МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Институт интеллектуальной робототехники

Кафедра Интеллектуальных систем теплофизики ИИР
Направление подготовки: 15.03.06 Мехатроника и робототехника
Направленность (профиль): Мехатроника и робототехника

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Шкоков Родион Олегович
(Фамилия, Имя, Отчество автора)

Тема работы Разработка алгоритмов по расшифровке архивных рукописных документов начала XX века

«К защите допущен(-а)»
и.о. зав. кафедрой ИСТИИР,
д.т.н, доцент

Назаров А.Д. /_____
(ФИО) / (подпись)

«……»……2024г.

Руководитель ВКР
доцент ФИТ НГУ
к.ф.н.

Хоцкина О.В. /_____
(М.П.) (ФИО) / (подпись)

«……»……2024г.

Новосибирск, 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Необходимость обнаружения текста в документах для последующей цифровизации этой информации является очень актуальной темой в наше время.

Во-первых, это позволит сэкономить время и силы сотрудников, которые в нашу эпоху тратят много времени на ручной ввод данных. Вместо использования ручного труда людей, система распознавания данных может автоматически извлекать информацию из документов и сохранять ее в удобном электронном формате.

Во-вторых, распознавание данных в старых документах позволит проводить более эффективный анализ информации. Благодаря использованию современных алгоритмов машинного обучения и искусственного интеллекта, системы распознавания данных смогут автоматически классифицировать и структурировать информацию, что позволит более точно анализировать ее.

Третья причина, по которой распознавание данных в старых документах является необходимой задачей – это сохранение и защита информации. Бумажные документы подвержены риску утраты или повреждения, что может привести к потере ценной информации. Кроме того, доступ к бумажным документам ограничен, что затрудняет обмен информацией между сотрудниками и организациями. Если же данные будут распознаны и сохранены в электронном формате, они будут легко доступны и защищены от потери или повреждения.

Детектирование рукописного текста является сложной задачей из-за ряда особенностей и вызовов, связанных с разнообразием стилей письма, нечеткостью и неоднородностью написания, а также проблемами, связанными с качеством и разрешением изображения. Выражается это в следующем:

1) Каждый человек имеет свой уникальный стиль письма, что делает анализ и распознавание рукописного текста сложным. Различия в наклоне, размере и форме букв, неровности и скорости письма могут привести к большому разнообразию внешнего вида одного и того же символа. Такие особенности текста создают сложности при попытке автоматической детектирования и распознавания рукописного текст, поскольку требуется учитывать все возможные варианты написания символов.

2) Рукописный текст часто имеет нечеткое и неоднородное написание, что дополнительно усложняет его распознавание. Размытость, неровности и перекрытия символов могут затруднить определение границ и формы букв. Кроме того, наличие шума и искажений на изображении также может плохо повлиять на точность и надежность детекции рукописного текста.

3) Ещё одна сложность, связанная с детектированием рукописного текста, заключается в проблемах с качеством и разрешением изображения. Низкое разрешение, сжатие или искажение изображения могут привести к потере деталей и информации о рукописном тексте. Это может затруднить обнаружение и распознавание символов и слов, особенно если они имеют малый размер или содержат мелкие детали.

Среди рукописных текстов особый интерес вызывают тексты начала 20 века. В эту эпоху количество людей, обладающих грамотностью, было значительно выше, чем в 18 или 19 веках, а значит выборка данных будет более разнообразной. А еще информация за 20 век является более актуальной для людей, которые обращаются в архивы в поиске нужной для них информации.

Данная работа заключается в распознании текстов начала 20 века, и главная сложность заключается в особенностях языка и стиле письма, используемого в прошлом. Старые документы могут содержать устаревшие слова, фразы и грамматические конструкции, которые могут быть непонятны или незнакомы для современных систем распознавания текста. Еще одной сложностью является то, что при записывании информации человек мог прибегать к скорописи, что может затруднить распознавание подобных текстов. Это создает дополнительные трудности при попытке правильной интерпретации и распознавания содержимого архивных документов.

Всё вышеперечисленное делает невозможным использование современных методов по распознанию старых рукописных текстов для перевода в удобочитаемую форму. В связи с этим, **целью данной работы** стало создание программно-аппаратного комплекса на основе нейронных сетей, который способен распознавать рукописный текст начала 20 века и переводить его в печатный вид, понятный пользователю. Для достижения данной цели были поставлены следующие задачи:

1. Сформировать собственный датасет на основе писем начала 20 века
2. Реализовать алгоритм поиска цепочек символов (букв, слов, знаков препинания, цифр) на изображениях с текстом.
3. Реализовать алгоритм распознавания найденных на изображении цепочек символов.
4. Используя реализованные алгоритмы, распознать рукописные письма начала 20 века.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

AUC – Площадь под кривой (Area Under the Curve).

Adam – Адаптивная оценка моментов (Adaptive Moment Estimation)

AdamW – Адаптивная оценка моментов с затуханием весов (Adaptive Moment Estimation with Weight Decay)

BiLSTM – Двунаправленная долгая краткосрочная память (Bidirectional Long short-term memory)

CER – Коэффициент ошибок в символах (Character Error Rate)

CNN – Сверточная нейронная сеть (Convolutional Neural Network)

CTC – Коннекционистская временная классификация (Connectionist Temporal Classification)

GeLU – Гауссовская линейная функция (Gaussian Error Linear Unit)

HTR – Распознание рукописного текста (Handwritten Text Recognition)

JIT – Точно в нужное время (Just In Time)

LSTM – Долгая краткосрочная память (Long short-term memory)

OCR – Оптическое распознание символов (Optical Character Recognition)

CRNN – Сверточная рекурентная нейронная сеть (Convolutional Recurrent Neural Network)

ReLU – Линейный выпрямитель (Rectified Linear Unit)

ResNet – Остаточная нейронная сеть (Residual neural network)

RNN – Рекурентная нейронная сеть (Recurrent Neural Network)

ROC – Рабочая характеристика приёмника (Receiver Operating Characteristic)

SGD – Стохастический градиентный спуск (Stochastic gradient descent)

FN – Ложно отрицательный (False Negative)

FP – Ложно положительны (False Positive)

TN – Правдиво отрицательный (True Negative)

TNR – Оценка правдиво отрицательных (True Negative Rate)

TP – Правдиво положительный (True Positive)

TPR – Оценка правдиво положительных (True Positive Rate)

WER – Коэффициент ошибок в словах (Word Error Rate)

ДИ – Допустимый Интервал

1 ЛИТЕРАТУРНЫЙ ОБЗОР

1.1 Особенности распознавания текста на изображении

Алфавит, которым мы пользуемся в данное время сформировался в 1918 году (105 лет назад). До этого времени в языке существовали 35 букв (а в славянской азбуке их вообще было 43).

Попытку реформирования первым предпринял Петр I в 1708-1710 годах. Был введен гражданский шрифт, была утверждена азбука. Были упразднены многие буквы («ом», «юс малый», «юс большой», «земля», «ферт», «ижица» и др.), что позволило «очистить язык» и сделать его проще. Своей реформой Петр I упростил процесс обучения грамотности и распространения знаний, а также это поспособствовало упрощению обмена информацией внутри государства.

Согласно декрету от 10 октября 1918 года «О введении новой орфографии», «все правительственные издания, периодические (газеты и журналы) и непериодические (научные труды, сборники и т. п.), все документы и бумаги должны с 15 октября 1918 года. печататься согласно при сем прилагаемому новому правописанию». [2]

Если правительственные издания можно обязать выполнять закон, то за всеми остальными проследить сложно. Так из-за больших противоречий и недовольства, многие люди до 1920х и 1930х писали так, как хотели. Русское зарубежье перешло на новую орфографию только в 1940-е – 1950-е годы, хотя некоторые и по сей день печатают издаваемые книги и газеты по-старому. [3]

Буквы, которые пропали из языка по реформе 1918 были - «ять», «фита», «и десятеричное» и «ижица», а пришли в алфавит «ё» и «й». Также исчез твердый знак на конце слов и изменилось окончание в некоторых падежных существительных (см. рисунок 1).

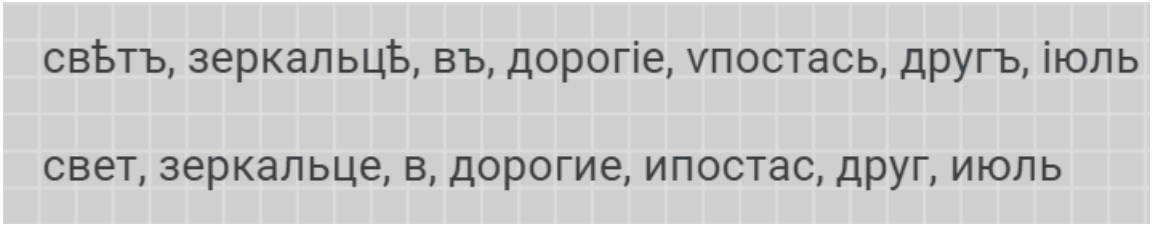


Рисунок 1 – Сопоставление слов с старыми буквами и новыми буквами

В результате реформ Петра I и реформы в 1918 года в языке произошли серьезные изменения. Во-первых, был нарушен строй языка (система алфавита), который представляет собой сложную философскую и математическую систему. Каждая буква в алфавите имела свое место, порядковый номер, своё имя и числовое значение. Во-вторых, изъяты названия букв, в них был завет каждому, кто начинал изучать русский язык: «Азъ буки ведаю, глаголю добро» и «Рцы слово твердо» (р, с, т). Будь крепок в слове. В-третьих, в результате реформ русский

язык уже потерял девять гласных букв, а это, по словам экспертов, колоссальная энергетика. В-четвертых, после отмены «Ъ» в конце слов многие слова стали оканчиваться на согласный звук, а раньше это были открытые слоги, оканчивающиеся на полугласный «Ъ» — и здесь опять-таки энергетика. В-пятых, сокращение алфавита вызвало обвальные процессы в правописании, семантике, в лексикологии. Все эти изменения нашли отражение на социально-экономической, образовательной сферах российского социума. [1]

Ещё стоит учесть, что неофициальным языком русской аристократии 19 века и начала 20 был французский и вообще высшее сословие любило блеснуть знанием иностранных языков, что предполагает наличие кусков, написанных на французском и немецком языках. А так как это конец 19 века – начало 20 века, то стоит учитывать, что возможно эти языки могут быть не похожи на современные (к примеру, раньше в английском было слово hath, теперь это has).

Текст бывает различных видов, самыми распространёнными из которых считают – печатные и рукописные. При решении задачи распознавания печатного текста на изображениях обычно применяют технологию оптического распознавания символов (OCR, optical character recognition), которая позволяет автоматически распознавать и извлекать текст с изображений.

OCR включает в себя разнообразные методы и алгоритмы для обработки изображений, извлечения текста и последующего его распознавания. В ходе работы с печатным текстом на изображениях, алгоритм сначала проводит предварительную обработку изображения для более точного выделения областей с текстом. После этого алгоритм просматривает найденные области и распознает символы, находящиеся на них.

В Python при работе с рукописным текстом используют такие библиотеки, как pytesseract (обертка над Tesseract OCR), OpenCV (библиотека компьютерного зрения) и другие. Используя эти библиотеки можно загружать изображения, выполнять их обработку, извлекать текст и проводить его распознавание с использованием соответствующих алгоритмов.

При анализе рукописного текста на изображениях возникают дополнительные сложности из-за его нестандартности и разнообразия почерка. Для решения этой задачи часто применяются методы глубокого обучения, такие как рекуррентные нейронные сети (RNN) или сверточные нейронные сети (CNN). Эти методы позволяют обучать модели на больших объемах данных рукописного текста и повышают эффективность распознавания.

1.2 Существующие датасеты

В открытом доступе удалось обнаружить несколько размеченных датасетов с примерами рукописного текста, вот некоторые из них:

1) IAM Handwritten Forms Dataset (набор форм рукописного текста на английском языке). [4]

2) George Washington (набор из 20 рукописных писем Джорджа Вашингтона 1755 года, всего 5000 слов). [5]

3) Digital Peter (набор из 9000 фраз из писем Петра I). [6]

4) CAPTCHA Images (Набор из 1070 изображений капчи). [7]

Также в доступе есть набор писем начала 20 века человека, проживавшего на тот момент в Петербурге. Для разметки писем был использован сервис Transkribus. [8]

Transkribus (см. рисунок 2) – это онлайн сервис для разметки текстовых датасетов. На сайте можно делать оцифровку и распознание текстов как раз на основе алгоритма HTR (HTR, hand text recognition). Перевод можно выполнить как собственноручно, так и автоматически, при помощи одной из “открытых” моделей (см. рисунок 3).

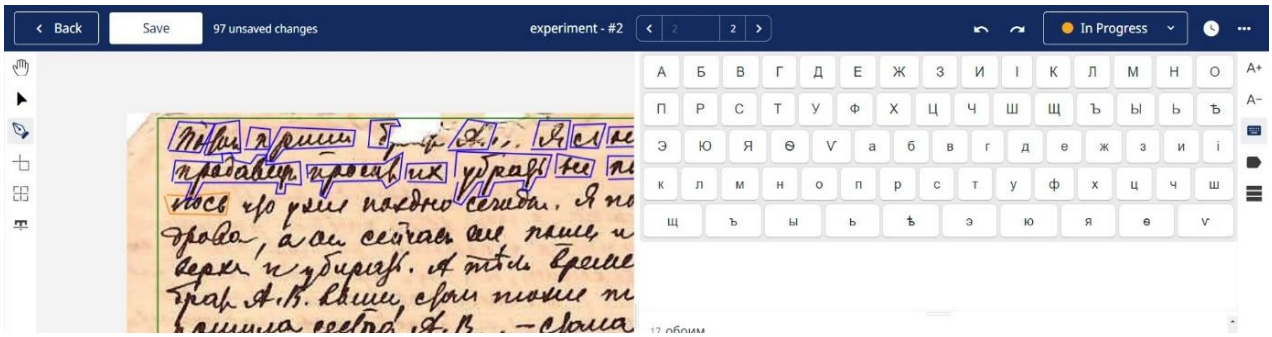


Рисунок 2 – Вид приложения Transkribus

Ради эксперимента было решено проверить существующую в общем доступе модель для оцифровки рукописных российских текстов 19-20 веков. Результат автоматической разметки имеет значительное количество ошибок и недочетов, на тестовых данных алгоритм показал следующие значения:

1) ДИ CER = [0.182, 0.290]

2) ДИ WER = [0.468, 0.765]

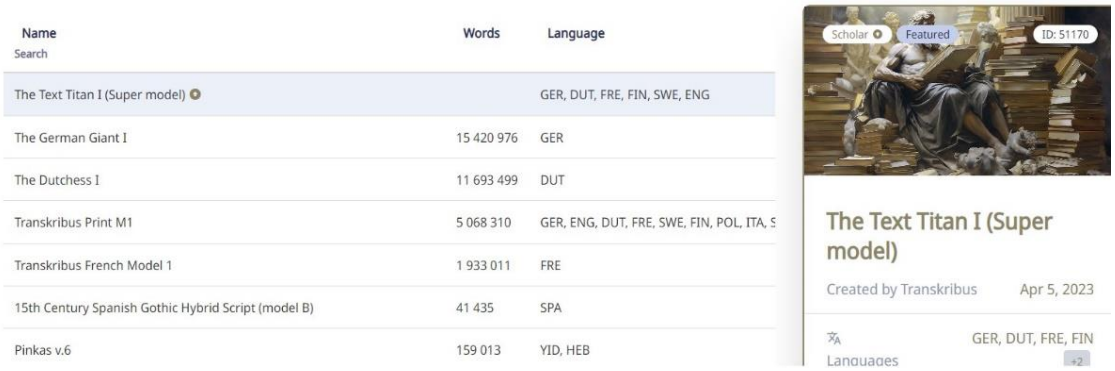


Рисунок 3 – Набор моделей по оцифровке текста в приложении Transkribus

Это еще раз подтверждает актуальность проблемы расшифровки рукописного текста, пример реального текста и расшифровки сервисом Transkribus изображено на рисунке 4.

