

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра технологий программирования

**СИСТЕМА РАСПОЗНАВАНИЯ РУКОПИСНОГО ТЕКСТА НА ОСНОВЕ
НЕЙРОННЫХ СЕТЕЙ**

Курсовой проект

Солодуха Дмитрия
Владимировича

студента 3 курса,
специальность
«прикладная
информатика»

Научный руководитель:
старший преподаватель
Ю.В. Ветров

Минск, 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
ГЛАВА 1 ПОСТАНОВКА ЗАДАЧИ И ПОДХОДЫ К ЕЁ РЕШЕНИЮ	6
1.1. <i>Режимы онлайн и оффлайн.....</i>	<i>6</i>
1.1.1. <i>Онлайн-распознавание</i>	<i>6</i>
1.1.2. <i>Оффлайн-распознавание</i>	<i>6</i>
1.2. <i>Основные подходы решения задачи распознавания в режиме оффлайн....</i>	<i>7</i>
1.2.1. <i>Предобработка</i>	<i>7</i>
1.2.2. <i>Процесс распознавания</i>	<i>10</i>
1.2.3. <i>Постобработка.....</i>	<i>11</i>
ГЛАВА 2 НЕЙРОННЫЕ СЕТИ.....	13
2.1. <i>Полносвязные нейронные сети</i>	<i>13</i>
2.1.2. <i>Искусственный нейрон.....</i>	<i>13</i>
2.1.3. <i>Функции активации</i>	<i>14</i>
2.1.4. <i>Обучение нейронной сети</i>	<i>16</i>
2.2. <i>Свёрточные нейронные сети.....</i>	<i>16</i>
2.2.1. <i>Свёрточные слои.....</i>	<i>17</i>
2.2.2. <i>Слой субдискретизации.....</i>	<i>18</i>
ГЛАВА 3 РАЗРАБОТКА ПРОТОТИПА СИСТЕМЫ	19
3.1. <i>Проектирование системы.....</i>	<i>19</i>
3.1.1. <i>Предобработка</i>	<i>19</i>
3.1.2. <i>Распознавание</i>	<i>19</i>
3.1.3. <i>Постобработка.....</i>	<i>20</i>
3.1.4. <i>Вход и выход системы.....</i>	<i>20</i>
3.2. <i>Реализация.....</i>	<i>20</i>
3.2.1. <i>Предобработка</i>	<i>20</i>
3.2.2. <i>Распознавание</i>	<i>21</i>
3.2.3. <i>Постобработка.....</i>	<i>24</i>
3.2.4. <i>Ввод текста.....</i>	<i>24</i>
3.3. <i>Оценка работы системы</i>	<i>25</i>
3.3.2. <i>Недостатки</i>	<i>26</i>
3.3.3. <i>Выводы</i>	<i>26</i>
ЗАКЛЮЧЕНИЕ	28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	29
ПРИЛОЖЕНИЯ	31
<i>Приложение А</i>	<i>31</i>

ВВЕДЕНИЕ

В современном мире рукописные тексты появляются все реже и реже в повседневной жизни человека. Может показаться, что они и вовсе могут быть вытеснены цифровыми текстами. Однако проблема распознавания рукописного текста всё ещё является непременно важной. Ведь имеется множество различных источников информации, которые фактически полностью состоят из рукописного текста, а текст, который получен в результате сканирования данных источников хранится в виде изображения, что очень часто не позволяет работать с ним в полной мере. Поэтому их распознавание является достаточно важной проблемой. Сюда также можно отнести и распознавание различных форм и бланков, которые части заполнены «от руки», и использования распознавания в электронных записных книжках, и многое другое. Как видно, в современном мире задача распознавания непрерывного рукописного текста легко находит применение. Также важно отметить, что если область распознавания печатных текстов (OCR – optical character recognition) компьютером можно считать практически исследованной, то для рукописных текстов (HWR – handwriting recognition) такого сказать нельзя.

Если раньше такая задача могла выглядеть неподъёмно, как с теоретической точки зрения — не было придумано достаточно хороших алгоритмов, так и с практической — для реализации алгоритмов не хватало вычислительных ресурсов, то на данном этапе разработано уже множество подходов для её решения. Существует целый класс методов, которые решают схожие задачи — машинное обучение и компьютерное зрение.

В данной работе основное внимание было уделено распознаванию рукописного текста в режиме оффлайн. Были рассмотрены основные методы, которые позволяют решить данную задачу. Целью работы ставится разработка прототипа системы распознавания рукописного текста на основе нейронных сетей.

Задачами работы являются:

1. Изучение существующих подходов при распознавании рукописного текста и основных алгоритмов, используемых в данных подходах
2. Рассмотрение свёрточных нейронных сетей в контексте задачи распознавания
3. Разработка прототипа системы распознавания рукописного текста

В рамках курсового проектирования были применены следующие методы исследования: анализ, моделирование, эксперимент. В частности, в первой главе

был проведен анализ задачи и существующих подходов для её решения, описаны методы предобработки, распознавания и постобработки изображений рукописного текста. Во второй главе проведено исследование нейронных сетей, даны основные определения и понятие нейронных сетей, рассмотрены основные составные части архитектуры свёрточных нейронных сетей. В третьей главе рассмотрена реализация прототипа системы и архитектура свёрточной сети, используемой для распознавания, проведены и описаны эксперименты, дана оценка качества полученных результатов.

ГЛАВА 1

ПОСТАНОВКА ЗАДАЧИ И ПОДХОДЫ К ЕЁ РЕШЕНИЮ

Чаще всего в системы распознавания рукописного текста, которые описываются в литературе, можно разделить на системы, работающие в онлайн и оффлайн режиме. Это разделение строится на основании данных, с которыми работают данные системы.

1.1. Режимы онлайн и оффлайн

1.1.1. Онлайн-распознавание

При распознавании в режиме онлайн распознавание ведется параллельно с написанием рукописного текста. Так как процессы формирования изображения текста и его ввод в систему распознавания совпадают, мы можем отслеживать сам процесс начертания символов. И помимо графической информации мы сможем иметь и различную информацию о структуре написанного текста. Это может быть скорость движения стилуса, направление его движения, сила нажатия на стилус при написании и другие важные для нас признаки. Однако такой подход достаточно ограничен, так как использовать его можно только со специальными устройствами ввода, например, планшетных компьютерах. При этом задача онлайн-распознавания считается хорошо изученной и имеет достаточно много хороших решений [7].

1.1.2. Оффлайн-распознавание

При распознавании в режиме же мы владеем только графической информацией. И уже тут можно заметить, что такая задача гораздо сложнее, ведь мы должны работать только с подмножеством данных, которые доступны нам при решении задачи онлайн. И если в задаче онлайн распознавании мы должны были бороться с такими проблемами, как:

- Вариативность в написании символов
- Орфографические ошибки

То теперь сюда можно также добавить:

- Особенности начертания
- Наложение элементов друг на друга
- Дефекты изображения, различные исправления, пометки

И очевидно, что этот список можно продолжить.

При этом задача оффлайн-распознавания является более общей и находит своё применение в различных сферах. Сюда можно отнести даже само онлайн-распознавание, так как в нем мы практически всегда можем обладать данными нужным для оффлайн режима. Стоит отметить, что существуют системы совмещающие эти два режима [12]. Однако считается, что такая задача до сих пор не решена. Именно поэтому в этой работе я попытаюсь рассмотреть одно из возможных решений задачи распознавания рукописного текста в режиме оффлайн.

