Об автоматическом распознавании печатного текста

А. Д. Агафьина, А. А. Никитин

Аннотация. **В рамках статьи рассматриваются внутреннее устройство системы распознавания печатных данных и перевода их в удобный для использования формат. Разработанный проект представляет собой решение для конвертации изображений и PDF-файлов с печатным текстом и формулами в формат LaTeX, используя бесплатные, открытые библиотеки, а также собственную обученную модель для классификации, детекции и сегментации данных.**

Подробно рассмотрены этапы разработки системы, начиная с анализа существующих алгоритмов и заканчивая созданием собственной модели для распознавания текста и формул. Основное внимание уделено выбору инструментов и библиотек, подходящих для решения задач, связанных с автоматизацией распознавания и конвертации печатных документов в формат LaTeX, попытке улучшения работы библиотек. Описан процесс создания и подготовки набора данных для обучения модели, включающий разметку изображений и текстов. Итоги обучения моделей представлены в виде таблиц с полученными точностями распознавания различных классов данных.

В качестве результатов в статье представлены результаты тестирования функционала системы для распознавания текста и формул на русском и английском языках, подробно описаны достоинства и недостатки разработанной системы, а также возникшие сложности в процессе её создания. На базе нашей разработки был создан сайт, включающий интерфейс по конвертации картинок/pdf в LaTeX-файл, а также Telegram-бот с аналогичным функционалом.

***Ключевые слова*— компьютерное зрение, машинное обучение, распознавание документов, сегментация данных, latex, OCR.**

I. Введение

В наше время потребность в эффективных инструментах для обработки текстовой информации становится всё более актуальной. С ростом объёмов информации возникает спрос на инструменты, которые могут облегчить и улучшить автоматизацию обработки данных.

LaTeX — один из наиболее популярных инструментов для создания научных и технических документов, благодаря своей гибкости и мощным возможностям форматирования. Однако процесс создания LaTeX-кода для изображений с печатным текстом может быть сложным и требовать значительных временных затрат, особенно когда документ содержит разнообразные математические формулы и иллюстрации.

Наша цель — разработать приложение, которое автоматически преобразует изображения и PDF-файлы с печатным текстом в формат LaTeX. Мы стремимся предоставить пользователям инструмент, который значительно упрощает процесс создания LaTeX-документов, делая его доступным для широкого круга пользователей, включая студентов, исследователей и профессионалов в области науки и техники. Сильные стороны нашего проекта обусловлены несколькими ключевыми особенностями:

- использование собственной обученной модели для сегментации данных;
- детектирование и извлечение графиков и изображений из входных данных, что позволяет получить полноценный результат без потери визуальной части отсканированного документа. Этот функционал не реализован в аналогичных приложениях.

Основные результаты нашей работы включают разработку функционала для максимально точного распознавания текста, полученного из изображений или печатных PDF-документов на русском и английском языках, и его преобразование в LaTeX-код. Нами использовались бесплатные и/или открытые библиотеки, такие как Open-CV, Pytesseract, LaTeX OCR [3], Nougat-LaTeX-OCR [13] и Ultralytics YOLOv8 [16]. Эти инструменты позволяют достичь высокой точности распознавания и обработки информации, что является ключевым требованием к системам распознавания документов.

II. Обзор литературы и схожих программных решений

Рассмотрим алгоритмы, которые позволяют разделить PDF-документ на различные блоки: текстовые, математические и блоки с изображениями. Например, в статье [14] описывается шесть популярных алгоритмов для сегментации страницы. Авторы также проводят сравнительный анализ этих алгоритмов и приходят к выводу, что наиболее эффективными являются алгоритмы, основанные на алгоритмах Voronoi, Docstrum и Text-line finding.

Еще один подход был описан в работе [10], где рассматривается вопрос распознавания отдельных

Анастасия Дмитриевна Агафьина, Национальный Исследовательский Университет “Высшая Школа Экономики”, студент, adagafina@edu.hse.ru.

Алексей Антонович Никитин, Московский Государственный Университет имени М. В. Ломоносова, доцент, nilitin@cs.msu.su.

символов в формуле с помощью свёрточной нейронной сети.

В статье [15] также рассматривается архитектура encoder-decoder. В этой архитектуре используется механизм внимания (attention), который позволяет улучшить работу с длинными последовательностями. Эта архитектура применяется для различных задач, включая распознавание изображений, сопоставление математических изображений с соответствующим кодом разметки LaTeX и преобразование текста в изображение.

Для рассмотрения оптического распознавания символов можно обратиться к статье [7]. В этой статье также используется архитектура encoder-decoder, но в контексте задачи OCR. Основной акцент делается на том, как обрабатывать презентационные аспекты математических выражений, такие как надстрочные и подстрочные обозначения, специальные символы и вложенные дроби.

Успешные системы OCR комбинируют специализированную сегментацию символов [12] с грамматиками базового языка математической структуры. Это включает систему INFTY, которая преобразует печатные математические выражения в LaTeX и другие форматы разметки. Статья также обращает внимание на увеличенный интерес к задачам, требующим совместной обработки изображений и текста, благодаря улучшениям в глубоких нейронных моделях. Это включает в себя распознавание почерка, OCR в естественных сценах и генерацию подписей к изображениям.

В последние годы было создано несколько программных комплексов, которые решают похожие задачи. Одним из таких инструментов является система Mathpix [11] - это современный инструмент для распознавания текста и формул в LaTeX-коде. Он может работать с PDF-документами и изображениями, которые в контексте этого приложения называются снипами.

Для идентификации объектов на снипах, Mathpix использует собственную технологию оптического распознавания символов (OCR), которая с высокой точностью переводит данные в текстовый формат, который можно использовать в работе с компьютером. Эта система может обрабатывать сложные математические уравнения, химические диаграммы и таблицы в LaTeX, а также поддерживает множество языков, включая русский, английский, немецкий, французский и другие. Mathpix доступен в виде мобильного и веб-приложения, расширения браузера и программы для компьютеров и ноутбуков. Разработчики программного обеспечения могут получить доступ к системе через API.

Из недостатков, на которые можно обратить внимание, отметим, что во время распознавания кириллических символов, Mathpix часто допускает ошибки, путая такие символы как, например, “п” и “л”, “н” и “п”. Однако, нельзя сказать, что используемый в нашей работе, Pytesseract лучше в этом вопросе, потому что данная библиотека имеет схожие ошибки распознавания. Ещё один недостаток Mathpix заключается в том, что математические графики и иллюстрации не добавляются в итоговый файл. Это означает, что пользователю приходится самостоятельно извлекать их из исходного файла и вставлять в код в сторонних программах. Это увеличивает время, необходимое для перевода документа в формат LaTeX. В отличие от Mathpix, наше программное решение автоматически распознаёт иллюстрации как отдельный класс данных, сегментирует их, сохраняет все пояснительные изображения в отдельные файлы и вставляет их в итоговый LaTeX-документ. Это позволяет значительно сократить время пользователя на работу с документом.

Ещё одна возможность для перевода отсканированных документов в LaTeX — это чат-бот Chat GPT [4] (GPT - Generative Pre-trained Transformer). Ранее в бесплатной версии не было возможности прикреплять файлы, но, на момент анализа данного инструмента, она появилась. Chat GPT — это большая языковая модель, которая работает в режиме диалогового окна. Пользователи могут отправлять запросы и получать ответы в режиме реального времени. Мы прикрепили отсканированный документ к диалогу с комментарием о необходимости перевода в LaTeX. Ответ был дан в течение нескольких секунд.

Выходной файл, полученный с помощью Chat GPT, дал результат идентичный оригиналу. Были правильно отображены все теги, включая правильно отображённые оглавления, отсутствие табуляции и формулы. В итоговый LaTeX-код была корректно добавлена информация об иллюстрациях, содержащихся в отсканированном файле. Однако самих изображений данная модель выдать не может. Таким образом, пользователю, который использует Chat GPT, нужно будет потратить время на заготовку файлов с сопутствующими изображениями. В то же время, пользователь, который работает с результатами нашего проекта, не будет тратить время на отделение иллюстраций от общего вида документа.