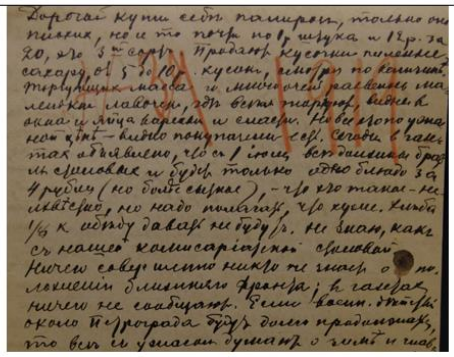
Перевод «Transkribus»	Текст	Оригинальный перевод
Дорогый кути себѣ папиравъ, только оно плохих, но ие то почта по орщчука и 12 р. за 20, это 3й сарх) продаювъ кусочний пиленые сахару, ов 3 до 10р. кусик, смотая по келичить, торгуящих масса и многоочень расведось ма¬ ленький лавочско, где все тарковов, видны в окна и яйца варскый и снасти. Но все допо у ной цень – видно покупаиди сеть. Сегоди в ган¬ тах объявлено, что с 1июня все долхины бра м стиловых и буде только одно бледо зе Урублу (но болде сызное), - что его такое-не ихаестно, но надо помагать, что куже. Хлеба 11 к обеду давазь не будуд, не знаю, как с нашей комисариатской своловой, ничего совершенно никго не знаер оДт¬ ложении ближнего фронга; в галчах ничего не сообщах Если воени детстви около перограда дуд долго продалонать то все с ужасой буман о зим и щав.		Дорогой купи себѣ папиросокъ, только он Плохих, но и то почти по 1р штука м 12р. за 20, это 3его сорта. Продаютъ кусочки пиленые сахару, от 5 до 10р кусокъ, смотри по величинѣ, торгующих масса и много развелось ма деньких лавочек, гдѣ всѣм торгуютъ, видно в окна и яйца вареныя и сласти. Но все это по ужас ной цѣнѣ - видно покупатели есть. сегоди в газе так объявлено, что съ 1 іюня всѣ должны брать из столовых и будетъ только одно блюдо за 4 рубля (но болѣе сытное) - что это такое – не извѣстно, но надо полагать, что хуже. Хлеба 1/8 к обѣду давать не будутъ, не знаю, какъ съ нашей камисаріарской столовой Ничего совер щенно никто не знаетъ о по доженіи ближняго фронта; въ газетах ничего не сообщаютъ. Если военн, Дѣйствія около Петрограда будутъ долго продолжаться, то всѣ съ ужасомъ думаютъ о зимѣ и щав.

Рисунок 4 – Результаты перевода рукописного текста переводчиком и сервисом Transkribus (в левой колонке корректный перевод по середине изначальный текст, в правой колонке перевод алгоритмом)

Если современные сервисы расшифровки не справляются с переводом рукописного текста начала 20 века, то надо посмотреть какие еще есть способы для расшифровки.

1.3 Существующие методы распознавания рукописного текста

1.3.1 Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN) – это класс нейронных сетей, спроектированный для работы с последовательными данными, такими как текст или временные ряды. Основная характеристика RNN заключается в наличии обратных связей между нейронами, что позволяет им запоминать и использовать информацию о предыдущих состояниях сети для обработки последующих входных данных. Это обеспечивает возможность модели учитывать контекст и последовательность данных, что делает их эффективными в задачах, где важно учитывать зависимости между элементами последовательности. Пример архитектуры RNN на рисунке 5.

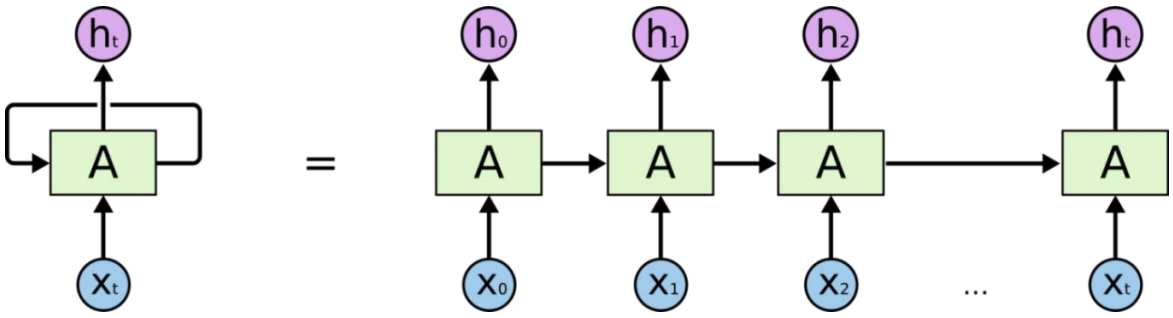


Рисунок 5 – Рекурентная нейронная сеть (взято из статьи «Упадок RNN и LSTM сетей») [29]

Еще одной отличительной особенностью рекуррентных нейронных сетей является то, что они умеют обрабатывать входные последовательности переменной длины из-за своей способности к автоматическому масштабированию входных данных. Благодаря обратным связям между нейронами, RNN имеют возможность адаптироваться к различным размерам входных последовательностей, что делает их гибкими и эффективными для решения задач, где данные могут быть различной длины или когда важны контекст и последовательность событий. Кроме того, RNN могут быть обучены как на данных с учителем, так и без учителя, что делает их мощным инструментом для моделирования последовательных данных.

RNN широко используют в задачах, где важна последовательность, например, в машинном переводе, анализе текста [9, 10, 13], распознавании рукописного текста и других. Однако у обычных RNN может возникать проблема затухания или взрыва градиента, что может затруднить обучение модели на длинных последовательностях.

1.3.2 Долгая краткосрочная память

С увеличением расстояния между элементами последовательности обычные рекуррентные нейронные сети (RNN) не справляются проблемой ухудшения эффективности передачи контекста. Однако долгая краткосрочная память (LSTM) способна эффективно обучаться долгосрочным зависимостям, позволяя модели учитывать информацию на больших расстояниях в последовательности. [29]

LSTM (Long Short-Term Memory) – это разновидность рекуррентной нейронной сети, которая хорошо подходит для работы с последовательными данными, такими как рукописный текст на изображениях. Основное преимущество LSTM перед обычными рекуррентными нейронными сетями заключается в ее способности запоминать и использовать информацию на протяжении длительных временных интервалов. За счет наличия особых "воротных" (gate) узлов в своей структуре, LSTM способна эффективно управлять потоком информации и избегать проблемы затухания градиента, что часто является проблемой для обычных рекуррентных сетей.

На рисунке 6 показано, что LSTM содержит не один, а целых четыре слоя нейронной сети. В начальной части сети изображение разбивается на k вертикальных сегментов вдоль

горизонтальной оси. После применения свертки каждому сегменту назначается вектор признаков размерности (1, n). Удаление одного из измерений позволяет уменьшить размерность карты признаков ((k, 1, n) -> (k, n)) без потери информации. Это приводит к получению двумерной матрицы признаков размерности (k, n) на выходе, которую затем можно передать в рекуррентную часть сети (RNN). Здесь n определяет размер вектора признаков для каждого окна и представляет собой количество карт признаков, полученных после последней свертки.

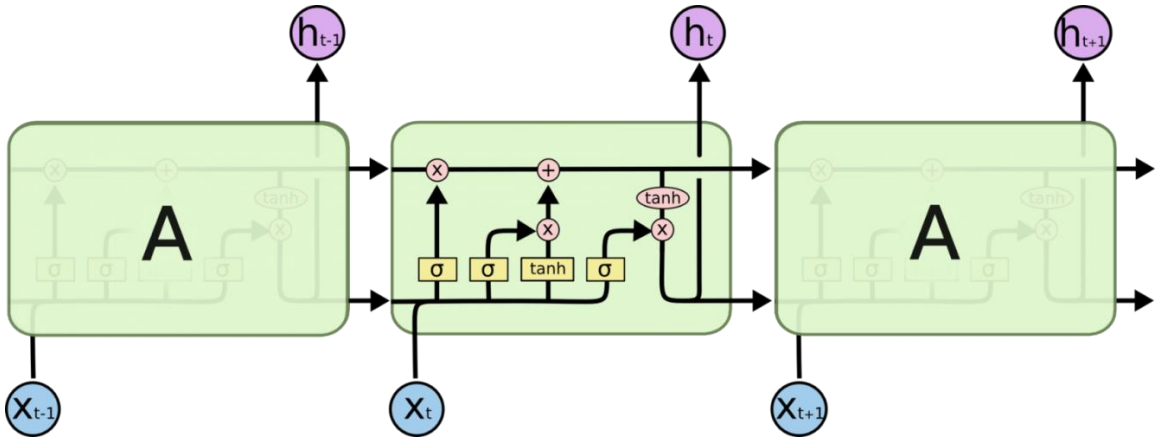


Рисунок 6 – Рекурентная нейронная сеть (взято из статьи «Упадок RNN и LSTM сетей»)

Используя LSTM, возможно учитывать долгосрочные зависимости в данных, что придает этой модели эффективность в ситуациях, где важны контекст и последовательность, как, например, в задачах распознавания текста на изображениях. Благодаря способности сохранять информацию на протяжении длительных периодов, LSTM может успешно работать с рукописным текстом и предсказывать следующие символы в последовательности. [11]

1.3.3 Трансформеры

Трансформеры – это архитектура моделей глубокого обучения, которая была разработана для обработки последовательных данных, таких как тексты или изображения. Они состоят из нескольких слоев, включающих механизм внимания (attention), который позволяет модели обращать внимание на разные части входных данных при генерации выхода. Механизм внимания (attention) стал популярным в области машинного перевода и представляет собой современную альтернативу LSTM. В отличие от LSTM, где каждый шаг RNN оперирует с фиксированным объемом информации, механизм внимания позволяет модели обращаться к более обширному "хранилищу" информации на каждом шаге. Однако главным недостатком рекуррентных нейронных сетей с механизмом внимания является отсутствие возможности параллельных вычислений. Таким образом в архитектуре трансформеров, рекуррентные нейронные сети были заменены механизмом внимания, что позволило решить эту проблему и улучшить производительность моделей.

Трансформеры быстро стали популярными в области обработки естественного языка и машинного перевода (см. рисунок 7) благодаря своей способности эффективно моделировать долгосрочные зависимости в последовательностях и работать параллельно, что улучшает скорость обучения и вывода результатов.

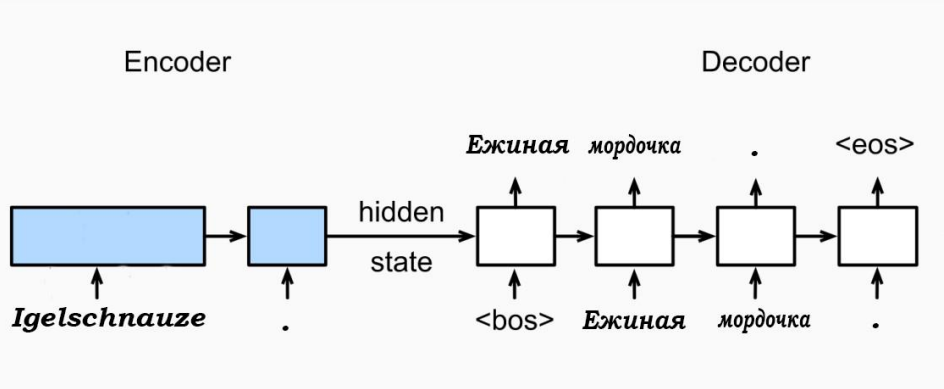


Рисунок 7 – Пример использования Трансформеров для перевода

А теперь немного подробнее. Как и подобные нейронные сети для машинного перевода, Трансформеры также имеют двухмодульную структуру (см. рисунок 8).

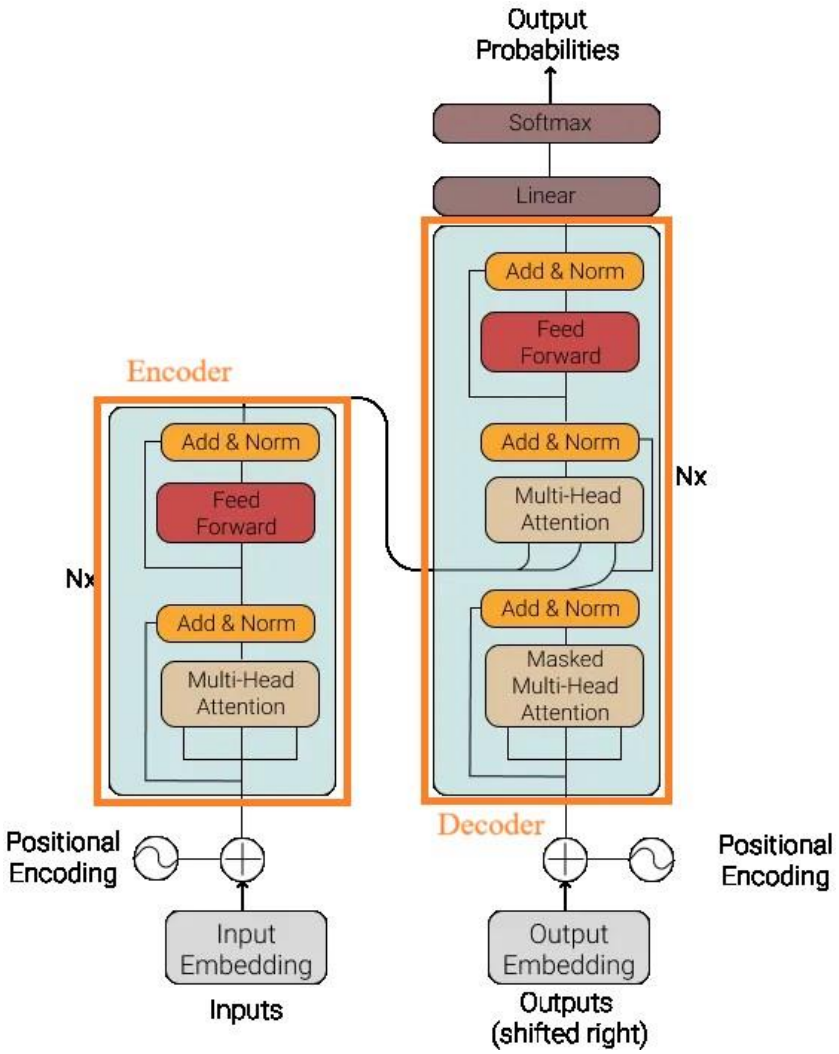


Рисунок 8 – Архитектура Трансформера.

1) Кодировщик (Encoder): этот компонент принимает последовательность входных данных и генерирует их представление, используя механизм внимания. Кодировщик анализирует и кодирует информацию из входных данных, выделяя важные аспекты и создавая их векторные представления.

2) Декодировщик (Decoder): после получения векторного представления входных данных от кодировщика, декодер обрабатывает эту информацию и генерирует выходную последовательность. Декодировщик также задействует механизм внимания для сосредоточения на различных частях входных данных в процессе генерации выходной последовательности.

В современных исследованиях по обработке естественного языка широко используют архитектура трансформеров [12]. Она нашла применение в задачах машинного перевода, генерации текста и других областях. Благодаря высокой эффективности и возможности параллельной обработки данных, трансформеры стали одним из наиболее популярных подходов в современных нейронных сетях для работы с последовательными данными.