1.2. Основные подходы решения задачи распознавания в режиме оффлайн

Задачу распознавания в режиме оффлайн обычно решают в несколько этапов:

1. Предобработка изображения
 - a. Нормализация
 - b. Перевод в черно-белый формат с отделением фона
 - c. Выделение предложений, слов, отдельных букв
2. Распознавание выделенных компонент
3. Постобработка распознанного текста

При этом для каждого из этапов можно применять свой подход, коих существует достаточно много.

1.2.1. Предобработка

Наиболее частыми операциями, которые применяются перед непосредственным распознаванием являются: очистка изображения от дефектов, пороговая бинаризация исходного изображения, отделение фона, сегментация предложений, слов и, если возможно отдельных букв. Раскрою наиболее важные операции.

Пороговая бинаризация является чуть ли не основной частью, так как от ее правильности сильно зависят дальнейшие операции. Но данная задача не является новой и для нее существует множество хороших методов. Достаточно часто её решают с помощью гистограмм яркости. Строится гистограмма яркости пикселей и по ней определяется некоторое «оптимальное» пороговое значение. На основании этого порогового значение пиксели с яркостью выше принимаются за фон, а ниже – за искомый текст. Таким образом вся суть сводится к нахождению данного

порогового значения. Пример такой пороговой бинаризации можно увидеть на рисунке (Рисунок 1.1) Наиболее продвинутым алгоритмом, который находит такое пороговое значение является алгоритм Оцу [6].

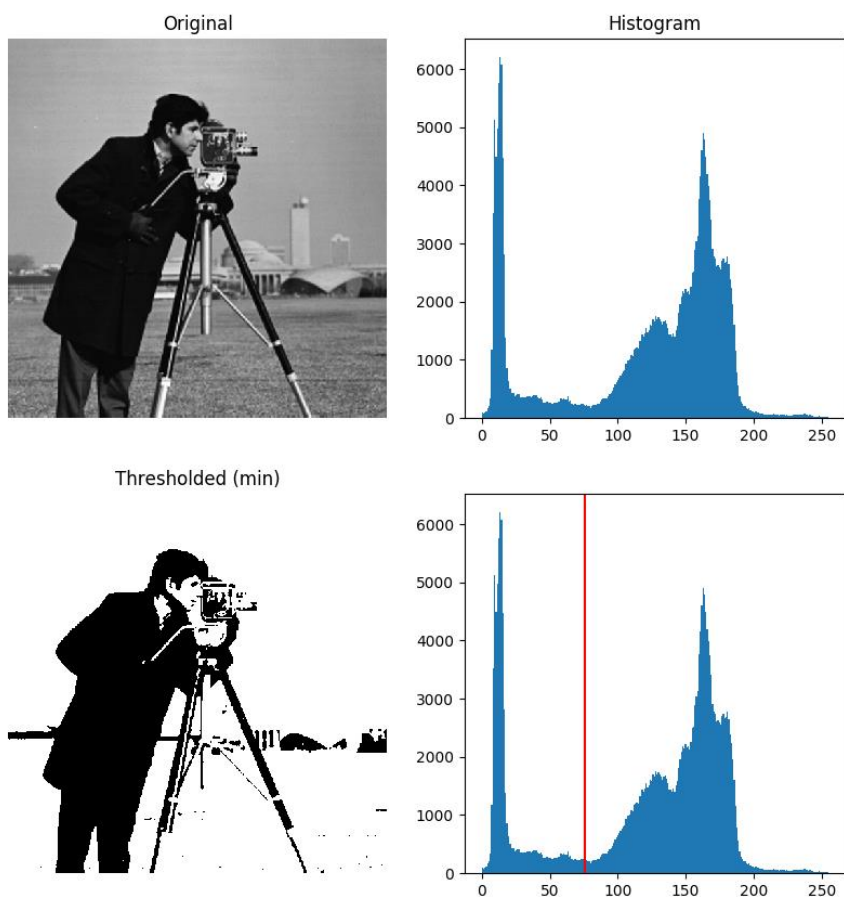


Рисунок 1.1 Пример пороговой бинаризации с помощью гистограмм

Также стоит отметить, что если изначальное изображение сильно зашумлено, то можно использовать не глобальную бинаризацию, а локальную или адаптивную. В самом простом случае она аналогична глобальной, но только производится на меньших участках.

После бинаризации обычно выделяют часть, где локализован текст. Проще всего это сделать с помощью алгоритма связанных компонентов. После данные компоненты можно классифицировать необходимым образом и оставить только то, что относится к тексту. Обычно для классификации применяют различные геометрические методы или эвристики.

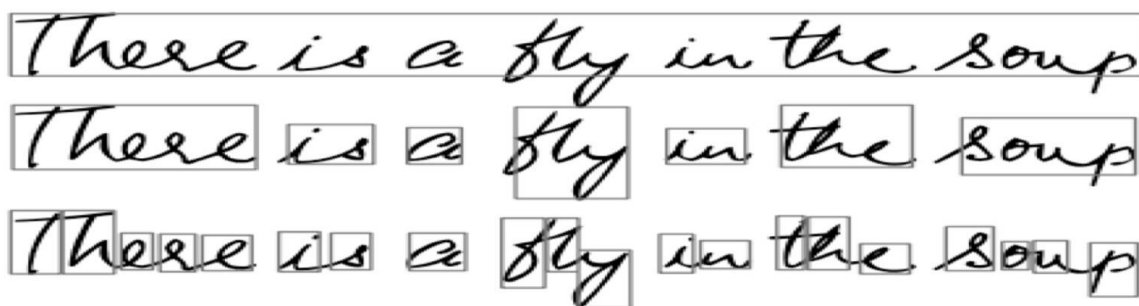


Рисунок 1.2 Сегментация сверху вниз. Сначала сегментируются отдельные строки, а после слова и буквы

Следующим этапом является сегментация текста на строки, слова и буквы (Рисунок 1.2). Здесь стоит отметить, что данная задача для машинных текстов считается хорошо изученной и решенной, но вот с рукописным текстом дела обстоят гораздо хуже. В очень редких случаях здесь можно применить алгоритмы пригодные для машинных текстов. Обычно этому мешает высокая вариативность в написании, слитность в написании букв, отсутствие параллельности строк и тому подобное. Если говорить про сегментацию строк, то наиболее часто описываем методом в литературе является метод профиля горизонтальной проекции. Опять-таки строится диаграмма пикселей изображений вдоль горизонтальных линий, а после не ней ищутся локальный минимумы, которые и принимаются за условные разделители строк. Примером работы данного метода может служить следующая иллюстрация (Рисунок 1.3).

Julian's mother performed
act of
motherly love-she put her
children's education ahead
of her own life.

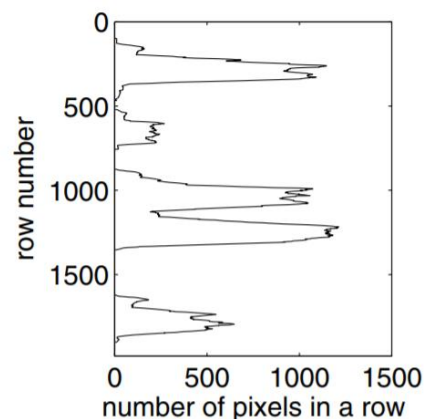


Рисунок 1.3 Диаграмма черных пикселей для рукописного текста при применении метода профиля горизонтальной проекции

Однако, как уже было сказано ранее, не всегда строки в тексте могут быть параллельными, поэтому чаще используется модификация данного метода, в которой минимумы ищутся адаптивно и локально [8][5].

Иногда также пробуют применять схожий метод к разделению букв и слов, но уже с вертикальными проекциями. Для слов такой метод получается достаточно хорошим, а вот отдельные буквы сегментировать им уже практически невероятно.