III. Разработка модели распознавания текста для сегментированного посимвольно файла

Работа над распознаванием текста начинается с задачи распознавания текстовой информации. Для этого мы решили написать код на языке Python, который бы сегментировал файл на отдельные символы. Мы использовали функцию поиска прямоугольных контуров на изображении из библиотеки OpenCV. Также мы применили цветовую инверсию чёрно-белого изображения, изменили контрастность и применили фильтр размытия по Гауссу. Это позволило нам получить координаты отдельных абзацев, строк и символов на тестовых файлах. Благодаря этому мы смогли выполнять сегментацию на необходимом уровне.

Затем нами был создан собственный набор данных, который аналогичен базе данных MNIST. База данных MNIST состоит из 70000 изображений рукописных цифр размером 28 на 28 пикселей. Наш набор данных состоял

из 28730 изображений букв русского алфавита заглавного (кроме “ь”, “ъ” и “ы”) и строчного вида начертаний, символов и знаков препинания размером 28 на 28 пикселей каждое. Всего было 104 вида типов данных. Для каждого символа мы использовали все доступные встроенные шрифты в ОС Windows, которые применимы к кириллице. Также мы аугментировали датасет, растягивая изображения в ширину и/или в высоту, размывая их, наклоняя буквы в разные стороны и перемещая центр буквы в разные места полотна. Вес полученного датасета составил 144 МБ.

На полученном датасете мы обучили собственную модель распознавания русских текстовых символов с помощью библиотеки tensorflow. Мы написали нейронную сеть сверточного типа для классификации изображений. Свёрточные нейронные сети отличаются тем, что состоят из нескольких слоёв различного типа, через которые информация проходит в одном направлении. Наша модель содержит один слой свёртки (Conv2D), слой пулинга (MaxPooling2D), слой выравнивания (Flatten), два полносвязанных скрытых слоя (Dense) и слой Dropout (Dropout). Размер входного вектора был 784, а по результатам работы модели на выходе было 107 нейронов, которые позволяли определить, какой именно символ был подан для распознавания.

Для обучения было использовано 80% датасета, остальные 20% - для валидации. Во избежание переобучения модели распознавания букв, с помощью графиков (Рисунок 1 и Рисунок 2) было подобрано количество эпох (количество проходов через весь набор данных) обучения.

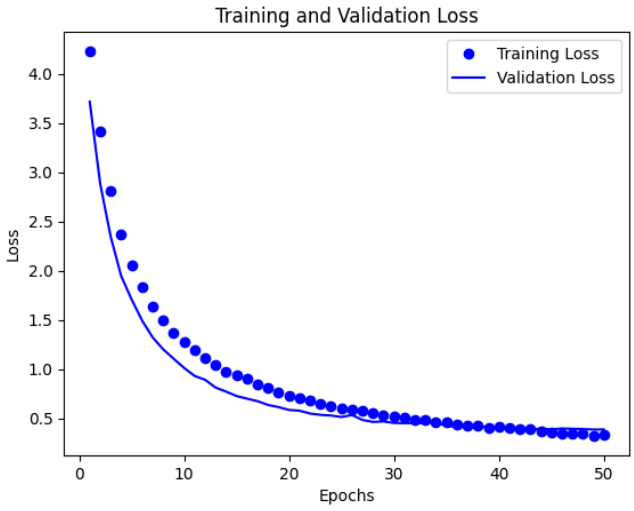

Рис. 1 - График потерь. Убывающая функция потерь показывает улучшение качества модели. Точка пересечения графиков показывает, что обучение необходимо остановить во избежание переобучения.

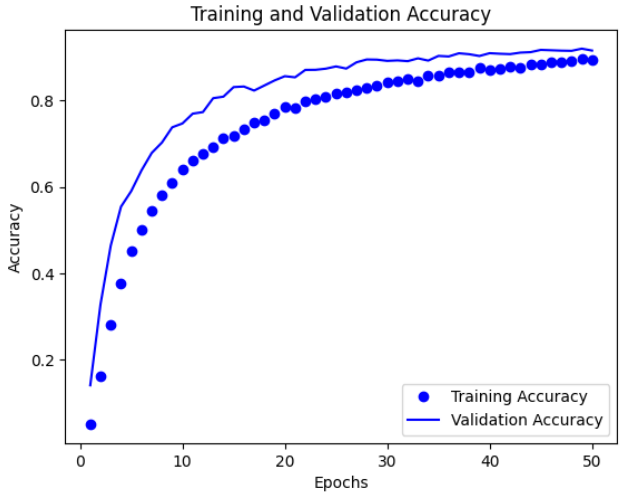

Рис. 2 - График точности. Возрастающая функция точности показывает улучшение качества модели. Точка пересечения графиков показывает, что обучение необходимо остановить во избежание переобучения.

Обучение происходило с помощью бесплатной версии Google Colab. За 50 эпох обучения, были получены значения точности, представленные в Таблице I.

Полученная модель показывала валидационную точность порядка 91%, однако, при тестировании на реальных символах из сегментированного изображения, данные которого не участвовали в обучении, модель давала точность около 50%, что сделало невозможным её дальнейшее использование в данном проекте.

Таблица I - Результаты обучения модели для классификации буквенных символов.

Параметр	Тренировочный набор	Валидационный набор
Потери (Loss)	0.3338	0.3870
Точность (Accuracy)	0.8935	0.9156

На основе данной неуспешной стратегии принято решение использовать сегментацию не по отдельным буквам, а по абзацам, формулам и отдельно стоящим словам, чтобы, с использованием готовых библиотек (Pytesseract, LaTeX OCR и Nougat-LaTeX-OCR), переводить данные в текст и формулы. Данные библиотеки на высоком уровне справляются со своими задачами, что позволило нам заниматься другими прикладными задачами данного проекта.

IV. Детекция, сегментация и классификация объектов изображения с помощью модели YOLOv8

Столкнувшись с проблемой классификации частей сегментированнного документа, был произведён поиск средств, с помощью которых, определение к какому типу данных принадлежит каждый конкретный символ, происходило бы в автоматическом режиме. Так как вариант работы с каждым отдельным символом вызывал трудности, то идея обработки информации сегментами бОльшего размера, а затем последовательная их отправка, в зависимости от класса данных, на обработку различными библиотеками, представлялась более жизнеспособным вариантом развития нашего проекта.

A. Особенности работы Ultralitics YOLOv8

Семейство алгоритмов компьютерного зрения YOLOv8 [16] предназначено для обнаружения объектов на различных типах визуальной информации. Алгоритм разделяет входное изображение на небольшие ячейки, которые затем анализируются на наличие потенциальных объектов.

Каждой ячейке присваивается вероятность наличия в ней объектов определённого класса (например, людей, автомобилей и т. д.). Элементы изображения рассматриваются в разных масштабах, что позволяет исследовать его максимально подробно. Кроме того, применяется аугментация — процесс изменения изображений для увеличения разнообразия данных, на которых обучается модель.

Одним из преимуществ YOLOv8 является то, что для обучения модели детекции, классификации и сегментации не нужно заранее определять количество эпох, после которых модель не будет переобучена и результаты будут наилучшими. В процессе обучения автоматически сохраняются два файла с весами модели: результат после обучения последней эпохи и наилучший результат, который даёт наилучшие статистические показатели и также не является переобученным.

Исследователю, который обучает свою модель для сегментации, достаточно установить в конфигурационном файле высокое значение числа эпох обучения.

B. Средства, использованные для создания наборов данных

Наша работа с данной системой началась с создания датасета. Для этого был выбран инструмент для создания аннотаций CVAT [6], который является бесплатным для работы в одиночку или вдвоём и при разметке небольшого количества данных при работе на сайте или при запуске через Docker Compose [8].

Для удобства работы был выбран именно второй вариант работы с данным инструментом разметки. Но, так как при запуске контейнера со стационарного ноутбука команда столкнулась с прерыванием работы CVAT, было принято решение настроить доступ к нему через контейнер Docker Compose, развёрнутый на базе виртуальной машины AWS [1] на операционной системе Ubuntu 23.10. Использован инстанс t3.xlarge (ЦПУ: 4, память: 16 Гб). Инстанс – это серверный ресурс, предоставляемый облачным сервисом.

Для развёртывания CVAT были испробованы разные инстансы от самых дешёвых и простых до более сложных, но, так как для работы данной системы разметки запускается целый набор взаимосвязанных Docker образов, тратится очень много ресурсов. Путём подбора был обнаружен инстанс, минимальной мощности, который работал стабильно при использовании докер образов.