1.3.4 Сверточные рекурентные нейронные сети

CRNN (Convolutional Recurrent Neural Network) представляет собой архитектуру нейронной сети, которая объединяет в себе сверточные (Convolutional) и рекуррентные (Recurrent) слои для эффективного распознавания текста на изображениях. [9, 10, 13]

В начале процесса работы CRNN применяет сверточные слои для извлечения признаков из входного изображения. Затем извлеченные признаки подаются на вход рекуррентным слоям, которые способны учитывать контекст и последовательность символов. Рекуррентные слои помогают модели понимать структуру текста и предсказывать следующий символ в последовательности. Пример такой модели иллюстрируется на рисунке 9.

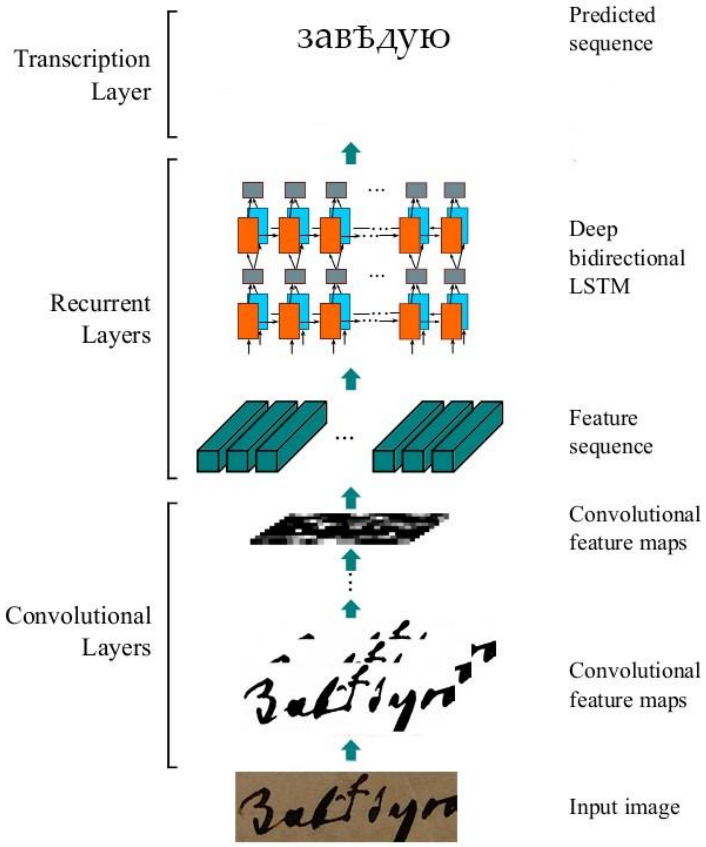


Рисунок 9 – Пример архитектуры CRNN

CRNN состоит из трех основных частей: сверточные слои, рекуррентные слои и транскрипционный слой, организованные в порядке последовательности обработки данных от нижнего к верхнему. На начальном уровне CRNN сверточные слои автоматически извлекают последовательность признаков из каждого входного изображения. Рекуррентная сеть, построенная поверх сверточных слоев, используется для предсказания каждого кадра последовательности признаков. Транскрипционный слой в верхней части CRNN превращает предсказания на уровне кадров в последовательность меток. Несмотря на то, что CRNN объединяет различные архитектуры сетей, такие как сверточные и рекуррентные, их можно обучать совместно, используя общую функцию потерь.

К данному моменту в публичном доступе не было найдено рабочих способов для корректного и точного распознания рукописных российских текстов начала 20 века, поэтому было решено самостоятельно написать нейронную сеть для выполнения поставленной задачи.

2 МАТЕРИАЛЫ И МЕТОДЫ

2.1 Анализируемые датасеты

2.1.1 Датасет CAPTCHA

Набор данных представляет собой набор вариантов Капчи, состоящей из пяти символов [7]. Объем датасета не велик (1,070 изображений), однако этот набор данных содержит перекрытые и зачеркнутые символы, что делает его удобным для проведения экспериментов. Пример изображения из этого набора данных можно увидеть на рисунке 10.

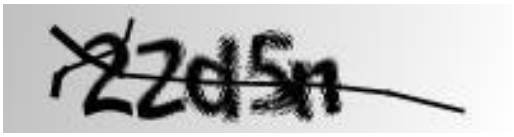


Рисунок 10 – Изображение из датасета CAPTCHA

2.1.2 Датасет Digital Peter

Это датасет рукописных документов Петра I периода 1709-1713, который предоставил ПАО Сбербанк на международном соревнование AI Contest в рамках конференции Artificial Intelligence Journey в 2020 году [6]. Датасет насчитывает 9,694 изображений текста, с его переводом в текстовом формате. Данный датасет был выбран по той причине, что он максимально близок к теме диплома и все данные в нем уже размечены. Пример изображения из датасета можно увидеть на рисунке 11.

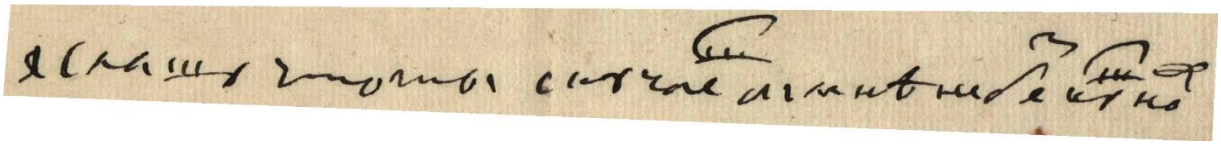


Рисунок 11 – Изображение из датасета Digital Peter

(перевод – я слышу что ты скучаеш а і мнѣ не безкушно ж)

2.1.3 Письма начала 20 века

Данный датасет предоставляет из себя сборник писем (90 страниц) 1919 года. Реформа письма была в 1918 году, но автор писем использует все убранные реформой символы (такие как ѣ - ять, і – и десятеричная, ѳ - фита, ѵ – ижица). Но буквы в нем более различимые чем в датасете Петра I. Главная проблема заключается в том, что автор часто менял стиль написания той или иной буквы, что вызывало некоторые сложности при разметке. Пример изображения из датасета можно увидеть на рисунке 12.

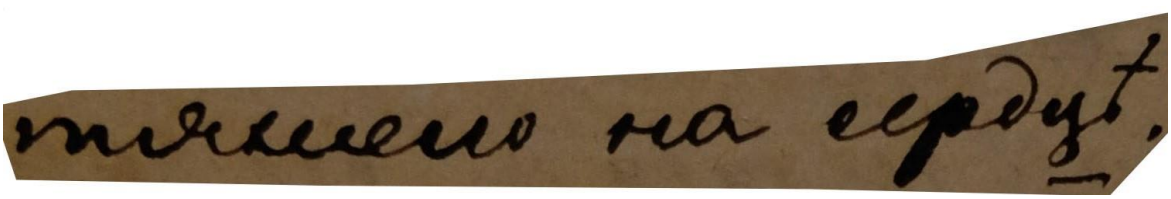


Рисунок 12 – Изображения из датасета писем 1919 года (перевод – тяжело на сердцѣ)

2.2 Предобработка данных

2.2.1 Аугментация данных

В области обработки изображений методы аугментации представляют собой приемы модификации исходных изображений для расширения набора данных, используемых при обучении моделей машинного обучения. Эти техники играют важную роль в повышении эффективности и обобщающей способности моделей. Существует несколько причин, по которым аугментация – это неотъемлемая часть процесса обучения нейронной сети:

1) Расширение обучающего набора данных является важной составляющей в машинном обучении. Аугментации помогают создавать большее разнообразие изображений из ограниченного объема данных, что особенно важно, если количество доступных обучающих примеров ограничено.

2) Увеличение устойчивости модели также играет важную роль. Применение различных методов аугментации помогает модели учиться на разнообразных данных и делает ее более устойчивой к изменениям в тестовых данных. Например, вращение, смещение и изменение размера изображения способствуют лучшей обобщенной способности модели к объектам на изображениях.

3) Использование аугментаций помогает предотвратить переобучение модели путем увеличения разнообразия данных, используемых для обучения, что в конечном итоге способствует улучшению способности модели к обобщению на новых данных.

4) Аугментации могут увеличить инвариантность модели к различным изменениям в данных, таким как изменения освещения, угла обзора или масштабирования.

5) Применение методов аугментации способствует улучшению реализма моделей, поскольку позволяет обучаться на данных, более точно отражающих реальный мир.

Поскольку работа связана с обработкой текстовых изображений, использование аугментаций, таких как отражение, изменение цветовой гаммы, вращение и увеличение резкости, может быть нецелесообразным или даже негативно сказаться на конечных результатах. При обработке текста аугментация изменение цветовой гаммы, часто бесполезна, поскольку текст не зависит от цвета и изменения в цветовой палитре, а вращение бессмысленно, поскольку все изображения, находящиеся в датасете либо заранее выравнены, либо дополнительно проходят процедуру выравнивания. Поэтому целесообразно использовать следующие методы аугментации: смещение, изменение масштаба и добавление шума. Работа с рукописным текстом представляет собой сложную задачу, и иногда даже для человека бывает трудно разобрать написанное. Поэтому важно следить за тем, чтобы при применении

аугментаций сохранить информативную часть изображений и обеспечить четкость и различимость текста.

Также после анализа рукописных писем, был сделан вывод, что одной из важных аугментаций является добавление имитаций исправлений, зачеркиваний и других изменений текста, к примеру как сделали авторы статьи «Mixup Without Hesitation» [14].

Человек по своей натуре часто может допускать различные ошибки на письме или делать это целенаправленно, поэтому исправления и зачеркивания могут помочь модели научиться распознавать текст, который содержит ошибки или имеет некоторые дефекты. Это может сделать модель более устойчивой к различным вариациям текста и помочь ей лучше обобщать данные.

Для создания подобной аугментации были нарисованы 20 вариаций зачеркиваний и исправлений, 6 из которых представлены на рисунке 13.

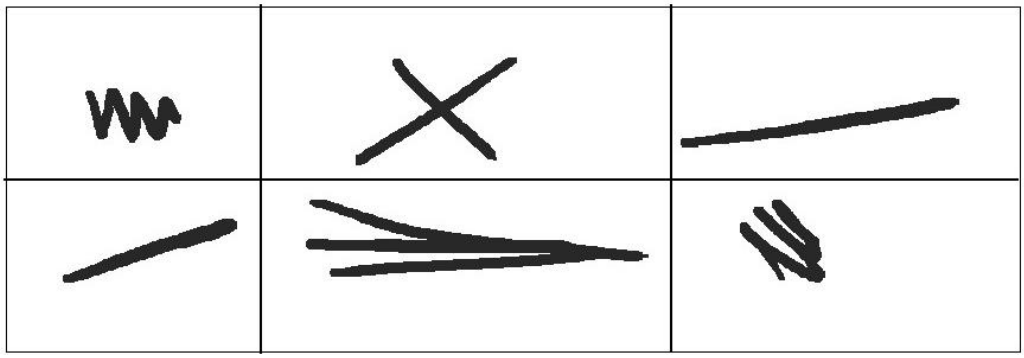


Рисунок 13 – Примеры аугментаций для имитации исправлений и зачеркиваний

Была проведена работа по созданию модуля, способного автоматически и правильно применять различные аугментации в разных положениях и объемах к изображениям слов. Работа алгоритма заключается в обнаружении контуров слова, определении области, в которой находится слово (с координатами верхнего левого угла), и затем применении аугментации в зависимости от формы слова и формы аугментации. Результат работы алгоритма представлен на рисунке 14.

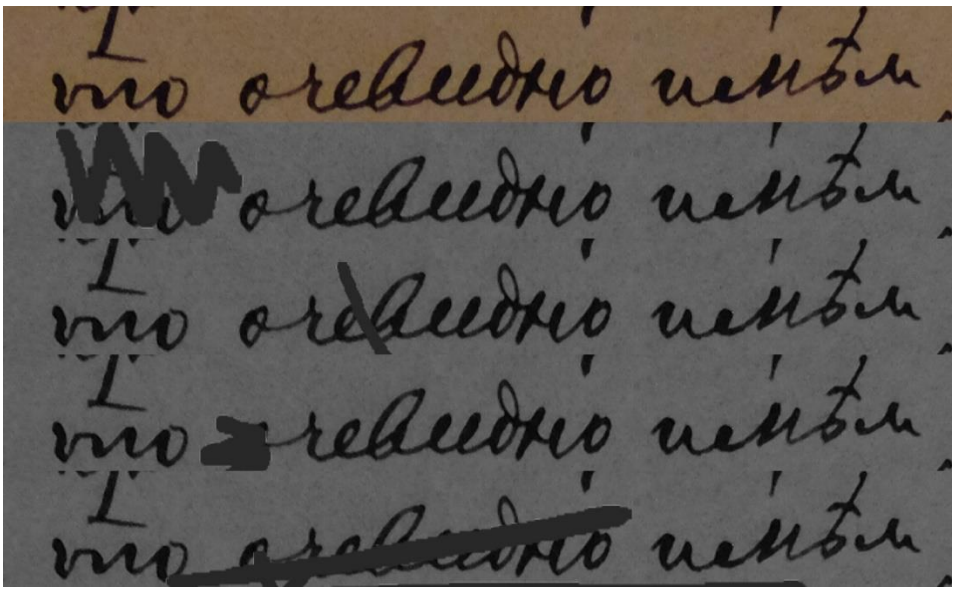


Рисунок 14 – Сверху изначальные данные,
ниже предоставлены варианты применения аугментаций

В языке Python доступно несколько модулей и библиотек для добавления аугментаций к изображениям и тексту. Некоторые из популярных вариантов включают:

1) Imgaug – мощная библиотека, предназначенная для аугментации изображений. Она предлагает разнообразные методы аугментации, такие как повороты, смещения, масштабирование, изменение яркости и контрастности, а также более сложные техники, включая аффинные трансформации и деформации.

2) Albumentations – еще одна популярная библиотека для аугментации изображений. Она отличается высокой производительностью и поддерживает широкий спектр методов аугментации, включая изменение яркости, контрастности, цветовых каналов, а также геометрические трансформации.

3) Augmentor – удобная и простая в использовании библиотека для аугментации изображений. Она позволяет создавать цепочки аугментаций с помощью предопределенных операций, таких как повороты, смещения, масштабирование и другие.

4) TorchVision – библиотека машинного обучения для компьютерного зрения, содержащая набор методов для аугментации изображений. Она предоставляет простые функции для изменения размера, поворотов, отражений и других преобразований.

Выбор был сделан в пользу Albumentations и TorchVision из-за своей простоты и удобства использования, также в данных модулях возможно использование собственных аугментаций и у них хорошая интеграция с фреймворком PyTorch, который впоследствии понадобится для написания нейронной сети.

2.2.2 Добавление синтетических данных

Внесение синтетических данных в датасет может привести к нескольким преимуществам и повысить эффективность модели машинного обучения:

1) Синтетические данные представляют различные вариации исходных данных, что помогает обучать модель на более широком спектре сценариев и улучшает ее способность к обобщению. [15]

2) При работе с несбалансированными данными добавление синтетических примеров для реже встречающихся классов может улучшить способность модели распознавать их, повышая точность классификации.

3) Введение разнообразных синтетических данных помогает уменьшить вероятность переобучения модели, так как модель обучают на более обобщенных и разнообразных примерах.

4) Расширение датасета с помощью синтетических данных позволяет модели обучаться эффективнее, что способствует достижению более высокой точности и эффективности.

В нашем случае можно сделать новые данные вручную, но это займет очень много времени, что очень непродуктивно. Поэтому для данной задачи можно использовать CTC Loss в нашей нейронной сети.