Другим часто встречающимся методом, который можно применять для сегментации является группировочный метод, который основан на объединении связанных компонент по их различным геометрическим факторам. Используя его можно просто объединять буквы в слова или слова в строки. Он очень прост в понимании и отлично проявляет себя при распознавании машинных текстов. Но опять же его использование для рукописного распознавания не очень эффективно. Так как он может объединять несколько близко расположенных слов или строк. Существует также некоторая его модификация, которая основана на нахождении эмпирической плотности распределения черных пикселей (метод вероятностных карт) [5].

1.2.2. Процесс распознавания

Метод, который будет применяться в распознавании во многом зависит от, того на как именно мы сегментировали текст. Чаще всего сегментация рукописного текста заканчивается на отделении различных слов, так как отделить букву является гораздо более сложной задачей. В таком случае применяется два основных подхода:

- Скрытая марковская модель
- Искусственная нейронная сеть

В современной литературе первый метод отходит на задний план, хотя ранее считался достаточно перспективным. Наибольший же интерес сейчас представляют искусственные нейронные сети, а особенно сети, использующие Connectionist Temporal Classification (CTC) – специальную функцию потерь, которая позволяет работать сразу со всей входной последовательностью и используется при обучении рекуррентных нейронных сетей [3]. В настоящее время именно такие сети чаще всего используют для распознавания речи и рукописного текста.

Рассмотрим случай, когда у нас получилось сегментировать рукописный текст до уровня букв. Обычно так получается если в нашей задаче изначально были некоторые ограничения, которые обусловлены её спецификой. Например, при анализе бланков, где буквы записаны в отдельных клеточках такое проверить очень просто. В таком случае мы здесь решается простая задача классификация, которую уже можно решить многими методами машинного обучения. Но тут уже можно точно сказать, что из изученных на сегодняшний день моделей наиболее точными будут нейронные сети. Можно просто проанализировать точность

методов на задаче распознавания цифр на наборе данных MNIST [4]. Таблица с результатами различных моделей машинного обучения представлена ниже (Таблица 1.1).

Таблица 1.1 Точность методов на наборе данных MNIST. Данные с официального сайта

Тип	Структура	Предварительная обработка	Ошибка (%)
Линейный классификатор	Одноуровневый перцептрон	Нет	12
Метод k ближайших соседей	K-NN с нелинейной деформацией (P2DHMDM)	Shiftable edges	0.52
Метод опорных векторов	Виртуальная система опорных векторов deg-9 poly, 2-pixel jittered	Выравнивание	0.56
Нейронная сеть	2-уровневая сеть 784-800-10	Нет	1.6
Глубокая нейронная сеть	6-уровневая сеть 784-2500-2000-1500-1000-500-10	Нет	0.35
Свёрточная нейронная сеть	6-уровневая сеть 784-50-100-500-1000-10-10	Расширение данных для обучения	0.27

Стоит также отметить, что выбор метода распознавания зависит не только от способа и результатов предобработки, но и от алгоритма последующей постобработки, наличия языковой модели и других факторов.

1.2.3. Постобработка

После того, как мы получили текст можно проверить его на очевидные ошибки и попытаться их исправить. Самым простым способом является просто разбиение

текста на слова и проверка на наличие их в словаре. Если такой словарь хорошо покрывает предметную область, из которой взят рукописный текст, то если данного слова нет в словаре, то можно с высокой вероятностью говорить о том, что была допущена ошибка в распознавании или в самом тексте. Также можно пытаться находить исправления для таких случаев. Для этого можно искать в исходном словаре слова достаточно близкие к нашему относительно какой-либо метрики. Наиболее популярной является редакционное расстояние Левенштейна. Также можно рассматривать и некоторые метрики Минковского.

В этап постобработки можно отнести и различные методы улучшения процесса распознавания на основе моделей языка. С помощью такой модели можно сразу же отсекаать маловероятные случаи в процессе распознавания. Очень хорошо такие языковые модели показывают себя вместе с марковской моделью письма, так как возможен так же ввод марковской модели языка, которая основана на *n-граммах*. Если же говорить про нейронные сети, работающие с СТС, то для них существует целый ряд методов, которые помогают корректно распознавать результаты работы сети с помощью языковой модели. Наиболее продвинутым алгоритмом в этом случае является Word Beam Search algorithm [9].

ГЛАВА 2

НЕЙРОННЫЕ СЕТИ

По вышеописанным причинам основным компонентом разрабатываемой системы должны стать нейронные сети. Поэтому в этой главе я попытаюсь раскрыть основу теории искусственных нейронных сетей.

2.1. Полносвязные нейронные сети

Нейронная сеть – это математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологические нейронных сетей [14]. В самом простом случае нейронная сеть представляет собой систему соединённых и взаимодействующих между собой искусственных нейронов (Рисунок 2.1). Искусственные нейроны же представляют собой упрощенную модель биологического нейрона.

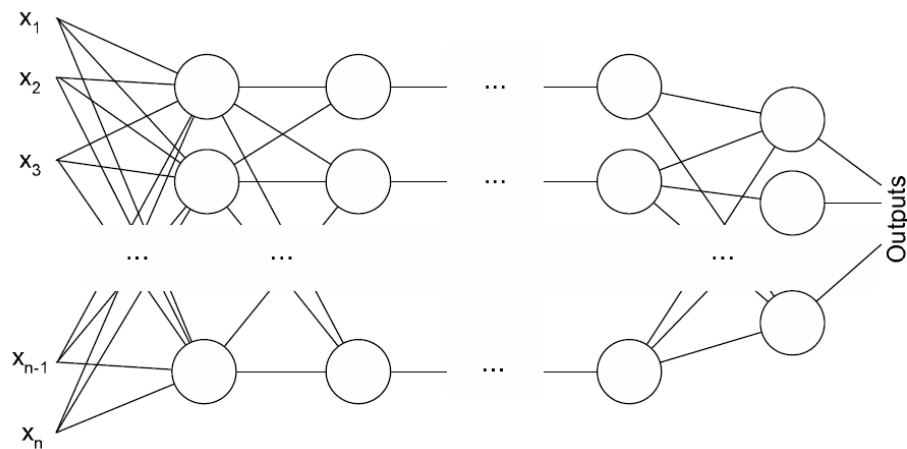


Рисунок 2.1 Организация простейшей полносвязной нейронной сети

2.1.2. Искусственный нейрон

Искусственная модель нейрона устроена аналогично модели своего «живого» собрата: входные сигналы от одних нейронов поступают на входы других с каким-либо весом и после, в зависимости от силы взвешенного входного сигнала, нейрон генерирует единственный выходной сигнал, поступающий на входы следующих нейронов.