Выгрузка датасета из CVAT возможна в различных форматах, но выгрузка для YOLO происходит в устаревшем формате, где в одной папке содержатся все размеченные изображения вместе с файлами .txt, содержащие информацию о классах объектов и их координатах. Для запуска обучения необходимо было привести набор данных к виду, отвечающему требованиям YOLOv8. Данная задача была решена с помощью CVAT2YOLO Converter [9], данное средство в автоматическом режиме, с использованием командной строки, изменяет структуру хранения файлов для корректного использование с YOLOv8.

C. Результаты обучения собственных моделей

Нами было решено использовать 3 класса данных: формула, текст и изображения, в которую команда отнесла графики, изображения, таблицы и другое. На Рисунке 3 показан пример разметки данных в текущей работе.

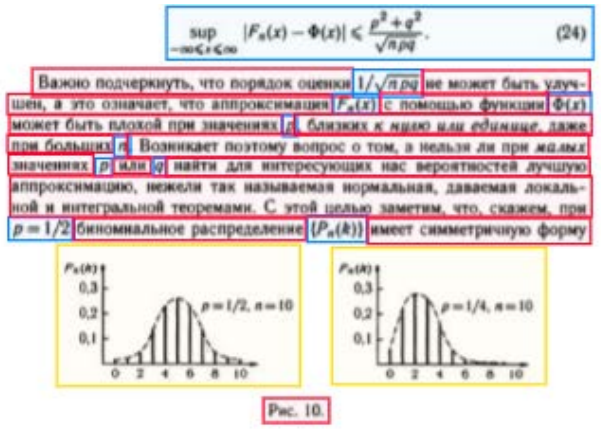

Рис. 3 - Пример разметки данных. Синим выделены формулы, красным - текст, а жёлтым - изображения (в данном случае графики).

Тестовая версия модели сегментации содержала в себе 10 размеченных изображений. Результат обучения на 20 эпохах представлен в Таблице II, где mAP50 и mAP50-95 (mean Average Precision) - это метрики, используемые для оценки точности моделей компьютерного зрения. mAP50 - среднее значение точности, вычисленное при IoU (Intersection over Union, сравнение истинной ограничивающей рамки с прогнозируемой) равном 0.5, а mAP50-95 при различных порогах от 0.5 до 0.95 IoU. GFLOPs - единица, показывающая, сколько операций с плавающей запятой в секунду выполняется на данной машине.

Таблица I - Результаты работы нулевой (тестовой) модели YOLOv8.1.16 для детекции и классификации объектов.

Параметр	Значение	
Версия модели	YOLOv8.1.16	
Python	3.10.12	
Torch	2.1.0+cu121	
Процессор	Intel Xeon 2.20GHz	
Количество слоев	218	
Количество параметров	25,841,497	
GFLOPs	78.7	
Класс	**mAP50**	**mAP50-95**
Все	0.190	0.0731
Формула	0.262	0.11
Текст	0.307	0.11
Картинка	0.000	0.000

Обучение тестовой модели происходило в бесплатной версии Google Colab, наблюдая которое, стало ясно, что следующие версии предстоит обучать в иных облачных сервисах, в которых можно использовать бОльшие ресурсы. Результаты обучения показали, что такой способ получения собственной модели сегментации и классификации стоит улучшать и использовать в проекте.

Первая полноценная версия модели от 05.03.2024 включала в себя 240 размеченных изображений с помощью CVAT, вторая версия от 15.03.2024 - 502 изображения, а третья версия от 15.04.2024 - 983 изображения. Результаты работы предоставлены в Таблице II, Таблице III и Таблице IV соответственно.

Таблица II - Результаты работы первой модели YOLOv8.1.24 для детекции и классификации объектов от 05.03.2024.

Параметр	Значение	
Версия модели	YOLOv8.1.24	
Python	3.10.11	
Torch	2.2.0+cpu	
Процессор	Intel Core i3-3110M 2.40GHz	
Количество слоев	218	
Количество параметров	25,841,497	
GFLOPs	78.7	
Класс	**mAP50**	**mAP50-95**
Все	0.827	0.549
Формула	0.769	0.481
Текст	0.807	0.526
Картинка	0.905	0.64

Таблица III - Результаты работы второй модели YOLOv8.1.24 для детекции и классификации объектов от 15.03.2024.

Параметр	Значение	
Версия модели	YOLOv8.1.24	
Python	3.10.11	
Torch	2.2.0+cu118	
Процессор	Intel Core i3-3110M 2.40GHz	
Количество слоев	218	
Количество параметров	25,841,497	
GFLOPs	78.7	
Класс	**mAP50**	**mAP50-95**
Все	0.803	0.544
Формула	0.811	0.509
Текст	0.819	0.537
Картинка	0.78	0.587

Таблица IV - Результаты работы третьей модели YOLOv8.1.24 для детекции и классификации объектов от 15.04.2024.

Параметр	Значение	
Версия модели	YOLOv8.1.24	
Python	3.10.11	
Torch	2.2.0+cu118	
Процессор	Intel Core i3-3110M 2.40GHz	
Количество слоев	218	
Количество параметров	25,841,497	
GFLOPs	78.7	
Класс	**mAP50**	**mAP50-95**
Все	0.829	0.560
Формула	0.865	0.554
Текст	0.889	0.593
Картинка	0.733	0.534

Для обучения моделей первой, второй и третьей версии использовался инстанс c7a.24xlarge от AWS (ЦПУ: 64, память: 128 Гб). Причина выбора данного инстанса в том, что он самый мощный из доступных нашей команде.

В каждой версии обучения 20% данных отправлялись на валидацию. 51 изображение было выделено дополнительно для тестирования результатов, эта часть датасета была неизменна от версии к версии, что позволяет сравнивать точности корректно между собой.

Для запуска моделей необходимо иметь конфигурационный файл .yaml, в котором описаны классы, которые модель детектирует и файл с весами .pt. В папке проекта должен лежать сам набор данных, на которых обучена модель.

V. Внутренняя логика обработки файлов и получение LaTeX-кода на их основе

Постобработка сегментированного документа выполнена посредством языка программирования Python. Процесс начинается с загрузки файла в форматах .pdf, .jpg или .png. Если загружен .pdf, он разбивается на отдельные .png страницы. На начальном этапе также задаются параметры окружения для LaTeX-документа, чтобы обеспечить корректную компиляцию с учетом возможного наличия формул и изображений.

Каждая страница отсканированного документа анализируется на предмет необходимости её разворота. Ротация происходит с помощью библиотеки Open CV, инструменты которой позволяют, на основе вычисленных углов наклона строк, определить на какой угол необходим разворот, а затем выполняет его.

Далее для каждой страницы определяются сегменты текста, формул и изображений. Модель, обученная на основе YOLOv8, определяет координаты сегментов, которые затем сортируются на основе срединных точек с учётом погрешности, что позволяет восстановить данные в хронологическом порядке. Учитывается также возможное наличие дубликатов частей изображения.

Обработка каждого сегмента зависит от класса данных, обнаруженных в нём. Так, формулы обрабатываются двумя схожими ocr, которые имеют различные параметры точности. В зависимости от

размера формулы, выбирается или LaTeX OCR, или Nougat-LaTeX-OCR. Параметры точности указаны в Таблице V. Обе модели распознавания формул обучены на датасете, состоящем из порядка 100000 изображений [17], но у Nougat-LaTeX-OCR другие параметры обучения, что позволило получить результат для больших формул лучше, чем у LateX OCR.

Таблица V - Сравнение точности и нормированного расстояния редактирования моделей LaTeX OCR и Nougat-LaTeX-OCR.

Model	Token Accuracy ↑	Normed Edit Distance ↓
LaTeX OCR	0.60000	0.10000
Nougat-LaTeX-OCR	0.62385	0.06180

Token Accuracy - параметр точности модели, а Normed Edit Distance - это мера, используемая для оценки сходства между двумя строками текста, в основе которой лежит расстояние Левенштейна, которое определяется как минимальное количество операций вставки, удаления и замены символов, необходимых для преобразования одной строки в другую. Таким образом точность (accuracy) должна быть как можно выше, а нормированное расстояние редактирования (normed edit distance) должно быть как можно ниже.

Сегменты с текстовой информацией обрабатываются библиотекой Pytesseract, которая одновременно распознаёт слова на русском и английском языках. После получения текста, код обрабатывает его, чтобы убрать знаки переноса, которые будут мешать красивому выводу информации, добавляет окружение для того, чтобы не ставилась табуляция вначале неполных абзацев, происходит центрирование заголовков, разбиение на абзацы и страницы, правильное расположение фрагментов текста и формул по отношению друг к другу.

