|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Теоретическая информатика и компьютерные технологии

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

**ПО КУРСУ АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ГРАФИКИ**

**НА ТЕМУ:**

Распознавание рукописного текста

Студент

*подпись, дата фамилия, и.о.*

Научный руководитель

*подпись, дата фамилия, и.о.*

*2021*

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc96614203)

[1. Обзор нейросетевых моделей 6](#_Toc96614204)

[1.1 Основные понятия 6](#_Toc96614205)

[1.1.1 Сверточная нейронная сеть 6](#_Toc96614206)

[1.1.2 Рекуррентная нейронная сеть 8](#_Toc96614207)

[1.1.3 Временная классификация CTC 10](#_Toc96614208)

[2. Разработка критериев оценки модели, подготовка набора данных 16](#_Toc96614209)

[2.1 Критерии оценки 16](#_Toc96614210)

[2. 2 Набор данных 16](#_Toc96614211)

[2. 3 Генерация синтетических данных 18](#_Toc96614212)

[3. Реализация модулей 21](#_Toc96614213)

[3.1 Реализация нейросетевой модели 21](#_Toc96614214)

[3. 2 Предобработка изображений 22](#_Toc96614215)

[3. 3 Реализация цикла обучения 24](#_Toc96614216)

[4. Тестирование 26](#_Toc96614217)

[4. 1 Оценка качества модели на тестовых данных 26](#_Toc96614218)

[4. 2 Оценка качества модели на реальных данных 28](#_Toc96614219)

[ЗАКЛЮЧЕНИЕ 29](#_Toc96614220)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc96614221)

# ВВЕДЕНИЕ

С каждым днем все больше достижений в области информатики и, особенно, в машинном обучении и анализе изображений находят свое применение в реальной жизни. Компьютерное зрение, зарекомендовав себя как один из самых эффективных способов анализа видео и изображений, давно интегрировано во множество областей, облегчая или заменяя работу человека. Распознавание текста – одна из ключевых задач компьютерного зрения. Это способность компьютера получать и интерпретировать записи из таких источников, как бумажные документы, фотографии, сенсорные экраны и другие устройства.

Данную задачу можно отнести к разделу распознавания образов. В последние десятилетия, благодаря использованию современных достижений компьютерных технологий, были разработаны новые методы обработки изображений и распознавания образов, благодаря чему стало возможным создание удачных промышленных систем распознавания рукописного текста, как, например, Tesseract by Google или PDFelement Pro by Iskysoft. Тем не менее, создание приложения в данной области по-прежнему остается творческой задачей и требует дополнительных исследований в связи со специфическими требованиями по разрешению, быстродействию, надежности распознавания и объему памяти, которыми характеризуется каждая конкретная задача. Проблема действительно актуальна, ведь оцифрованный вид рукописных записей позволил бы автоматизировать бизнес-процессы множества компаний в самых различных сферах.

Цель компьютерного зрения заключается в формировании полезных выводов относительно объектов и сцен реального мира на основе анализа изображений, полученных с помощью датчиков.

Следует разделить распознавание рукописного текста на два класса: офлайн-распознавание с листа бумаги или другой поверхности с помощью оптического сканирования и онлайн - с помощью взаимодействия с поверхностью экрана компьютера или планшета.

При онлайн-распознавании текста слова и символы распознаются во время выполнения, и информация о процессе написания знаков хранится в буфере памяти. Самым распространенным способ онлайн-записи является использование пера и планшета. Используя знание о количестве, наборе и скорости штрихов, программа может эффективно и точно распознавать написанный пером текст.

При офлайн-распознавании обрабатывается статическое представление имеющих смысл записей на сцене. В виду вариативности символов и, зачастую, высокой степенью зашумления, данный тип распознавания требует более сильного и композитного алгоритма, с использованием машинного обучения и глубоких нейронных сетей.

На данный момент, во многих сферах жизни применяется оффлайн-распознавание рукописного текста. Например, огромные объемы исторических данных оцифровываются для доступа по всему миру, но несут малый практический интерес, пока текст на изображении не может быть распознан и сохранен на носитель памяти. Во многих странах регулярно выписывают банковские чеки, они по-прежнему играют важную роль в безналичных транзакциях и процесс обработки может быть ускорен в разы с применением методов компьютерного зрения и машинного обучения. Определение автомобильных номеров и интерпретация типизированных документов с помощью алгоритмов искусственного интеллекта стало самым успешным применением технологий распознавания текста на сегодняшний день.

Существуют решения, позволяющие распознавать печатный текст на типизированных документах с помощью аналитических методов без применения нейросетевых технологий, например, с помощью эталонных изображений символов. Однако, данные модели нельзя назвать универсальными, ведь каждая из них подходит лишь под определенный тип документа и шрифта.

Задача офлайн-распознавания рукописного текста является более трудоемкой и нетривиальной задачей по сравнению с печатным по ряду причин:

1. Сильная вариативность почерка от человека к человеку,
2. Непостоянство почерка отдельного человека от времени,
3. Неоднозначность интерпретации штрихов
4. Зашумленные и неоднородные данные
5. Переменный наклон символов
6. Возможное негоризонтальное расположение строк
7. Трудноотделимые символы и строки
8. Отсутствие эталонного шаблона

Первоначальные подходы к распознаванию рукописного текста включали

методы машинного обучения, такие как скрытые марковские модели и метод опорных векторов. После процесса снижения размерности, в котором исходный набор исходных данных сокращается до более управляемых групп для дальнейшей обработки, такие как количество замкнутых участков, прямых или косых линий и т. д. признаки передаются на вход классификатору для получения результата. Такой подход к выделению признаков недостаточно устойчиво справляется с проблемами, приведенными выше.

Недавние достижения в области глубокого обучения, такие как появление архитектур с использованием сверточных, рекурретных слоев, механизмов внимания [1], трансформеров [1] ускорили наш прогресс в распознавании рукописного текста.

Целью данной курсовой работы является разработка нейросетевой модели, способной распознавать рукописный текст на изображении, цикла обучения и тестирования.

# Обзор нейросетевых моделей

## 1.1 Основные понятия

Основная проблема при обработке визуальных данных заключается в том, что каждое изображение представлено в виде двумерной или трехмерной матрицы, а не один одномерный вектор, который нам нужен для обучения типичных нейронных сетей. Всегда можно преобразовать изображение в формат 1D, «распрямив» его. Это означает, что мы держим каждый пиксель рядом друг с другом, чтобы сформировать единый вектор, который принимает на вход нейронная сеть. Но делая это, теряется много пространственной информации, присутствующей на изображении. Следовательно, нейронная сеть не будет фокусироваться на поиске закономерностей между соседними пикселями в изображении. С этими задачами справляются сверточные нейронные сети.

### 1.1.1 Сверточная нейронная сеть

CNN – сверточная нейронная сеть. Свертка — основная операция сверточной нейронной сети, позволяющая выделять признаки из изображения и задается формулой:

Где *K* – двумерное ядро свертки, I – изображение с одним цветовым каналом.



Рисунок 1 – фильтры сверточной нейронной сети, обученной на классификации изображений

Ядро проходит над изображением, поэлементно выполняя операцию свертки над покрываемым участком (рисунок 1). Таким образом, матрицы, полученные после операции свертки, называются картами признаков, потому что хранят в себе некие признаки предыдущих матриц, но уже в ином виде. Количество различных ядер сверток (фильтров) является параметром модели и настраивается в зависимости от сложности задачи.

Между сопоставлением карт признаков может быть произведена операции подвыборки, позволяющая уменьшить количество признаков, оставив самые значимые.

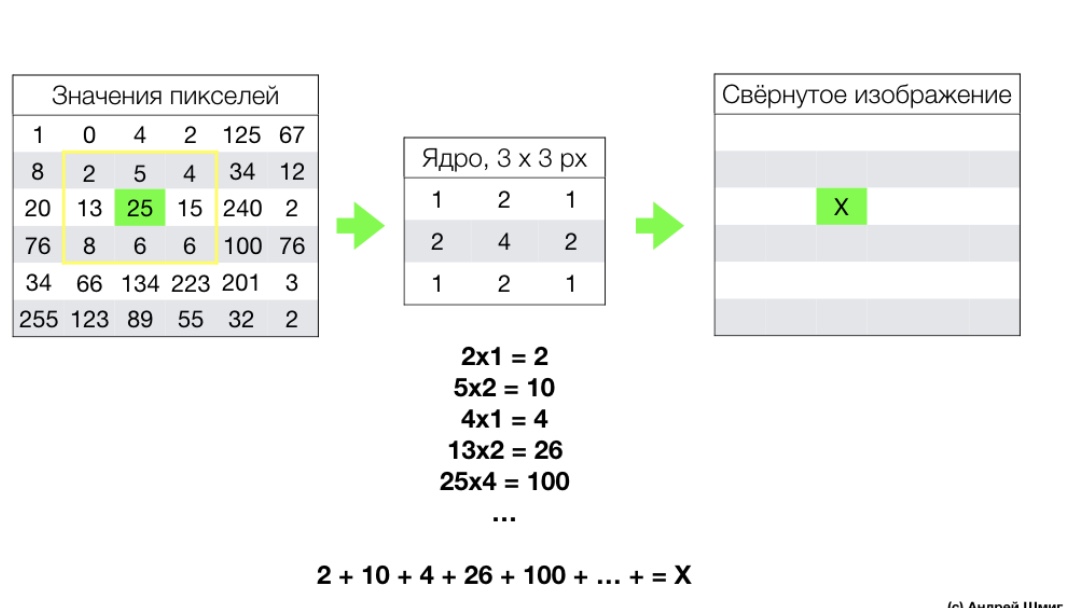


Рисунок 2 – Пример операции свертка

Существует несколько операций подвыборки, самые распространенные из них максимум и среднее. В ходе разработки выбор пал на функцию максимума как более часто используемую в решении задач подобного класса.