Математически это можно формализовать следующим образом:

$$y = f(u), \quad u = \sum_{i=1}^N w_i x_i + b$$

Здесь x_i и w_i – входные сигналы от других нейронов и их веса соответственно. А b – некоторый сигнал, называемый сдвигом или *bias*, который можно рассматривать как постоянный сигнал присутствующий в данном нейроне. Данный параметр в нейроне присутствует не всегда, но важен, когда нам нужна полнота пространства линейных функций, которые мы можем представлять с помощью u . И последним, но не по значимости, параметром является функция активации или передаточная функция, которая отображает полученную сумму в итоговый сигнал нашего нейрона. Принцип работы нейрона отображён на рисунке ниже (Рисунок 2.2):

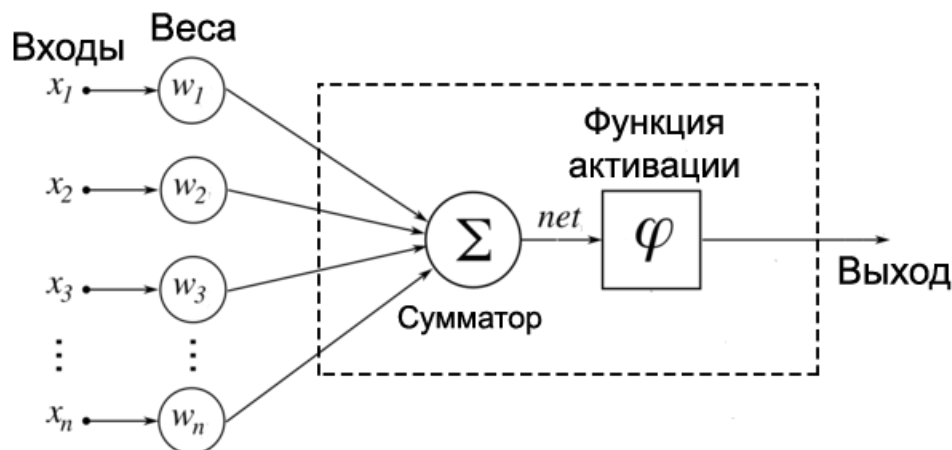


Рисунок 2.2 Принцип работы нейрона

2.1.3. Функции активации

Как было описано ранее, перед передачей сигнала другим нейронам применяется некоторая функция активации. Фактически она является способом нормализации данных перед дальнейшей отправкой сигнала. Строго говоря, на функции активации не накладывается никаких ограничений, и она может быть произвольной вещественной функцией. Но на самом деле выбор функции активации очень сильно влияет на свойства нашего нейрона, поэтому желательно, чтобы она обладала какими-то полезными для нас свойствами. Обычно таковыми являются:

- Нелинейность, благодаря которой наша нейронная сеть и может решать сложные задачи. Доказывается даже, что благодаря этому свойству функции активации двухуровневая нейронная сеть может сколь угодно близко аппроксимировать любую непрерывную ограниченную функцию [2].
- Непрерывная дифференцируемость, которая позволяет нам решать задачу оптимизации с помощью градиентного спуска.

- Монотонность, которая обуславливает выпуклость оптимизационной задачи.

Можно определить и ещё больше желательных свойств, но они меняются от задачи к задаче, а эти являются основными.

Наиболее часто применяющимися функциями активации являются:

- Полуполинейный элемент (англ. *Rectified linear unit, ReLU*)
- Линейная функция
- Гиперболический тангенс
- Сигмоидная функция

Их графический вид и уравнения можно видеть на рисунке (Рисунок 2.3):

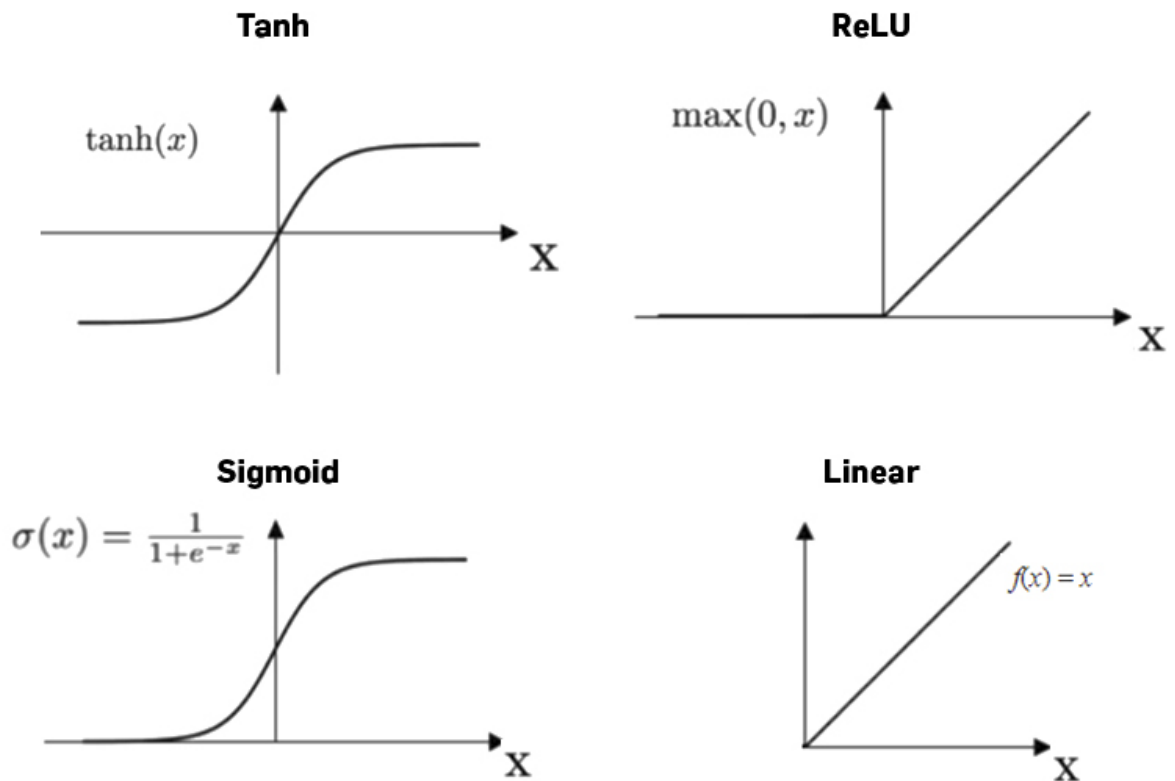


Рисунок 2.3 Функции активации. *Tanh* – гиперболический тангенс. *ReLU* – полуполинейный элемент. *Sigmoid* – сигмоидная функция. *Linear* – линейная функция.

Стоит также заметить, что наиболее перспективной в контексте глубоких нейронных сетей является функция ReLU. Это обусловлено простотой вычисления самой функции и её производной, а также меньшим затуханием градиента при

обратном распространении ошибки, что является очень важным фактором при построении глубоких нейронных сетей.

2.1.4. Обучение нейронной сети

Наиболее распространённым методом обучения нейронной сети является обучение с учителем. При таком методе мы работаем с уже размеченными данными, то есть для каждого обучающего прецедента у нас уже есть правильный ответ, который мы и хотим получить от нашей нейронной сети. Для подсчёта отклонения между правильными ответами и выходом нейронной сети вводится специальная функция – функция потерь (loss function). После этого задача сводится к тому, чтобы минимизировать функцию потерь на векторе весов нейронной сети. Чаще всего для этого применяется метод обратного распространения ошибки, который в свою очередь основан на градиентном спуске.

При обучении важными моментами являются выбор функции потерь и метода, которым мы будем производить оптимизацию. Функция потерь обычно выбирается исходя из нужд решаемой задачи. Для задачи регрессии чаще всего применяется средний квадрат отклонения, а для задачи классификации – перекрестная энтропия. Метод оптимизации обычно выбирается эмпирическим путём, поскольку мы обычно не обладаем какими-то априорными сведениями об целевой функции. Здесь можно отметить, что наиболее часто применяем методом является метод адаптивной оценки момента (Adam).

Важной также является борьба с переобучением. Переобучение, переподгонка (overtraining, overfitting) — нежелательное явление, возникающее при решении задач обучения по прецедентам, когда вероятность ошибки обученного алгоритма на объектах тестовой выборки оказывается существенно выше, чем средняя ошибка на обучающей выборке. Переобучение возникает при использовании избыточно сложных моделей [15]. Для борьбы с данным явлением применяется регуляризация, которая не даёт весам сети слишком сильно расти или каким-либо образом упрощает модель во время обучения (Dropout).

2.2. Свёрточные нейронные сети

Описанные выше полносвязные сети находят своё применения во многих практических задачах. Однако их применения в задачах распознавания образов являются очень ограниченными. Если попробовать целиком построить архитектуру на таких сетях при решении задач распознавания, то можно быстро столкнуться либо с вычислительными проблемами из-за огромного количества параметров нейронной сети, либо с недостаточной сложностью модели для целевой функции.