Сегменты с иллюстрациями добавляются в LaTeX-код в двух вариантах: как одиночная картинка с подписью и без подписи. Также, помимо описанных функций, есть несколько вспомогательных небольших частей, которые обеспечивают доступ к открытому API Nougat-LaTeX-OCR, визуализируют работу модели сегментации, что помогает в тестировании, а также создают вспомогательные файлы.

В общем виде, программа способна корректно работать с простыми файлами, без подачи информации в несколько столбцов, отлично обнаруживает и сохраняет графики и изображения для дальнейшей компиляции.

VI. Улучшение результатов распознавания LaTeX OCR

Параметры точности распознавания формул в Mathpix или Chat GPT неизвестны, поэтому сравнить их с точностью LaTeX OCR и Nougat-LaTeX-OCR не представляется возможным, однако пользователям наглядно заметно, что точность первых сильно выше. На гитхабе [3] указан способ как обучить аналогичную модель на пользовательских данных, что позволяет провести эксперименты с улучшением распознаванием формул.

A. Работа с набором данных

Известно, что для обучения LaTeX OCR использовался набор данных [17], поэтому было решено провести новое обучение на новом наборе данных, который более, чем в два раза больше предыдущего [18]. Найденный открытый датасет состоит из порядка 230000 изображений формул, .json файл с указанием словаря, использованного для записи результата, а также содержит в себе файлы .txt с расшифровкой формулы в LaTeX и с указанием порядка, в котором производилось расшифрование. Указанный датасет необходимо было переделать по аналогии с меньшим датасетом, на котором обучение уже было, иначе использовать его было нельзя.

Для начала необходимо было переименовать все файлы с изображениями формул в формате 0000000.png, где число, содержащееся в названии, соответствовало бы порядку формулы в файле с расшифровками. Затем были убраны лишние изображения, для которых не было расшифровки. Каждый файл был преобразован в 8-битную глубину цвета, а также переделаны размеры изображений таким образом, чтобы параметры ширины и высоты были бы кратными 32. Это было необходимо, чтобы код, который описывает обучение LaTeX OCR, не выдавал ошибку. Кроме того, минимальные и максимальные значения ширины и высоты были использованы в конфигурационном файле config.yaml, пример которого выложен на гитхабе LaTeX OCR, который необходимо было переделать под новую задачу и изменить некоторые строки. В нём же указывается сколько эпох обучения планируется. Используемое соотношение тренировочных данных к валидационным: 9 к 1.

B. Инструменты, использованные в процессе обучения

Обучение LaTeX OCR очень ресурсозатратно, поэтому было принято решение использовать сервис Yandex Cloud [5]. На виртуальной машине на платформе Intel Ice Lake, ЦПУ 32, ОС Ubuntu 22.04 было запущено 2 эксперимента по обучению, о результатах описано в следующем пункте.

Код, который написан для LaTeX OCR, использует для визуализации обучения сайт Wandb [2], на который автоматически посылаются данные о прохождении каждой эпохи, данные о полученных статистиках, что помогает разработчикам отслеживать процесс обучения.

C. Результаты

Нами было решено запустить 20 эпох обучения на 230000 изображений. Настройки обучения были изменены только необходимые, а именно параметры минимальных и максимальных размеров изображений и пути до файлов со значениями формул в LaTeX и к словарю. На Рисунке 4 представлена визуализация результатов. Bleu score - это параметр, показывающий, насколько близок машинный перевод формулы к

человеческому, то есть чем он выше, тем лучше. Данные выгружены с сайта Wandb.

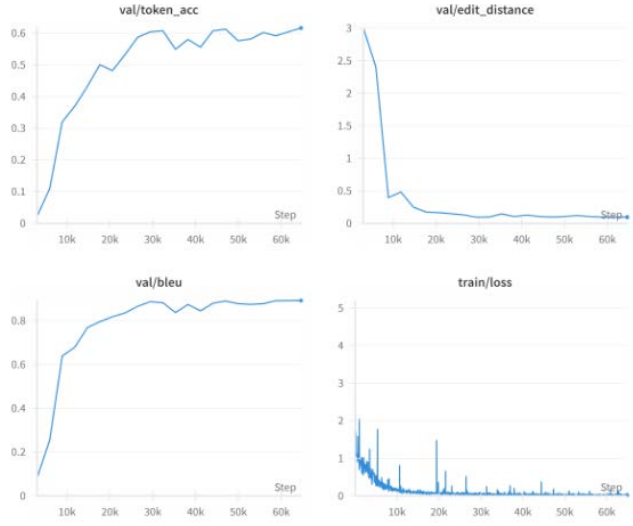

Рис. 4 - Графики для первого обучения LaTeX OCR. Слева сверху график точности (accuracy), справа сверху - расстояние редактирования (edit distance), слева снизу - bleu score, справа снизу - график потерь.

Таблица VI - Пример распознавания формул первой версии обучения LaTeX OCR.

Truth	Prediction
$\delta A_\mu^{(0)} = \tilde{D}_\mu \epsilon^{(0)}, \quad \delta A_\mu^{(n)} = 0$	$\delta A_\mu^{(0)} = \tilde{D}_\mu \epsilon^{(0)}, \quad \delta A_\mu^{(n)} = 0$
$\Gamma[\phi] = \Gamma_0[$	
$\mathrm{bar}\phi] + \hbar\Gamma_1[\phi] + O(\hbar^2)$	$\Gamma[\vec{\phi}] = \Gamma_0[\vec{\phi}] + \hbar\Gamma_1[\vec{\phi}] + O(\hbar^2)$
$\partial_- J_+ - \partial_+ J_- = -m^2 F_{-+}$	$\partial_- J_+ - \partial_+ J_- = -m^2 F_{-+}$
$ds^2 = -e^{2\nu(r)} dudv + r^2 d\varphi^2$	$ds^2 = -e^{2\nu(r)} dudv + r^2 d\varphi^2$
$r_{np}(\tau, \tilde{\lambda}_n) = \tau^{-(n+3)}\, e^{-\tilde{\lambda}_n \tau}\, r^0_{np}\,.$	$r_{np}(\tau, \tilde{\lambda}_n) = \tau^{-(n+3)}\, e^{-\tilde{\lambda}_{np}}\, r^0_{np}\,.$

По результатам обучения было получено несколько файлов весов .pth, сохранённые при прохождении некоторых эпох. Первые 5 эпох сохранялись автоматически, а далее сохранялись только лучшие варианты моделей. В Таблице VI представлены результаты сравнения между ожидаемым значением формулы и найденным. Видно, что обучение прошло успешно, ошибки минимальны. Наилучшие параметры, достигнутые за это обучение: точность (accuracy) - 0.6123, расстояние редактирования (edit distance) - 0.09715, bleu score - 0,8894. В то же время эти параметры у LaTeX OCR: 0.6, 0.1 и 0.8, что позволяло судить об успешности обучения. Данное обучение было произведено за счёт гранта, полученного от Yandex Cloud.

Далее, было принято решение попробовать ещё сильнее улучшить точность, несмотря на то, что графики уже начинали выглядеть полого, что означает, что, возможно, дальше продолжать не стоит. Была гипотеза, что на данном этапе они выглядят так, но дальше будет лучше. Для улучшения модели в коде LaTeX OCR была обнаружена возможность, как поставить флаг дообучения, однако этот способ не сработал и обучение запускалось с нуля, игнорируя флаг. Тогда было решено провести повторное обучение с нуля системы, но уже на 100 эпох при тех же вводных данных. Графики с результатами представлены на Рисунке 5. Можно увидеть, что линии на них более хаотичные, чем в предыдущей версии, однако общая тенденция к возрастанию или убыванию у них схожая. Результат сравнения ожидаемого значения формул и распознанного представлен в Таблице VII. Можно заметить, что довольно сложные формулы прекрасно распознаются.