CTC Loss (Connectionist Temporal Classification Loss) — это функция потерь, которую часто используют в задачах распознавания речи и оптического распознавания символов (OCR). Она позволяет обучать модель на данных без явного выравнивания между входными данными и метками. CTC Loss учитывает возможность неоднозначности в соответствии между входной последовательностью и целевой меткой, а также учитывает возможные повторения и пропуски в распознавании.

CTC Loss работает путем вычисления вероятности соответствия входной последовательности определенной метке (например, символа) с учетом всех возможных выравниваний [17, 18, 19]. Затем функция потерь суммирует вероятности для всех возможных путей, соответствующих той же целевой метке. Цель состоит в том, чтобы минимизировать потери и добиться наилучшего соответствия между входными данными и целевыми метками, даже при наличии неопределенности в соответствии.

После того как модель сделает предсказание последовательности символов, можно использовать эту информацию для разделения входного изображения по этим символам, учитывая их размеры и расположение на изображении. Для этого необходимо использовать координаты точек соединения различных букв в слове [16].

Результат работы такого разбиения можно увидеть на рисунке 15.

Рисунок 15 – Собранное слово «привет» из других слов автора письма

2.3 Реализация алгоритма по поиску цепочек символов на изображениях с текстом

Помимо написания нейронной сети по распознаванию рукописного текста на изображении, необходимо написать алгоритм, который будет искать слова на переданном ему изображении и отправлять их в нейронную сеть для последующего распознания. Для решения данной задачи были выдвинуты следующие идеи:

1) Обучить нейросеть с архитектурой Yolov8n для детекции текста.
2) Воспользоваться модулем easyocr.
3) Применить морфологические свертки для поиска слов и последующей группировки их в строки.

Yolov8n (You Only Look Once Version 8n) – это версия алгоритма компьютерного зрения YOLO (You Look Only Once), который используется для обнаружения объектов на изображениях и видео. Yolov8n представляет собой совершенствование предыдущих версий алгоритма, направленное на улучшение точности и скорости обнаружения объектов.

Идея использования Yolov8n для детекции элементов текста на изображении не привела к положительным результатам. Обученная нейронная сеть не справлялась с поиском текста, что было предсказуемо в виду его специфичности и инвариантности (см. рисунок 16). Текст на изображениях имеет меньший размер, чем объекты, которые Yolov8n обычно обнаруживает. Это приводит к тому, что Yolov8n просто не обнаруживает/неправильно обрабатывает небольшой текст.

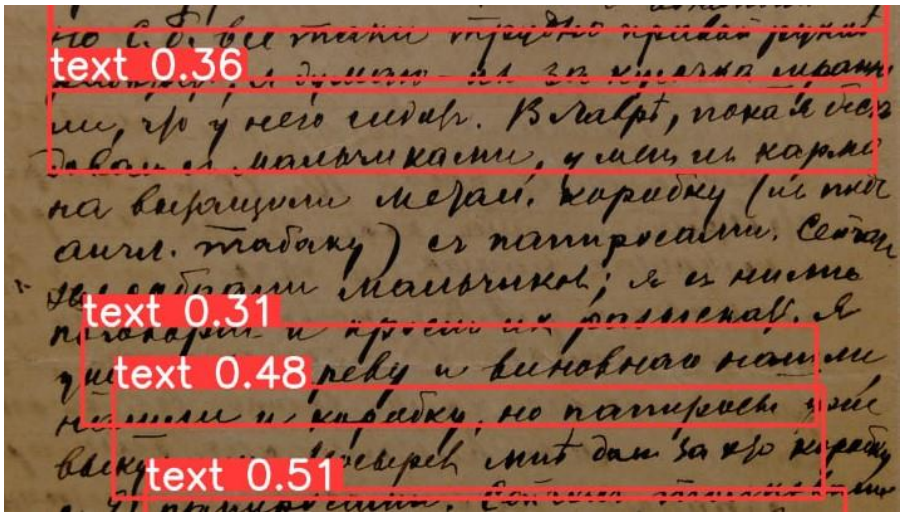


Рисунок 16 – Пример детекции текста моделью Yolov8n

Подход, подразумевающий работу с модулем easyocr [20], который применяют как систему оптического распознавания текста, оказался также неэффективным. Для данной задачи необходимо было лишь получить результаты детекции текста, поскольку текст рукописный и использует символы, которых уже нет в русском языке. Результат оказался лучше предыдущего, но не достаточным, пример разметки можно увидеть на рисунке 17.

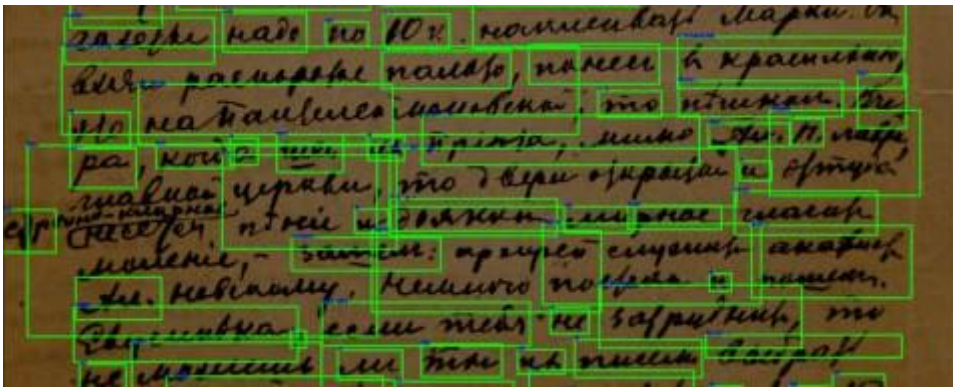


Рисунок 17 – Пример детекции текста модулем easyocr

Можно было и дообучить данную модель, о чем рассказывает автор статьи «Дообучение EasyOCR» [21], но было принято решение не тратить на это время.

Подход с применением морфологических сверток продемонстрировал лучший результат. Данный подход основан на следующей идее – изображение бинаризуют, затем применяют алгоритм по поиску контуров и обводят найденные объекты (впоследствии можно кластеризовать несколько объектов, находящихся на одной линии по оси y).

В нашем случае у изображений будут неравномерные яркость и контрастность, поэтому будет логично использовать адаптивную бинаризацию.

Адаптивная бинаризация – это метод обработки изображений, который позволяет автоматически определить порог бинаризации для каждого пикселя изображения на основе его локального окружения. В отличие от глобальной бинаризации, при которой один порог применяют ко всему изображению, адаптивная бинаризация учитывает изменения освещенности, контраста и шума в различных областях изображения.

Основная идея адаптивной бинаризации основана на пороге бинаризации, который вычисляют для каждого пикселя с учетом среднего значения яркости в небольшой окрестности этого пикселя. Таким образом, порог может быть настроен индивидуально для различных областей изображения, что позволяет лучше разделять объекты и фон при наличии неравномерной яркости и контрастности.

Применение адаптивной бинаризации может улучшить качество обработки изображений в различных областях, таких как распознавание текста, обнаружение объектов,

анализ медицинских изображений и многие другие области. Существуют различные методы адаптивной бинаризации, вот самые распространенные из них:

1) Метод Sauvola – этот метод учитывает среднее значение яркости пикселей и стандартное отклонение в небольшой области изображения.

2) Метод Niblack - он также использует среднее значение и стандартное отклонение, но с весовыми коэффициентами для адаптивного порога (threshold).

3) Метод Otsu – этот метод определяет порог бинаризации, минимизируя внутриклассовую дисперсию и максимизируя межклассовую дисперсию.

4) Метод Kittler – основан на оценке вероятности пикселя принадлежности к объекту и фону.

5) Метод Bredly-Roth – основан на идее вычисления глобального порога для каждого пикселя в рамке окна детекции.

Был проведен ряд экспериментов по бинаризации текста (см. рисунок 18) и его последующего распознавания. Для экспериментов было выбрано 3 письма 1919 года, после чего их бинаризовали 5 разными алгоритмами бинаризации (метод Sauvola, метод Niblack, метод Otsu, метод Kittler и метод Bredly-Roth), затем итоговые изображения загрузили в сервис автоматического распознания рукописного текста Transkribus и высчитали значения WER и CER. Самые лучшие результаты продемонстрировал метод Бредли-Рота. Поэтому было решено использовать данный алгоритм для бинаризации изображений. Результаты экспериментов отражены в таблице 1.

Таблица 1 – Результаты сравнения адгоритмов бинаризации на письмах 1919 года

	ДИ CER	ДИ WER
Sauvola	[0.177;0.262]	[0.511;0.696]
Niblack	[0.921;1.0]	[1.0;1.0]
Otsu	[0.186;0.294]	[0.556;0.674]
Kittler	[0.193;0.333]	[0.556;0.804]
Bredly-Roth	[0.168;0.219]	[0.511;0.649]

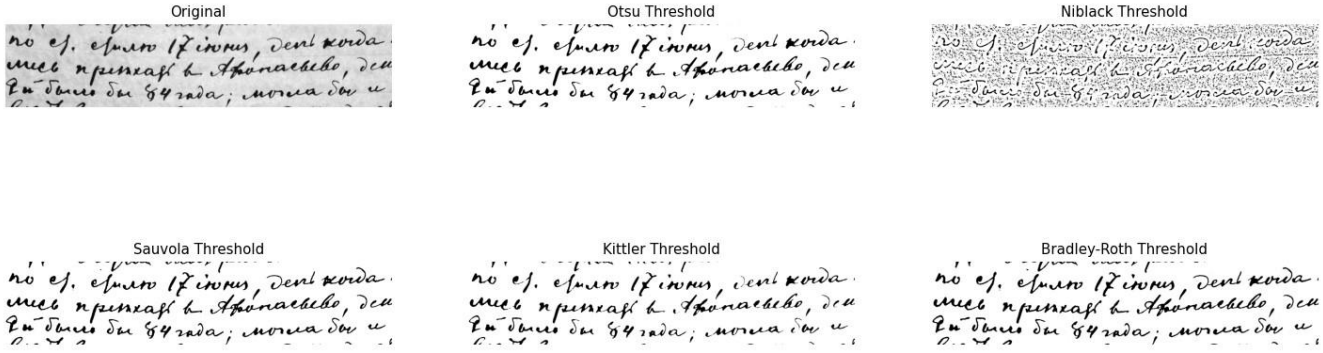


Рисунок 18 – Результаты бинаризации писем 1919 года

В результате была произведена адаптивная бинаризация алгоритмом Бредли-Рота [22], с применением морфологических сверток, после чего процесс бинаризации был ускорен при помощи онлайн-компилятора numba.

Numba – это библиотека для языка Python, которая позволяет ускорять выполнение кода за счет компиляции функций в машинный код. Библиотека numba работает путем преобразования Python-кода в оптимизированный машинный код, что позволяет выполнять операции быстрее, чем интерпретируемый язык Python. Это достигается благодаря использованию техник, таких как JIT (Just-In-Time) компиляция и векторизация, что повышает производительность кода, особенно при работе с массивами данных. В итоге, использование библиотеки numba позволяет значительно ускорить выполнение Python-скриптов, делая их более эффективными и быстрыми.

Как итог, были найдены контуры всех объектов на изображении и убраны самые маленькие и самые большие. Для всех оставшихся контуров были высчитаны параметры выделения объекта (bounding box). Были объединены все контуры, которые находились на близком расстоянии друг от друга. При помощи поиска центра масс каждого объекта удалось распределить их все по строчкам. Результат можно увидеть на рисунке 19.

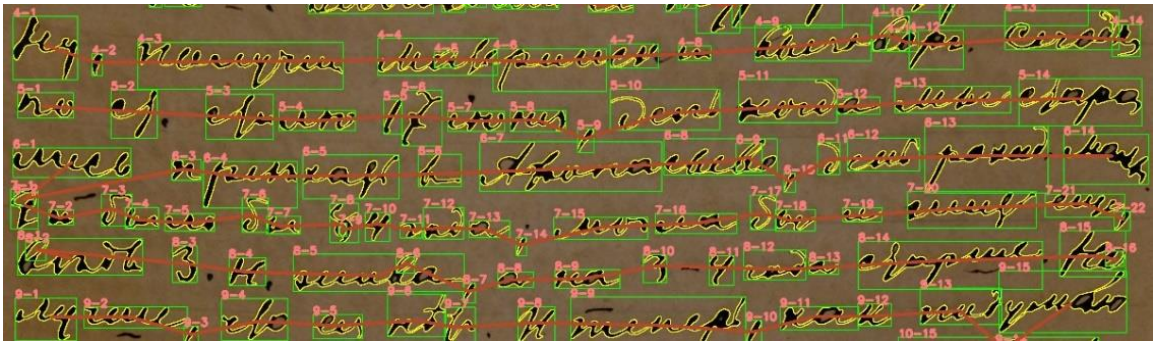


Рисунок 19 – Пример детекции текста с помощью морфологических сверток

2.4 Архитектуры моделей, используемых для решения, и общая стратегия обучения

2.4.1 Модель нейронной сети

Раннее в статье «StackMix and Blot Augmentations for Handwritten Text Recognition» [25] авторами была использована архитектура ResNet + BiLSTM для решения задачи распознания рукописных текстов на изображениях (ResNet – это архитектурн нейронной сети, в которой каждый слой принимает на вход не только данные из предыдущего слоя, но и "остаточную" информацию, которая обходит один или несколько слоев. Это позволяет более эффективно обучать глубокие нейронные сети, предотвращая проблему затухания градиентов и упрощая процесс обучения). По этой причине было решено в первую очередь попробовать данную архитектуру.

BiLSTM (Bidirectional Long Short-Term Memory) — это вид рекуррентной нейронной сети (RNN), который состоит из двух LSTM слоев: одного слоя, который обрабатывает последовательность в прямом направлении, и другого слоя, который обрабатывает последовательность в обратном направлении. [26, 30]

Таким образом, BiLSTM способен учитывать контекст как слева, так и справа от текущего элемента в последовательности данных, также модель хорошо подходит для работы с данными, упорядоченными в последовательности, такими как тексты, речь, временные ряды и другие. LSTM слои внутри BiLSTM обеспечивают способность обучаться на долгосрочные зависимости и предотвращать проблему затухания градиента, которая часто возникает в более простых рекуррентных сетях (см. рисунок 20).

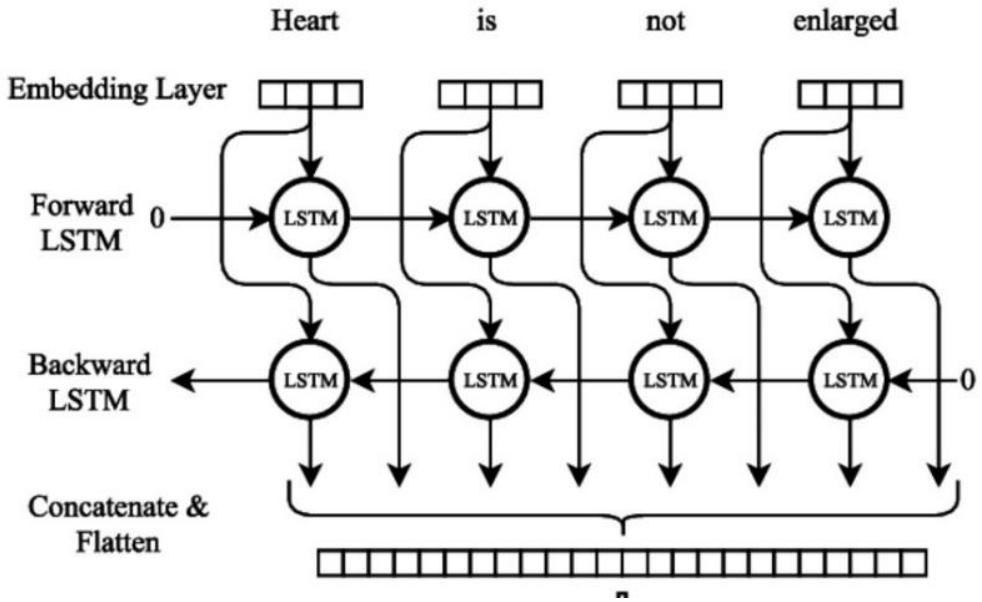


Рисунок 20 – Архитектура BiLSTM (взято из статьи Modelling Radiological Language with Bidirectional Long Short-Term Memory Networks)

Модель способна учитывать долгосрочные зависимости и контекст в последовательных данных. А модель ResNet является мощной сверточной архитектурой, которая хорошо справляется с извлечением признаков с изображений.