Поэтому при работе с изображениями чаще применяются свёрточные нейронные сети (Convolutional neural networks).

2.2.1. Свёрточные слои

Ключевой частью свёрточной нейронной сети являются свёрточные слои. Свёрточный слой состоит из некоторого набора карт, а каждая из карт в свою очередь обладает своим ядром, которое иногда также называют фильтром. Обработка выхода предыдущего слоя происходит следующим образом: ядро свертки фильтра проходит по всем фрагментам предыдущего слоя и для каждого такого фрагмента производится операция свёртки:

$$(I * K)[m, n] = \sum_{k, l} I[m - k, n - l] * K[k, l]$$

То есть просто вычисляется поэлементное произведение. В результате мы получаем матрицу результата (Рисунок 2.4).

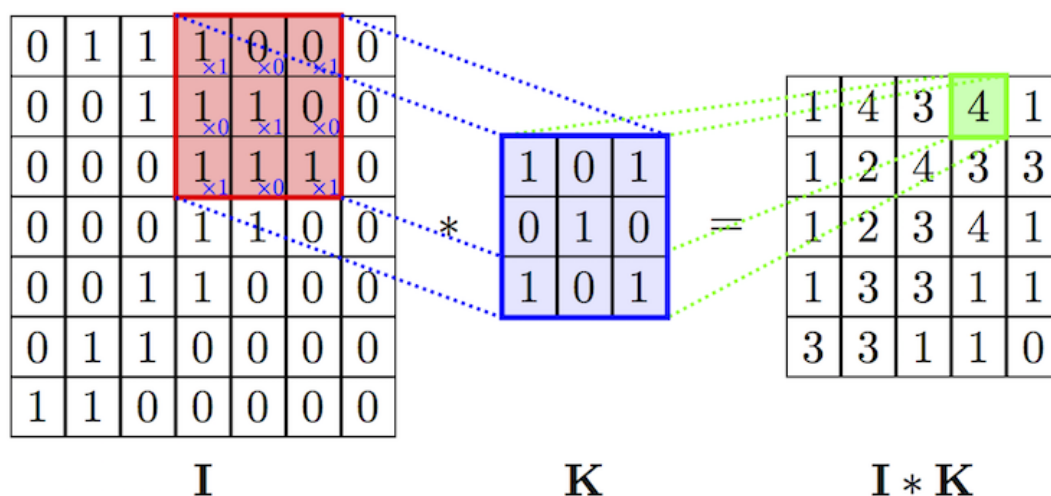


Рисунок 2.4 Демонстрация операции свертки

При этом ядро свёртки состоит из обучаемых параметров. Использование свёрточных слоев при распознавании изображений имеет множество плюсов. Можно отметить основные:

- Такая архитектура будет очевидным образом учитывать топологию входных данных
- Операция свёртки может быть устойчива к небольшим преобразованиям, что тоже является важным при распознавании изображений
- Свёрточные слои гораздо эффективнее полносвязных относительно вычислительной сложности

2.2.2. Слой субдискретизации

Слой субдискретизации (пулинга, подвыборки) аналогично свёрточному слою имеет карты, количество которых совпадает с числом карт предыдущего слоя. Он представляет собой некоторое уплотнение предыдущей карты признаков с целью уменьшения размерности. При такой операции группа пикселей уплотняется в один. Чаще всего в качестве нелинейного преобразования берется функция максимума, но можно встретить что-нибудь и более экзотическое, например, среднее значение. Если считать, что операция свертки помогала нам выделять признаки, то операция пулинга выбирает из этих признаков наиболее значимые, опуская остальные. Также стоит заметить, что наиболее часто в подвыборочном слое карты не пересекаются и имеют размерность 2×2 , что позволяет также уменьшить размерность в два раза. Пример работы такого слоя представлен на рисунке ниже (Рисунок 2.5).

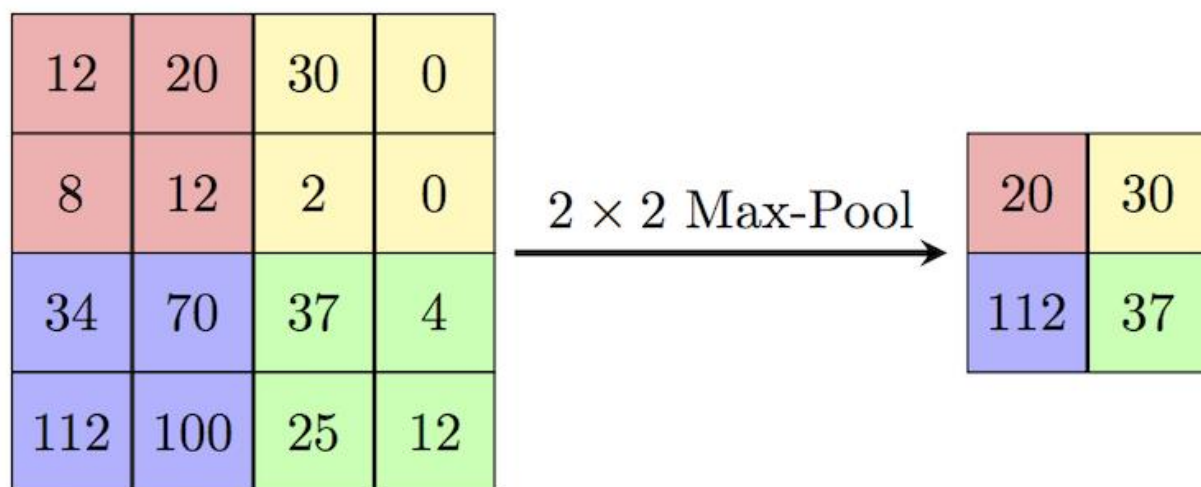


Рисунок 2.5 Исходная карта и результат работы подвыборочного слоя с функцией максимума.

Слой субдискретизации обычно следует за свёрточным слоем и такие комбинации чередуются, пока мы не достигнем нужной нам глубины сети.

ГЛАВА 3

РАЗРАБОТКА ПРОТОТИПА СИСТЕМЫ

3.1. Проектирование системы

Было решено реализовывать прототип системы на языке Python. Задача была декомпозирована в соответствии с этапами, описанными в первой главе. Система состоит из трех основных модулей:

- preprocessing – отвечает за предобработку входных данных
- model – отвечает за распознавание сегментированных изображений
- postprocessing – отвечает за постобработку полученных данных

3.1.1. Предобработка

Задача предобработки изображения решается при следующих допущениях:

- Текст содержит английские буквы и цифры
- Текст допускает сегментацию на отдельные буквы
- Строки в тексте являются параллельными и не пересекаются
- Фон отделим от текста

Пример текста, для которого соблюдены данные допущения можно увидеть на рисунке (Рисунок 3.1).

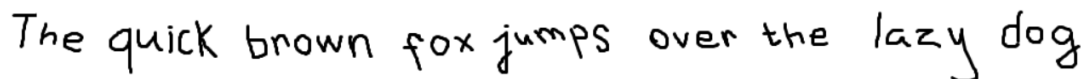


Рисунок 3.1 Пример корректного входного текста

При такой постановке задачи самым практически все методы сегментации будут хорошо работать, но самым эффективным будет метод связных компонент. С помощью него в результате предобработки должны выделяться отдельные буквы с сохранением их первоначальных координат.

Также на этапе предобработки должна решаться задача пороговой бинаризации. Для этого должна использоваться адаптивная гауссова бинаризация.