Рисунок 3 – Пример операций подвыборок

### 1.1.2 Рекуррентная нейронная сеть

RNN – рекуррентная нейронная сеть. Идея RNN заключается в последовательном использовании информации. В традиционных нейронных сетях подразумевается, что все входы и выходы независимы. Но для многих задач это не подходит. Например, если необходимо предсказать следующее слово в предложении, лучше учитывать предшествующие ему слова.

RNN называются рекуррентными, потому что они выполняют одну и ту же операцию для каждого элемента последовательности, причем выход зависит от предыдущих вычислений. Одна из интерпретаций RNN: это сети, у которых есть «память», учитывающая предшествующую информацию.

Теоретически RNN могут использовать информацию в произвольно длинных последовательностях, но на практике они ограничены лишь несколькими шагами.

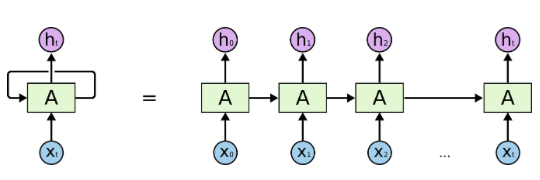
.

Рисунок 4 – архитектура классической рекуррентной сети

Рекуррентная формула:

Где *h* – вектор состояний, *x* – вектор входных данных, *f* – функция с параметрами *W*, *W* – матрица весов.

Для каждого момента времени (time step) используется одна функция и одна матрица весов. Обычно на месте *f* используются стандартные функции активации, такие как *tanh* или *sigmoid.* Для обучения рекуррентных нейросетей используется разворачивание во времени и обратное распространение ошибки.

Однако, при больших последовательностях возникает проблема затухающего градиента. При передаче от слоя к слою сигнал об изменении весов уменьшается. Поэтому на практике используются модификации RNN, использующие “ворота” памяти, такие как GRU (Gated recurrent unit) и LSTM (Long Short Term Memory) и их двунаправленные версии (временные шаги проходят через сеть в обратном порядке, как бы “подглядывая в будущее”).

Формула LSTM:

где – функция сигмоиды, *c* – состояние ячейки памяти, f – “ворота” забывания, g – “ворота” запоминания, *i, o* – вход и выход соответственно, \* - поэлементное умножение.

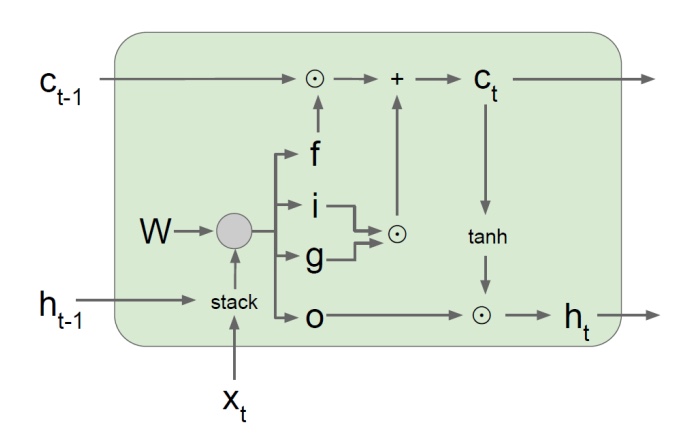


Рисунок 5 – схема ячейки LSTM

В таком виде рекуррентная сеть использует предыдущий контекст для вычисления выход. Однако, поскольку нейросети разрешается заглянуть в будущее, лучше использовать двунаправленную версию – BiLSTM [2]. Таким образом, модель будет учитывать как прошлый, так и предстоящий временной шаг.

Формула BiLSTM:

где верхний индекс *f* (forward) означает параметры скрытого слоя прямого прохождения, а *b* (backward) ­– обратного, y – результирующий вектор, составленная из выходов слоя прямого и обратного хождения.

### 1.1.3 Временная классификация CTC

Ключевая технология в распознавании последовательный данных, таких как распознавание речи и текста: CTC (Connectionist temporal classification)[3]

Определение границ символа отдельная и трудоемкая задача. Зачастую они имеют переменную ширину и вовсе трудноразделимы. Поэтому изображение текста следует воспринимать как непрерывную последовательность, что влечет за особой разбиение на временные шаги без явной сегментации. Но, принимая наиболее вероятный символ за временной шаг, можно получить текст вроде «HHHHHeeeellllllllloooo». Очевидная идея удаления повторов приводит к неправильному результату – Helo.

CTC решает эти проблемы: Сеть обучается с без указания начала и конца символа с помощью потери СТС (CTC loss), декодер СТС преобразовывает вывод нейросети в окончательный результат.

Как это достигается?

* Введем специальный символ СТС-пробел (blank), обозначаемый “~”, чтобы указать, что ни один из символов не найден на данный шаг времени
* Изменим исходный текст T на T`, вставляя CTC-пробелы и повторяя символы всеми возможными способами.
* Рассчитаем оценку для каждого возможного Т` и, просуммируем по всем оценкам для получения потери

Вычисление потери CTC могут оказаться очень дорогими по времени и памяти, если не оптимизировать их. Простой подход и вычисления значений для каждого пути, суммируя их все по мере продвижения, порождает огромное количество вариантов. Для большинства задач это было бы слишком медленно. Вычислить потери можно намного быстрее с помощью алгоритма динамического программирования. Ключевым моментом является то, что если два пути достигли одного и того же результата на одном и том же шаге, мы можем их объединить.

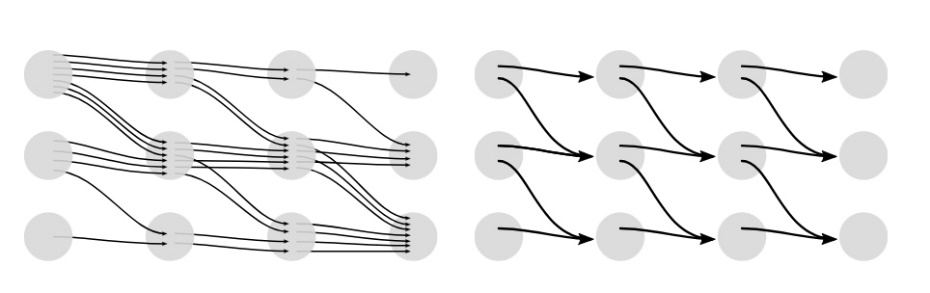


Рисунок 6 – граф вычислений CTC потери без оптимизации и с оптимизацией

Существует несколько способом расшифровать полученный результат. Самая тривиальная реализация – жадный алгоритм выбирает на каждом шаге выход с максимальной вероятностью.

Алгоритм лучевого поиска [4] https://arxiv.org/pdf/2110.05909.pdf] используется N (ширина луча) кандидатов и выдает результаты, основываясь на суммарной вероятности. Жадный алгоритм работает быстро и просто, но может дать сбой. Например, как показано на рисунке 6, жадный алгоритм выведет ‘~’, а алгоритм лучевого поиска – ‘a’, поскольку сложит вероятность путей ‘a~’, ‘~a’, ‘aa’, которые приведут к правильному результату. Имеется возможность использовать языковую модель на уровне символов вместе с лучевым поиском. То есть, при вычислении текущего результата, также оцениваются вероятности нахождения текущего символа и пар символов.

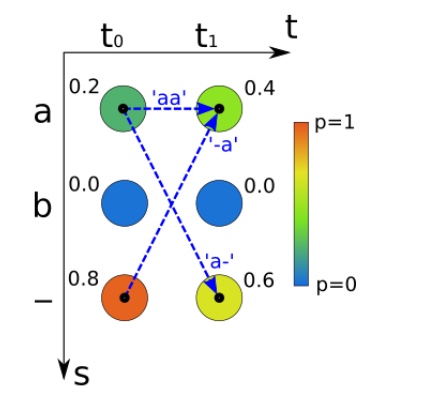


Рисунок 7 – пример нестабильный работы жадного алгоритма

Следующий алгоритм декодирования – лучевой поиск по словам (word beam search [5]). Распознаваемые слова ограничены словарем, при этом нет ограничений на цифры и знаки препинания и есть возможность дополнить языковой моделью на уровне слов.

