выбраны библиотеки Pillow и NumPy, так как их совместимость заключается в том, что Pillow может использовать массивы NumPy в качестве исходных данных для обработки изображений. В частности, Pillow может принимать массив NumPy в качестве входных данных. Также, Pillow может сохранять изображения в массиве NumPy.

Для создания нейронной сети, для распознавания рукописных букв было решено использовать библиотеки Tensorflow и Keras.

TensorFlow – это библиотека с открытым исходным кодом, разработанная и поддерживаемая компанией Google. Она является одной из самых популярных библиотек для машинного обучения и предоставляет широкие возможности для работы с данными и моделями. TensorFlow используется для создания глубоких нейронных сетей, обработки естественного языка, обработки речи, компьютерного зрения и других задач [5].

Keras – это высокоуровневый интерфейс для создания нейронных сетей, который может быть использован вместе с TensorFlow. Keras упрощает создание, обучение и оценку нейронных сетей за счет предоставления эффективных функций, которые позволяют пользователю быстро создавать сложные модели машинного обучения. С помощью Keras можно легко создавать слои и модели нейронных сетей, определять функции потерь, оптимизаторы и метрики [5].

Использование TensorFlow и Keras позволяет быстро конструировать и обучать сложные модели машинного обучения. Обе библиотеки широко используются в академической среде и индустрии и могут быть использованы для решения различных задач, таких как классификация, регрессия, кластеризация, определение объектов и т.д.

2.2 Набор данных для обучения нейронной сети

Для обучения нейронной сети необходим набор обучающих данных. Был выбран готовый набор данных – Cyrillic-oriented MNIST (CoMNIST) который содержит в себе изображения рукописных букв кириллицы. В настоящее время набор данных состоит из 15000+ изображений в формате png размером 278x278 пикселей, представляющих все 33 буквы русского алфавита (рисунок 2.1) и в настоящее время ведется расширение данного набора данных.

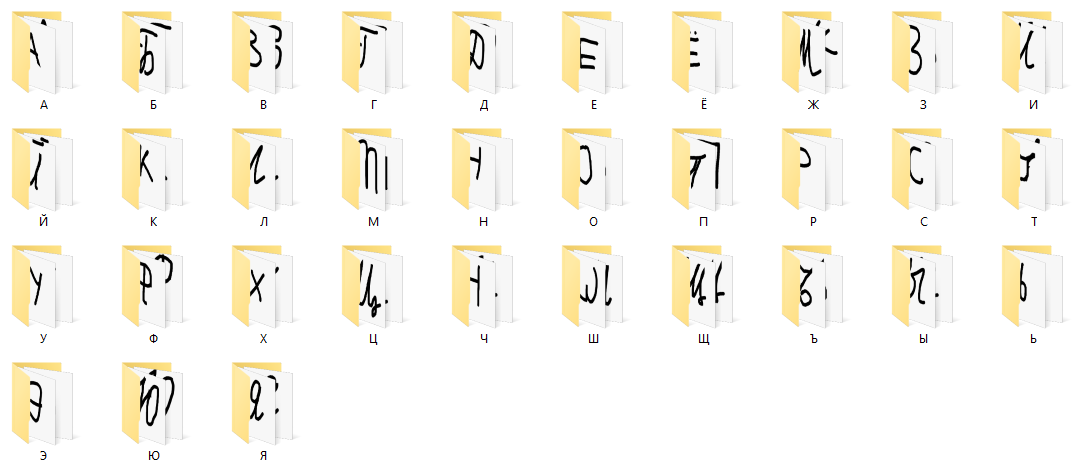


Рисунок 2.1 – Набор данных CoMNIST.

В каждом классе букв количество изображений варьируется от 400-500 изображений – это говорит о недостаточном количестве изображений для полноценного обучения нейронной сети и необходимо произвести дополнительную аугментацию (расширение) набора данных с помощью смещения, поворотов и уменьшения изображений, которые добавляются в один из двух наборов. Также набор данных не предоставляет тренировочные и тестовые данные. Так что необходимо самостоятельно разделить их на тренировочные и тестовые данные в соотношении 3:17 (15% тестового набора и 85% тренировочного набора).

Во время детального исследования набора данных были обнаружены некорректно написанные буквы, либо буквы, добавленные не в свой класс.

2.3 Создание и обучение сверточной нейронной сети распознавания рукописных букв

Для начала обучения сверточной нейронной сети необходимо создать новый проект на интегрированной среде разработки PyCharm. Для этого в приветственном окне необходимо выбрать пункт «New Project» (рисунок 2.2).

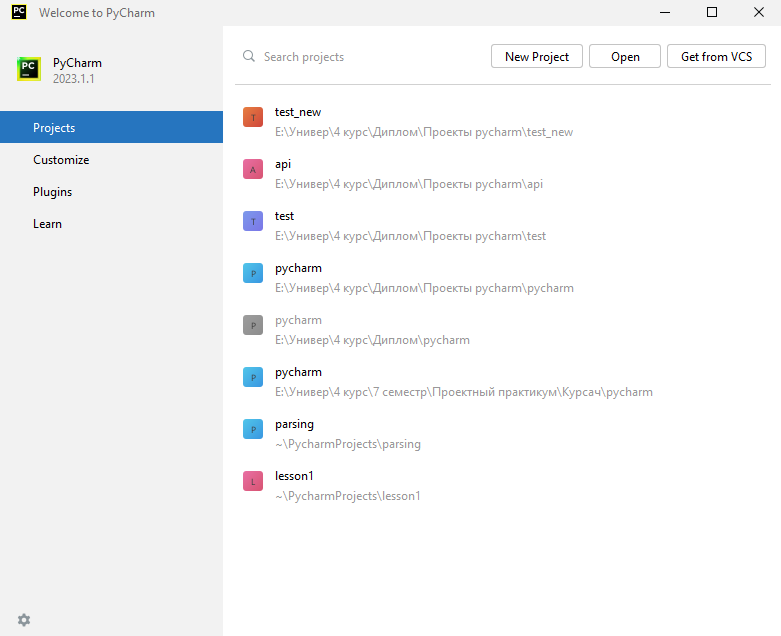


Рисунок 2.2 – Приветственное окно PyCharm.

Далее необходимо написать название проекта и выбрать его место нахождения (рисунок 2.3).

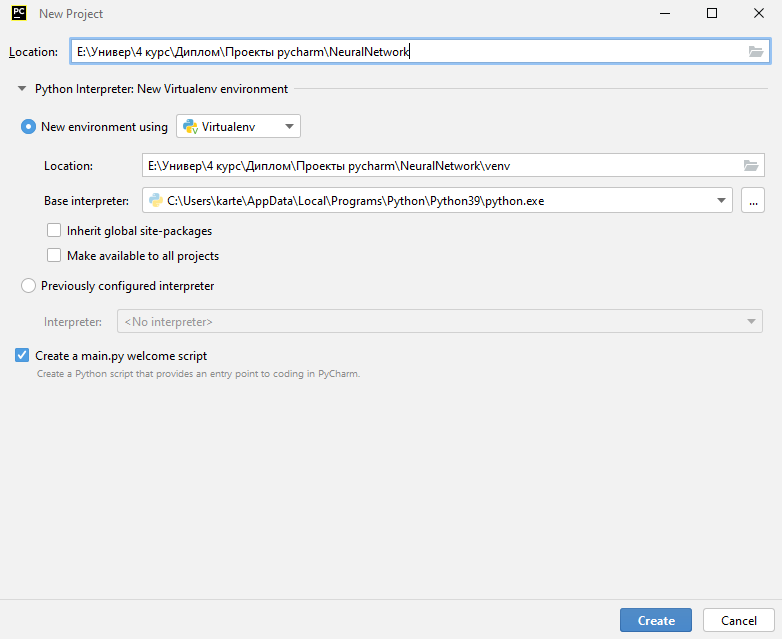


Рисунок 2.3 – Окно создания проекта.