2.4.2 Выбор оптимизатора

Для того чтобы хорошо обучить модель необходимо выбрать пару важных компонентов, одним из которых является Оптимизатор.

Оптимизатор (Optimizer) – это алгоритм, используемый в машинном обучении для минимизации функции потерь путем обновления параметров модели в процессе обучения. Оптимизаторы играют важную роль в процессе обучения нейронных сетей, помогая находить оптимальные значения параметров модели для достижения наилучшей производительности на задаче.

В качестве оптимизатора выбор был между алгоритмами Adam и SGD:

1) Алгоритм Adam находит локально-оптимальное решение быстрее, поскольку он комбинирует преимущества адаптивного шага обучения и момента. Оптимизатор автоматически регулирует скорость обучения для каждого параметра, а также обычно хорошо работает на задачах с большим количеством данных и/или параметров, так как он способен эффективно обновлять веса модели.

2) SGD – это более простой оптимизатор, который может стабильнее и более предсказуем в ряде задач. Этот оптимизатор может быть эффективным на небольших наборах данных и помочь избежать переобучения.

Датасет DigitalPeter насчитывает свыше 9000 изображений текста, поэтому выбор был сделан в пользу Adam. Впоследствии было решено проверить и AdamW.

Оптимизатор AdamW – это вариант оптимизатора Adam, который вводит корректировку весов (weight decay) непосредственно в формулу обновления весов.

2.4.3 Планировщик

Для эффективного обучения нейронной сети также необходим планировщик.

В машинном обучении планировщик (schedular) — это инструмент, который управляет гиперпараметрами в процессе обучения модели. Планировщики позволяют динамически изменять параметры обучения, такие как скорость обучения, шаг обучения и другие, в зависимости от заданного расписания или условий обучения. Использование планировщиков помогает улучшить процесс обучения модели и повысить ее производительность.

В качестве scheduler было решено выбрать:

1) Планировщик OneCycleLR – основная идея данного планировщика заключается в том, чтобы изменять темп обучения в течение одного цикла обучения, увеличивая его до максимального значения, а затем постепенно уменьшая обратно. Изменение шага обучения в данном случае изображено на рисунке 21.

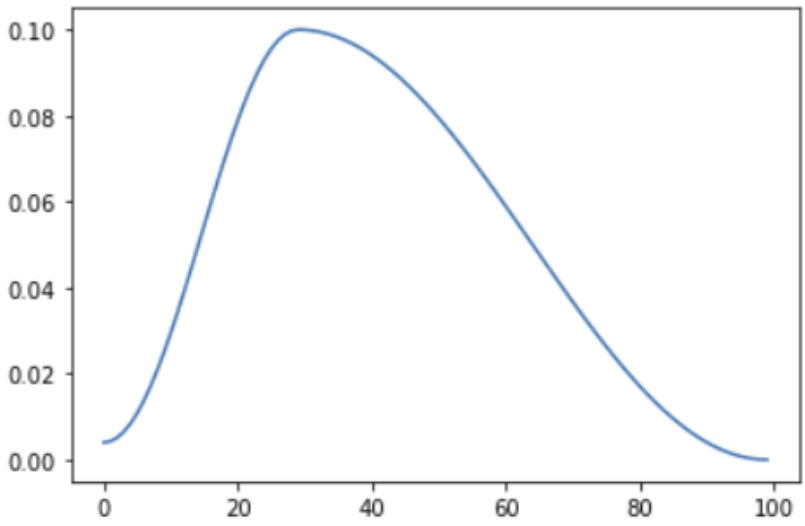


Рисунок 21 – График изменения шага обучения при использовании планировщика OneCycleLR (вертикальная ось – это размер шага обучения, горизонтальная ось – номер эпохи)

2) Планировщик CosineAnnealingWarmRestarts – основная идея данного планировщика заключается в том, чтобы изменять темп обучения по косинусоидальному закону с повторяемой активацией, что помогает модели быстрее сойтись к оптимальным значениям. Изменение шага обучения в данном случае изображено на рисунке 22.

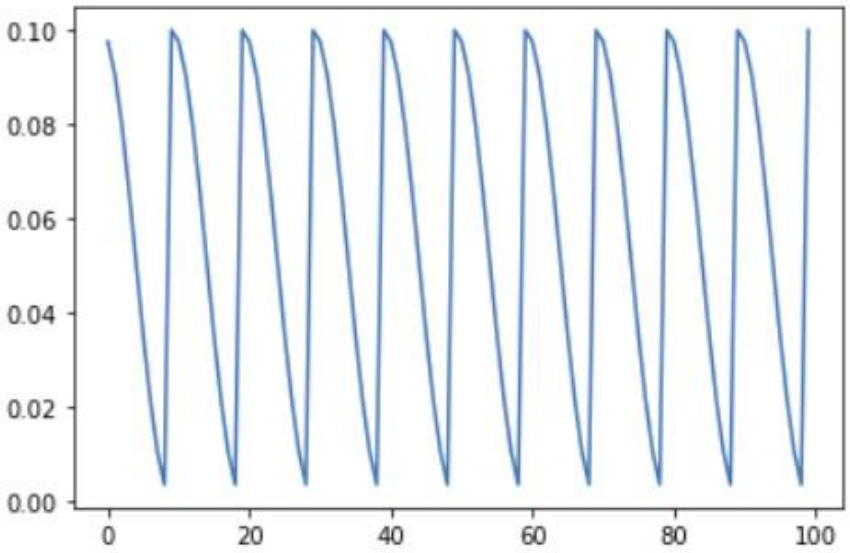


Рисунок 22 – График изменения шага обучения при использовании планировщика CosineAnnealingWarmRestarts (вертикальная ось – это размер шага обучения, горизонтальная ось – номер эпохи)

3) MultiplicativeLR - Основная идея данного планировщика заключается в том, чтобы умножать текущий темп обучения на заданный коэффициент каждую эпоху или каждый определенный шаг обучения. Изменение шага обучения в данном случаем изображено на рисунке 23.

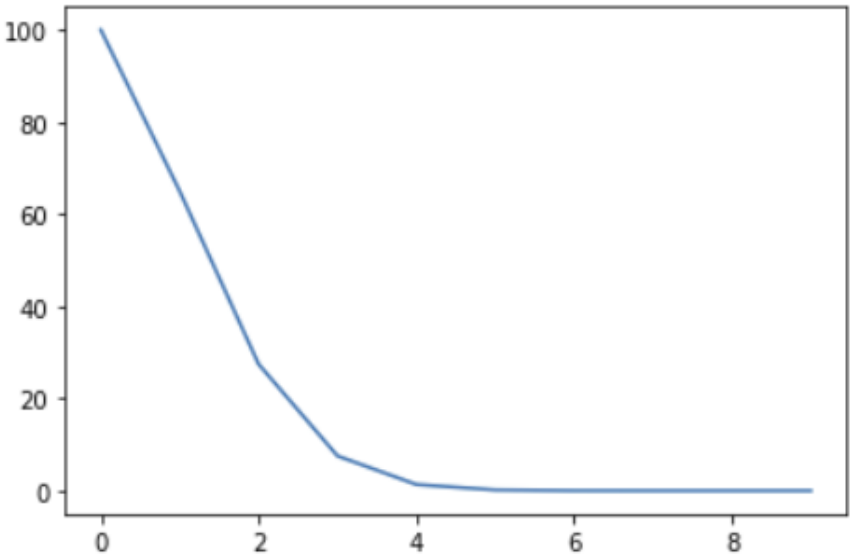


Рисунок 23 – График изменения шага обучения при использовании планировщика MultiplicativeLR (вертикальная ось – это размер шага обучения, горизонтальная ось – номер эпохи)

2.4.4 Функция потерь

Функция потерь (Loss function) – это самая важная компонента в процессе обучения нейронной сети, она оценивает расхождение между прогнозируемыми значениями модели и их реальными целевыми значениями. Цель функции потерь – минимизировать ошибку модели путем коррекции параметров в процессе обучения. Выбор подходящей функции потерь зависит от типа задачи (например, классификация, регрессия) и особенностей данных. Некоторые из наиболее распространенных функций ошибок включают в себя:

1) Функция Mean Squared Error (MSE): используют в задачах регрессии для измерения среднеквадратичного расхождения между прогнозами и фактическими значениями.

2) Функция Cross Entropy Loss: часто применяют в задачах классификации для оценки расхождения между предсказанными вероятностями классов и их истинными метками.

3) Функция Binary Cross Entropy Loss: аналогична Cross Entropy Loss, но используют в задачах бинарной классификации.

4) Функция Categorical Cross Entropy Loss: применяют в многоклассовой классификации, где каждый пример может быть отнесен к одному из нескольких классов.

5) Функция Connectionist Temporal Classification (CTC) Loss: используют в задачах распознавания последовательностей символов без явного соответствия между входом и выходом.

В процессе работы было необходимо решить задачу классификации, поэтому в качестве функции потерь использованы функции потерь – Cross Entropy Loss и CTC Loss.

2.4.4.1 Функция потерь Cross Entropy Loss

Функция потерь Cross Entropy Loss — это популярный выбор для задач классификации [27], так как он хорошо подходит для оценки ошибки между предсказанными вероятностями классов и истинными метками. В контексте распознавания рукописного текста, где каждая картинка представляет символ или слово, функция потерь Cross Entropy loss может быть эффективным выбором, помогая модели правильно классифицировать входные изображения на различные классы.

Для расчета функции потерь Cross Entropy Loss необходимо сравнить вероятности, предсказанные моделью для каждого класса, с реальными метками классов:

1) Сначала модель прогнозирует вероятности принадлежности каждому классу для каждого примера в данных. Эти вероятности обычно находятся в диапазоне от 0 до 1.

2) Затем для каждого примера сравнивают истинную метку класса с предсказанными вероятностями модели. С использованием функции Cross Entropy Loss вычисляют ошибку для каждого класса.

3) Ошибки для всех классов суммируют и делят на общее количество примеров. Полученное значение представляет собой Cross Entropy Loss для модели.

2.4.4.2 Функция потерь CTC Loss

Однако, в зависимости от конкретной задачи и особенностей датасета, также следует рассмотреть альтернативные функции потерь, которые учитывают специфику задачи распознавания рукописного текста. Например, функция потерь Connectionist Temporal Classification (CTC) Loss [17, 18, 19] может быть полезным выбором для задачи распознавания последовательностей символов без необходимости точного соответствия с изображением. Это позволяет обучать модель на данных с переменной длиной последовательностей и учитывать неопределенность в распознавании текста.

Например, необходимо распознать на изображении рукописное слово «Завѣдую». Для такого слова нейронная сеть выдает в качестве предсказания последовательность, например, «ЗЗ-ав-ѣдую» («-» обозначает blank - элемент, который необходим для разделения последовательностей). Затем идёт декодирование последовательности: сначала алгоритм убирает все повторяющиеся элементы, которые отделяют «blank» символы, после чего алгоритм удаляет пустые символы (см. рисунок 24).

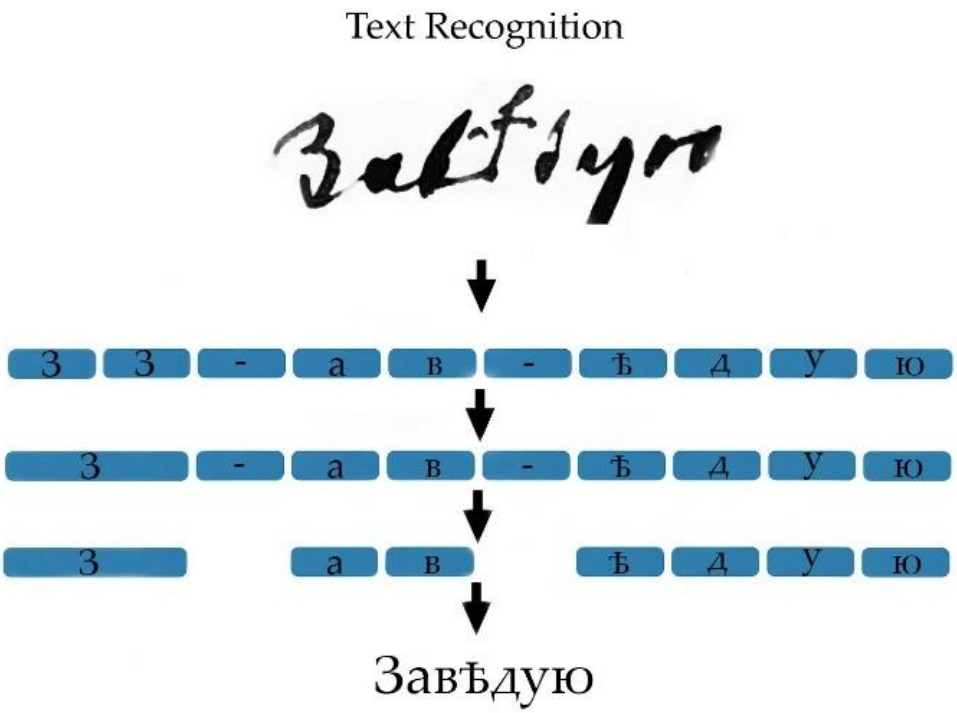


Рисунок 24 – Пример декодирования выданной нейронной сетью последовательности

В качестве примера предскажем первую букву «З». Модель декодирует три позиции, каждая из которых даёт вероятность нахождения там всех букв (в нашем случае это 7 букв + blank), это будет матрица 3x8 (3 позиции и 8 элементов), после чего алгоритм выбирает элемент с наибольшей вероятностью (см. рисунок 25).

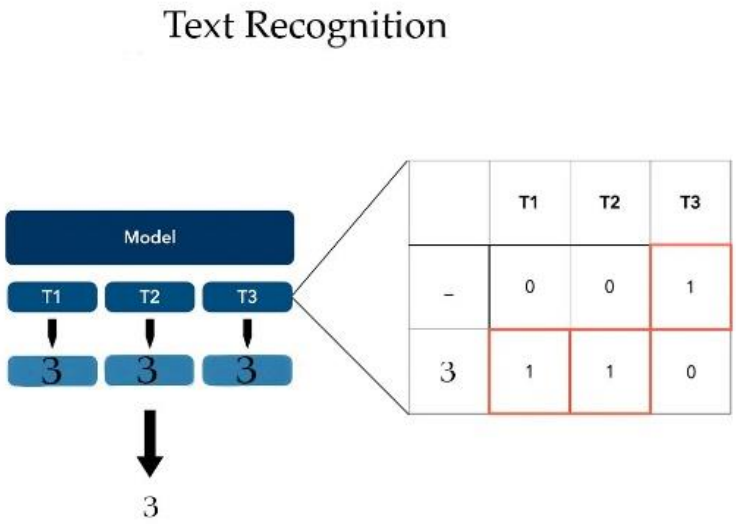


Рисунок 25 – Декодирование выходного результата функции потерь CTC Loss для слова «Завѣдую»

Существует много комбинаций, которые могут дать подобный результат (результат приводит к букве «З»). Модель необходимо научить воспроизводить один из подобных

результатов. Логичный способ решения проблемы – пронумеровать все подобные комбинации и посчитать функцию потерь для каждой из таких комбинаций (см. рисунок 26).