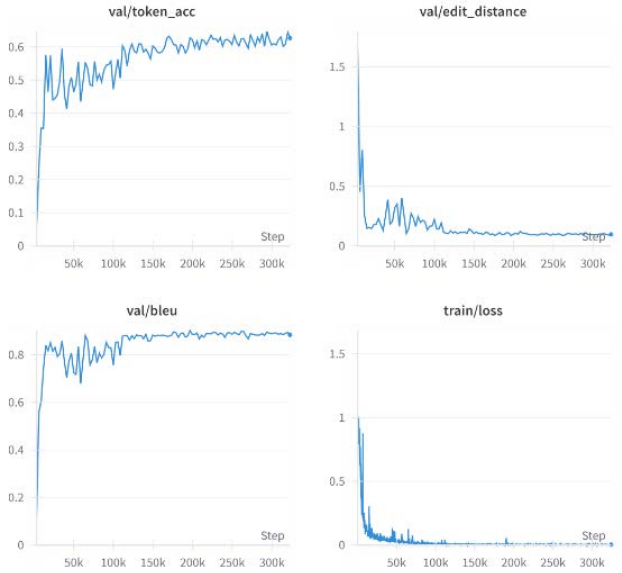

Рис. 5 - Графики для второго обучения LaTeX OCR. Слева сверху график точности (accuracy), справа сверху - расстояние редактирования (edit distance), слева снизу - bleu score, справа снизу - график потерь.

Таблица VII - Пример распознавания формул второй версии обучения LaTeX OCR.

Truth	Prediction
$\delta A_\mu^{(0)} = \tilde{D}_\mu \epsilon^{(0)}, \quad \delta A_\mu^{(n)} = 0$	$\delta A_\mu^{(0)} = \tilde{D}_\mu \epsilon^{(0)}, \quad \delta A_\mu^{(n)} = 0$
$\Gamma[\phi] = \Gamma_0[\phi] + \hbar\Gamma_1[\phi] + O(\hbar^2)$	$\Gamma[\vec{\phi}] = \Gamma_0[\vec{\phi}] + \hbar\Gamma_1[\vec{\phi}] + O(\hbar^2)$
$\partial_- J_+ - \partial_+ J_- = -m^2 F_{-+}$	$\partial_- J_+ - \partial_+ J_- = -m^2 F_{-+}$
$ds^2 = -e^{2\nu(r)} dudv + r^2 d\varphi^2$	$ds^2 = -e^{2\nu(r)} dudv + r^2 d\varphi^2$
$r_{np}(\tau, \tilde{\lambda}_n) = \tau^{-(n+3)}\, e^{-\tilde{\lambda}_n \tau}\, r^0_{np}\,.$	$r_{np}(\tau, \tilde{\lambda}_n) = \tau^{-(n+3)}\, e^{-\tilde{\lambda}_{np}}\, r^0_{np}\,.$
$dq(t) = [A(t) + B(t)q(t)]dt + \nu^{1/2}(t)dw(t)\,.$	$dq(t) = [A(t) + B(t)q(t)]dt + \nu^{1/2}(t)dw(t)\,.$
$p \in D(q) \equiv \{000 \circ p' \mid U(p', q) \text{ is defined}\}\,,$	$p \in D(q) \equiv \{000 \circ p' \mid U(p', q) \text{ is defined}\}\,,$
$\Phi^m \Phi^m = \rho^2 \cdot \mathbf{1}_{n\times n}\,, \qquad \Phi^A \Phi^A = v^2 \cdot \mathbf{1}_{n\times n}\,.$	$\Phi^m \Phi^m = \rho^2 \cdot \mathbf{1}_{n\times n}\,, \qquad \Phi^A \Phi^A = v^2 \cdot \mathbf{1}_{n\times n}\,.$
$b = 2\,, \quad 2a = 2 + b\,, \quad \text{and} \quad c = b - a + 1\,,$	$b = 2\,, \quad 2a = 2 + b\,, \quad \text{and} \quad c = b - a + 1\,,$
$z^1 = x^4 + ix^5,\ z^2 = x^6 + ix^7,\ z^3 = x^8 + ix^9.$	$z^1 = x^4 + ix^5,\ z^2 = x^6 + ix^7,\ z^3 = x^8 + ix^9.$

Параметры лучшей эпохи обучения: точность (accuracy) - 0.6469, расстояние редактирования (edit distance) - 0.09406, bleu score - 0,8954. Для наглядности сравнение всех версий приведено в Таблице VIII. Обучение производилось платно за счёт пожертвований участников паблика «Ёжик в матане».

Таблица VIII - Лучшие параметры моделей для распознавания формул. Таблица содержит значения точности (accuracy), расстояния редактирования (edit distance) и BLEU score для двух обучений и сравнение с LaTeX OCR

Модель	Точность (accuracy)	Расстояние редактирования (edit distance)	BLEU score
LaTeX OCR	0.6000	0.10000	0.8000
Обучение 1	0.6123	0.09715	0.8894
Обучение 2	0.6469	0.09406	0.8954

При учёте позитивных статистических изменений, воспользоваться обновлёнными моделями LaTeX OCR так и не удалось. При использовании результатов обучений, модели выдавали результаты, которые совершенно не похожи на то, что пользователь ожидает увидеть. Нельзя описать это как небольшие ошибки, это именно что-то очень далёкое от реальности, скорее даже результат вообще не имеет ничего общего с реальностью. По нашему мнению, это произошло по причине переобучения модели, она слишком хорошо натренировалась на наборе данных, что на новых изображениях не может справиться со своей задачей.

Была попытка поиска файла .pth без признаков переобучения и имеющем меньшую точность. Для этого перебором проверялись файлы, полученные после более ранних эпох. К сожалению, не нашлось варианта модели, который был бы лучше LaTeX OCR статистически и при этом не был переобучен.

Из полученных результатов следует вывод, что тестировать на сторонних данных модель надо было еще после первого обучение.

VII. Выбор датасета и комплексная разработка моделей для сегментации изображений

A. *Выбор подходящего датасета*

Перед началом работы над созданием модели распознавания LaTeX формул был проведён тщательный анализ потенциальных источников данных для обучения. В результате, было принято решение сформировать датасет, который включал бы в себя пары LaTeX формул и соответствующих им изображений. Эти данные были разбиты на тренировочный, валидационный и тестовый наборы. Источником для формул послужили LaTeX-исходники, взятые из открытых архивов сайта arXiv.

В процессе создания датасета первым шагом стало извлечение формул из материалов KDD Cup 2003, содержащего статьи по физике и математике. Эти статьи содержали математические выкладки, представленные в формате .tex, что хорошо подходило для данной задачи. С помощью регулярных выражений из текстов были извлечены формулы, при этом отсеивались ненужные паттерны и игнорировались несоответствующие выражения.

$$S_0 = \sum_l \frac{1}{2\Delta_l^2} \mathrm{Tr}\, \phi_l^a \phi_{-l}^a + \sum_l \frac{1}{2\epsilon_l^2} \mathrm{Tr}\, f_l^a f_{-l}^a + \sum_r \frac{1}{g_r} \mathrm{Tr}\, \bar{\psi}_r^a \psi_r^a \,.$$

```
S_{0} = \sum_{l} \frac{1}{2\Delta_{l}^{2}} \mathrm{Tr}\, \phi_{l}^{a} \phi_{-l}^{a} + \sum_{l} \frac{1}{2\epsilon_{l}^{2}} \mathrm{Tr}\, f_{l}^{a} f_{-l}^{a} + \sum_{r} \frac{1}{g_{r}} \mathrm{Tr}\, \bar{\psi}_{r}^{a} \psi_{r}^{a} \,.
```

Рис. 6 - Пример разбиения датасета на пары вида картинка-формула.

Следующим этапом была нормализация формул: токенизация и разбор с использованием библиотеки KaTeX, удаление избыточных токенов и приведение всего к единому формату. Для преобразования текстовых формул в визуальный формат, в начале использовалась команда XeLaTeX для конвертации их в PDF, которая обеспечивает гибкую поддержку современных шрифтов и международных языков и улучшает обработку документов. Затем, с помощью инструмента ImageMagick, эти PDF-файлы были преобразованы в изображения формата PNG. Полученные изображения были организованы и индексированы в базе данных, присваивая каждой формуле уникальный идентификатор. Таким образом получился датасет, состоящий из 100 тысяч изображений, однако данный подход не был применен в дальнейшей работе по причине нахождения более эффективных способов.