В открывшемся проекте необходимо импортировать необходимые библиотеки (рисунок 2.4) и модули для дальнейшей разработки и обучения сверточной нейронной сети. Приведу их название и краткую характеристику:

* Библиотека «NumPy» – это библиотека для работы с многомерными массивами и матричными операциями, входит как расширение языка программирования Python;
* Модуль «OS» – это модуль для работы с операционной системой, служит проводником к наборам данных;
* Модуль «Random» – данная функция необходима для генерации случайных чисел;
* Библиотека «PIL» – это библиотека для работы с изображениями;
* Библиотека «Keras» была отмечена в параграфе 2.1;
* Модуль «SkLearn.utils» (Scikit-learn) – это модуль для различных утилит машинного обучения, в данном случае он используется для перемешивания данных.

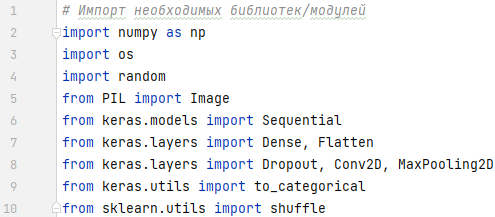


Рисунок 2.4 – Импорт необходимых библиотек и модулей в главном файле.

Первым шагом к обучению нашей модели нейронной сети будет указание пути к набору данных CoMNIST и создание пустых массивов, в которые будут храниться данные обучающей и тестовой выборки – изображения и их метки соответственно (рисунок 2.5).

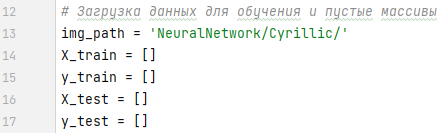


Рисунок 2.5 – Загрузка данных для обучения и пустые массивы данных.

Далее необходимо сделать инициализацию папок внутри набора данных Cyrillic содержащий изображения букв русского алфавита, а также проиндексировать их (рисунок 2.6) и вывести на экран, результат проиллюстрирован на рисунке 2.7.

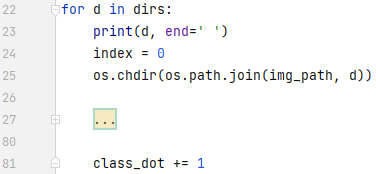


Рисунок 2.6 – Инициализация папок.



Рисунок 2.7 – Полученный результат.

Так как в наборе данных недостаточное количество изображений для полноценного обучения нейронной сети, необходимо произвести дополнительную аугментацию (расширение) набора данных с помощью смещения, поворотов и уменьшения изображений и разделить их на тренировочные и тестовые наборы в соотношении 3:17 (рисунок 2.8).

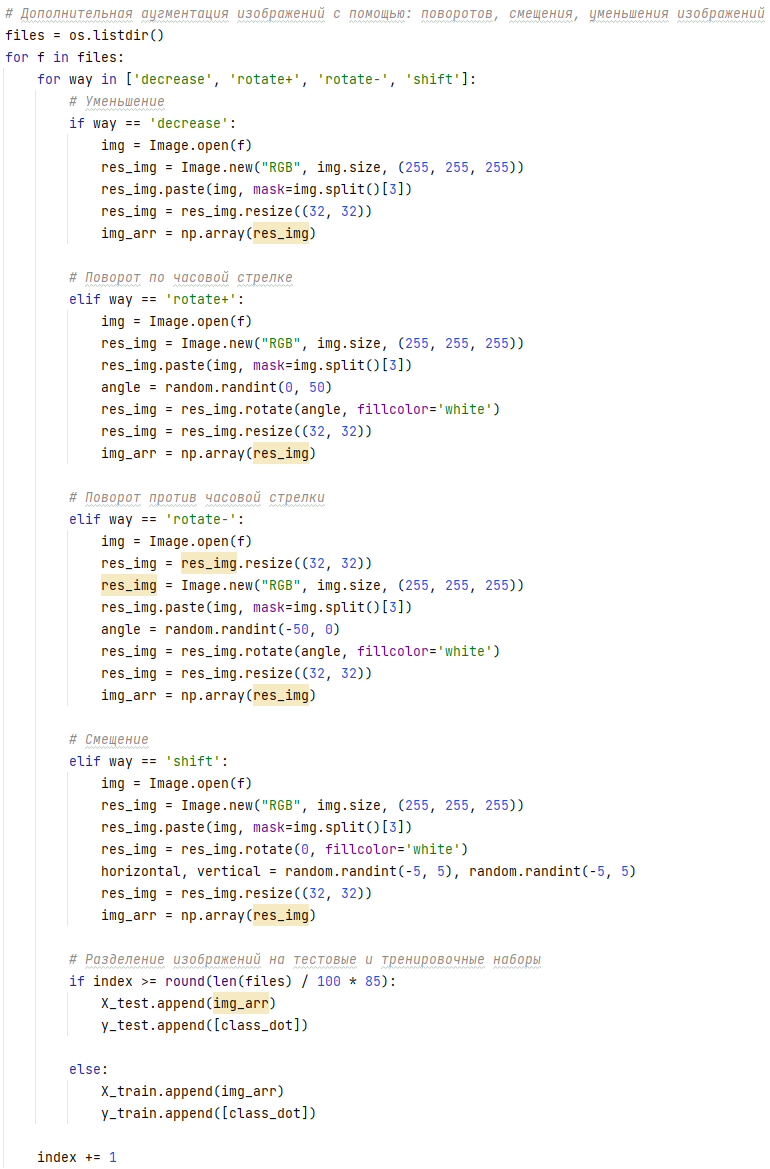


Рисунок 2.8 – Дополнительная аугментация изображений.

Далее следует разделение их на тренировочные и тестовые массивы данных, преобразование в бинарную матрицу, с помощью функции «to\_categorical()», и нормализация тренировочных данных путем деления на 255, так как нейронные сети не могут обрабатывать входные данные с большим весом, это может привести к нарушению процесса обучения и необходимо приводить значения пикселей изображения от 0 до 1, и перемешивание тестовой выборки (рисунок 2.9).

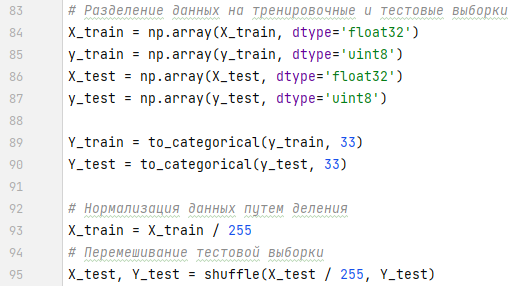


Рисунок 2.9 – Разделение, нормализация, перемешивание данных.

На рисунке 2.10 представлено получившееся количество тренировочных и тестовых данных после аугментации и перемешивания тестовой выборки.

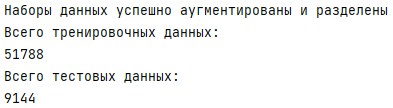


Рисунок 2.10 – Результат аугментации и перемешивания тестовой выборки.

Теперь приступим к созданию архитектуры модели, для этого необходимо сделать описание архитектуры модели с помощью библиотеки «Keras» определяется последовательная модель сети («Sequential»), в которой последовательно добавляются слои обработки данных (рисунок 2.11).



Рисунок 2.11 – Архитектура модели.

Слои «Conv2D» являются сверточными слоями, каждый из них содержит 16, 32, 64, 128, 256 ядер размером 3 на 3 пикселя, с активационной функцией «ReLU» и параметром «padding='same'», что означает «одинаковое заполнение». Без заполнения размерность выходного изображения после свертки будет меньше, чем у входного. В результате использования параметра заполнения размерность выходного изображения сохраняется, что упрощает дальнейшую обработку.

ReLU (Rectified Linear Unit) – это одна из самых популярных активационных функций в нейронных сетях. Она применяется к выходным значениям каждого нейрона, чтобы добавить нелинейность и увеличить способность модели обучаться более сложным функциям.