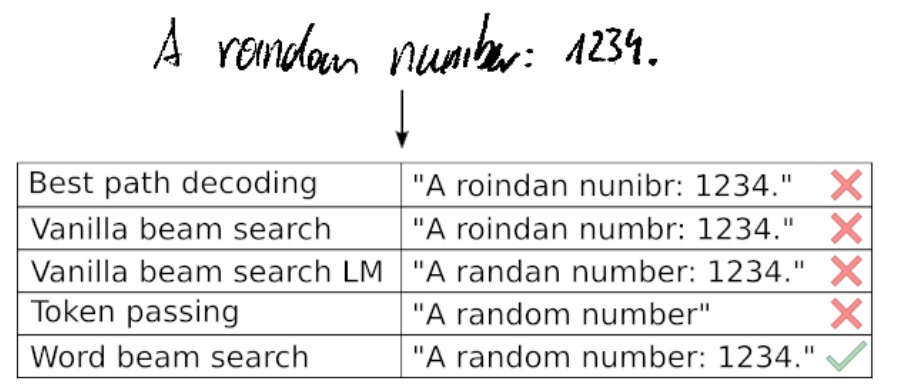


Рисунок 8 – пример применения различных эвристик

**1.2** **Обзор моделей, выбор модели**

Многомерные рекуррентные нейронные сети (MDLSTM).

RNN/LSTM справляются с данными, имеющими последовательный характер, но они ограничены размерностью входных данных и не могут обрабатывать 2D или 3D объекты как изображения с текстом. Для решения этой проблемы было предложено использовать архитектура с использованием многомерных рекурретных нейронных сетей [6]

Основная идея состоит в том, чтобы заменить единственное рекуррентное соединение, на столько соединений, сколько пространственно-временных измерений в данных.

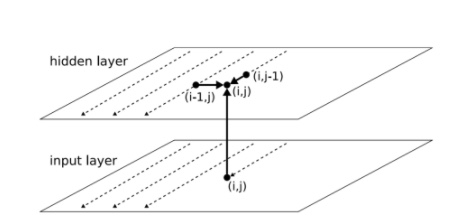


Рисунок 9 – связь между измерениями

Эти соединения позволяют создавать гибкое внутреннее представление окружающего контекста, которое устойчиво к локальным искажениям. Скрытый слой MDRNN сканирует входные данные в 1D полосах, сохраняя свои активации в буфере. Полосы упорядочены таким образом, чтобы в каждой точке слой уже посетил точки на один шаг назад по каждому измерению. Для обучения используется CTC-потеря. Однако, данная реализация модели распознавания рукописного текста имеет критических недостаток – отсутствие слоев подвыборки и проход всего размера изображения увеличивает количество весов модели в несколько десятков или сотен раз при таком же качестве, по сравнению с другими.

Модели-трансформеры (Transformer-OCR).

По аналогии с рекуррентными нейросетями, модели-трансформеры [7] предназначены для обработки последовательностей и последние несколько лет показывают высокие результаты в таких задачах как машинный перевод и автоматическое аннотирование [1] и на данный момент используются в Яндекс и Google переводчиках. Они также нашли применение и в распознавании образов на изображении. В отличие от RNN, трансформеры не требуют обработки данных последовательности по порядку благодаря механизму самовнимания [1], что позволяет производить эффективные параллельные вычисления. Сперва, изображения разбивается на блоки, которые в виде одномерного вектора с позиционной информацией поступают на вход кодировщику, состоящего из нескольких слоев с механизмом самовнимания.

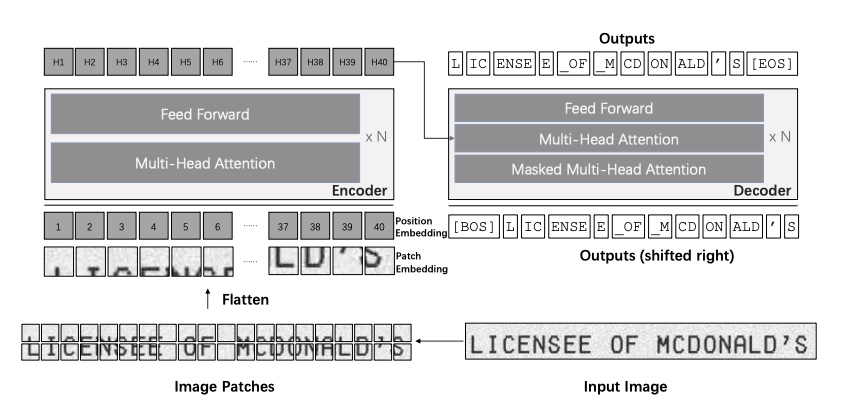


Рисунок 10 – процесс работы трансформера

Из преимуществ стоит отметить возможность встроить предобученную языковую модель на слои декодера. Главным отличием от существующих решений является замена CNN на кодировщика изображений. Однако, подобные модели сложны в реализации и требуют значительных вычислительных затрат.

Конволюционно-рекурретные модели (CRNN).

Данный тип модели является синтезом двух ключевых технологий – конволюционной и рекуррентной нейронной сети [8]. CNN выделяет признаки и понижает размерность, RNN использует выходную карту признаков как последовательность для обучения с помощью CTC-потери. Такой подход позволяет значительно сэкономить время работы и вес модели за счет более компактного представления данных в виде набора карт признаков.

Данная архитектура не позволяет встроить языковую модель в цикл обучения, однако может показывать лучшие результаты CTC-расшифровки с применением языковой модели.

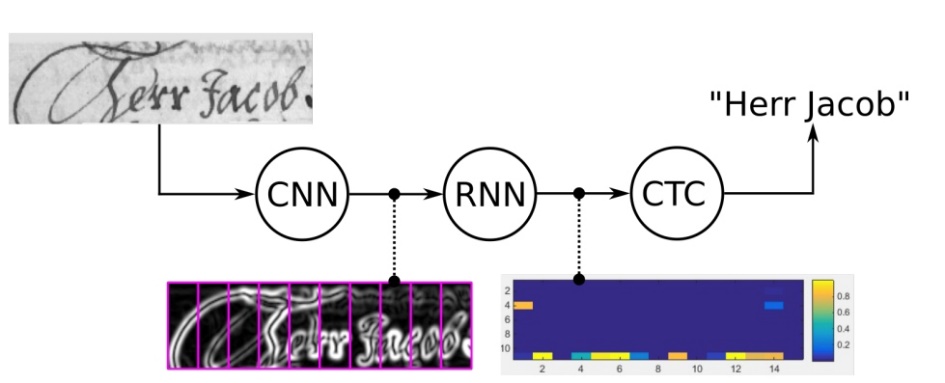


Рисунок 11 – процесс работы CRNN модели

CNN+BiLSTM – самая популярная архитектура для решение данный задачи, Компании мирового масштаба используют ее в своих сервисах, например, Tesseract by Google и OCR by Yandex.Cloud, Поэтому, выбор падает именно на данный тип архитектуры.

# 2. Разработка критериев оценки модели, подготовка набора данных

# 2.1 Критерии оценки

Необходимо ввести метрики качества, используемые для оценки эффективности модели. Имеет смысл производить оценку модели на разных уровнях: на уровне символов, слов и предложений.

Рассмотрим метрики CER, WER и SER (character, word and sentence error rate).

где I + S + D — минимальное количество операций замены соответствующего структурного элемента, необходимых для получения верного результата (расстояние Левенштейна),

I — количество вставок элементов,

S — количество замен элементов,

D — количество удалений элементов,

N — количество элементов.

CER, WER и SER можно интерпретировать как процент неверно угаданных символов, cлов и предложений соответственно. Имеет смысл считать метрики без постобработки результата с помощью сложных декодирующих эвристик, поэтому самой показательной метрикой в нашем случае будет CER, поскольку улучшить показатели WER и SER можно благодаря языковой модели и внешнего словаря.

## 2. 2 Набор данных

HKR [9]— закрытая база данных, содержащая фрагменты изображений русского и казахского рукописного текста в трехцветной формате с расширением jpg. Помимо 33 символов русского алфавита в наборе присутствуют 9 символов казахского алфавита. HKR представляет собой набор форм. Все формы были созданы с помощью LATEX и впоследствии были заполнены людьми. База данных состоит из более чем 1400 заполненных форм. Всего около 63000 предложений, более 715000 символов, написанных примерно 200 разными авторами.

Набор данных соответствует требованиям для решения поставленной задачи, поскольку имеет распределение по символам, примерно аналогичное словарю русского языка.