B. *Анализ архитектуры encoder-decoder*

В процессе разработки системы распознавания формул LaTeX была также изучена архитектура кодировщика-декодировщика. Изначально, кодировщик выполняет стандартизацию изображений, приводя их к единообразному размеру и диапазону значений пикселей от -0.5 до 0.5, что упрощает последующую обработку. После этого, свёрточная нейронная сеть (CNN) анализирует подготовленное изображение и формирует сетку визуальных признаков A, преобразуя её в матрицу векторов признаков, отражающих ключевые области изображения. Эти векторы объединяются с использованием метода пулинга, что ведёт к уменьшению размерности данных и созданию объединённого вектора признаков для каждой зоны. В конечном итоге, карта признаков превращается в одномерную последовательность a.

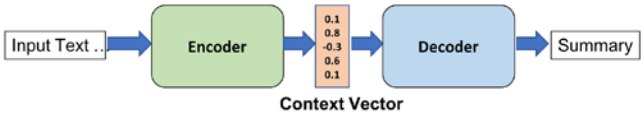

Рис. 7 - Схема Encoder-Decoder архитектуры.

Декодировщик, в свою очередь, применяет рекуррентную нейронную сеть (RNN) для обработки полученной закодированной информации, а также предыдущих слов, чтобы генерировать последующие элементы выходной последовательности. Стек LSTM, занимающийся учетом долговременных зависимостей в данных, в сочетании с моделью внимания, позволяет фокусироваться на соответствующих участках изображения. Таким образом, система последовательно предсказывает следующее слово кода LaTeX. В дополнение, модель начального состояния создаёт исходные данные для RNN, а матрица встраивания трансформирует выходные вероятности в плотные векторы представления.

C. Сверточные и рекуррентные нейронные сети

Был проведен подробный анализ работы сверточных (CNN) и рекуррентных (RNN) нейронных сетей с последующей целью применения этих знаний и навыков при дальнейшем выполнении работы.

Говоря о сверточных нейронных сетях (CNN), они представляют собой архитектуру, которая эффективно распознает различные образы. Их основной принцип работы заключается в применении свертки и субдискретизации для извлечения и обобщения признаков из входных данных. Сверточные нейронные сети состоят из двух основных типов слоев: сверточных и субдискретизирующих. Сверточный слой – это ключевой компонент CNN. Он выполняет операцию свертки над данными, поступающими из предыдущего слоя. Операция свертки включает в себя применение ядра к входным данным для извлечения признаков, что позволяет сети выявлять важные элементы изображений. Субдискретизирующий слой - это слой, который уменьшает размерность карт признаков, полученных на предыдущем сверточном слое. Это помогает уменьшить количество параметров и вычислительную нагрузку, сохраняя при этом важную информацию о признаках.

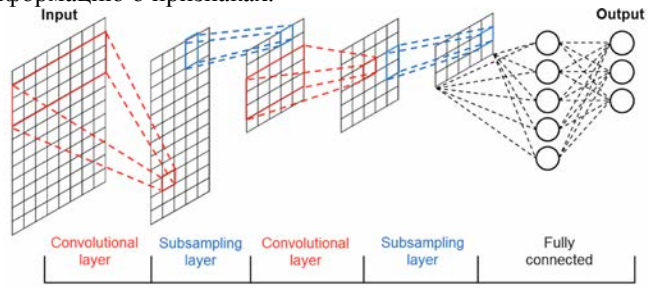

Рис. 8 – Пример одной из схем сверточной нейронной сети.

Рекуррентные нейронные сети (RNN) представляют собой архитектуру, в которой связи между элементами образуют направленные циклы. В отличие от обычных нейронных сетей, рекуррентные сети передают свои выходные данные обратно на свои входы. Эта обратная связь позволяет сети сохранять и использовать информацию из предыдущих шагов, что делает RNN особенно полезными для задач, связанных с последовательными данными, такими как обработка текста или временных рядов.

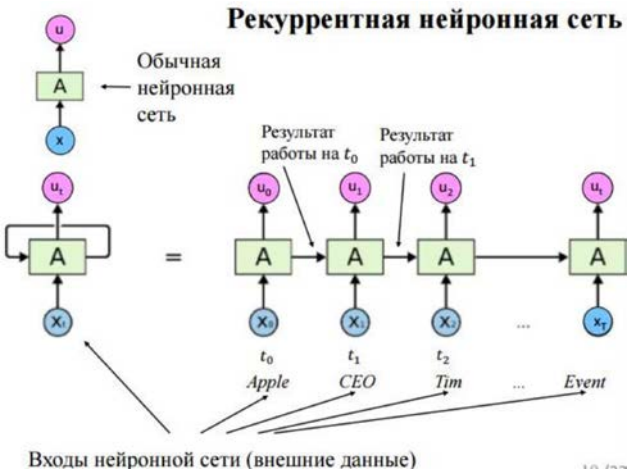

Рис. 9 - Схема рекуррентной нейронной сети.

D. Разметка данных в CVAT: процесс, преимущества и сравнение с альтернативными методами

Разметка данных играет ключевую роль в разработке и обучении моделей машинного обучения, особенно в задачах компьютерного зрения. Computer Vision Annotation Tool (CVAT), разработанный Intel, представляет собой открытое программное обеспечение для аннотирования видео и изображений. Этот инструмент предлагает удобный интерфейс для точной разметки объектов, что является критически важным фактором для обучения точных и эффективных моделей.

Процесс начинается с загрузки изображений или видео на сервер. Затем пользователи могут настраивать проект, определять классы объектов и выбирать различные типы аннотаций, такие как прямоугольники, полигоны и кубоиды. Разметка осуществляется в интерактивном режиме: аннотатор выделяет объекты на изображении и классифицирует их согласно заранее определенным категориям. Объекты можно делить на типы, такие как картинка, формула и текст.

CVAT поддерживает экспорт аннотаций в популярные форматы, такие как PASCAL VOC, YOLO и COCO, широко используемые для обучения моделей машинного обучения. Это позволяет пользователям экспортировать данные в формате, совместимом с их инструментами и библиотеками машинного обучения. Преобразование аннотаций в требуемые структуры данных осуществляется, например, через преобразование в JSON-файлы для COCO или текстовые файлы для YOLO.

Основное преимущество CVAT — его гибкость. Платформа позволяет настраивать проекты под конкретные задачи, определяя нужные классы объектов и выбирая методы аннотации. Это обеспечивает точную адаптацию к специфике данных проекта. Коллективная работа над проектами ускоряет процесс аннотации и способствует более эффективному распределению рабочей нагрузки. Интеграция с предобученными моделями машинного обучения автоматизирует и ускоряет процесс аннотации, сокращая время подготовки данных.

Однако для повышения точности модели требуется значительное увеличение объема датасета. Работа с несколькими аннотаторами требует строгого контроля за единообразием разметки, что может потребовать дополнительных усилий. Качество аннотации напрямую влияет на производительность модели, и ошибки в данных могут серьезно сказаться на её эффективности. Разметка данных — процесс, требующий времени, и при небольшом количестве аннотаторов подготовить объемный и качественный датасет сложнее. Важно также, чтобы датасет отражал всё многообразие сценариев, с которыми модель столкнётся в реальной жизни.

При работе над проектами в области машинного обучения и компьютерного зрения, доступ к качественным данным является ключевым. Например, два других популярных метода это:

Публично доступные датасеты: эти датасеты предлагают удобный доступ к обширным наборам данных, что значительно экономит время и ресурсы. Они особенно полезны для начальных этапов исследования или когда требуется быстро проверить гипотезу. Однако, такие датасеты могут не всегда точно соответствовать специфическим требованиям проекта, что ограничивает их применимость в специализированных задачах.

Веб-скрапинг: этот метод позволяет собирать данные напрямую из интернета, что идеально подходит для получения больших объемов разнообразной информации. Однако, веб-скрапинг связан с определенными рисками, такими как юридические ограничения и вопросы этичности, а также требует тщательной проверки и обработки данных, чтобы обеспечить их качество и релевантность.

Computer Vision Annotation Tool (CVAT) зарекомендовал себя как один из лучших инструментов в сфере аннотации данных благодаря своей способности обеспечивать высокий контроль над качеством и точностью данных. Особенностью CVAT является возможность тщательной настройки процесса аннотации в соответствии с требованиями каждого проекта, а также автоматизация многих процессов, что особенно важно в задачах компьютерного зрения. Перечисленные функции делают CVAT подходящим инструментом для нашего проекта, где основной задачей было достижение высокой точности обработки данных в условиях большой вариабельности документов. Используя CVAT была создана основа для разработки эффективных и точных моделей машинного обучения, способных справляться с широким спектром задач.