Функция ReLU имеет очень простой вид:

где, – входное значение нейрона.

Если значение положительное, то активация будет передана без изменений. Если же значение отрицательное, то активация будет обнулена (выход активации равен 0) [21].

Преимущества использования функции ReLU заключаются в ее простоте, высокой скорости вычисления и способности предотвращать затухание градиента при обратном распространении ошибки. Кроме того, эта функция может помочь сети обучаться разреженным представлениям данных, что может быть полезно при работе с большими наборами данных.

Далее следует слой «MaxPooling2D», который выполняет подвыборку изображения и уменьшает его размеры в два раза. Это позволяет сократить количество параметров для обработки и делает модель более легковесной. В каждом таком слое также присутствует слой «Dropout», который предотвращает переобучение модели. Метод «Dropout(0.1)» означает, что 10% случайно выбранных нейронов будут игнорироваться на каждой обучающей эпохе. Рассмотрим данный метод поподробнее.

«Dropout» – это метод регуляризации для искусственных нейронных сетей, который заключается в случайном «выключении» определенного числа нейронов во время обучения. В процессе обучения каждый нейрон имеет вероятность быть удаленным (отключенным) сетью. Это означает, что входные данные не будут передаваться через этот нейрон, и его связи с другими нейронами не будут участвовать во время этой конкретной итерации обучения. Таким образом, «Dropout» может предотвратить переобучение сети и улучшить ее обобщающую способность [19].

Формула метода «Dropout» выглядит следующим образом:

где, – входные данные, – вероятность удаления элементов, – случайная бинарная маска, которая принимает значения 0 или 1 с вероятностью и соответственно.

Далее следует слой «Flatten», который выполняет преобразование двумерного массива данных в одномерный. Затем добавляются три слоя полносвязных многослойных перцептрона («Dense»), каждый из которых содержит набор нейронов. Нейроны в этих слоях активируются функцией «ReLU».

Последний слой перцептрона содержит количество нейронов, равное количеству классов и использует активационную функцию «softmax», функция «softmax» помогает получить вероятности для каждого класса, а значение на выходе этого слоя является вероятностью принадлежности объекта к каждому из классов.

Также стоит рассмотреть операцию «Flatten» более подробно. Flatten – это операция в нейронных сетях, которая преобразует многомерный тензор в одномерный тензор путем «расплющивания» всех измерений изображения [20].

Функция «softmax» – это нелинейная активационная функция, которая обычно используется в многослойных нейронных сетях для преобразования выходных значений каждого нейрона в вероятности отнесения входного сигнала к различным классам [16].

Функция «softmax» определяется следующей формулой:

где, – это вектор значений линейной комбинации входных данных и параметров модели (без акткивации), а – номер элемента вектора, соответствующий конкретному классу.

Эта функция берет на вход вектор , который может быть любой длины, и преобразует его в новый вектор можно рассматривать как вероятность того, что взодной сигнал относится к соответствующему классу.

Таким образом, модель обучается на входных изображениях, после чего может проводить классификацию новых изображений на соответствующие классы.

Далее нам необходимо настроить параметры обучения модели методом «compile» и обучение модели методом «fit» (рисунок 2.12), где, метод «compile» состоит из функции потерь (основанная на «categorical\_crossentropy» – кросс-энтропии), оптимизатора – «adam» и метрики показателя качества для оценки производительности модели – «accuracy» – точность, далее метод «fit», использует данные из массивов «X\_train» – входные данные и «Y\_train» – целевые значения для обучения. «Batch\_size» – определяет размер пакета данных равный 80, то есть количество тренировочных примеров, используемых одновременно в каждой итерации обучения. «Epochs» – число эпох, то есть количество циклов обучения, установлено на 10. Также указываются данные («X\_test» и «Y\_test») для проверки модели после каждой эпохи, которые не используются для обучения модели и активируем режим перемешивания тренировочных данных перед каждой эпохой («shuffle=True»).

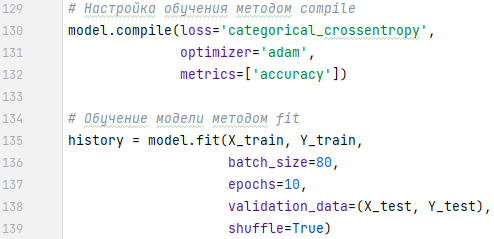


Рисунок 2.12 – Настройка и обучение модели методами «compile» и «fit».

На рисунке 2.13 изображен пример обучения модели на 10 эпохах. Здесь можно увидеть количество эпох для прохождения обучения по тренировочным данным, за какое время выполняется одна эпоха и сколько занимает времени одна эпоха.

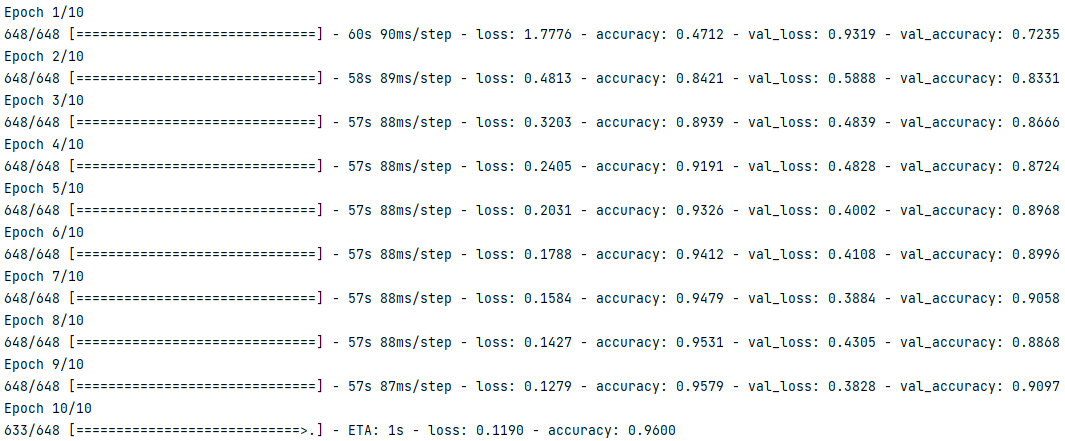


Рисунок 2.13 – Пример обучения модели.

После завершения обучения необходимо сохранить обученную модель с расширением h5. После сохранения модели необходимо приступить к созданию графического интерфейса для нейронной сети. Полноценный программный код обучения нейронной сети представлен в приложении А.

2.4 Создание графического интерфейса для нейронной сети

Чтобы проверить результат обучения модели сверточной нейронной сети необходимо создать графический интерфейс.

Для этого создадим новый файл «gui.py», выбрав File, далее пункт «New», «Python File» и задать ему имя.

Далее снова импортируем необходимые библиотеки и модули (рисунок 2.14). Краткое описание импортированных библиотек и модулей:

* Модули OS и NumPy были отмечены в параграфе 2.3;
* Библиотека PyQt5 – это набор Python-модулей, позволяющих создавать программы с графическим интерфейсом пользователя. PyQt5 строится на основе библиотек Qt, разработанной компанией Trolltech в 1991 году [8];
* Библиотеки Tensorflow и Keras была отмечена в параграфе 2.1.

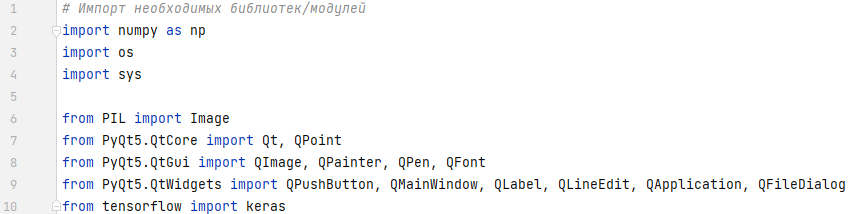


Рисунок 2.14 – Импорт необходимых библиотек и модулей в файле «gui.py».