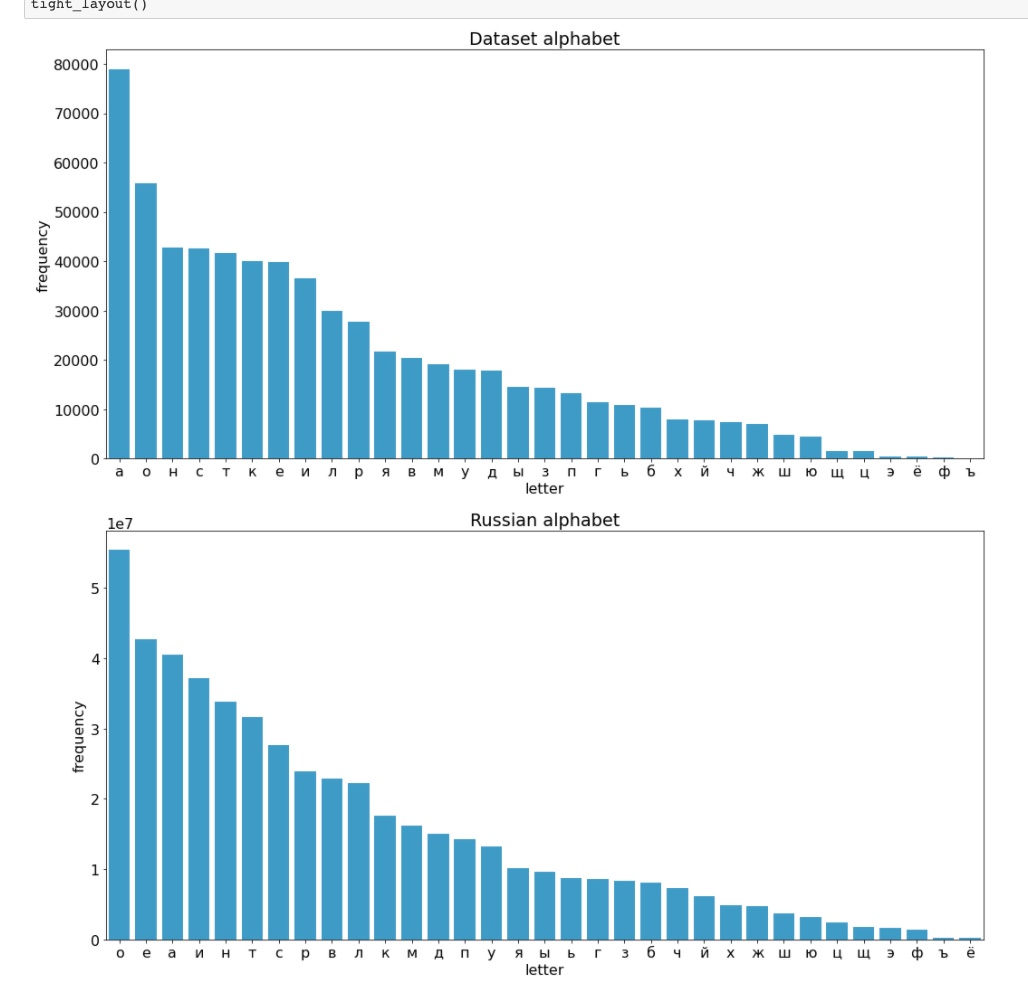


Рисунок 12 – распределения символов в наборе данных и русском алфавите

Ограничим задачу распознаванием только русского текста, поэтому удалим из набора данные, содержащие казахские символы – их 0.37%.

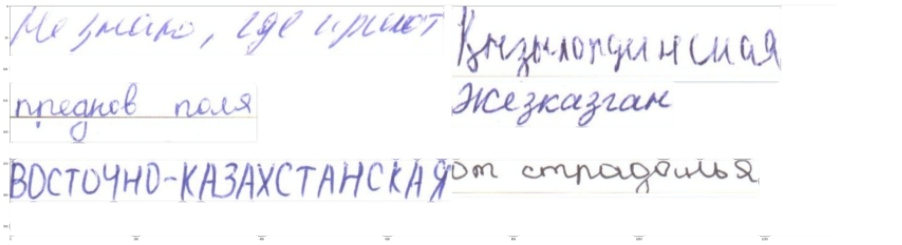


Рисунок 13 – пример изображений, составляющих набор данных

Рассмотрим краткую статистику по набору данных, изображенную на рисунке 14. Нейросеть принимает изображения фиксированного размера, оптимальным размером является 64 на 758 после изменения размера. Для компактности данных, объединим короткие изображения в единую строчку.



Рисунок 14 – краткая статистика по набору данных

В результате получим чуть более 48 тысяч изображений, составленных от 1 до 4 исходных, примеры изображены на рисунке 15.

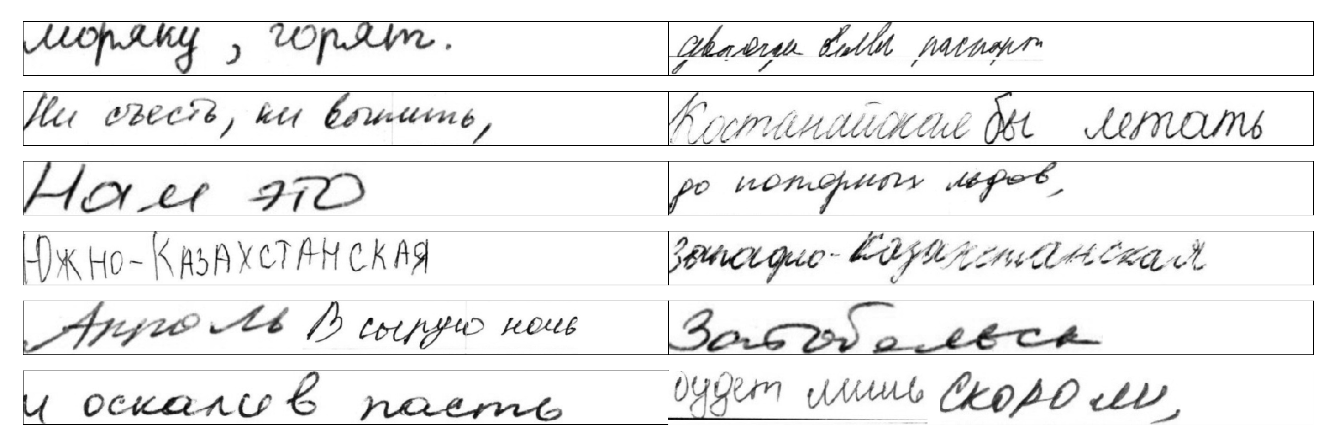


Рисунок 15 – результирующий набор данных

## 2. 3 Генерация синтетических данных

В итоге, набор данных составляет более 48000 изображений, составленных из набора данных HKR. Однако, это не покрывает все требования - для полного решения необходимо присутствие рукописных цифр. Для решения проблемы используем набор из рукописных шрифтом и с помощью стандартной библиотеки PIL сгенерируем синтетические изображения рукописного текста с добавлением цифр. Всего было использовано 97 различных рукописных шрифтов для генерации 100 тысяч изображений различной длины в диапазоне от 5 до 30.

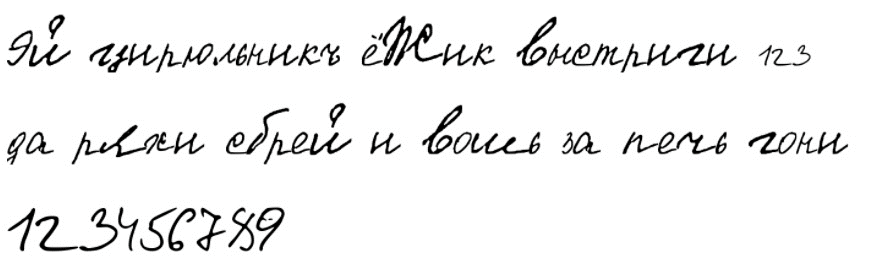


Рисунок 16 – пример синтетических данных

Аугментация данных — это методика создания дополнительных данных из имеющихся. Чаще всего, проблема ограниченного набора данных возникает при решении задач, связанных с обработкой изображений. Для устойчивости модели и расширения набора данных была применена аугментация с помощью библиотеки imgaug[10]. На каждое изображение приходилось две случайные функции аугментации из пяти возможных:

- подчеркивание черной линией

- инвертирование пикселей

- сжатие

- локальные аффинные преобразования

- случайные прямоугольные вырезы

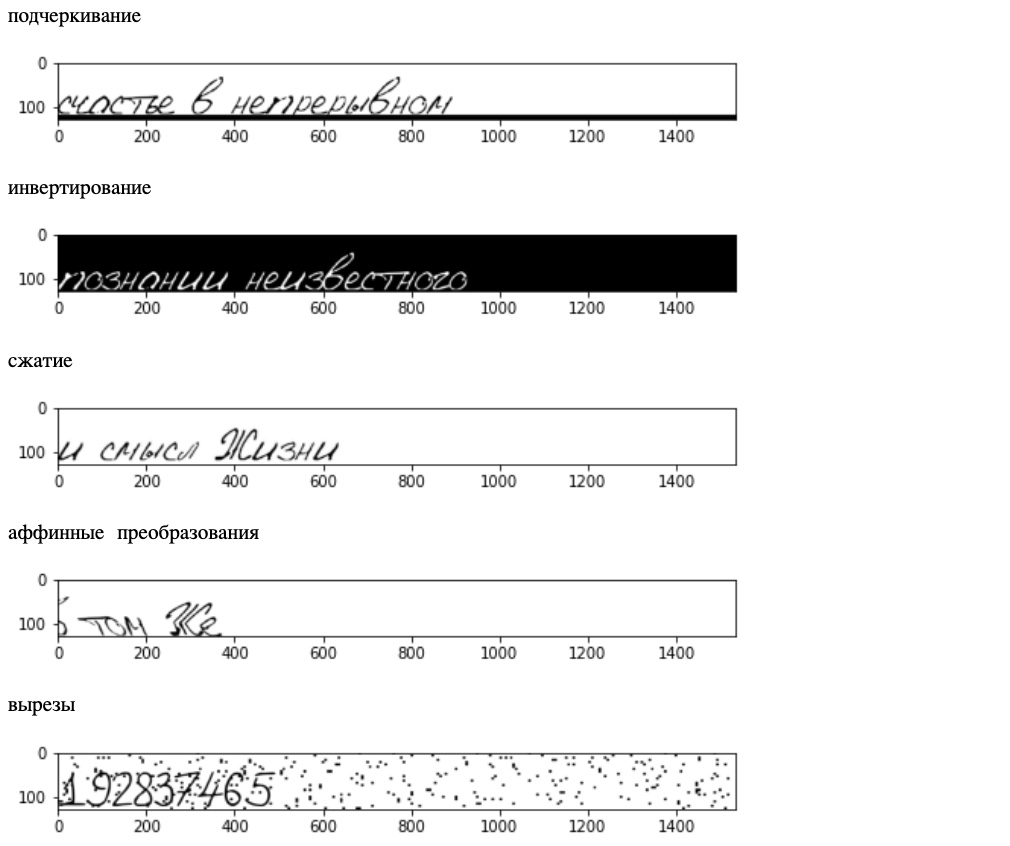


Рисунок 17 – пример применения функций аугментации изображений.

В итоге получаем 470 тысяч изображений в тренировочной выборке, 9 тысяч в валидационной и 5 тысяч в тестовой. Тестовая выборка не содержит синтетических данных. Набор данных хранится в виде csv таблицы, хранящей пути до изображений, текст на них, принадлежность в выборке и набору данных.

# 3. Реализация модулей

## 3.1 Реализация нейросетевой модели

Для реализации модели распознавания рукописного текста с применением конволюционных и рекурретных слоев, а также цикла обучения с применением CTC-потери, воспользуемся фреймворком машинного обучения с открытым исходным кодом PyTorch [11]. PyTorch поддерживается CUDA, что позволяет выполнять эффективные вычислительные операции с использованием графических процессоров. Для промежуточных вычислений воспользуемся библиотеками NumPy[12] и SciPy[13], для визуализации и аналитики воспользуемся Matplotlib[14], Pandas[15] и OpenCV[16].

Разработку модели можно разбить на следующие этапы:

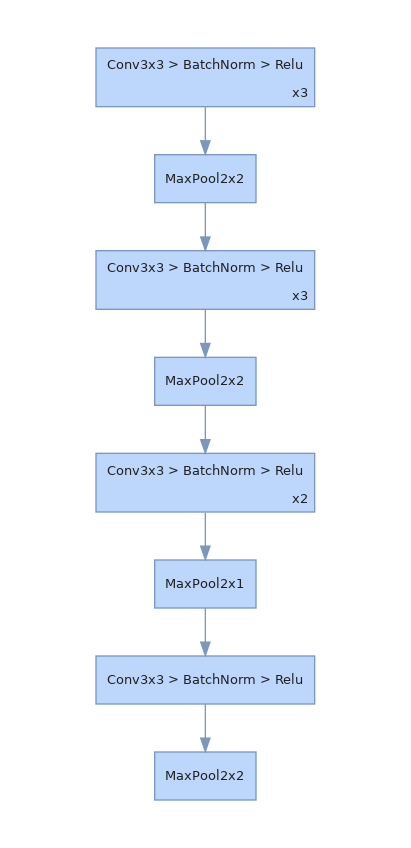
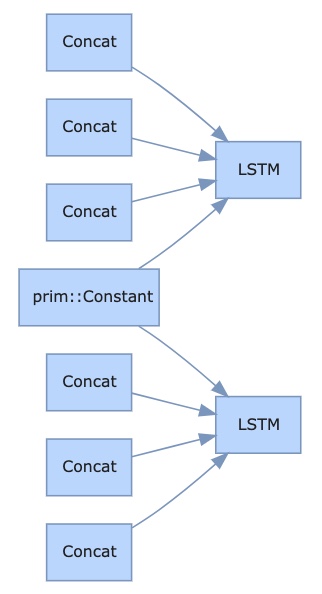
1. Разработка класса CNN, отвечающего за выделение признаков из изображений
2. Разработка класса BiLSTM, отвечающего за анализ временных шагов
3. Разработка класса CTCModel, отвечающего за синтез конволюционной и рекуррентной части с помощью полносвязных слоев, а также выходов рекуррентной части с каждым символов

Классы наследуется от родительского класса torch.nn.Module. Входом CNN является тензор формы (32, 1, 64, 768), где 32 – размер батча

Выходом CNN является тензор формы (32, 256, 4, 96), где 256, 4, 96 – количество каналов (признаков), высота и ширина карты признаков.

Далее, в процессе прогона батча, полносвязный слой переводит выходной тензор из CNN в входной тензор для BiLSTM, который имеет вид (32, 96, 512). То есть, конкатенировав признаки на оси, отвечающей за высоту карты признаков, и пропустив через полносвязный, мы получаем 512 признаков для каждого из 96 временных шагов.

BiLSTM сохраняет размерность тензора, поэтому последним слоем является полносвязный, преобразовывая вектор признаков в вектор из 87 значений, то есть длины используемого алфавита.



1. б)

Рисунок 17 – a) схема RNN б) схема CNN

## 3. 2 Предобработка изображений

Поскольку модель ориентирована на работу с однострочным текстом, стоит позаботится о выделение области рукописного текста на изображении. Для этого воспользуемся моделью детекции CRAFT[17] через фреймворк Easy OCR[18].

Модель детекции и модель распознавания текста лучше работают с обработанными данными без лишним шумов, поэтому воспользуемся адаптивной бинаризцией с применением взвешенной суммы Гаусса.

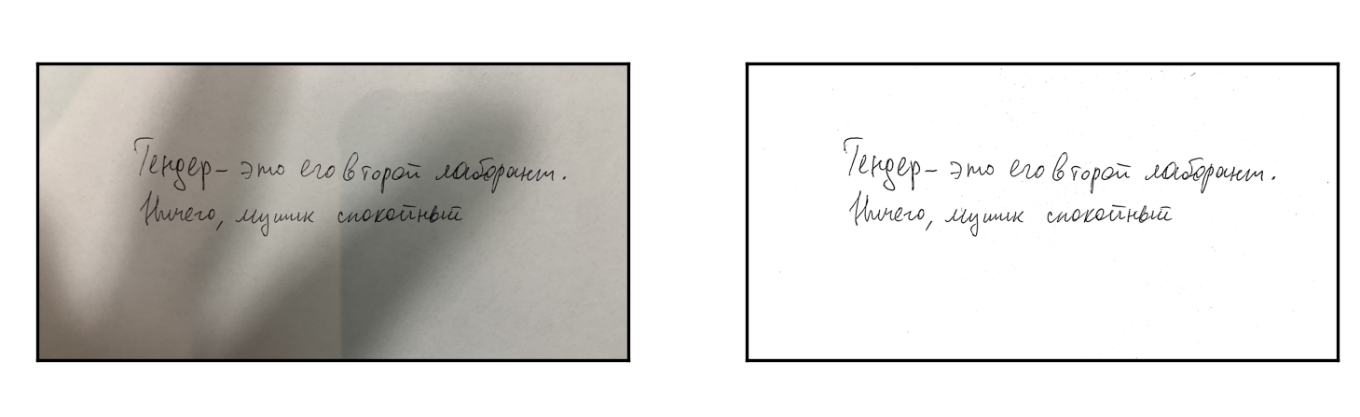


Рисунок 18 – пример работы алгоритма адаптивной бинаризации.