3.1.2. Распознавание

Распознавание отдельных букв должно производиться с помощью свёрточной нейронной сети. В результате распознавания каждому распознанному сегменту с предыдущего шага должен быть присвоен класс.

3.1.3. Постобработка

Так как мы работаем с отдельными буквами, то на этапе постобработки помимо коррекции возможных ошибок должно происходить разделение текста на слова. Для этого логичнее всего применять факторы из геометрического расположения, а также естественность полученных слов.

Задача коррекции же должна решаться на основании минимизации редакционного расстояния Левенштейна с регулируемым порогом.

3.1.4. Вход и выход системы

На вход подается изображение, соответствующее условиям предобработки, а на выход – распознанный текст. Также для более удобного взаимодействия с системой должен быть реализован графический интерфейс с возможностью ввода.

3.2. Реализация

В соответствии с проектированием было разработано приложение на языке Python.

3.2.1. Предобработка

Предобработка производилась с использованием OpenCV – библиотеки компьютерного зрения с открытым исходным кодом. В данной библиотеке реализованы все необходимые для реализации процесса предобработки методы.

Пороговая бинаризация проводилась с помощью с помощью метода Оцу. Перед этим на изображении также накладывается размытие с гауссовым ядром. Результат такой бинаризации можно видеть на рисунке (Рисунок 3.2).

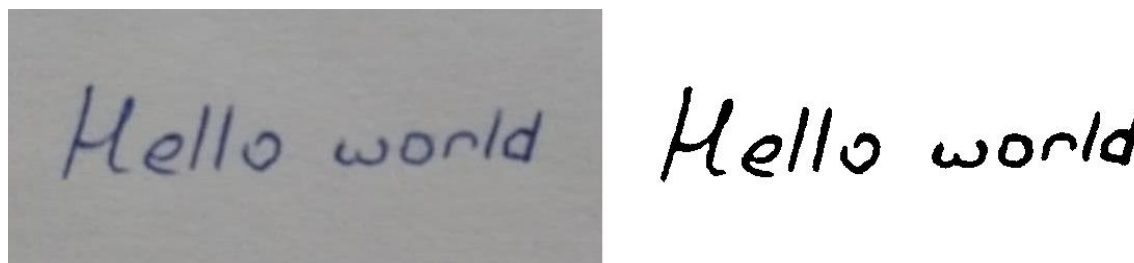


Рисунок 3.2 Результат пороговой бинаризации

Для возможности корректной сегментации букв, имеющих отдельное написание, перед применением метода связных компонент буквы происходит расширение всех черных пикселей с помощью морфологической операции эрозии, а также избавление от различных мелких дефектов с помощью морфологической операции замыкания. Результат сегментации корректного входного изображения представлен на рисунке (Рисунок 3.3). Код с реализацией сегментации находится в приложении Б.

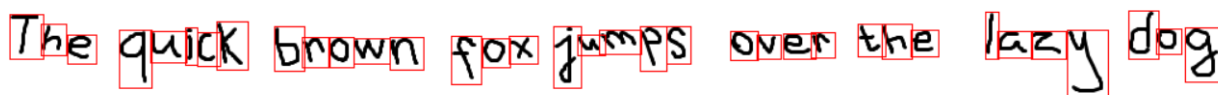


Рисунок 3.3 Пример сегментации букв

На этапе предобработки помимо этого производилось преобразование сегментированных букв и цифр до необходимого размера, который диктовался возможным входом нейронной сети.

3.2.2. Распознавание

Центральной частью системы стала реализации нейронной сети, которая была реализована с помощью открытой библиотеки Keras (Tensorflow). В качестве архитектуры была выбрана модификация свёрточной сети Lenet-5. Это было мотивировано тем, что задача классификации букв с цифрами в целом практически не отличается от задачи классификации цифр. Основная архитектура сети, которая была использована при распознавании представлена на рисунке (Рисунок 3.4).

Обучение нейронной сети происходило на наборе данных EMNIST [1], который представляет из себя расширенную версию набора данных MNIST, состоящего только из цифр. Всего в наборе данных порядка 800 тысяч размеченных изображений. При обучении выборка разбивалась на обучающую и тестовую в соотношении 7:1. В качестве функции потерь была выбрана перекрёстная энтропия, а за основную метрику качества была взята точность. Также при работе отслеживалась и метрика точности попадания в первые два из наиболее вероятных ответов сети (Top K categorical accuracy). Такая метрика качества для данной задачи может быть достаточно показательной, так как в некоторые буквы и цифры очень похожи друг на друга и сети трудно их различать.

Для предотвращения переобучения в сети использовались слои Dropout и аугментация обучающего набора данных посредством различных аффинных преобразований. В целом процедура аугментации в данном контексте носит не совсем обязательный характер, так как набор данных и так достаточно большой и вариативный. Также в Keras поддерживается удобная система callback'ов, которая позволяет использовать раннюю остановку, если результат перестают улучшаться.

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 28, 28, 32)	320
conv2d_7 (Conv2D)	(None, 26, 26, 64)	18496
batch_normalization_3 (Batch Normalization)	(None, 26, 26, 64)	256
max_pooling2d_3 (MaxPooling2D)	(None, 13, 13, 64)	0
dropout_4 (Dropout)	(None, 13, 13, 64)	0
conv2d_8 (Conv2D)	(None, 13, 13, 128)	73856
conv2d_9 (Conv2D)	(None, 13, 13, 128)	147584
batch_normalization_4 (Batch Normalization)	(None, 13, 13, 128)	512
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout_5 (Dropout)	(None, 6, 6, 128)	0
conv2d_10 (Conv2D)	(None, 6, 6, 256)	295168
conv2d_11 (Conv2D)	(None, 6, 6, 512)	1180160
batch_normalization_5 (Batch Normalization)	(None, 6, 6, 512)	2048
max_pooling2d_5 (MaxPooling2D)	(None, 3, 3, 512)	0
dropout_6 (Dropout)	(None, 3, 3, 512)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_2 (Dense)	(None, 256)	1179904
dropout_7 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 62)	15934
Total params: 2,914,238		
Trainable params: 2,912,830		

Рисунок 3.4 Архитектура сети

В результате обучения были достигаемые результаты точности лежали в интервале от 85 до 88%. А значение Top K categorical accuracy достигало 99.5%. Графики точности и значения функции потерь представлены на рисунках ниже (Рисунок 3.5 и Рисунок 3.6). Исходный код лучшей из моделей приведен в приложении А.

Для сравнения также была реализована полносвязная нейронная сеть. Точность на тестовой части выборки для такой сети достигала только лишь 69%, что еще раз доказывает слабую применимость таких сетей для решения такого класса задач.