Первым шагом будет создание класса «Window», который будет являться главным окном приложения и содержит:

1. Метод init настраивает размер и расположение окна, кисть для рисования, холст, кнопки и поле «результат»;
2. Метод «print\_letter» принимает результат распознавания буквы и выводит ее на поле «результат»;
3. Метод «predicting» загружает модель нейронной сети и изображение буквы, преобразовывает его в массив и передает в модель, получая результат распознавания буквы.
4. Методы «mousePressEvent», «mouseMoveEvent», «mouseReleaseEvent», «paintEvent» реагируют на движение и клик мыши на холсте и отображают изменения на холсте.
5. Методы «save» и «clear» используются для сохранения изображения буквы и очистки холста соответственно;
6. Метод «load\_image» используется для загрузки изображения из папки на выбор пользователя;
7. Методы «dragEnterEvent» и «dropEvent» используются для обработки события перетаскивания файлов («Drag and Drop») на окно приложения.

Методы «print\_letter» и «predicting» изображены на рисунке 2.15.

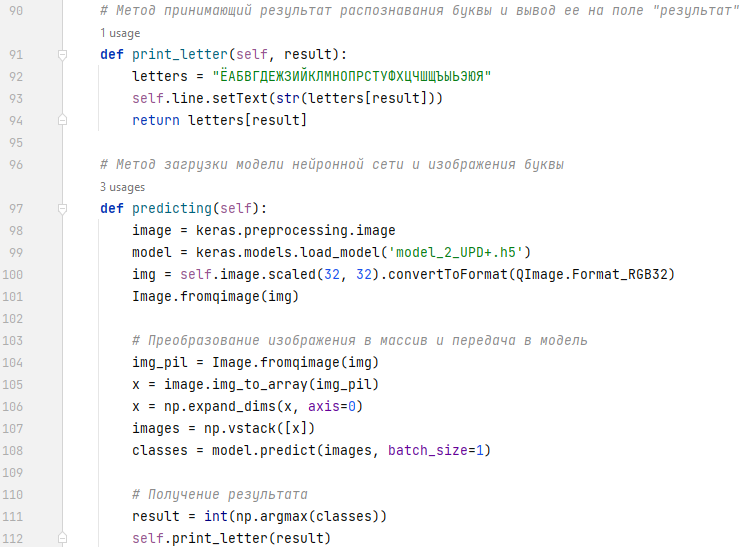


Рисунок 2.15 – Методы «print\_letter» и «predicting».

Подробно разберем методы «print\_letter» и «predicting». Метод «print\_letter» предназначен для вывода распознанной буквы на поле "результат". Он принимает один аргумент «result», который представляет из себя номер буквы в списке «letters» (об этом далее). Внутри метода объявляется переменная «letters», которая содержит все буквы русского алфавита, и номер «result» используется для извлечения соответствующей буквы из списка «letters». Далее метод устанавливает полученную букву в виджет self.line и возвращает ее.

Метод «predicting» представляет собой процесс распознавания русской буквы на изображении, используя нашу обученную модель сверточной нейронной сети. Сначала метод загружает модель с помощью функции «load\_model» из библиотеки «keras.models», передав ей название файла модели. Затем данное изображение (self.image) приводится приводится к размеру 32 на 32 пикселя и конвертируется в формат RGB32 – это нужно для того чтобы преобразовать изображение в трехканальное, так как нейронная сеть обучена на изображениях формата RGB, где каждый пиксель кодируется тремя 8-битными значениями для цветов красного, зеленого и синего (RGB). Далее изображение преобразуется в массив, затем массив передается в модель с помощью функции «predict», которая возвращает вероятности принадлежности изображения к каждому классу (букве) на выходе модели. Результатом метода является номер буквы с наибольшей вероятностью, возвращаемый функцией «argmax» из библиотеки «NumPy». Этот результат передается в метод «print\_letter», который выводит распознанную букву на экран.

Графический интерфейс и результат его работы изображен на рисунках 2.16, 2.17.

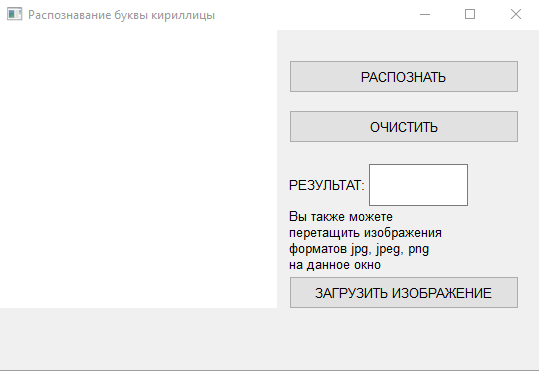


Рисунок 2.16 – Графический интерфейс.

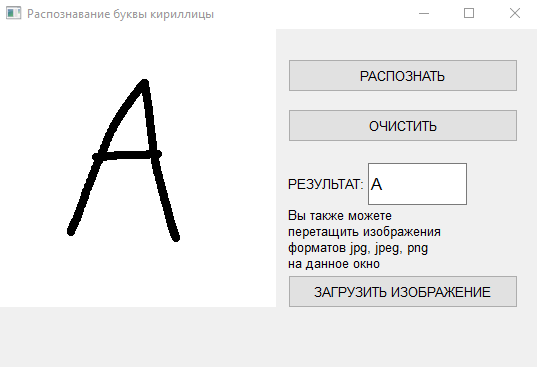


Рисунок 2.17 – Результат работы.

Полноценный программный код графического интерфейса представлен в приложении Б.

2.5 Выводы к главе 2

Таким образом, в данной главе была обучена сверточная нейронная сеть и создан графический интерфейс для взаимодействия с нейронной сетью. Перед этим были описаны и выбраны применяемые средства разработки и набор данных для разработки и обучения нейронной сети с графическим интерфейсом, также были рассмотрены функции активации «ReLU», «softmax».

ГЛАВА 3 ТЕСТИРОВАНИЕ МОДЕЛИ НЕЙРОННОЙ СЕТИ

В данной главе проводились различные тестирования над обученной сверточной нейронной сетью:

* Общая оценка модели нейронной сети;
* Оценка распознавания нейронной сетью при повороте буквы;
* Сравнение аналогов, Pen to Print, MyScript Write.

Для этого были проведены наблюдения на графическом интерфейсе. Тестирование проводилось на обученной нейронной сетью с общей точностью 91%.

3.1 Общая оценка модели нейронной сети

Для оценки были проведены тестирования в лице написания с помощью графического планшета всех букв русского алфавита на приложении графического интерфейса в 10 подходов.

Результаты тестирования представлены в таблице 3.1.

Таблица 3.1 – Результаты общего тестирования

|  |  |
| --- | --- |
| Буква алфавита | Точность распознавания, % |
| 1 | 2 |
| А | 70 |
| Б | 90 |
| В | 100 |
| Г | 90 |
| Д | 80 |
| Е | 100 |
| Ё | 100 |
| Ж | 70 |
| З | 100 |
| 1 | 2 |
| И | 60 |
| Й | 100 |
| К | 80 |
| Л | 70 |
| М | 80 |
| Н | 90 |
| О | 100 |
| П | 60 |
| Р | 100 |
| С | 100 |
| Т | 100 |
| У | 90 |
| Ф | 80 |
| Х | 90 |
| Ц | 70 |
| Ч | 60 |
| Ш | 90 |
| Щ | 80 |
| Ъ | 90 |
| Ы | 90 |
| Ь | 90 |
| Э | 100 |
| Ю | 90 |
| Я | 90 |
| Средняя точность | 86,36 |

По результатам оценки общего тестирования (таблица 3.1) точность оказалась – 86,36%. Были выявлены буквы, в которых нейронная сеть была не очень точна. Ими оказались буквы: А, Ж, И, Л, П, Ц, Ч. В букве А нейронная сеть некорректно распознавала буквы Д и Л, в букве Ж с буквой Х, в букве И с буквами Н, Ц, Ы, в букве Л с буквами А, П, в букве П с буквой Л, в букве Ц с буквой И, в букве Ч с буквами: Н, Ц, Я. Это связано с особенностями почерка и данных, на которых обучалась нейронная сеть, в основном нейронная сеть «дорисовывала» у себя буквы и получались не очень точные результаты.