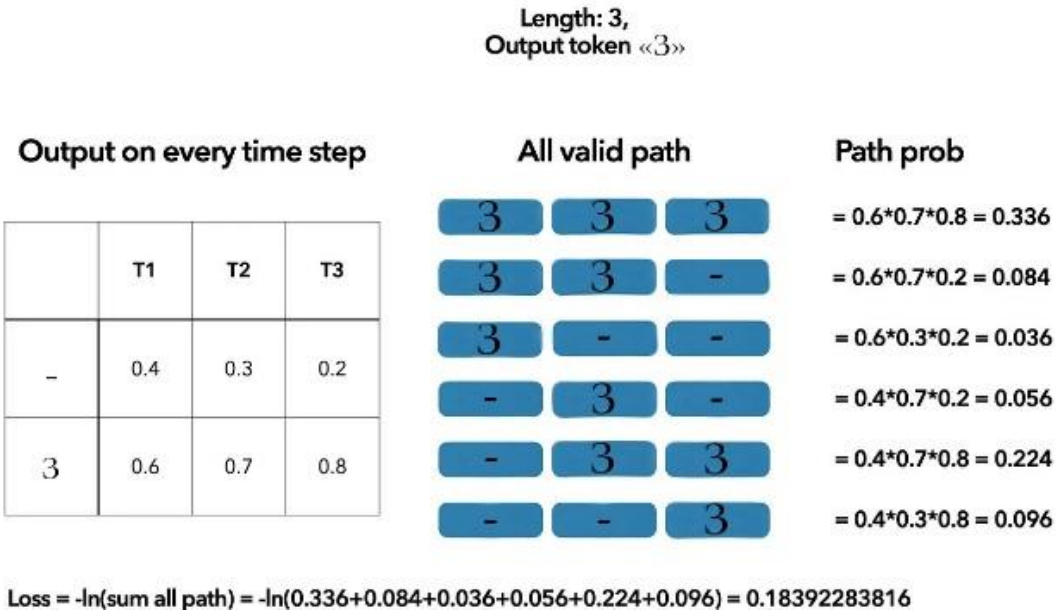


Рисунок 26 – Вычисление функции потерь для каждой из подходящих комбинаций

Алгоритм реализации функции потерь CTC Loss может приводить к следующей проблеме: В случае, когда элементов будет больше и размер последовательности будет больше, то и таких комбинаций будет намного больше. Эффективность алгоритма будет низкой на большом наборе данных.

Для повышения эффективности функции потерь CTC Loss использует метод динамического программирования (см. рисунок 27).

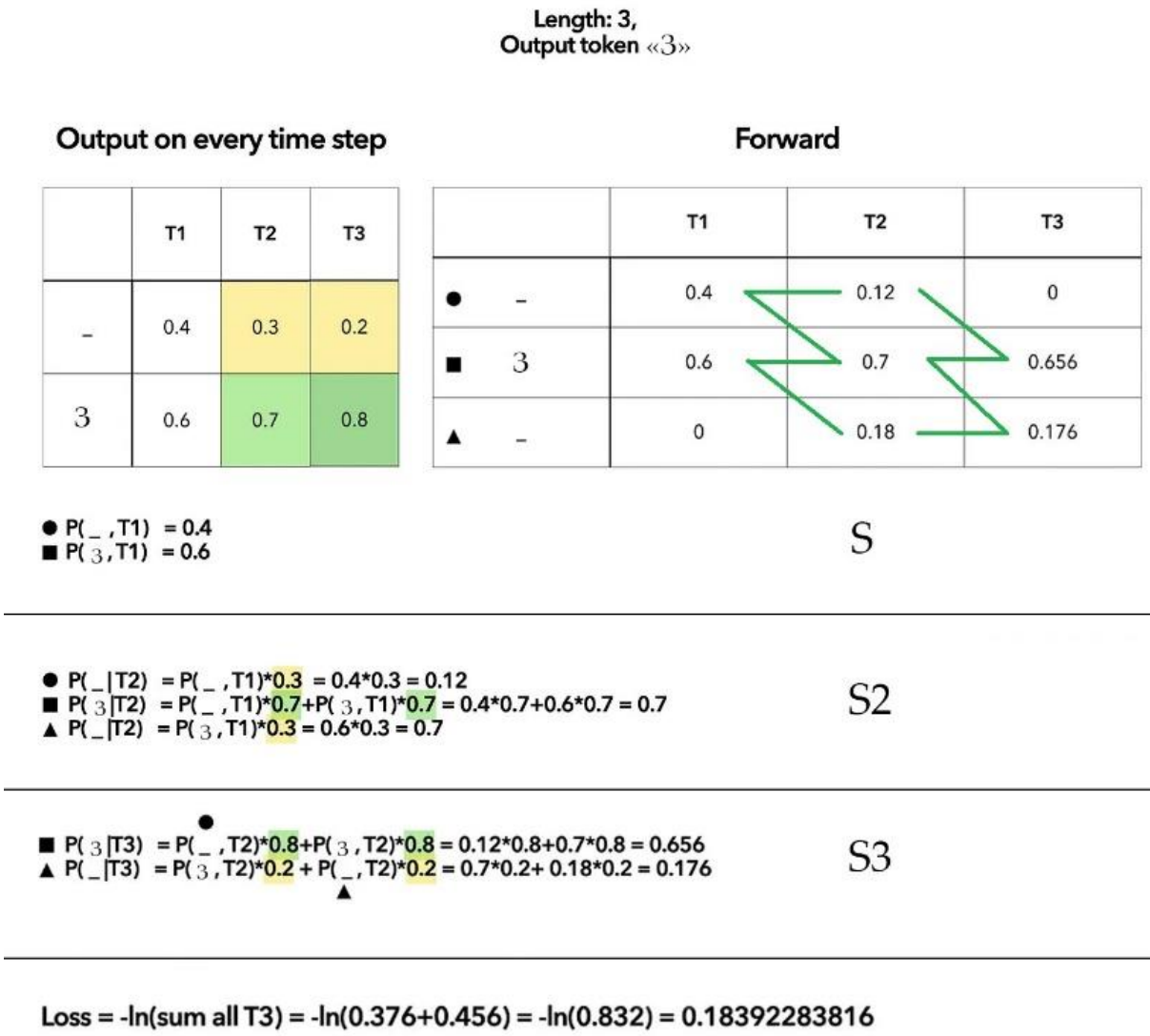


Рисунок 27 – Прямой проход CTC Loss,

где P – это вероятность, Sn – это номер шага алгоритма,

Tn – это временной промежуток n, «-» – это обозначение пустого символа.

Сначала алгоритм записывает все выходные вероятности для каждого элемента на каждом шаге алгоритма и делает отдельную таблицу. В первый момент времени T1 (см. рисунок 27), алгоритм дает каждому элементу соответствующую ему вероятность из левой таблицы (см. рисунок 27). Реализацию алгоритма возможно начать либо с первого пустого символа, либо с буквы «З», иначе не удастся получить искомый результат. Затем алгоритм вычисляет T2 (см. рисунок 27), используя значение T1 (см. рисунок 27), чтобы узнать в какие ячейки T2 (см. рисунок 27) можно попасть из ячеек столбца T1 (см. рисунок 27), при этом учитывая искомый результат. По итогу алгоритм вычисляет значение T3 (см. рисунок 27), используя результат T2 (см. рисунок 27). Если два предыдущих символа, полученной цепочки пустые, то из такой позиции можно попасть лишь в букву «З». В результате, вычисляют функцию потерь.

Алгоритм обратного пути реализуют аналогично алгоритму прямого пути, но в обратном направлении: начало будет в нижней ячейке столбца T3 (см. рисунок 27), а конец в верхней ячейке столбца T1 (см. рисунок 27).

2.5 Показатели эффективности

2.5.1 Метрики CER и WER

Для оценки модели необходима определенная метрика. Возможно рассчитать, какой процент слов прогнозирует нейронная сеть, но даже при единичной ошибке прогноза (например, вместо буквы «а» модель предсказывает букву «о») результат предсказания последовательности символов будет некорректным. В подобных случаях целесообразно использовать расстояния Левинштейна – данная метрика показывает степень сходства между двумя строковыми последовательностями, то есть она вычисляет, какой кратчайший путь из цепочки замен и удалений надо пройти первой последовательности, чтобы быть эквивалентной второй последовательности. Такую метрику называют CER (Character Error Rate):

$$CER = \frac{I+D+S}{N} \quad (1)$$

где S - количество замен, D – количество удалений, I - число вставок, N — символов в исходной строке.

А если смотреть на число замен в последовательности не по символам, а по словам, то возможно рассчитать еще одну метрику – WER (Word Error Rate). [28]

Чем ближе CER к 0, тем лучше алгоритм умеет различать символы на изображении, а чем ближе WER к 0, тем лучше алгоритм умеет распознавать слова на изображении.

2.5.2 Матрица ошибок

Метрики для оценки эффективности модели могут быть основаны на матрице ошибок (confusion matrix) для каждого из символов. Подобный подход позволяет получить более точный результат – конкретную информацию о том, какие символы у нас хуже всего удалось распознать. Тогда для каждого подобного случая у нас будет стандартный набор метрик:

1) ROC-кривая (Receiver Operating Characteristic curve) – это график, который отображает зависимость между долей верно классифицированных положительных примеров (True Positive Rate) и долей ложно классифицированных отрицательных примеров (False Positive Rate) при варьировании порога классификации.

2) AUC (Area Under the Curve) – это показатель, используемый для оценки качества модели классификации на основе ее ROC-кривой. AUC представляет собой площадь под ROC-

кривой и может принимать значения от 0 до 1. Чем ближе значение AUC к 1, тем модель лучше способна различать классы. Если AUC равен 0.5, это означает, что модель классифицирует случайно, если AUC менее 0.5 это значит, что модель склонна к обратной классификации (предсказывает всё наоборот), и ее предсказания нужно пересмотреть или инвертировать.

3) Полнота (Recall)/Чувствительность (Sensitivity)/True Positive Rate (TPR) – это метрика в области классификации, которая измеряет способность модели правильно классифицировать положительные примеры. Sensitivity выражает долю истинно положительных примеров, которые были корректно определены моделью. Формула для вычисления Sensitivity выглядит следующим образом:

$$Recall(TPR) = \frac{TP}{TP+FN} \quad (2)$$

где: TP (True Positives) - количество истинно положительных примеров, которые были правильно классифицированы как положительные; FN (False Negatives) - количество ложно отрицательных примеров, которые были неправильно классифицированы как отрицательные.

4) Специфичность (Specificity) – это метрика в области классификации, которая измеряет способность модели правильно классифицировать отрицательные примеры. Specificity выражает долю истинно отрицательных примеров, которые были корректно определены моделью. Формула для вычисления Specificity выглядит следующим образом:

$$Specificity = \frac{TN}{TN+FP} \quad (3)$$

где: TN (True Negatives) - количество истинно отрицательных примеров, которые были правильно классифицированы как отрицательные; FP (False Positives) - количество ложно положительных примеров, которые были неправильно классифицированы.

5) Точность (Precision) в контексте классификации – это метрика, оценивающая долю корректно предсказанных положительных классов от общего количества предсказанных положительных классов. Формула для расчета Precision выглядит следующим образом:

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

где, TP (True Positives) - количество верно предсказанных положительных примеров, которые были правильно классифицированы как положительные; FP (False Positives) - количество

неверно предсказанных положительных примеров, которые были неправильно классифицированы как положительные.

6) F1 Score – это гармоническое среднее между Precision и Recall (полнотой), которое используется в качестве метрики для оценки качества моделей классификации. F1 мера стремится балансировать между Precision и Recall, учитывая как точность, так и полноту модели при классификации данных. Формула для расчета F1 меры выглядит следующим образом:

$$F1\ score\ = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (5)$$

7) Достоверность (Accuracy) – это метрика, которая измеряет долю правильно классифицированных примеров от общего числа примеров в наборе данных. Формула для расчета Accuracy выглядит следующим образом:

$$Accuracy\ = \frac{TP+TN}{TP+FP+TN+FN} \quad (6)$$

где: TP (True Positives) - количество верно предсказанных положительных примеров; TN (True Negatives) - количество верно предсказанных отрицательных примеров; FP (False Positives) - количество неверно предсказанных положительных примеров; FN (False Negatives) - количество неверно предсказанных отрицательных примеров.

3 РЕЗУЛЬТАТЫ

Для решения задачи была использована архитектура BiLSTM + ResNet34 (ResNet34 отличается от других архитектур ResNet, таких как ResNet18, ResNet50 и т. д., в первую очередь, количеством слоев и глубиной сети, цифра на конце названия архитектуры означает количество слоев в нейронной сети). Первый слой ResNet был изменен на последовательность:

Conv2d(in_channels=3, out_channels=64, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3)) – это сверточный слой. При операции свертки применяется фильтр (ядро) к входным данным путем перемещения его по всей области входа. Произведение значений входных данных и значений фильтра суммируют, формируя выходной пиксель в выходном изображении. Параметры слоя представлены ниже:

1) In_channels – количество каналов входного изображения.
2) Out_channels – количество каналов после применения сверточного слоя.
3) Размер ядра (kernel_size) – это размер матрицы, используемой для применения операции свертки к изображению.
4) Шаг (stride) – это расстояние между применениями ядра свертки к изображению.
5) Заполнение (padding) – это добавление пустого пространства вокруг границ изображения перед применением сверточного слоя.

BatchNorm2d(num_features=64, eps=$1.0*10^{-5}$, momentum=0.1) представляет собой слой, который применяют для регуляризации модели в машинном обучении и помогает избежать переобучения (переобучение возникает, когда модель слишком точно подстраивается под обучающие данные, вместо того чтобы обобщать закономерности на новых, неизвестных данных). Основные параметры слоя Batch Normalization:

1) num_features – это количество каналов у входного объекта.
2) eps - это число, добавляемое к знаменателю, чтобы избежать деления на ноль.
3) momentum - параметр, который позволяет балансировать скорость обучения и стабильность модели в процессе обучения, влияя на скорость сходимости алгоритма и степень учета новых данных при обновлении статистики слоя.

ReLU(inplace=True) – это слой активации, где параметр inplace указывает, будет ли при выполнении операции создаваться новый объект или операция будет выполняться на месте.

MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False) – это слой, который используют для уменьшения размерности входных дынных, а также он способен извлекать ключевые признаки из входных данных. Параметры слоя представлены ниже:

1) Kernel_size – размер ядра, которым проходят вдоль изображения.
2) Stride – шаг с которым ядро свертки проходит по изображению.
3) Padding – толщина пустого пространства генерируемого вокруг изображения.

4) Dilation – это расстояние между ячейками ядра, которое проходит вдоль изображения во время операции свертки
5) Ceil_mode – параметр который говорит, что числа выходного значения будут округлять алгоритмом ceil (ceil_mode=True) или floor (ceil_mode=False).

У первого слоя ResNet34 параметр Stride делают равным 2 с целью уменьшения размера изображения и увеличения скорости обучения. Это позволяет уменьшить количество пикселей в выходных данных, сократить вычислительную сложность модели и ускорить процесс обучения. Для задачи распознания рукописного текста было решено установить параметр Stride равным 1, так как это, вероятно, повысит точность модели за счет более детального рассмотрения данных. Также были использованы три первых блока из ResNet-34, где каждый блок ResNet (см. рисунок 28) состоит из 3, 4 и 6 остаточных блоков с выходными слоями 64, 128 и 256. В начале и конце нейронной сети стоит AdaptiveAvgPool2d, который стандартизирует размеры карт признаков (feature maps), что помогает улучшить процесс обучения и уменьшает вероятность переобучение.