На вход могут поступать и негоризонтально-ориентированные изображения. Для решения этой проблемы разработаем функцию, вычисляющую угол и поворачивающее изображение. На вход принимается изображение, максимальный угол и шаг угла. Далее, в заданном диапазоне, наклоняем изображение и вычисляем оценку по формуле:

Где *h* – вектор гистограммы. Таким образом, мы отбираем гистограмму с наиболее ярко выраженными пиками, которые и отображают отдельные строки.

Поворот реализуется с помощью аффинных преобразований и матрицы поворота, реализованные в opencv.

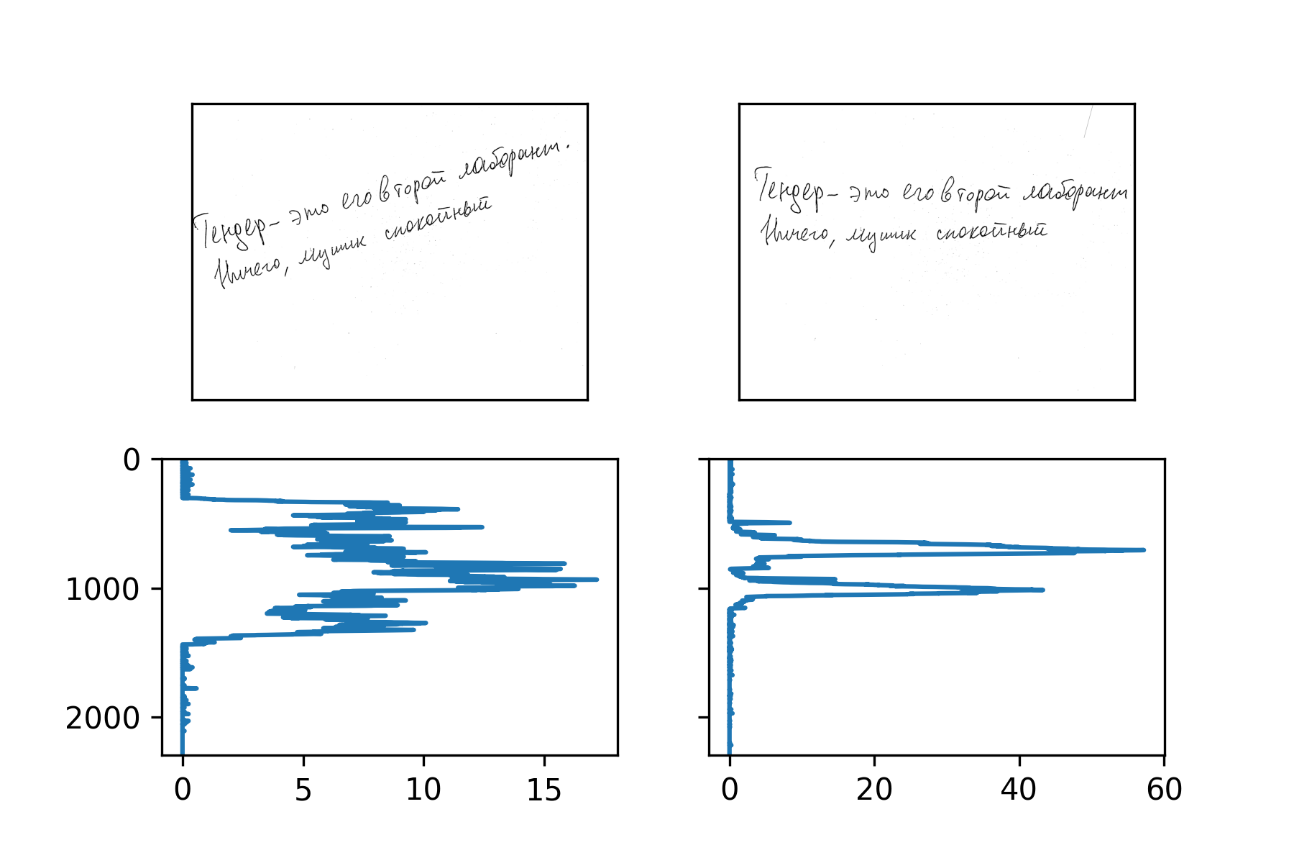


Рисунок 19 – пример работы выравнивания текста

На данном этапе изображение готово для детекции. Модель детекции CRAFT в составе фреймворка EasyOCR не требует специальной предобработки изображений.

## 3. 3 Реализация цикла обучения

Обучение нейронной сети – процесс настройки параметров посредством моделирования среды. Изначально веса задаются случайно. Далее, после прохода данных путем прямого прохождения, вычисляется значение ошибки. С помощью цепного правила вычисляются градиенты, которые показывают, насколько необходимо изменить параметр для минимизации ошибки. Распознавание рукописного текста относится к классу задач обучения с учителем посредством минимизации функции потерь. Воспользуемся стандартными для решения подобного класса задач оптимизатором Adam[19] с начальной скоростью обучения 0.0005, подобранным экспериментально и его снижаем в 0.8 раз каждые 5 эпох без улучшений.

Цикл обучения состоит из функции вычисления CTC-потери, шага оптимизатора, валидации и главной функции обучения. Модули работают независимо от модели, что позволяет с легкостью производить эксперименты над архитектурой.

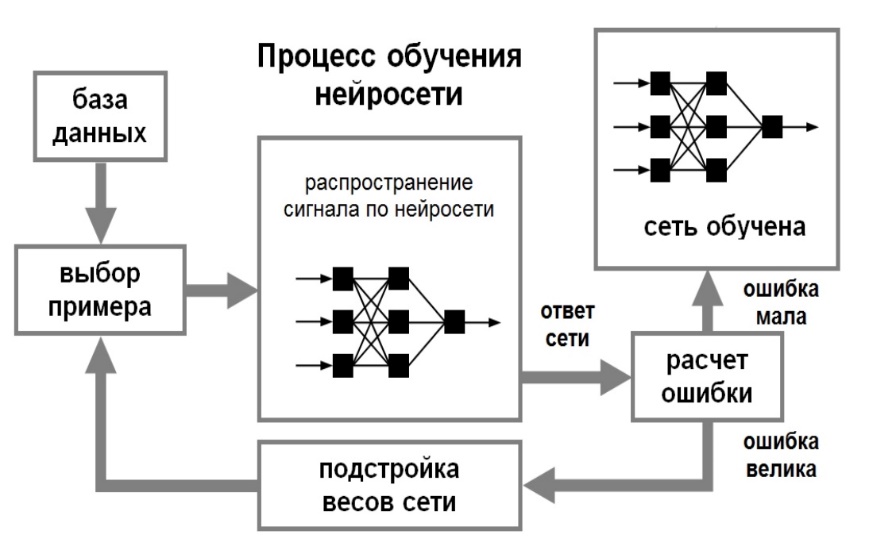


Рисунок 20 – процесс обучения нейросети

Для визуализации процесса обучения воспользуемся набором инструментов TensorBoard[20], выводя каждую эпоху метрики на валидационных данных, тренировочную и валидационную потерю. Цикл обучения проводится на сервере с графическим ускорителем Tesla A100. Каждая эпоха заняла примерно 30 минут, для полного обучения модели понадобилось 50 эпох.