3.2 Оценка распознавания нейронной сетью при повороте буквы

Для проведения тестирования, в наборе данных были выбраны изображения букв по 10 штук, которые нейронная сеть правильно распознал буквы, далее был написан цикл для поворота изображений, часть данного цикла показан на рисунке 3.1.

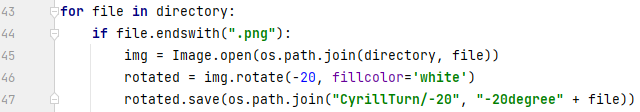


Рисунок 3.1 Цикл для поворота изображений.

Тестирование проводилось со вставкой изображения в приложение с повернутыми буквами от -20 до +20 градусов с шагом в 5 градусов.

По результатам тестирования (Приложение В) итоговая средняя точность – составила 83,83%, это говорит, о том, что нейронная сеть вполне способна правильно распознавать при повороте буквы и обеспечивает точность распознавания около 90% при углах поворота от -5 до -15 градусов с таким поворотом пишет человек, что позволило нейронной сети правильно распознать буквы, но для -20, +5 и +10 градусах точность составила 88,18%, 87,88% и 81,82% соответственно, что вполне хорошо, но на углах +15 и +20 градусах процент распознавания резко падает до 74,24% и 67,58% соответственно, это говорит, о том что нейронная сеть недостаточно обучена и некорректно распознает буквы на таких углах поворота. Наибольший процент букв, в которых нейронная сеть была не очень точна оказались буквы: А, Ж, Й, Л, Ф, Ш, Щ, Ъ, Ь, Ю. В букве А нейронная сеть чаще всего неправильно распознавала буквы Д, Л и Я, в букве Ж с буквами Н и Х, в букве Й с буквами Б, И, П, в букве Л с буквой Х, в букве Ф с буквами Д, П, Р, в букве Ш с буквами Щ, И, в букве Щ с буквами Ц, Ж, в букве Ъ с буквами Е, З, Х, в букве Ь с буквами Ъ, Б, в букве Ю с буквами Н, полная таблица указана в приложении Г.

3.3 Сравнение аналогов Pen to Print, MyScript Write

Тестирование-сравнение проводилось на онлайн-сервисах, использующих нейронные сети – Pen to Print и MyScript Write.

Pen to Print был разработан компанией Serensia Labs и запущен в 2017 году. Компания Serensia Labs является стартапом, специализирующимся на разработке технологий распознавания и преобразования изображений в текст. Сервис доступен для использования, как с персональных компьютеров (веб-версия), так и с мобильных устройств (мобильное приложение). Была использована веб-версия, принимает изображения форматов jpg, jpeg, а также документы формата PDF, сервис является платным, но предоставляется 10 бесплатных попыток, которые обновляются каждый день, тесты проводились в 3 дня по 1 букве алфавита. Из результатов тестирования следует, что сервис не распознает букву Й, а букву Ы распознает как «6 1», либо «6 /», буквы Ь и Ъ как «6 и «Мо» соответственно. Общая точность составила 70%.

MyScript Write создан компанией MyScript и был запущен в 2014 году во Франции. Сервис доступен на персональных компьютерах (веб-версия) и мобильных устройствах (мобильное приложение). В настоящее время MyScript является одним из ведущих производителей технологий распознавания рукописного ввода в мире. Так как MyScript Write позволяет использовать рукописное письмо прямо на сайте, было решено провести его детальное тестирование. Результаты сравнения MyScript и сверточной нейронной сети, созданной в главе 2, пункте 2.3 можно посмотреть на таблице 3.2.

Таблица 3.2 – Результаты сравнения аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Буква алфавита | MyModel, % | MyScript, % | Разница |
| 1 | 2 | 3 | 4 |
| А | 70 | 100 | 30 |
| Б | 90 | 90 | 0 |
| В | 80 | 100 | 20 |
| Г | 100 | 100 | 0 |
| Д | 80 | 100 | 20 |
| Е | 90 | 90 | 0 |
| Ё | 100 | 90 | 10 |
| Ж | 80 | 90 | 10 |
| З | 100 | 100 | 0 |
| И | 70 | 90 | 20 |
| Й | 80 | 80 | 0 |
| К | 80 | 90 | 10 |
| Л | 80 | 100 | 20 |
| М | 70 | 100 | 30 |
| Н | 90 | 90 | 0 |
| О | 90 | 100 | 10 |
| П | 100 | 100 | 0 |
| Р | 100 | 100 | 0 |
| С | 100 | 100 | 0 |
| Т | 90 | 100 | 10 |
| У | 90 | 100 | 10 |
| Ф | 80 | 100 | 20 |
| Х | 100 | 100 | 0 |
| Ц | 80 | 90 | 10 |
| Ч | 100 | 100 | 0 |
| Ш | 80 | 80 | 0 |
| Щ | 70 | 100 | 30 |
| 1 | 2 | 3 | 4 |
| Ъ | 80 | 90 | 10 |
| Ы | 100 | 100 | 0 |
| Ь | 80 | 90 | 10 |
| Э | 100 | 100 | 0 |
| Ю | 100 | 100 | 0 |
| Я | 80 | 100 | 20 |
| Средняя точность, % | 87,27 | 96,06 | 8,79 |

Из результатов тестирования следует, что сервис Pen to Print не распознает букву Й, а букву Ы распознает как «6 1», либо «6 /», буквы Ь и Ъ как «6» и «Мо» соответственно. Сервис MyScript Write распознает все буквы русского алфавита и очень точен, однако сервис тоже допускает ошибки, иногда некорректно распознает буквы Ш и Й. В букве Ш с буквой М, в букве Й с буквами И, Н.

3.4 Выводы к главе 3

Таким образом, в главе 3 были проведены тестирования и сравнения, также описаны результаты тестирований.

Было выявлено, что по результатам оценки общего тестирования точность оказалась – 86,36%. В основном это было связано с особенностями почерка и данных, на которых обучалась нейронная сеть, также она «дорисовывала» у себя буквы и получались не очень точные результаты.

По результатам оценка распознавания при повороте буквы итоговая средняя точность составила - 83,83%, однако точность распознавания достигало около 80-90% при углах поворота +10 градусов и от +5 до -20 градусов, но на углах +15 и +20 градусах процент распознавания резко падает до 67%.

Сравнение аналогов показало, что существуют сильные и более точные нейронные сети, созданные другими компаниями из разных стран. Но созданная нами нейронная сеть показала не плохие результаты, однако требуются доработки в точности распознавании.

ЗАКЛЮЧЕНИЕ

Таким образом, в ходе выпускной квалификационной работы была достигнута поставленная цель, а именно – разработка модели нейронной сети для распознавания рукописных букв.

Т.е. выполнены поставленные задачи:

* Изучение основных принципов функционирования искусственных нейронных сетей, их видов, преимуществ и недостатков;
* Изучение свойств почерка человека и проблем его распознавания;
* Разработка и обучение модели нейронной сети;
* Тестирование модели нейронной сети на точность при распознавании рукописных букв и сравнение с аналогами.

В результате изучения основных принципов функционирования искусственных нейронных сетей была выбрана сверточная нейронная сеть. Перед обучением был кратко описан выбранный набор данных для обучения модели нейронной сети. После обучения нейронной сети, чтобы проверить результат обучения, был разработан графический интерфейс.

В главе 3 были реализованы:

* Общая оценка модели обученной нейронной сети;
* Оценка распознавания при повороте буквы;
* Сравнение аналогов.