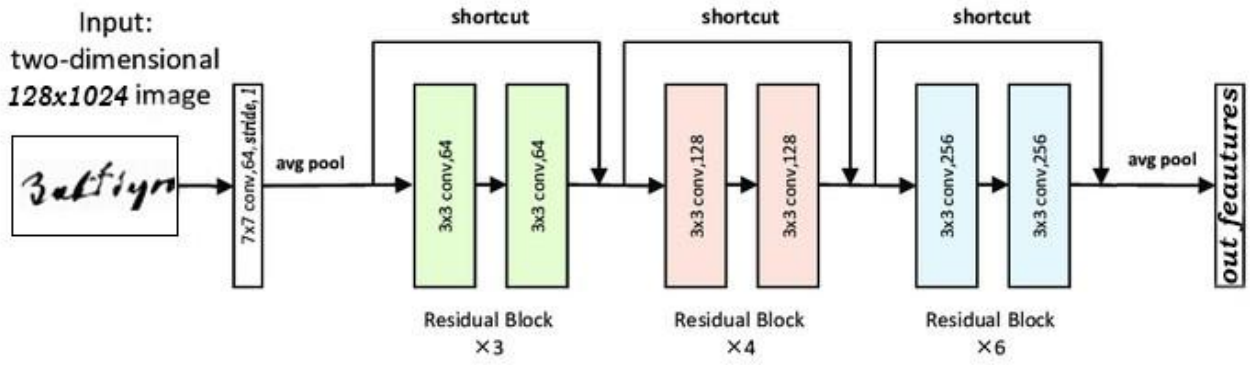


Рисунок 28 – Измененная архитектура ResNet34 под условия поставленной задачи

Полученные признаки идут на вход трех BiLSTM-слоев для обработки последовательностей признаков. Для классификации используют два полностью связанных слоя с функцией активации GELU (Gaussian Error Linear Units – функция активации, которая обеспечивает более плавное и непрерывное изменение значений активации в нейронной сети, это позволяет избежать проблемы затухания градиентов) и слоем Dropout (слой нейронной сети, который применяется для борьбы с переобучением модели) между ними (см. рисунок 29).

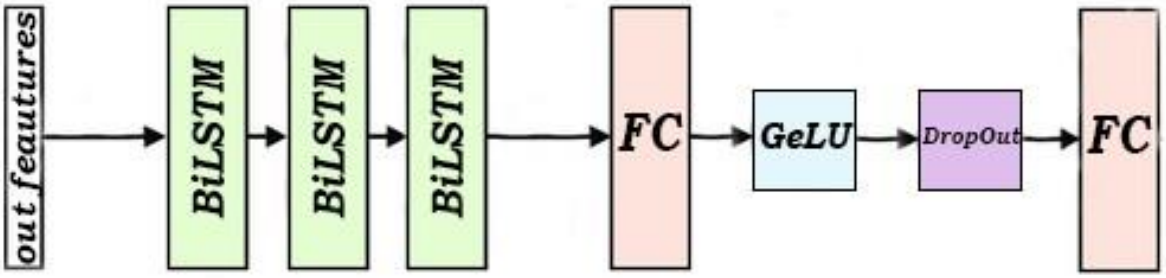


Рисунок 29 – Вторая часть нейронной сети, которая отвечает за классификацию полученных с изображения признаков

В качестве функции потерь была использована функция Cross Entropy Loss, а в качестве оптимизатора был взят оптимизатор Adam. Результаты, полученные с использованием данной архитектуры без дополнительных модификаций и с данным набором параметров инструментов для обучения, отображены в строке 1 таблицы 2.

Предыдущий эксперимент дал неудовлетворительные результаты (см. строка 1, таблица 2), поэтому функция ошибок Cross Entropy Loss была заменена функцией ошибок CTC Loss, а вместо оптимизатора Adam был выбран оптимизатор AdamW. Изменение тактики обучения не дало ожидаемого результата (см. строка 2, таблица 2), поэтому было решено изменить архитектуру нейронной сети.

Вместо ResNet была написана собственная CNN для получения карт признаков из изображения, архитектура представлена ниже:

Conv2d(3, 256, kernel_size=(9, 9), stride=(1, 1), padding=(1, 1))

ReLU()

BatchNorm2d(256, eps=$1.0*10^{-5}$, momentum=0.1)

MaxPool2d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)

Conv2d(256, 256, kernel_size=(4, 3), stride=(1, 1), padding=(1, 1))

ReLU()

BatchNorm2d(256, eps=1e-05, momentum=0.1)

Эта нейронная сеть выполняет операции свертки, активации, нормализации и пулинга для обработки входных изображений. Полученные признаки проходят через линейный слой, который уменьшает количество каналов, после чего полученные признаки идут на вход одной BiLSTM, для классификации используют линейный слой. Архитектура нейронной сети представлена на рисунке 30.

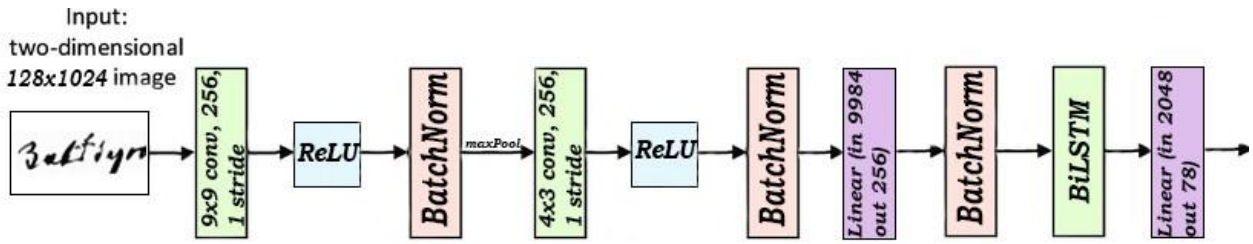


Рисунок 30 – Новая архитектура нейронной сети по распознанию рукописных текстов на изображении

Изменение архитектуры с ResNet34 на авторскую помогло эффективно обучить нейронную сеть. Стратегия обучения осталась без изменения – CTC Loss в качестве функции потерь и AdamW в качестве оптимизатора. Результаты, полученные с использованием новой архитектуры без дополнительных модификаций и данным набором параметров инструментов для обучения, отображены в строке 3 таблицы 2 и в таблице 3. Краткая таблица ошибок представлена на Рисунке 31.

Дальнейшие изменения траектория обучения (добавление планировщика добавление аугментаций) не помогли улучшить полученный результат (см. строка 4, таблица 2).

Таблица 2 – Результаты проверенных алгоритмов (в строках – номера экспериментов, в столбцах – названия критериев оценки эффективности и названия датасетов)

	Digital Peter		Письма 1919 года	
	ДИ CER	ДИ WER	ДИ CER	ДИ WER
1	[0.563;0.893]	[0.746;0.985]	[0.631;0.848]	[0.782;0.973]
2	[0.481;0.665]	[0.631;0.869]	[0.514;0.713]	[0.702;0.923]
3	[0.041;0.094]	[0.178;0.263]	[0.055;0.083]	[0.183;0.235]
4	[0.045;0.109]	[0.201;0.292]	[0.059;0.102]	[0.216;0.267]

Для более точного сравнения ДИ был построен график дисперанализа (график дисперсного анализа, или ANOVA – Analysis of Variance, является статистическим методом, используемым для сравнения средних значений между тремя или более группами данных. График дисперсного анализа определяет, есть ли статистически значимые различия между средними значениями групп или нет). График на рисунке 32 позволяет сделать следующие выводы:

1) Смена функции ошибки на CTC Loss повысила эффективность детекции

2) Смена архитектуры сети на BiLSTM + авторская CNN снизила число ошибок в три раза.

3) Добавление аугментаций и планировщика не дало никаких улучшений

	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	ѣ	–	⊕	⊗
а	1287	82	174	61	187	160	39	80	153	18	218	192	164	279	241	70	179	171	303	68	21	33	39	56	48	2	44	37	8	1	27	47	49	0	0	0
б	82	323	34	13	42	74	7	16	53	2	17	48	26	48	152	21	48	24	49	69	5	5	1	17	6	1	13	37	3	1	11	16	15	0	0	0
в	174	34	633	23	63	152	17	39	80	12	52	57	44	63	239	43	68	99	121	45	11	12	5	18	23	1	66	63	5	1	13	52	39	0	0	0
г	61	13	23	218	49	57	5	14	35	3	14	32	11	34	172	11	36	27	31	31	4	0	1	8	5	1	10	6	1	0	1	10	11	0	0	0
д	187	42	63	49	545	158	28	30	85	13	40	74	44	99	226	32	69	62	96	94	13	13	2	12	10	1	21	33	3	0	27	34	69	0	0	0
е	160	74	152	57	158	1201	104	69	129	36	77	179	139	312	237	114	210	143	209	83	20	17	18	80	50	7	51	47	4	0	20	57	51	0	0	0
ж	39	7	17	5	28	104	225	3	32	7	20	25	24	26	77	11	16	12	27	27	0	2	0	6	5	1	8	7	1	0	3	7	7	0	0	0
з	80	16	39	14	30	69	3	244	31	6	15	35	24	40	83	13	29	19	24	23	2	2	1	11	4	1	8	11	0	1	3	17	6	0	0	0
и	153	53	80	35	85	129	32	31	751	6	88	174	100	190	180	97	126	112	168	71	16	25	22	60	34	4	23	28	4	1	35	47	26	0	1	0
й	18	2	12	3	13	36	7	6	6	126	13	9	3	17	78	11	12	20	25	6	0	1	1	1	1	1	0	4	0	0	4	4	5	0	0	0
к	218	17	52	14	40	77	20	15	88	13	561	53	48	55	248	27	68	88	84	72	14	8	6	12	16	1	27	18	5	1	11	20	28	0	0	0
л	192	48	57	32	74	179	25	35	174	9	53	609	40	85	237	66	54	104	96	64	16	6	5	37	23	1	41	46	7	0	35	59	67	0	0	0
м	164	26	44	11	44	139	24	24	100	3	48	40	541	86	209	35	57	83	75	50	6	12	8	16	11	2	17	40	2	0	10	21	52	0	2	0
н	279	48	63	34	99	312	26	40	190	17	55	85	86	902	385	67	76	88	125	68	14	27	15	46	25	2	31	79	4	0	25	48	58	0	1	0
о	241	152	239	172	226	237	77	83	180	78	248	237	209	385	1947	239	296	224	488	112	42	50	17	83	40	4	46	68	7	0	50	68	65	0	0	0
п	70	21	43	11	32	114	11	13	97	11	27	66	35	67	239	541	117	75	85	38	6	9	1	16	12	2	49	12	2	0	11	28	25	0	0	0
р	179	48	68	36	69	210	16	29	126	12	68	54	57	76	296	117	652	58	150	97	18	19	8	20	13	1	55	41	3	0	11	37	44	0	1	0
с	171	24	99	27	62	143	12	19	112	20	88	104	83	88	224	75	58	709	188	55	13	11	8	15	9	2	27	33	11	3	22	37	50	0	1	0
т	303	49	121	31	96	209	27	24	168	25	84	96	75	125	488	85	150	188	1184	95	12	23	27	83	23	1	39	47	12	1	22	65	82	0	2	0
у	68	69	45	31	94	83	27	23	71	6	72	64	50	68	112	38	97	55	95	423	7	13	4	30	21	2	12	18	2	1	21	9	10	0	0	0
ф	21	5	11	4	13	20	0	2	16	0	14	16	6	14	42	6	18	13	12	7	89	2	3	2	2	0	7	1	0	0	1	4	2	0	0	0
х	33	5	12	0	13	17	2	2	25	1	8	6	12	27	50	9	19	11	23	13	2	137	3	4	4	0	11	19	1	0	2	9	20	0	0	0
ц	39	1	5	1	2	18	0	1	22	1	6	5	8	15	17	1	8	8	27	4	3	3	83	2	3	0	2	1	1	0	2	5	0	0	0	0
ч	56	17	18	8	12	80	6	11	60	1	12	37	16	46	83	16	20	15	83	30	2	4	2	231	3	0	7	5	4	0	13	13	8	0	0	0
ш	48	6	23	5	10	50	5	4	34	1	16	23	11	25	40	12	13	9	23	21	2	4	3	3	133	0	0	11	1	0	3	6	6	0	0	0
щ	2	1	1	1	1	7	1	1	4	1	1	1	2	2	4	2	1	2	1	2	0	0	0	0	0	24	1	0	0	0	1	4	0	0	0	0
ъ	44	13	66	10	21	51	8	8	23	0	27	41	17	31	46	49	55	27	39	12	7	11	2	7	0	1	214	10	3	0	6	22	13	0	0	0
ы	37	37	63	6	33	47	7	11	28	4	18	46	40	79	68	12	41	33	47	18	1	19	1	5	11	0	10	279	0	0	5	30	8	0	1	0
ь	8	3	5	1	3	4	1	0	4	0	5	7	2	4	7	2	3	11	12	2	0	1	1	4	1	0	3	0	27	0	0	0	2	0	0	0
э	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	3	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
ю	27	11	13	1	27	20	3	3	35	4	11	35	10	25	50	11	11	22	22	21	1	2	2	13	3	1	6	5	0	0	135	7	5	0	0	0
я	47	16	52	10	34	57	7	17	47	4	20	59	21	48	68	28	37	37	65	9	4	9	5	13	6	4	22	30	0	0	7	279	8	0	0	0
ъ	49	15	39	11	69	51	7	6	26	5	28	67	52	58	65	25	44	50	82	10	2	20	0	8	6	0	13	8	2	0	5	8	277	0	0	0
–	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
⊕	0	0	0	0	0	0	0	0	1	0	0	0	2	1	0	0	1	1	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	3	0
⊗	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Рисунок 31 – Таблица ошибок для букв для датасета писем начала 20 века (в столбцах таблицы записаны истинные значения, в строках записаны предсказанные значения, главная диагональ таблицы содержит TP для соответствующего символа, соотношение предсказанных значений – строки с реальными – столбцы является параметром FP, а соотношение реальных – столбцы с предсказанными – строки является параметром FN)

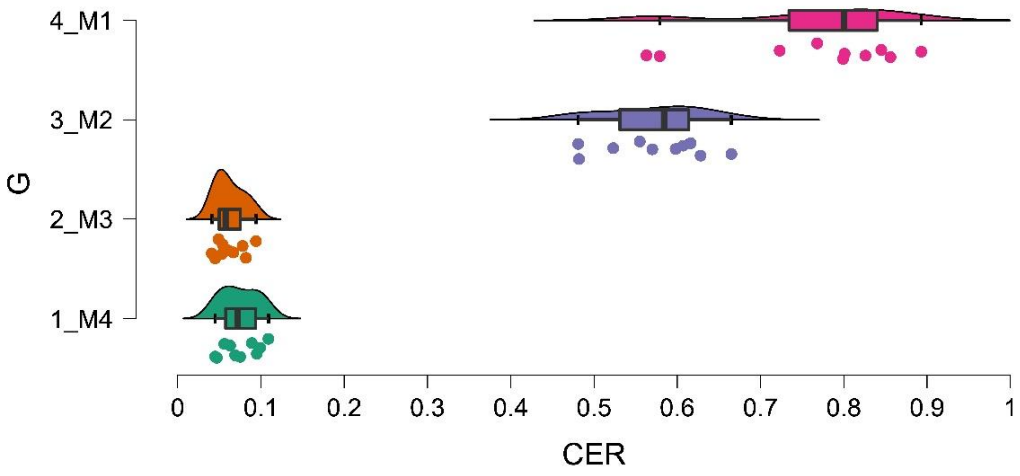


Рисунок 32 – График дисперанализа (вертикальная ось – номер проведенного эксперимента, горизонтальная ось – значение CER)