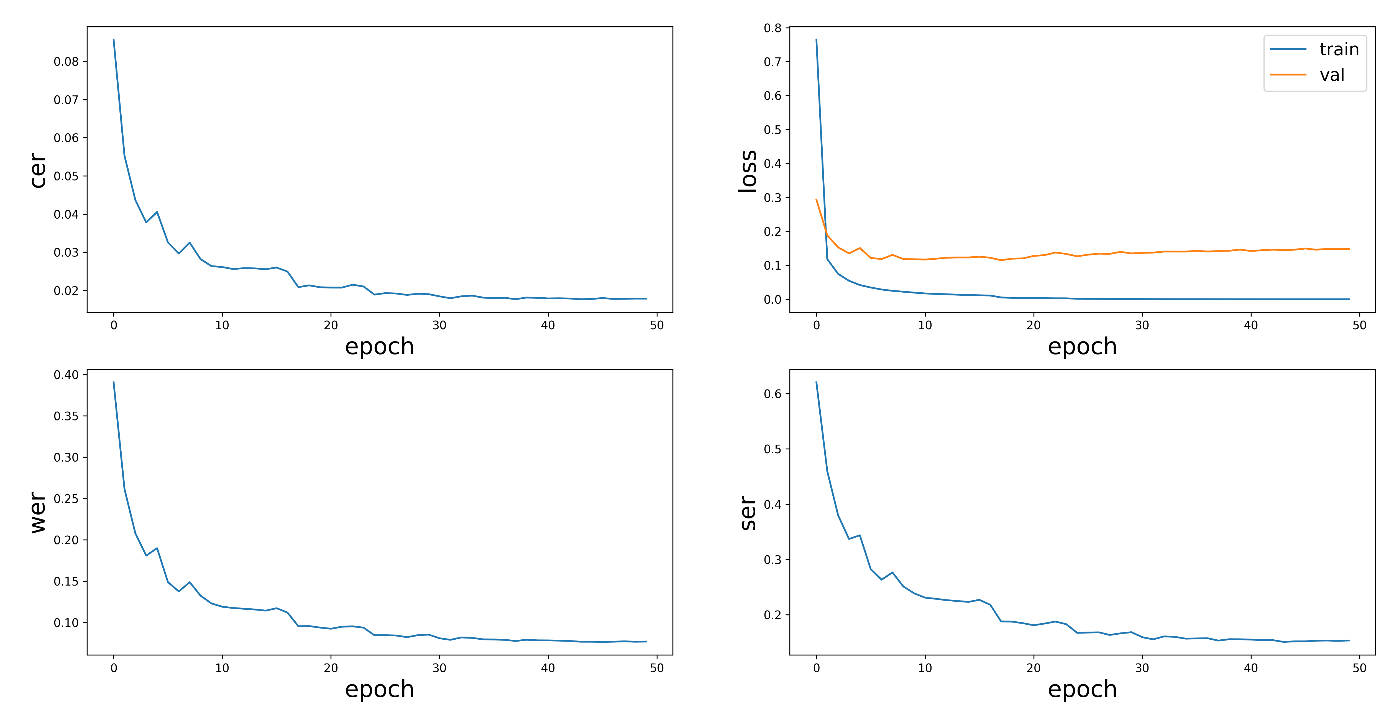


Рисунок 21 – показатели во время обучения

Как видно из графиков, отсутствуют явные признаки недообучения или переобучения модели. Валидационная потеря несколько возрастает после первых эпох и встает на плато, что может быть следствием неоднородности данных среди обучения и валидации. Потеря может интерпретироваться как уверенность модели. Несмотря на меньшую уверенность на валидации, валидационные метрики сходятся к нулю. Крутые спуски в районе 19 и 27 эпохи свидетельствуют о снижении скорости обучении, ведущее к более точному приближения к глобальному минимуму функционала ошибки.

# 4. Тестирование

## 4. 1 Оценка качества модели на тестовых данных

Для начала, рассмотрим метрики на тестовой части набора данных, содержащий исключительно рукотворный текст. Использование лучевого поиска CTC без языковой модели не улучшило показатели метрик.

Таблица 1 – сравнение декодирующих эвристик

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Эвристика декодирования | Точность по символам, % | Точность по словам, % | Точность, % | Время работы, мс |
| Жадный | 97 | 86 | 72 | 0.05 |
| Лучевой поиск по символам | 97 | 86 | 72 | 5.75 |
| Лучевой поиск по словам | 96 | 93 | 65 | 43 |
| Лучевой поиск по н-граммам | 70 | 37 | 43 | 22 |

Таким образом, жадный алгоритм справляется лучше всего и многократно выигрывает по скорости по сравнению с другими. Лучевой поиск по словам показывает наилучший результат точности по словам.

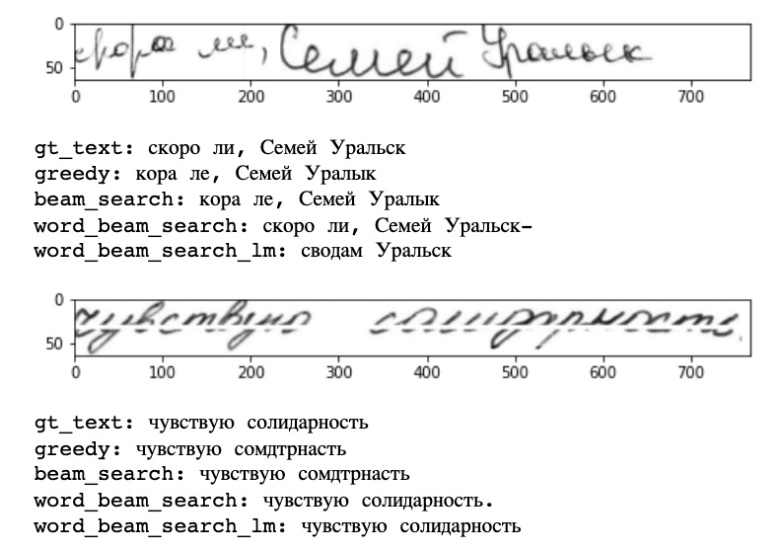


Рисунок 22 – пример распознавания и использования разных эвристик

Рассмотрим матрицу CTC выхода, чтобы наглядно увидеть каким образом происходит классификация по временным шагам. По оси x отложены наиболее вероятные символы, по оси y – временные шаги.

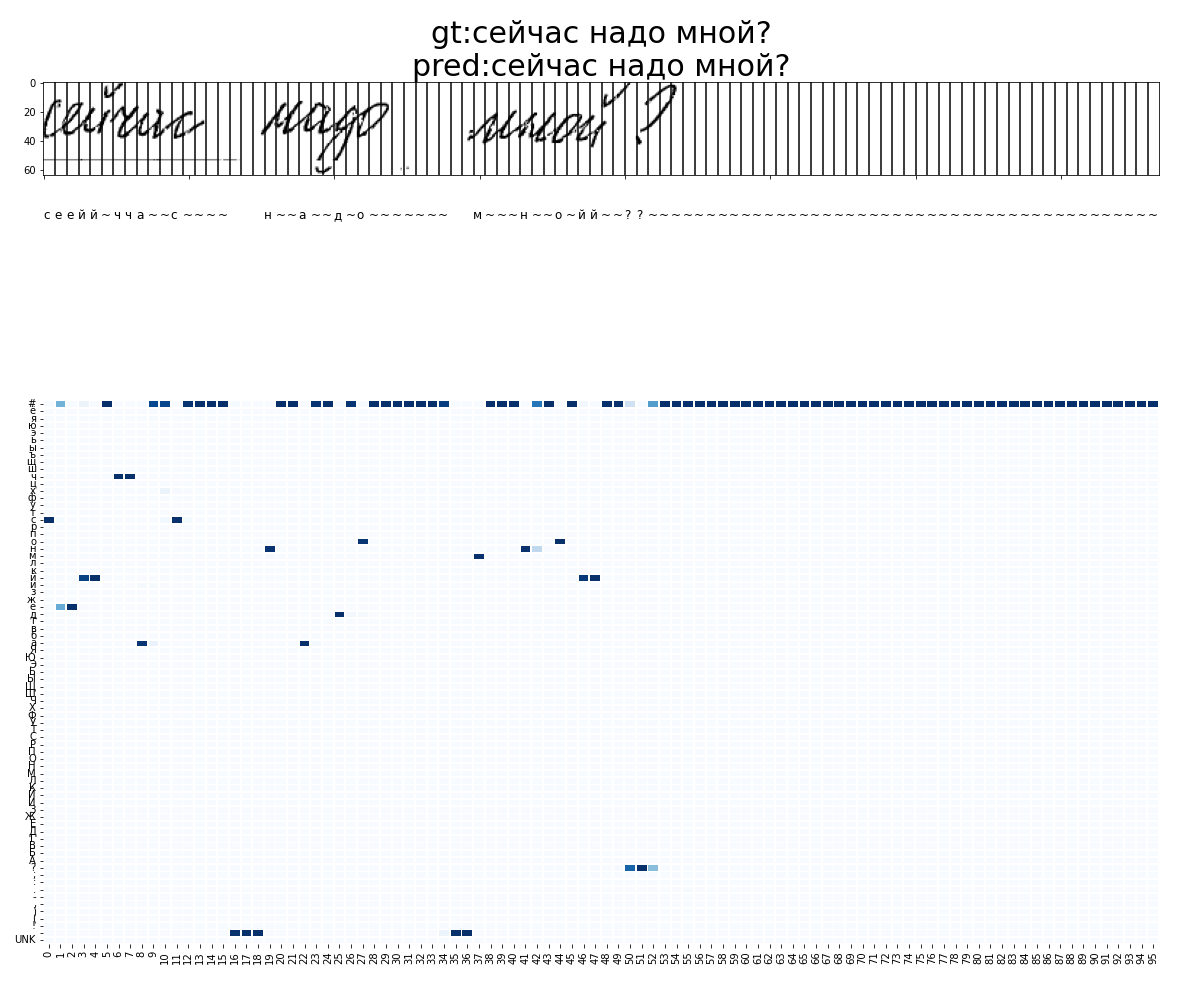


Рисунок 23 – визуализации CTC-матрицы

Тот факт, что модель предсказывает не предсказывает символ под каждый временной шаг (например, “м” занимает 3 временных шага, но имеет выводится лишь на одном) объясняется тем, как вычисляется CTC-потеря. Модель “штрафуют” за неуверенные решения, поэтому, рекуррентная сеть аккумулирует вероятности по соседним временным шагам и выдает самый вероятный результат, чаще всего занимающий один временной шаг.