Во время общей оценки модели нейронной сети в графическом интерфейсе путем написания всех букв русского алфавита на графическом интерфейсе в 10 подходов была достигнута точность распознавания в 86,36%. Далее во время оценки распознавания при повороте буквы была достигнута средняя точность распознавания в 83,83%, однако при углах поворота +10 градусов и от +5 до -20 градусов точность распознавания достигало около 80-90%, но на углах +15 и +20 градусах процент распознавания резко падает до 67%. Также во время сравнения аналогов показало, что существуют сильные и более точные нейронные сети, созданные другими компаниями из разных стран. Однако созданная нейронная сеть показала не плохие результаты, но требуются доработки в точности распознавания.

Результаты данной работы могут быть использованы для дальнейшей разработки модели нейронной сети, которая будет распознавать слова с использованием приемов сегментации букв. Распознавание рукописных букв, может быть использовано, например:

* Для перевода архивного рукописного текста на электронный.
* Распознавание адресов и почтовых индексов на конвертах и посылках.

Для улучшения разработанной нейронной сети предлагается несколько способов решения задачи:

* Оптимизировать нейронную сеть с помощью дополнения ранее выбранного набора данных большим количеством разнообразных изображений букв, либо выбрать аналогичный набор данных, либо совместить несколько наборов данных;
* Для более точного распознавания букв можно использовать метод препроцессинга изображений, например, с помощью нормализации яркости, поворотов изображений или обрезки. Это может устранить шумы и перекосы в изображении, что может привести к более точному распознаванию;
* Разработать нейронную сеть, которая будет распознавать полноценные слова и тексты с помощью сегментации букв;
* Разработать полноценно работоспособный программное приложение, либо веб-сайт.

Таким образом, вышеперечисленные методы могут быть использованы в будущих исследованиях по этой теме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Аггарвал, Ч. Нейронные сети и глубокое обучение: учебный курс [Текст] / Ч. Аггарвал. – Санкт-Петербург : ООО «Диалектика», 2020. – 752 с.
2. Гафаров, Ф.М. Искусственные нейронные сети и приложения: учеб. пособие [Текст] / Ф.М. Гафаров, А.Ф. Галимянов. – Казань : Изд-во Казан. Ун-та, 2018. – 121 с.
3. Джулли, А. Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow [Текст] / А. Джулли, С. Пал. – Москва : ДМК Пресс, 2018. – 294 с.
4. Лекун, Я. Как учится машина: Революция в области нейронных сетей и глубокого обучения [Текст] / Ян Лекун. – Пер. с фр. – Москва : Альпина ПРО, 2021. – 324 с.
5. Николенко, С. Глубокое обучение. [Текст] / С. Николенко, А. Кадурин, Е. Архангельская. – Санкт-Петербург : Питер, 2018. – 480 с.
6. Омельяненко, Я. Эволюционные нейросети на языке Python [Текст] / Я. Омельяненко. – Москва : ДМК Пресс, 2020. – 310 с.
7. Орельен, Ж. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем [Текст] / Ж. Орельен. – Санкт-Петербург : ООО «Диалектика», 2020. – 1040 с.
8. Прохоренок, Н.А. Python 3 и PyQt 5. Разработка приложений [Текст] / Н.А. Прохоренок, В.А. Дронов. – Санкт-Петербург : БХВ-Петербург, 2016. – 832 с.
9. Ростовцев, В.С. Искусственные нейронные сети: учебник [Текст] / В.С. Ростовцев. – Киров : Изд-во ВятГУ, 2014. – 208 с.
10. Рышард, Тадеусевич. Элементарное введение в технологию нейронных сетей с примерами программ [Текст] / Рышард Тадеусевич, Барбара Боровик, Томаш Гончаж, Бартош Леппер. – Москва : Горячая линия – Телеком, 2011. – 408 с.
11. Саттон, Р.С. Обучение с подкреплением: Введение [Текст] / Р.С. Саттон, Э. Дж. Барто. – 2-е изд. – Москва : ДМК Пресс, 2020. – 552 с.
12. Тарик, Р. Создаем нейронную сеть [Текст] / Р. Тарик. – Санкт-Петербург : ООО «Альфа-книга», 2017. – 272 с.
13. Хайкин, С. Нейронные сети: полный курс [Текст] / С. Хайкин. – 2-е изд. – Москва : Издательский дом «Вильямс», 2006. – 1104 с.
14. Хапке, Х. Разработка конвейеров машинного обучения. Автоматизация жизненных циклов модели с помощью TensorFlow [Текст] / Х. Хапке, К. Нельсон. – Москва : ДМК Пресс, 2021. – 346 с.
15. Шолле, Ф. Глубокое обучение на Python [Текст] / Ф. Шолле. – Санкт-Петербург : Питер, 2018. – 400 с.
16. Ian, Goodfellow. Deep Learning / Goodfellow Ian, Bengio Yoshua, Courville Aaron : MIT Press, 2016. – 800 с. – Текст : электронный. – URL: <https://www.deeplearningbook.org/contents/TOC.html#pf9> (дата обращения: 19.05.2023).
17. SLY\_G. Нейросети и глубокое обучение / SLY\_G. – Текст : электронный. – URL: <https://habr.com/ru/articles/456738/> (дата обращения 15.05.2023)
18. Softmax URL: <https://towardsdatascience.com/softmax-function-simplified-714068bf8156>
19. Srivastava Nitish. Dropout: A Simple Way to Prevent Neural Networksfrom Overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. – Toronto, 2014. – 30 p. – URL: https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf
20. Tensorflow : [сайт]. – URL: <https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten>
21. Vinod Nair. Rectified Linear Units Improve Restricted Boltzmann Machines. / Nair Vinod, Hinton E. Geoffrey. – Текст: электронный. – URL: <https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf> (дата обращения 21.05.2023)

ПРИЛОЖЕНИЕ А

# Импорт необходимых библиотек/модулей

import numpy as np

import os

import random

from PIL import Image

from keras.models import Sequential

from keras.layers import Dense, Flatten

from keras.layers import Dropout, Conv2D, MaxPooling2D

from keras.utils import to\_categorical

from sklearn.utils import shuffle

# Загрузка данных для обучения и пустые массивы

img\_path = 'E:/Универ/4 курс/Диплом/Проекты pycharm/diplom/Cyrillic/'

X\_train = []

y\_train = []

X\_test = []

y\_test = []

os.chdir(img\_path)

dirs = os.listdir()

class\_dot = 0

for d in dirs:

print(d, end=' ')

os.chdir(os.path.join(img\_path, d))

index = 0

os.chdir(os.path.join(img\_path, d))

# Дополнительная аугментация изображений с помощью: поворотов, смещения, уменьшения изображений

files = os.listdir()

for f in files:

for way in ['decrease', 'rotate+', 'rotate-', 'shift']:

# Уменьшение

if way == 'decrease':

img = Image.open(f)

res\_img = Image.new("RGB", img.size, (255, 255, 255)) # (255, 255, 255) (145, 145, 145)

res\_img.paste(img, mask=img.split()[3])

res\_img = res\_img.resize((32, 32))

img\_arr = np.array(res\_img)

# Поворот по часовой стрелке

elif way == 'rotate+':

img = Image.open(f)

res\_img = Image.new("RGB", img.size, (255, 255, 255))

res\_img.paste(img, mask=img.split()[3])

angle = random.randint(0, 30)

res\_img = res\_img.rotate(angle, fillcolor='white')

res\_img = res\_img.resize((32, 32))

img\_arr = np.array(res\_img)

# Поворот против часовой стрелки

elif way == 'rotate-':

img = Image.open(f)

res\_img = res\_img.resize((32, 32))

res\_img = Image.new("RGB", img.size, (255, 255, 255))

res\_img.paste(img, mask=img.split()[3])

angle = random.randint(-30, 0)

res\_img = res\_img.rotate(angle, fillcolor='white')

res\_img = res\_img.resize((32, 32))

img\_arr = np.array(res\_img)