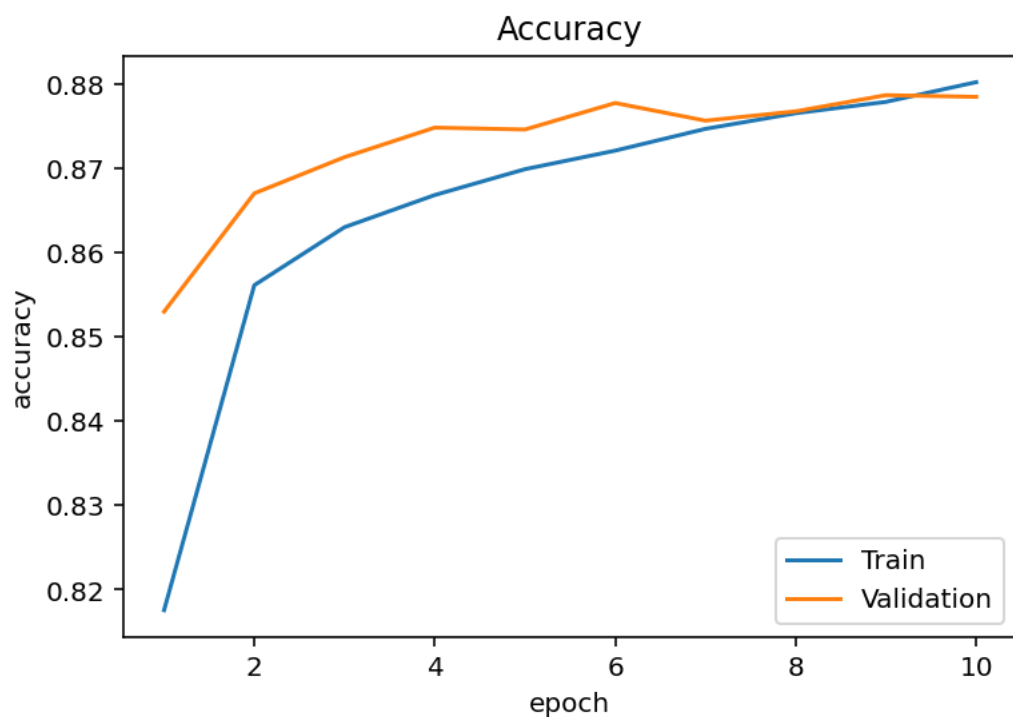


Рисунок 3.5 Зависимость точности на тестовой выборке от количества эпох обучения

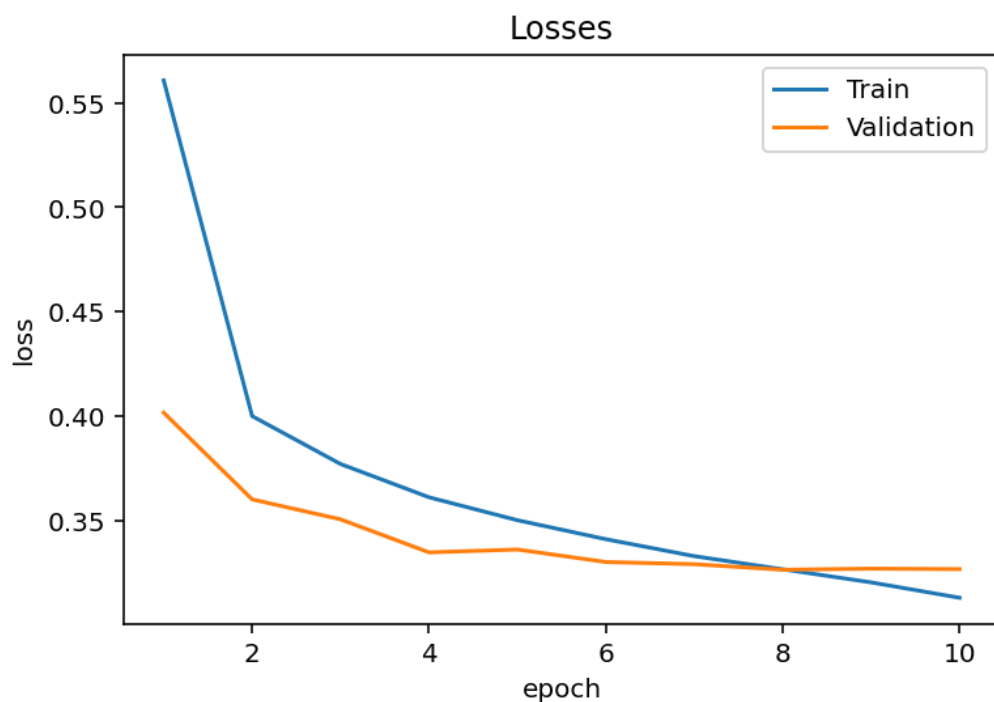


Рисунок 3.6 Зависимость потерь на тестовой выборке от количества эпох обучения

С учётом последующей постобработки результаты получились хорошими. По матрице ошибок можно было увидеть, что сеть ошибалась только на действительно

сложных прецедентах. К сожалению, из-за большого количества классов здесь она не приводится.

3.2.3. Постобработка

В реализованной системе сложно было бы обойтись без постобработки в виде реализации словаря с допустимыми словами. Достаточно часто обученная сеть путала схожие буквы и цифры: “0” и “о”, “1” и “l” и так далее. Для реализации поиска по словарю и исправления ошибок была взята открытая библиотека FuzzySet с реализацией одноименной структуры данных. При поиске в ней слова разбиваются на граммы и ищутся ближайшие слова относительно косинусного коэффициента. Для лучших результатов вычисляется редакционное расстояние Левенштейна и на основании этого выдается результат.

На этапе постобработки также строятся отдельные слова. Разбиение основывается на расстоянии между двумя соседними буквами. Считаем, что если оно больше некоторого значения, то буквы находятся в разных словах и нужно вставить разделитель. В качестве такого значения лучше всего подходила половина от средней ширины всех букв. Обычно такое эвристическое разбиение на слова не работало, только если специальным образом вводить текст.

3.2.4. Ввод текста

Для возможности ввода текста была разработана графическая часть приложения, которая позволит вводить текст в специальном окне и распознавать его. Интерфейс реализован с помощью библиотеки Tkinter, входящей в состав стандартной библиотеки языка Python (Рисунок 3.7).

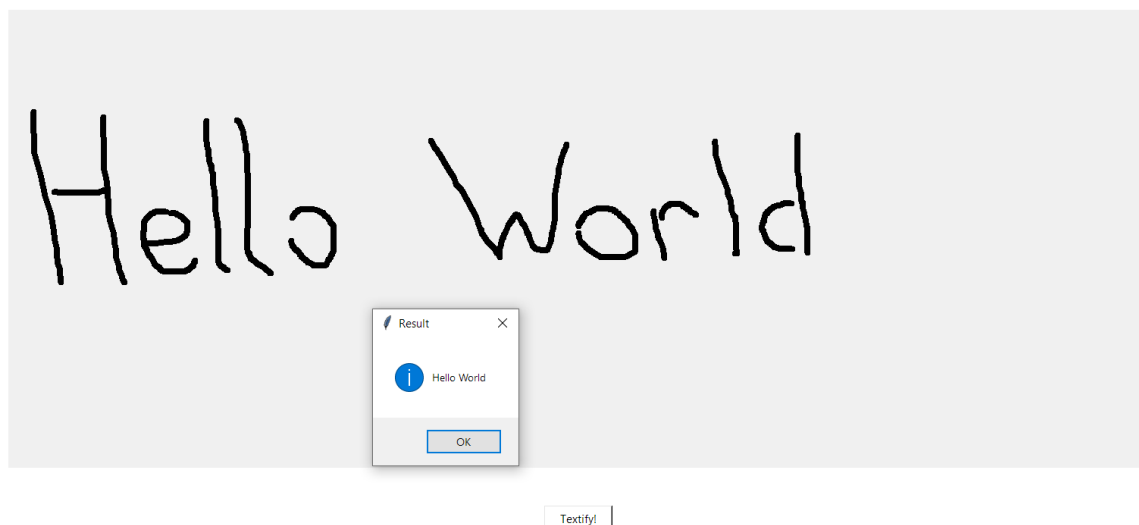


Рисунок 3.7 Пример ввода и вывода при использовании графического интерфейса

Помимо этого, прототип поддерживает работу в консоли, что позволяет подавать на вход изображения, полученные извне и при этом выбирать модель,

3.3. Оценка работы системы

Разработанный прототип системы распознавания текста отлично проявляет себя на данных, для которые соблюдены принятые ранее допущение. Ошибки допускаются только при явных нарушениях и использовании неизвестных для словаря слов. Примеры распознавания изображений представлены ниже (Рисунок 3.8, Рисунок 3.9).