VIII. Тестирование и результаты

После завершения настройки системы, было проведено тестирование работоспособности модели. Тестирование проводилось на основе 150 картинок, в процессе была оценена эффективность, точность и надежность модели, в результате чего были выявлены как сильные, так и слабые стороны ее работы.

A. Основные сильные стороны

Точность воспроизведения математических символов и формул.

Начиная с явных преимуществ этой модели, стоит отметить, что она достаточно эффективно преобразует текст, представленный на изображениях, в LaTeX формат, особенно хорошо это видно при работе с базовыми математическими уравнениями.

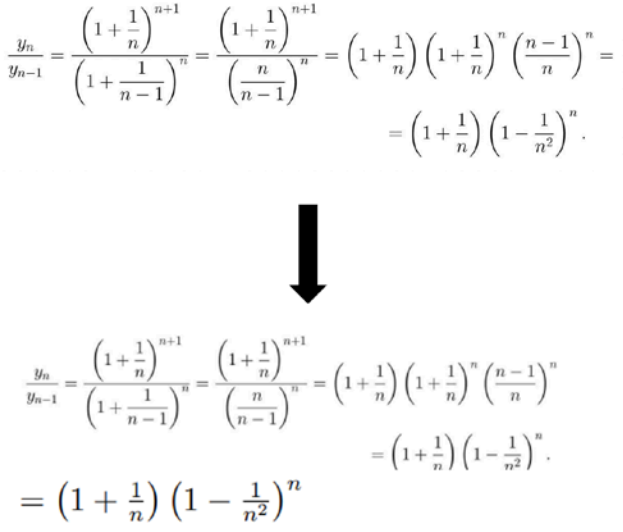

Рис. 10 – Точность математических представлений.

Помимо основных математических формул (Рис. 10 модель также достаточно точно распознает специфические математические символы, такие как знаки модуля, обозначения пределов и буквы греческого алфавита (Рис. 11).

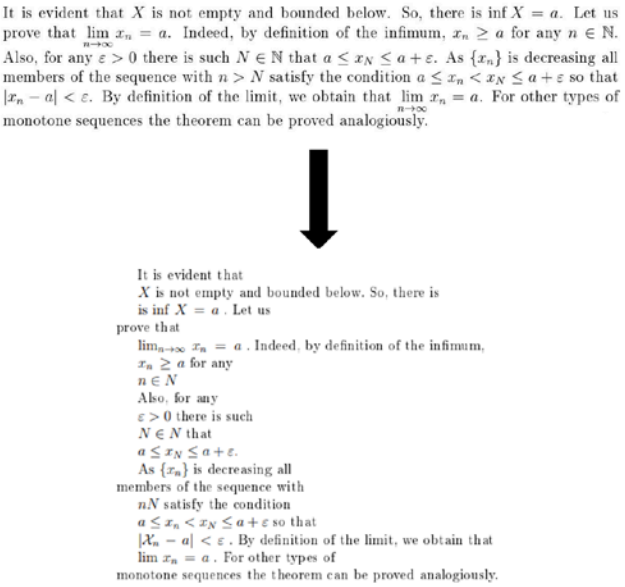

Рис. 11 – Воспроизведение символов и формул.

Эффективность обработки различных объемов текста.

На основе представленных выше Рисунках 10 и 11, можно также сделать вывод, что модель способна обрабатывать картинки как с небольшим количеством текста, так и более объемные текстовые изображения. При этом для модели не имеет значения, присутствуют

ли на картинке только математические формулы или же они даны совместно с текстовой частью.

Высокая производительность.

Еще одним немаловажным преимуществом модели является ее эффективность, ответ на запрос пользователя поступает в течение нескольких секунд. Однако время ожидания ответа может быть увеличено в зависимости от объема исходного изображения и нагрузки сервера. Тем не менее, модель достаточно быстро справляется с поставленной задачей.

Способность интегрирования сложных комппонентов.

Данная модель показывает возможность распознавания и обработки сложных элементов, таких как таблицы, различные рисунки и графики. Эти факторы свидетельствуют о гибкости модели в использовании, и ее способности адаптироваться под различные типы входных данных.

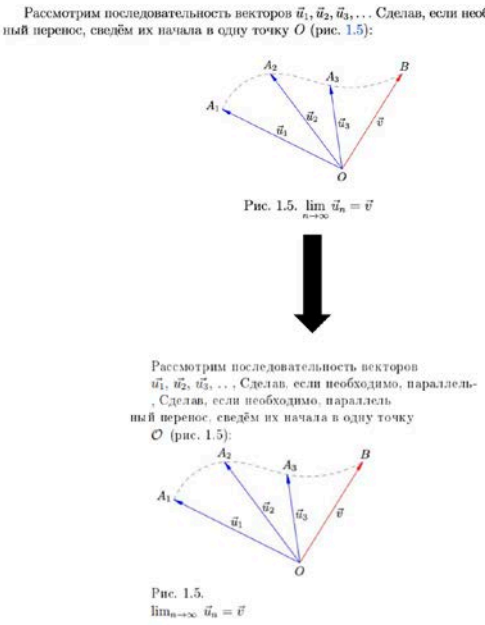

Рис. 12 – Интеграция сложных компонентов.

B. Основные слабые стороны

Ошибки форматирования.

Данная модель может некорректно интерпретировать команды выравнивания в LaTeX, что приводит к смещению текста или математических формул относительно ожидаемой позиции. Это может существенно снижать читаемость документа, особенно в сложных академических работах, где точное выравнивание важно для понимания структуры документа. Более того, некорректное применение или полное игнорирование отступов в тексте может привести к нарушению структуры абзацев и разделов, что затрудняет восприятие информации и делает текст визуально менее читаемым.

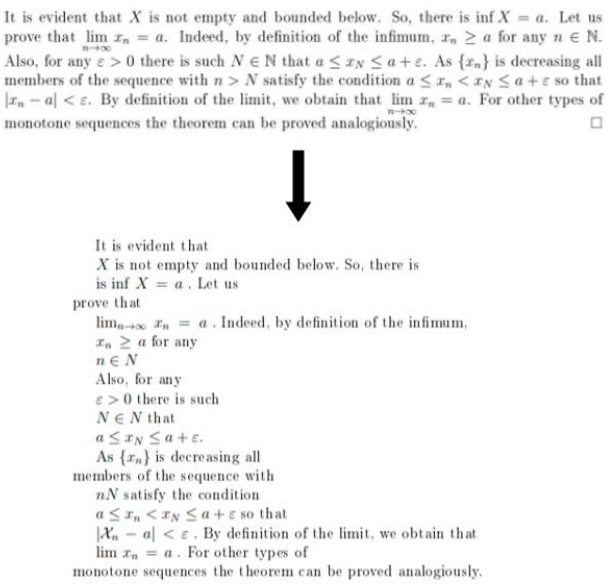

Рис. 13 – Неправильное выравнивание.

Ошибки в обработке команд для стилизации текста, такие как курсив или жирный шрифт, могут привести к потере важных акцентов в тексте. Например, в академических текстах часто используется выделение для указания специфических терминов, поэтому ошибки здесь могут исказить первоначальное намерение автора.

Рис. 14 – Игнорирование стилизации.

Ошибки с Перекодировкой Символов и сложные элементы форматирования.

Модель может некорректно обрабатывать специальные или региональные символы, особенно если они встречаются редко. Чаще всего это отображается на документах, содержащие технические или научные символы, которые должны быть точно воспроизведены для сохранения смысла.

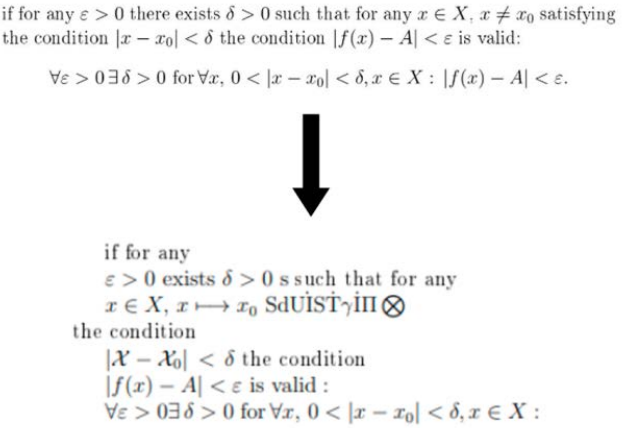

Рис. 15 – Сложные элементы форматирования.