# Смещение

elif way == 'shift':

img = Image.open(f)

res\_img = Image.new("RGB", img.size, (255, 255, 255))

res\_img.paste(img, mask=img.split()[3])

res\_img = res\_img.rotate(0, fillcolor='white')

horizontal, vertical = random.randint(-5, 5), random.randint(-5, 5)

res\_img = res\_img.resize((32, 32))

img\_arr = np.array(res\_img)

# Разделение изображений на тестовые и тренировочные наборы

if index >= round(len(files) / 100 \* 85):

X\_test.append(img\_arr)

y\_test.append([class\_dot])

else:

X\_train.append(img\_arr)

y\_train.append([class\_dot])

index += 1

class\_dot += 1

# Разделение данных на тренировочные и тестовые выборки

X\_train = np.array(X\_train, dtype='float32')

y\_train = np.array(y\_train, dtype='uint8')

X\_test = np.array(X\_test, dtype='float32')

y\_test = np.array(y\_test, dtype='uint8')

Y\_train = to\_categorical(y\_train, 33)

Y\_test = to\_categorical(y\_test, 33)

# Нормализация данных путем деления

X\_train = X\_train / 255

# Перемешивание тестовой выборки

X\_test, Y\_test = shuffle(X\_test / 255, Y\_test)

print('\nНаборы данных успешно аугментированы и разделены')

print('Всего тренировочных данных:')

print(len(X\_train))

print('Всего тестовых данных:')

print(len(X\_test))

# Архитектура модели

model = Sequential()

model.add(Conv2D(16, kernel\_size=(3, 3), padding='same', input\_shape=(32, 32, 3), activation='relu'))

model.add(Conv2D(16, kernel\_size=(3, 3), activation='relu'))

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

model.add(Dropout(0.1))

model.add(Conv2D(32, kernel\_size=(3, 3), padding='same', activation='relu'))

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

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

model.add(Dropout(0.3))

model.add(Conv2D(128, kernel\_size=(3, 3), padding='same', activation='relu'))

model.add(Conv2D(256, kernel\_size=(3, 3), activation='relu'))

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

model.add(Dropout(0.5))

model.add(Flatten())

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

model.add(Dense(512, activation='relu'))

model.add(Dense(33, activation='softmax'))

# Настройка обучения методом compile

model.compile(loss='categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

# Обучение модели методом fit

model.fit(X\_train, Y\_train,

batch\_size=80, #256,

epochs=200, #10

validation\_data=(X\_test, Y\_test),

shuffle=True)

# Сохранение модели нейронной сети для дальнейшего использования

os.chdir(r'E:/Универ/4 курс/Диплом/Проекты pycharm/diplom/')

model.save(f'model\_2\_CoMNIST.h5')

print('Модель успешно обучена и сохранена')

ПРИЛОЖЕНИЕ Б

# Импорт необходимых библиотек/модулей

import numpy as np

import os

import sys

from PIL import Image

from PyQt5.QtCore import Qt, QPoint

from PyQt5.QtGui import QImage, QPainter, QPen, QFont

from PyQt5.QtWidgets import QPushButton, QMainWindow, QLabel, QLineEdit, QApplication, QFileDialog

from tensorflow import keras

# Установка переменной окружения QT\_QPA\_PLATFORM\_PLUGIN\_PATH для работы с плагином PyQt5

os.environ['QT\_QPA\_PLATFORM\_PLUGIN\_PATH'] = 'venv/Lib/site-packages/PyQt5/Qt5/plugins/'

class Window(QMainWindow):

# Графический интерфейс

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Наименование и размерность окна

title = "Распознавание буквы кириллицы"

top = 200

left = 200

width = 540

height = 340

# Шрифты

font = QFont('Arial', 10)

fontBig = QFont('Arial', 12)

# Кисть для рисования

self.drawing = False

self.brushSize = 8

self.brushColor = Qt.black

self.lastPoint = QPoint()

# Холст

self.image = QImage(278, 278, QImage.Format\_RGB32)

self.image.fill(Qt.white)

# Лейбл "результат"

label = QLabel('РЕЗУЛЬТАТ:', self)

label.setFont(font)

label.move(290, 140)

# Размерность и расположение поля "результат"

self.line = QLineEdit(self)

self.line.setFont(fontBig)

self.line.move(370, 134)

self.line.resize(99, 42)

# Кнопка "распознать"

recognize\_button = QPushButton('РАСПОЗНАТЬ', self)

recognize\_button.setFont(font)

recognize\_button.move(290, 30)

recognize\_button.resize(230, 33)

recognize\_button.clicked.connect(self.save)

recognize\_button.clicked.connect(self.predicting)

# Кнопка "очистить"

clean\_button = QPushButton('ОЧИСТИТЬ', self)

clean\_button.setFont(font)

clean\_button.move(290, 80)

clean\_button.resize(230, 33)

clean\_button.clicked.connect(self.clear)

clean\_button.clicked.connect(self.line.clear)

# Кнопка "загрузить изображение"

load\_button = QPushButton('ЗАГРУЗИТЬ ИЗОБРАЖЕНИЕ', self)

load\_button.setFont(font)

load\_button.move(290, 246)

load\_button.resize(230, 33)

load\_button.clicked.connect(self.load\_image)

load\_button.clicked.connect(self.save)

load\_button.clicked.connect(self.predicting)

# Лейбл "drag n drop"

drag\_n\_drop = QLabel('Вы также можете\nперетащить изображение на окно', self)

drag\_n\_drop.setFont(font)

drag\_n\_drop.move(290,190)

drag\_n\_drop.resize(230,30)

# Разрешаем принимать файлы на окно GUI

self.setAcceptDrops(True)

self.setWindowTitle(title)

self.setGeometry(top, left, width, height)

# Метод принимающий результат распознавания буквы и вывод ее на поле "результат"

def print\_letter(self, result):

letters = "ЁАБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

self.line.setText(str(letters[result]))

return letters[result]

# Метод загрузки модели нейронной сети и изображения буквы

def predicting(self):

image = keras.preprocessing.image

model = keras.models.load\_model('model\_2\_UPD+.h5')

img = self.image.scaled(32, 32).convertToFormat(QImage.Format\_RGB32)

Image.fromqimage(img)

# Преобразование изображения в массив и передача в модель

img\_pil = Image.fromqimage(img)

x = image.img\_to\_array(img\_pil)

x = np.expand\_dims(x, axis=0)

images = np.vstack([x])

classes = model.predict(images, batch\_size=1)

# Получение результата

result = int(np.argmax(classes))

self.print\_letter(result)

# Методы mousePressEvent, mouseMoveEvent, mouseReleaseEvent, paintEvent

# отображают реагирование на движение и нажатие кнопок мыши и отображение изменения на экране

def mousePressEvent(self, event):

if event.button() == Qt.LeftButton:

self.drawing = True

self.lastPoint = event.pos()

def mouseMoveEvent(self, event):

if (event.buttons() & Qt.LeftButton) & self.drawing:

painter = QPainter(self.image)

painter.setPen(QPen(self.brushColor, self.brushSize, Qt.SolidLine, Qt.RoundCap, Qt.RoundJoin))

painter.drawLine(self.lastPoint, event.pos())

self.lastPoint = event.pos()

self.update()

def mouseReleaseEvent(self, event):

if event.button() == Qt.LeftButton:

self.drawing = False

def paintEvent(self, event):

canvas\_painter = QPainter(self)

canvas\_painter.drawImage(0, 0, self.image)

# Метод сохранения изображения буквы

def save(self):

self.image.save('res.jpg')

# Метод очищения холста

def clear(self):

self.image = QImage(278, 278, QImage.Format\_RGB32)

self.image.fill(Qt.white)

self.update()