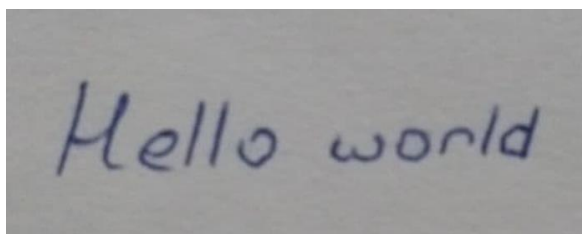


Рисунок 3.8 Распознавание снимка рукописного текста. Распознано как «Hello World»

The quick brown fox jumps over the lazy dog 387

Рисунок 3.9 Распознавание изображения полученного с использование графического интерфейса приложения. Распознано как «The quick brown Fox jumps over the lazy dog 387»

3.3.2. Недостатки

В работе прототипа системы можно выделить следующие недостатки:

- Условия, при которых решается задача слишком сильные, вряд ли в реальном тексте получится сегментировать буквы по отдельности
- В прототипе никак не обрабатываются крупные дефекты изображений: большие кляксы, точки. Они становятся помехой при работе программы, так как от них сложно избавиться только с помощью комбинации наложения гауссова фильтра и нахождения связных компонент
- Не используется вся информация, получаемая с выходов нейронной сети. Так как выходы имеют вероятностный смысл, то логично использовать не только самый вероятный случай, но и следующий за ним. Не делается это из-за того, что сложность такого алгоритма росла бы экспоненциально от количества букв, поэтому опять-таки лучше рассматривать методы распознавания, которые работают с целыми словами или хотя бы с последовательностью.

3.3.3. Выводы

Исходя из описанных выше недостатков, можно сделать вывод, что метод распознавания в системе лучше всего заменить на метод, который работает сразу с последовательностью символов. В качестве такового можно рассматривать рекуррентную нейронную сеть с CTC-loss. В целом можно заменить только метод распознавания, благодаря тому, что задача легко декомпозируется на достаточно независимые части. То есть возможно оставить этапы предобработки и

постобработки практически такими же, изменив в них только части, связанные с распознаванием.

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **Cohen Gregory [et al.]** EMNIST: an extension of MNIST to handwritten letters [Journal]. - 2 17, 2017.
2. **Cybenko G.** Approximation by superpositions of a sigmoidal function [Journal] // Mathematics of Control, Signals and Systems. - 1989. - Vol. 2. - pp. 303-314.
3. **Graves Alex [et al.]** Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks [Conference] // Proceedings of the 23rd International Conference on Machine Learning. - New York, NY, USA : Association for Computing Machinery, 2006. - pp. 369–376. - ISBN: 1595933832.
4. **Lecun Yann** [В Интернете] // MNIST. - 12 04 2020 г.. - <http://yann.lecun.com/exdb/mnist/>.
5. **Li Y. [et al.]** Script-Independent Text Line Segmentation in Freestyle Handwritten Documents [Journal] // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 2008. - Vol. 30. - pp. 1313-1329.
6. **Otsu N.** A Threshold Selection Method from Gray-Level Histograms [Journal] // IEEE Transactions on Systems, Man, and Cybernetics. - 1979. - Vol. 9. - pp. 62-66.
7. **Plamondon R. and Srihari S. N.** Online and off-line handwriting recognition: a comprehensive survey [Journal] // IEEE Transactions on Pattern Analysis and Machine Intelligence. - [s.l.] : IEEE, 2000. - 1 : Vol. 22. - pp. 63–84. - ISSN: 1939-3539.
8. **Ptak Roman, Żygadło Bartosz and Unold Olgierd** Projection–Based Text Line Segmentation with a Variable Threshold [Journal] // International Journal of Applied Mathematics and Computer Science. - 3 2017. - Vol. 27.
9. **Scheidl H., Fiel S. and Sablatnig R.** Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm [Conference] // 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). - Los Alamitos, CA, USA : IEEE Computer Society, 2018. - pp. 253-258.
10. **Schmidhuber Jürgen** Deep learning in neural networks: An overview [Journal] // Neural Networks. - [s.l.] : Elsevier BV, 1 2015. - Vol. 61. - pp. 85–117. - ISSN: 0893-6080.

11. **Szegedy Christian [et al.]** Going Deeper with Convolutions // Going Deeper with Convolutions. - 2014.
12. **Vinciarelli Alessandro and Perrone Michael** Combining Online and Offline Handwriting Recognition. [Conference]. - 2003. - pp. 844-848.
13. **Воронцов К. В.** Комбинаторная теория надёжности обучения по прецедентам: Дис. док. физ.-мат. наук: 05-13-17 [Report] : Ph.D. dissertation / Вычислительный центр РАН. - 2010. - p. 271.
14. **Осипов Ю. С.** Большая российская энциклопедия [Книга]. - [б.м.] : Рос. акад. наук. - Т. 2 : 35.
15. Переобучение [Online] // machinelearning.ru. - 04 22, 2020. - <http://www.machinelearning.ru/wiki/index.php?title=%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5&oldid=14284>.

ПРИЛОЖЕНИЯ

Приложение А

```
def get_model (hp):
    model = Sequential()
    model.add(Convolution2D(filters=32, kernel_size=(3, 3),
                            padding='same',
                            input_shape=(HEIGHT, WIDTH, 1),
                            activation='relu'))
    model.add(Convolution2D(filters=64, kernel_size=(3, 3),
                            activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Convolution2D(filters=128, kernel_size=(3, 3),
                            padding='same', activation='relu'))
    model.add(Convolution2D(filters=128, kernel_size=(3, 3),
                            padding='same', activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Convolution2D(filters=256, kernel_size=(3, 3),
                            padding='same', activation='relu'))
    model.add(Convolution2D(filters=512, kernel_size=(3, 3),
                            padding='same', activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(256, activation='relu'))

    if hp:
        model.add(Dropout(hp.Float('dropout', 0, 0.5, step=0.1,
                                   default=0.5)))
    else:
        model.add(Dropout(0.5))
    model.add(Dense(CLASSES, activation='softmax'))

    metrics = [keras.metrics.TopKCategoricalAccuracy(k=3),
               keras.metrics.CategoricalAccuracy()]
    if hp:
        optimizer = optimizer=keras.optimizers.Adam(
            hp.Float('learning_rate', 1e-4, 1e-2, sampling='log'))
```

```

else:
    optimizer = optimizer=keras.optimizers.Adam(learning_rate=3e-3)

model.compile(loss='categorical_crossentropy',
              metrics=metrics, optimizer=optimizer)
return model

```

Приложение Б

```

def extract_letters(self, height, width, eroding=4,
save_artifacts=False) -> List[Any]:
    opening = cv2.morphologyEx(self.thresh, cv2.MORPH_OPEN,
cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5,5)))
    img_erode = cv2.erode(opening, np.ones((eroding, eroding),
np.uint8), iterations=3)

    contours, hierarchy = cv2.findContours(img_erode, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

    output = None
    if save_artifacts:
        output = self.image.copy()

    letters = []
    for idx, contour in enumerate(contours):
        # checking for parent contour
        if hierarchy[0][idx][3] == 0:
            (x, y, w, h) = cv2.boundingRect(contour)
            if save_artifacts:
                cv2.rectangle(output, (x, y), (x + w, y + h), (0, 0,
255), 1)

            mask = np.zeros_like(img_erode)
            cv2.drawContours(mask, [contour], 0, 255, -1)

            out = 255 * np.ones_like(img_erode)
            out[mask == 255] = self.thresh[mask == 255]

            letter_crop = out[y:y + h, x:x + w]
            size_max = max(w, h)

```