Можно сделать вывод, что модель обучилась успешно и показывает достойные результаты на тестовой выборке

## 4. 2 Оценка качества модели на реальных данных

Протестируем распознавание в комбинации с детекцией на тексте, не применяемым при обучении модели.

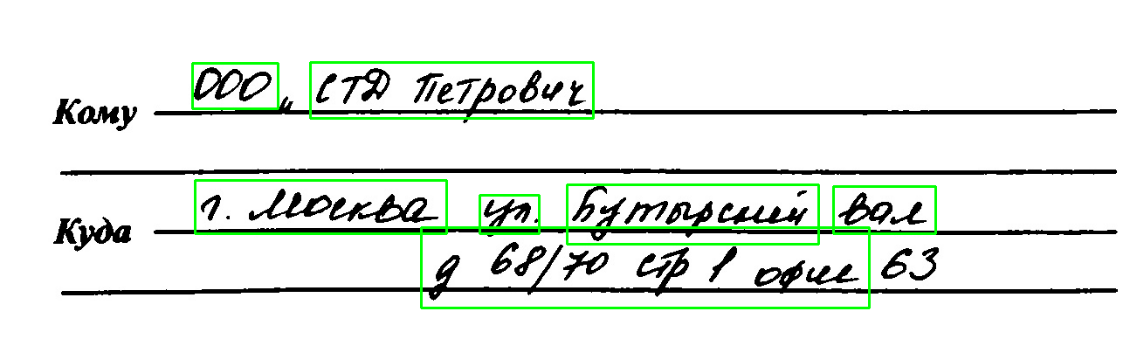


Рисунок 24 – применение детекции.

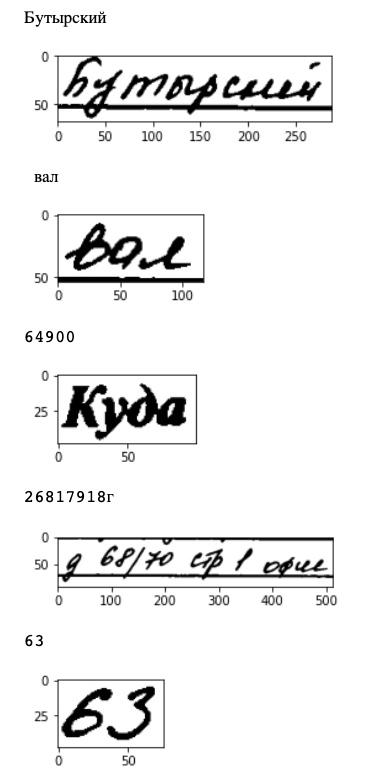
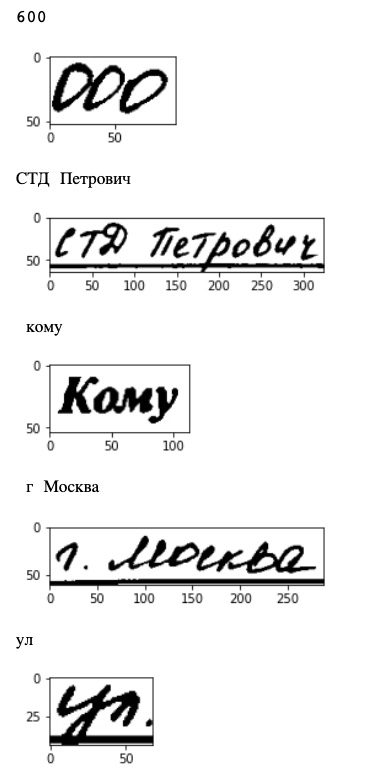


Рисунок 25 – применение распознавания с лучевым поиском по словам.

Модель справляется с поставленной задачей, но требует дополнительной постобработки результатов. Особые проблемы возникают при сочетании цифр и текста. И это неслучайно, поскольку в одном наборе данных цифры отсутствуют, а при генерации синтетики из-за особенностей шрифта не удается реализовать возможность сочетания цифр как на рисунке 25.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы были рассмотрены архитектуры нейронных сетей, решающие задачи распознавания рукописного текста. Также были рассмотрены базовые компоненты нейронных сетей и различные способы расшифровки полученных результатов.

На основе изученной информации, была разработана архитектура на основе конволюционной и рекуррентной нейросети и цикл обучения. А также, модули предобработки текста, включающий в себя устранение шума и выравнивание, детекции рукописного текста на сцене, генерации синтетических данных и декодировании различными эвристиками.

Распознавание кириллицы затруднено тем фактом, что в открытом доступе нет наборов данных. В дальнейшем, стоит лучше проработать языковую модель. Лучевой поиск по словам показывает хорошие результаты, если не обращать внимание на пунктуацию. Также, узким местом является модель детекции, но собственная реализация подобной системы трудоемкая отдельная задача. Несмотря на это, при помощи генерации синтетических данных и применения функций аугментации, получилось добиться достаточного набора данных для успешного решения поставленной задачи.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin, Attention Is All You Need. [Электронный ресурс]. – URL: https://arxiv.org/abs/1706.03762 (дата обращения 18.10.2021)
2. M. Schuster and K. K. Paliwal, Bidirectional recurrent neural networks // IEEE Transactions on Signal Processing – 1997. p. 2673-2681
3. Alex Graves and Santiago Fernandez, Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. [Электронный ресурс]. – URL: https://www.cs.toronto.edu/~graves/icml\_2006.pdf (дата обращения 18.10.2021)
4. Freitag Markus and Al-Onaizan Yaser, Beam Search Strategies for Neural Machine Translation // Proceedings of the First Workshop on Neural Machine Translation - 2017
5. H. Scheidl, S. Fiel and R. Sablatnig, Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm //16th International Conference on Frontiers in Handwriting Recognition (ICFHR), - 2018, pp. 253-258
6. Alex Graves, Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks // Guide to OCR for Arabic Scripts – 2012. P. 297-313
7. Minghao Li and Tengchao Lv and Lei Cui and Yijuan Lu and Dinei Florencio and Cha Zhang and Zhoujun Li, TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. [Электронный ресурс]. – URL: https://arxiv.org/abs/2109.1028262 (дата обращения 18.10.2021)
8. Baoguang Shi and Xiang Bai and Cong Yao, An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. [Электронный ресурс]. – URL: https://arxiv.org/abs/1507.05717 (дата обращения 18.10.2021)
9. Nurseitov, Daniyar and Bostanbekov, Kairat and Kurmankhojayev, Daniyar and Alimova, Anel and Abdallah, Abdelrahman and Tolegenov, Rassul, Handwritten Kazakh and Russian (HKR) database for text recognition // Multimedia Tools and Applications – 2021, p 1-23
10. Документация библиотеки ImgAug. [Электронный ресурс]. – URL: https://imgaug.readthedocs.io/en/latest/ (дата обращения 12.11.2021)
11. Документация фреймворка PyTorch. [Электронный ресурс]. – URL: https://pytorch.org/docs/stable/index.html (дата обращения 12.11.2021)
12. Документация библиотеки NumPy. [Электронный ресурс]. – URL: https://numpy.org/doc/stable/reference (дата обращения 12.11.2021)
13. Документация библиотеки SciPy. [Электронный ресурс]. – URL: https://docs.scipy.org/doc/scipy/reference/ (дата обращения 12.11.2021)
14. Документация библиотеки MatPlotLib. [Электронный ресурс]. – URL: https://matplotlib.org/stable/api/\_as\_gen/matplotlib.pyplot.html (дата обращения 12.11.2021)
15. Документация библиотеки Pandas. [Электронный ресурс]. – URL: https://pandas.pydata.org/docs/ (дата обращения 12.11.2021)
16. Документация библиотеки python-OpenCV. [Электронный ресурс]. – URL: https://docs.opencv.org/3.4/d6/d00/tutorial\_py\_root.html (дата обращения 12.11.2021)
17. Документация модели детекции CRAFT. [Электронный ресурс]. – URL: https://github.com/clovaai/CRAFT-pytorch (дата обращения 12.11.2021)
18. Документация фреймворка EasyOCR. [Электронный ресурс]. – URL: https://github.com/JaidedAI/EasyOCR (дата обращения 12.11.2021)
19. Diederik P. Kingma and Jimmy Ba, Adam: A Method for Stochastic Optimization [Электронный ресурс]. – URL: https://www.cs.toronto.edu/~graves/icml\_2006.pdf (дата обращения 18.10.2021)
20. Документация библиотеки TensorBoard. [Электронный ресурс]. – URL: <https://www.tensorflow.org/tensorboard/get_started> (дата обращения 12.11.2021)