Таблица 3 – Таблица показателей эффективности распознания для каждого символа из датасета писем начала 20 века

	Recall	Specificity	Precision	F1	Accuracy
	[0.389;0.421]	[0.901;0.905]	[0.42;0.451]	[0.404;0.435]	[0.822;0.829]
(	[0.2;0.5]	[1.0;1.0]	[0.5;1.0]	[0.286;0.667]	[1.0;1.0]
+	[0.386;0.5]	[0.999;0.999]	[0.385;0.542]	[0.386;0.512]	[0.998;0.999]

	Recall	Specificity	Precision	F1	Accuracy
0	[0.378;0.75]	[1.0;1.0]	[0.386;0.75]	[0.382;0.75]	[0.999;1.0]
1	[0.308;0.487]	[0.999;0.999]	[0.333;0.5]	[0.32;0.493]	[0.997;0.998]
2	[0.433;0.5]	[0.999;1.0]	[0.368;0.467]	[0.4;0.483]	[0.998;0.999]
3	[0.357;0.667]	[1.0;1.0]	[0.345;0.571]	[0.351;0.615]	[0.999;1.0]
4	[0.333;0.571]	[1.0;1.0]	[0.348;0.5]	[0.34;0.533]	[0.999;0.999]
5	[0.25;0.75]	[1.0;1.0]	[0.4;0.75]	[0.308;0.75]	[1.0;1.0]
7	[0.357;0.432]	[0.999;1.0]	[0.382;0.421]	[0.37;0.426]	[0.999;0.999]
8	[0.2;0.333]	[1.0;1.0]	[0.167;0.357]	[0.182;0.333]	[0.999;1.0]
9	[0.333;0.471]	[1.0;1.0]	[0.333;0.5]	[0.333;0.485]	[1.0;1.0]
[	[0.333;0.493]	[0.999;1.0]	[0.364;0.578]	[0.348;0.532]	[0.999;0.999]
]	[0.2;0.364]	[0.999;1.0]	[0.2;0.41]	[0.2;0.381]	[0.999;0.999]
i	[0.361;0.411]	[0.989;0.989]	[0.351;0.4]	[0.356;0.405]	[0.978;0.98]
а	[0.395;0.415]	[0.956;0.957]	[0.378;0.404]	[0.386;0.409]	[0.92;0.922]
б	[0.38;0.403]	[0.988;0.989]	[0.373;0.398]	[0.376;0.4]	[0.978;0.978]
в	[0.378;0.413]	[0.979;0.98]	[0.384;0.418]	[0.381;0.415]	[0.958;0.962]
г	[0.335;0.387]	[0.99;0.991]	[0.331;0.38]	[0.333;0.383]	[0.981;0.983]
д	[0.367;0.398]	[0.98;0.981]	[0.37;0.399]	[0.368;0.398]	[0.961;0.963]
е	[0.405;0.441]	[0.961;0.964]	[0.397;0.432]	[0.401;0.436]	[0.927;0.933]
ж	[0.374;0.404]	[0.993;0.994]	[0.38;0.411]	[0.377;0.406]	[0.986;0.988]
з	[0.375;0.41]	[0.992;0.993]	[0.389;0.419]	[0.384;0.414]	[0.983;0.986]
и	[0.386;0.426]	[0.973;0.974]	[0.377;0.42]	[0.381;0.423]	[0.949;0.951]
й	[0.384;0.43]	[0.996;0.996]	[0.399;0.46]	[0.391;0.444]	[0.991;0.992]
к	[0.385;0.408]	[0.98;0.982]	[0.391;0.409]	[0.388;0.408]	[0.962;0.965]
л	[0.396;0.446]	[0.979;0.98]	[0.398;0.447]	[0.397;0.446]	[0.959;0.961]
м	[0.345;0.411]	[0.981;0.982]	[0.338;0.406]	[0.341;0.408]	[0.964;0.966]
н	[0.406;0.424]	[0.97;0.97]	[0.395;0.419]	[0.4;0.421]	[0.943;0.944]
о	[0.439;0.453]	[0.938;0.94]	[0.43;0.443]	[0.434;0.448]	[0.889;0.893]
п	[0.372;0.409]	[0.979;0.983]	[0.367;0.405]	[0.369;0.407]	[0.959;0.967]
р	[0.389;0.422]	[0.975;0.977]	[0.384;0.418]	[0.386;0.42]	[0.953;0.956]
с	[0.354;0.394]	[0.977;0.978]	[0.358;0.394]	[0.356;0.394]	[0.955;0.957]
т	[0.403;0.427]	[0.961;0.963]	[0.391;0.417]	[0.397;0.422]	[0.929;0.932]
у	[0.388;0.415]	[0.983;0.985]	[0.371;0.399]	[0.379;0.407]	[0.968;0.971]
ф	[0.167;0.365]	[0.997;0.997]	[0.163;0.373]	[0.165;0.369]	[0.993;0.994]

	Recall	Specificity	Precision	F1	Accuracy
х	[0.356;0.396]	[0.995;0.996]	[0.368;0.414]	[0.362;0.405]	[0.99;0.992]
ц	[0.377;0.409]	[0.997;0.997]	[0.377;0.404]	[0.377;0.406]	[0.994;0.995]
ч	[0.381;0.42]	[0.992;0.993]	[0.388;0.423]	[0.386;0.421]	[0.984;0.985]
ш	[0.4;0.433]	[0.995;0.996]	[0.384;0.409]	[0.392;0.421]	[0.991;0.992]
щ	[0.455;0.525]	[0.999;1.0]	[0.438;0.547]	[0.452;0.531]	[0.999;0.999]
ъ	[0.382;0.415]	[0.991;0.992]	[0.363;0.389]	[0.372;0.402]	[0.983;0.985]
ы	[0.393;0.44]	[0.991;0.992]	[0.387;0.426]	[0.39;0.433]	[0.982;0.985]
ь	[0.257;0.329]	[0.999;0.999]	[0.31;0.393]	[0.281;0.355]	[0.998;0.998]
э	[0.323;0.343]	[0.999;1.0]	[0.35;0.378]	0[.355;0.363]	[0.998;1.0]
ю	[0.358;0.44]	[0.995;0.996]	[0.358;0.429]	[0.358;0.434]	[0.99;0.992]
я	[0.399;0.419]	[0.991;0.992]	[0.382;0.407]	[0.39;0.412]	[0.983;0.985]
ѣ	[0.387;0.427]	[0.989;0.99]	[0.377;0.414]	[0.382;0.42]	[0.98;0.982]

4 ВЫВОДЫ

В результате работы были достигнуты следующие цели:

1) Сформирован датасет рукописных писем начала 20 века (размер датасета – 90 страниц писем, 8700 изображений слов).

2) Реализован новый вид аугментаций, который имитирует зачеркивания и ошибки на изображениях с рукописным текстом.

3) Написан рабочий алгоритм по поиску цепочек символов на изображении при помощи алгоритма бинаризации Бредли-Рота.

4) Для распознавания писем разработаны алгоритмы на основе архитектуры нейронной сети BiLSTM+CNN/.

5) С использованием алгоритма распознаны письма, при этом достигнут уровень распознавания: CER = ДИ [0.055;0.083] и WER = ДИ [0.183;0.235].

6) С использованием обученной нейронной сети распознаны письма начала 20 века

ЗАКЛЮЧЕНИЕ

В результате использования авторской архитектуры сверточной нейронной сети разработан эффективный алгоритм по распознанию рукописных текстов начала 20 века на изображениях, который смог качественно распознать рукописные письма 1919 года.

Для написания нейронной сети по распознанию рукописного текста на изображении лучше всего использовать функцию ошибок CTC Loss и архитектуру BiLSTM+CNN.

Проведенный анализ итоговых метрик Precision, Recall и Specificity показал, что алгоритм лучше предсказывает каких символов не может быть на определенных позициях. Была выдвинута гипотеза, что алгоритм работает таким образом, что он предсказывает с какой вероятностью и на какой позиции, не будут находиться определенные символы.

Полученный алгоритм может быть использован в сферах, где требуется работа с текстом, к примеру, в архивах. Это поможет сэкономить время затрачиваемое на обработку рукописного текста (при составлении датасета писем начала 20 века, на расшифровку одного письма уходило 30-45 минут, алгоритм способен расшифровать такое письмо за 21 секунду), а также перевод рукописных текстов в цифровой формат поможет качественнее анализировать эти тексты, а главное, данные находящиеся в цифровом формате будут более сохранными.

СПИСОК ЛИТЕРАТУРЫ

1) Раньше буквы могли говорить: эксперт ЮФУ о реформе русской орфографии 1918 года. [Электронный ресурс] // https://sfedu.ru/press-center/news/69879 (дата обращения: 21.10.2023)

2) Реформа русской орфографии 1918 года. [Электронный ресурс] // https://clck.ru/HPSZN (дата обращения: 21.10.2023)

3) Реформа орфографии 1918 г. [Электронный ресурс] // http://www.dates.gnpbu.ru/3-8/Orfografiya/orfografiya.html (дата обращения 21.10.2023)

4) IAM Dataset [Электронный ресурс] // https://www.kaggle.com/datasets/naderabdalghani/iam-handwritten-forms-dataset (дата обращения 21.10.2023)

5) George Washington Dataset [Электронный ресурс] // https://paperswithcode.com/dataset/george-washington (дата обращения 21.10.2023)

6) Digital Peter Dataset [Электронный ресурс] // https://paperswithcode.com/dataset/digital-peter (дата обращения 21.10.2023)

7) CAPTCHA Dataset [Электронный ресурс] // https://www.kaggle.com/datasets/fournierp/captcha-version-2-images (дата обращения 21.10.2023)

8) Transkribus. [Электронный ресурс] // https://readcoop.eu/transkribus/ (дата обращения: 17.02.2024)

9) Kleopatra Markou, A Convolutional Recurrent Neural Network for the Handwritten Text Recognition of Historical Greek Manuscripts // K. Markou, L. Tsochatzidis, K. Zagoris, A. Papazoglou, X. Karagiannis, S. Symeonidis, I. Pratikakis, 2021, Democritus University of Thrace // https://www.researchgate.net/publication/349470666_A_Convolutional_Recurrent_Neural_Network_for_the_Handwritten_Text_Recognition_of_Historical_Greek_Manuscripts

10) Qiang Guo, Memory Matters: Convolutional Recurrent Neural Network for Scene Text Recognition // Qiang Guo∗, Dan Tu, Guohui Li and Jun Lei, National University of Defense Technology // https://arxiv.org/pdf/1601.01100

11) Baoguang Shi, An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition // Baoguang Shi, Xiang Bai, Cong Yao, 2015, Huazhong University of Science and Technology // https://arxiv.org/pdf/1507.05717

12) Denis Coquenet, End-to-end Handwritten Paragraph Text Recognition Using a Vertical Attention Network // Denis Coquenet, Clément Chatelain, Thierry Paquet, 2021 // https://arxiv.org/pdf/2203.12273

13) Puneeth Prakash, CRNN model for text detection and classification from natural scenes // Puneeth Prakash, Sharath Kumar Yeliyur Hanumanthaiah, Somashekhar Bannur Mayigowda, 2023, Visvesvaraya Technological University //

https://www.researchgate.net/publication/378640431_CRNN_
model_for_text_detection_and_classification_from_natural_scenes

14) Hao Yu, Mixup Without Hesitation // Hao Yu1, Huanyu Wang, Jianxin Wu, 2021 // Nanjing University // https://arxiv.org/pdf/2101.04342

15) Синтетические данные для машинного обучения. [Электронный ресурс] // https://habr.com/ru/articles/721170/ (дата обращения: 21.02.2024)

16) De-An Huang Connectionist Temporal Modeling for Weakly Supervised Action Labeling. [Электронный ресурс] // De-An Huang, Li Fei-Fei, Juan Carlos Niebles, Connectionist Temporal Modeling for Weakly Supervised Action Labeling, Computer Science Department, Stanford University, 2016 // https://arxiv.org/pdf/1607.08584.pdf (дата обращения: 23.11.2023)

17) Hannun, Awni Sequence Modeling with CTCLoss [Электронный ресурс]// Hannun, "Sequence Modeling with CTC", Distill, 2017 // https://distill.pub/2017/ctc/ (дата обращения: 2.12.2023)

18) Hairong Liu Gram-CTC: Automatic Unit Selection and Target Decomposition for Sequence Labelling [Электронный ресурс] // Liu, H., Zhu, Z., Li, X. and Satheesh, S., 2017. Proceedings of the 34th International Conference on Machine Learning. // http://arxiv.org/pdf/1703.00096.pdf (дата обращения: 22.11.2023)

19) Olah C Attention and Augmented Recurrent Neural Networks. [Электронный ресурс] // Olah, Chris and Carter Shan "Attention and Augmented Recurrent Neural Networks", Distill, 2016 // http://distill.pub/2016/augmented-rnns (дата обращения: 13.11.2023)

20) Sana'a Zahra, OCR Segmentation with Python Code [Электронный ресурс] //Sana'a Zahra, 2023 // https://sanaazahra.medium.com/ocr-segmentation-with-python-code-f3251114ee48

21) Дообучение easyocr. [Электронный ресурс] // 2022 // https://habr.com/ru/articles/691598/ (дата обращения: 11.03.2024)

22) Бинаризация изображений: алгоритм Брэдли. [Электронный ресурс] // https://habr.com/ru/articles/278435/ (дата обращения: 14.03.2024)

23) Paramasivam M E, The Effect Of Binarization Algorithms Considering Color-To-Gray Scale Conversion Methods On Historic Document Images // Paramasivam M E, Sabeenian R S, Dinesh P M, 2020, International journal of science & technology research // https://www.researchgate.net/publication/342876902_The_Effect_Of_Binarization_Algorithms_Considering_Color-To_-Gray_Scale_Conversion_Methods_On_Historic_Document_Images

24) Risab Biswas, A Layer-Wise Tokens-to-Token Transformer Network for Improved Historical Document Image Enhancement // Risab Biswas, Swalpa Kumar Roy, Umapada Pal, 2023, Indian Statistical Institute // https://arxiv.org/html/2312.03946v1

25) Alex Shonenkov, StackMix and Blot Augmentations for Handwritten Text Recognition // Alex Shonenkov, Denis Karachev, Max Novopoltsev, Mark Potanin, Denis Dimitrov, 2021 // https://arxiv.org/pdf/2108.11667

26) Почему BiLSTM лучше, чем LSTM [Электронный ресурс] // https://medium.com/@souro400.nath/why-is-bilstm-better-than-lstm-a7eb0090c1e4

27) Zecheng Xie, Aggregation Cross-Entropy for Sequence Recognition // Zecheng Xie, Yaoxiong Huang, Yuanzhi Zhu, Lianwen Jin, Yuliang Liu, Lele Xie, 2019, South China University of Technology // https://arxiv.org/pdf/1904.08364

28) Enrique Vidal , End-to-End page-Level assessment of handwritten text recognition // Enrique Vidal, 2023, Universitat Politècnica de València // https://www.sciencedirect.com/science/article/pii/S003132032300393X#sec0009

29) Упадок RNN и LSTM сетей [Электронный ресурс] // https://habr.com/ru/articles/561082/

30) Savelie Cornegruta, Modelling Radiological Language with Bidirectional Long Short-Term Memory Networks // Savelie Cornegruta, Robert Bakewell, Samuel Withey, Giovanni Montana, Department of Biomedical Engineering, King’s College London // https://arxiv.org/pdf/1609.08409

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлен с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

_____	_____
ФИО студента	*Подпись студента*

« ____ »_____20 __г.
(заполняется от руки)