# Метод загрузки изображения из папки

def load\_image(self):

fname = QFileDialog.getOpenFileName(self, 'Выбрать изображение', '', 'Изображения (\*.jpg \*.jpeg \*.png)')[0]

if fname:

img = Image.open(fname)

img = img.resize((278, 278), Image.BILINEAR)

img.save(fname)

self.image = QImage(fname)

self.update()

# Методы dragEnterEvent и dropEvent обрабатывают события перетаскивания файлов на окно приложения

def dragEnterEvent(self, event):

if event.mimeData().hasUrls():

event.accept()

else:

event.ignore()

def dropEvent(self, event):

for url in event.mimeData().urls():

fname = url.toLocalFile()

if fname.lower().endswith(('.jpg', '.jpeg', '.png')):

img = Image.open(fname)

img = img.resize((278, 278), Image.BILINEAR)

img.save(fname)

self.image = QImage(fname)

self.predicting()

self.update()

break

# Конструкция для запуска приложения с GUI

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

window = Window()

window.show()

app.exec()

ПРИЛОЖЕНИЕ В

Угол

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | -20 | -15 | -10 | -5 | 5 | 10 | 15 | 20 | Средняя точность |
| 1  Буква | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| А | 30 | 70 | 80 | 90 | 90 | 70 | 40 | 40 | 63,75 |
| Б | 100 | 100 | 100 | 100 | 100 | 100 | 80 | 80 | 95,00 |
| В | 90 | 100 | 80 | 80 | 100 | 80 | 100 | 80 | 88,75 |
| Г | 90 | 100 | 90 | 90 | 90 | 100 | 80 | 70 | 88,75 |
| Д | 90 | 90 | 100 | 90 | 80 | 90 | 80 | 80 | 87,50 |
| Е | 70 | 80 | 90 | 100 | 100 | 80 | 90 | 70 | 85,00 |
| Ё | 80 | 90 | 100 | 100 | 100 | 80 | 80 | 70 | 87,50 |
| Ж | 100 | 80 | 90 | 90 | 70 | 60 | 60 | 30 | 72,50 |
| З | 100 | 90 | 90 | 100 | 90 | 90 | 90 | 90 | 92,50 |
| И | 100 | 90 | 90 | 70 | 90 | 90 | 90 | 90 | 88,75 |
| Й | 70 | 90 | 90 | 70 | 90 | 80 | 80 | 60 | 78,75 |
| К | 100 | 90 | 90 | 100 | 90 | 90 | 80 | 60 | 87,50 |
| Л | 80 | 90 | 90 | 80 | 80 | 70 | 70 | 70 | 78,75 |
| М | 90 | 90 | 90 | 90 | 90 | 80 | 70 | 70 | 83,75 |
| Н | 100 | 100 | 90 | 100 | 100 | 80 | 60 | 50 | 85,00 |
| О | 80 | 90 | 80 | 90 | 100 | 100 | 90 | 90 | 90,00 |
| П | 100 | 80 | 80 | 80 | 90 | 80 | 80 | 70 | 82,50 |
| Р | 90 | 100 | 100 | 90 | 90 | 90 | 90 | 80 | 91,25 |
| С | 100 | 90 | 100 | 100 | 100 | 100 | 90 | 90 | 96,25 |
| Т | 100 | 90 | 90 | 100 | 90 | 90 | 60 | 80 | 87,50 |
| У | 90 | 70 | 90 | 90 | 100 | 100 | 80 | 90 | 88,75 |
| Ф | 80 | 80 | 90 | 80 | 70 | 70 | 60 | 50 | 72,50 |
| Х | 90 | 100 | 100 | 100 | 90 | 90 | 70 | 70 | 88,75 |
| Ц | 70 | 100 | 100 | 100 | 90 | 90 | 80 | 80 | 88,75 |
| Ч | 90 | 90 | 90 | 90 | 100 | 100 | 90 | 80 | 91,25 |
| Ш | 100 | 100 | 100 | 100 | 90 | 60 | 40 | 10 | 75,00 |
| Щ | 80 | 100 | 100 | 90 | 80 | 70 | 50 | 50 | 77,50 |
| Ъ | 80 | 100 | 70 | 70 | 70 | 70 | 70 | 40 | 71,25 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Ы | 100 | 100 | 100 | 80 | 80 | 70 | 60 | 60 | 81,25 |
| Ь | 90 | 70 | 50 | 70 | 50 | 50 | 40 | 60 | 60,00 |
| Э | 100 | 90 | 90 | 100 | 100 | 90 | 100 | 90 | 95,00 |
| Ю | 100 | 90 | 90 | 90 | 60 | 60 | 70 | 60 | 77,50 |
| Я | 80 | 100 | 100 | 100 | 90 | 80 | 80 | 70 | 87,50 |
| Средняя точность | 88,182 | 90,606 | 90,303 | 90 | 87,879 | 81,818 | 74,242 | 67,576 | 83,826 |

ПРИЛОЖЕНИЕ Г

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 20  Угол | 15 | 10 | 5 | -5 | -10 | -15 | -20 |
| 1  Буква | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| А | Л, Я, Д | Л, Я, Д, | Я | Я | Я | Л, Д | Л, Д | Д, Х |
| Б | Е, П | Е, Ъ |  |  |  |  |  |  |
| В | Ь |  | Б |  | Б | Б |  | Б |
| Г | П, С, | Т, С |  | П | П | П |  | П |
| Д | Л | Л | Л | Ъ | Ъ |  | Ф | А |
| Е | Ё | Ё | Ё |  |  | Ё | Ё | Ё, С |
| Ё | Е | Е | Е |  |  |  | Е | Е |
| Ж | Н, И, Х, | Й, Х, Н | Й, Н | Н, Х | Н | Х | Х |  |
| З | Ъ | Е | Е | Е |  | Э | Э |  |
| И | Й | Ц | Ц | Ц | Ц, Й | Й | Й |  |
| Й | Ё, Х, П, Ъ | Ъ, П | И | И | Ъ, Б, П | И | И | И, Б |
| К | Х, А, П | А, Х | Х | А |  | Х | Х |  |
| Л | Х | Х | Х, П | Х | Х | Х | Х | Г, Х |
| М | Л, Д | Л, Д, | Л | У | У | Л | Н | Л |
| Н | У, И, Л | И, Л | И, Л |  |  | У |  |  |
| О | С | С |  |  | С | С | С | С |
| П | Г, В, С | В, Л, | В, Л | Л | Л, Т | Г | Г | Г |
| Р | Д, Л | В | В | В | В |  |  | Д |
| С | Е | Е |  |  |  |  | Е | Е |
| Т | Г | Г | Г | Г |  | Г | Г | Г |
| У | Э | Ч |  |  | Л | С | Х, Э | Э |
| Ф | Х, Д, П | Д | Д, Р | Х, П | Х, П | Р | Р | Р, Д |
| Х | Э, Л, Р | Э, Д | Э | Ж |  |  |  | Ж |
| Ц | Ъ | Ч | Ч | И |  |  |  | В, Ъ, И |
| Ч | Ъ, Е | Ъ |  |  | Ц | Ц | И | Ц |
| Ш | И, Ъ, Ц, Щ, Е | Щ, И | Щ, И | O |  |  |  |  |
| Щ | Ц, Ч | Ц, И | Ц, И | Ц | Ж | Ж | Ж | Ш |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Ъ | З, Е | Х | Х | Х, Е | Х, Е | З | З | М |
| Ы | Х, Б, Ь | Х, Ъ | Х, Ъ | Ь | Ь |  |  |  |
| Ь | Ъ, Б | Ъ, Б, | Ъ, Б, | Ъ | Ъ | Ъ | Ъ | Ъ |
| Э | З |  | З |  |  | С | З |  |
| Ю | М, Й, Е | Н | Н | Н | Н | Н | Д |  |
| Я | У, Ф, Х | У, Ё | У, Ё | У |  | У | У | У |