Проблемы с кодировками могут привести к тому, что символы отображаются как неразборчивые знаки, что делает текст нечитаемым.

Анализ слабых сторон модели подчеркивает необходимость точности и надежности в обработке академического и технического текста. Ошибки в форматировании, стилях шрифтов, перекодировке символов и верстке ухудшают визуальное восприятие текста и могут существенно исказить смысл передаваемой информации, что является большой проблемой. Это подчеркивает важность усовершенствования существующих методов обработки данных, чтобы обеспечить высокий уровень точности системы. Также необходимо уделить особое внимание обучению модели на более разнообразных и комплексных данных, что поможет улучшить её способность адаптироваться к различным форматам и структурам текста.

IX. Заключение

В рамках данного проекта была разработана система автоматизации документооборота на базе искусственного интеллекта, с помощью которой можно преобразовывать отсканированные одностраничные или многостраничные отсканированные файлы с печатным текстом в код LaTeX. По итогам работы над проектом были достигнуты такие результаты как: созданы удобные формы работы с программным решением; разработаны новые модели детекции, классификации и сегментации на основе YOLOv8, которые показали высокую эффективность в работе; обеспечена возможность обработки как простых текстовых документах на двух языках, так и сложных, содержащих в себе формулы и графики, что значительно упрощает работу над созданием или редактированием научных и технических документов.

Отрицательные результаты работы нашей команды: выявлены проблемы с корректностью форматирования распознанного текста в некоторых случаях, что требует дальнейшей доработки; имеются ошибки в распознавании некоторых сложных символов, из-за чего снижается читаемость итогового документа; попытки улучшить качество распознавания формул не увенчались успехом.

Перспективы дальнейшего развития проекта: доработка моделей для улучшения точности распознавания отдельных частей отсканированного изображения; включение возможности распознавания рукописного текста, что расширит сферы применения нашего продукта; добавление дополнительных языков для удобства различных категорий пользователей; улучшение производительности и скорости обработки данных за счёт модернизации и оптимизации кода и использования более мощных серверных решений; дальнейшее улучшение интерфейса веб-приложения, добавление новых полезных функций.

Таким образом, наша работа является важным шагом на пути к созданию доступных и эффективных решений для перевода печатного текста в LaTeX формат и обладает значительным потенциалом для дальнейшего улучшения и развития.

Благодарности

Авторы выражают благодарность Кокориной М.В., Лазаревой Е.Н. и Селезнёву Г.И. за ценные обсуждения, помощь в выборе подходящего датасета, разметке данных, разработке телеграмм-бота и сайта проекта.

Библиография

[1] AWS. AWS Free Tier. 2024. url: https://aws.amazon.com/ru/free/ (дата обр. 22.05.2024).
[2] Weights Biases. Weights Biases. 2024. url: https://wandb.ai/site (дата обр. 24.04.2024).
[3] Lukas Blecher. LaTeX OCR. 2024. url: https://github.com/lukas-blecher/LaTeX-OCR (дата обр. 22.05.2024).
[4] ChatGPT. ChatGPT: AI Language Model. url: https://chatgpt.com (дата обр. 18.05.2024).
[5] Yandex Cloud. Yandex Cloud. 2024. url: https://yandex.cloud/ru/?utm_referrer=https%3A%2F%2Fwww.google.com%2F (дата обр. 24.04.2024).
[6] CVAT. CVAT: Computer Vision Annotation Tool. 2024. url: https://www.cvat.ai (дата обр. 22.05.2024).
[7] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling и Alexander M. Rush. “Image-to-Markup Generation with Coarse-to-Fine Attention”. В: arXiv preprint (2016). url: https://arxiv.org/pdf/1609.04938.pdf.
[8] Docker. Compose: Define and Run Multi-Container Applications. 2024. url: https://docs.docker.com/compose/ (дата обр. 22.05.2024).
[9] Ankush Hafizov. CVAT2YOLO: Convert CVAT annotations to YOLO format. 2024. url:https://github.com/ankhafizov/CVAT2YOLO (дата обр. 22.05.2024).
[10] Yann LeCun, Leon Bottou, Yoshua Bengio и Patrick Haffner. “Gradient-Based Learning Applied to Document Recognition”. В: Proceedings of the IEEE 86.11 (1998), с. 2278—2324. doi: 10.1109/5.726791. url:http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf.
[11] Mathpix. Mathpix: Convert Images to LaTeX, AsciiMath, MathML, and more. url: https://mathpix.com (дата обр. 10.05.2024).
[12] Erik G. Miller и Paul A. Viola. “Ambiguity and Constraint in Mathematical Expression Recognition”. В: Proceedings of the AAAI Conference on Artificial Intelligence. 1998. url: https://people.cs.umass.edu/~elm/papers/AAAI-98.pdf.
[13] Norm. nougat-latex-base. 2024. url: https://huggingface.co/Norm/nougat-latex-base (дата обр. 22.05.2024).
[14] Faisal Shafait, Daniel Keysers и Thomas M. Breuel. “Performance Comparison of Six Algorithms for Page Segmentation”. В: Lecture Notes in Computer Science. Т. 3872. Springer, 2006, с. 368—379. doi: 10.1007/11669487_33. url: https://link.springer.com/chapter/10.1007/11669487_33.
[15] Sumeet Sohan Singh. “Teaching Machines to Code: Neural Markup Generation with Visual Attention”. В: arXiv preprint (2018). url: https://arxiv.org/abs/1802.05415.
[16] Ultralytics. YOLOv8: Real-Time Object Detection and Segmentation. 2024. url: https://ultralytics.com/yolov8 (дата обр. 21.05.2024).
[17] Zenodo. Record 56198. 2024. url: https://zenodo.org/records/56198#.V2px0jXT6eA (дата обр. 22.05.2024).
[18] Zenodo. Record 7738969. 2024. url: https://zenodo.org/records/7738969 (дата обр. 20.05.2024).

Агафьина Анастасия Дмитриевна родилась в г. Сергиев Посад 21.08.1995. В 2013 году окончила Физико-математический лицей г. Сергиева Посада. С 2022 года является студентом факультета ФКН НИУ ВШЭ. Область научных интересов: машинное обучение, распознавание данных.

Никитин Алексей Антонович родился в г. Москва 14.02.1983, В 2000 году окончил Лицей информационных технологий №1533, в 2005 году -- факультет ВМиК МГУ им. М.В. Ломоносова, в 2008 году

защитил диссертацию на соискание степени к.ф.-м.н. на тему «Граничное управление третьим краевым условием» под руководством В.А. Ильина. С 1 мая 2008 года работает на кафедре Общей математики факультета ВМиК МГУ в должности ассистента, и доцента с октября 2013 года. В 2009 - 2021 годах работал в НИУ ВШЭ в должности доцента департамента математики. С 2019 года работает в совместном университете МГУ-ППИ, г. Шэньчжэнь. Область научных интересов: математическое моделирование, нелинейные интегральные уравнения, математическая биология, вопросы модернизации и информатизации системы образования, использование визуальных образов в процессе образования.

About automatic recognition of printed text

A. D. Agafina, A. A. Nikitin

Abstract. **This article examines the internal structure of a system for recognizing printed data and converting it into a convenient format for use. The developed project provides a solution for converting images and PDF files containing printed text and formulas into LaTeX format by using free, open-source libraries and a custom-trained model for data classification, detection and segmentation.**

This study provides a detailed overview of the system development stages starting from the analysis of existing algorithms to the creation of a custom model for text and formula recognition. The focus is on selecting tools and libraries suitable for tasks related to the automation of recognizing and converting printed documents into LaTeX format and on attempts to improve the performance of these libraries. The process of creating and preparing the dataset for model training which includs image and text annotation. The outcomes of the model training are presented in tables showing the achieved recognition accuracies for various data classes.

The article presents the results of testing the system's functionality for recognizing text and formulas in both Russian and English. It also details the strengths and weaknesses of the developed system and the challenges encountered during its development. Based on our development, we have created a website that includes an interface for converting images and PDF files into LaTeX format, as well as a Telegram bot with similar functionality.

***Keywords*— computer vision, machine learning, document recognition, data segmentation, LaTeX, OCR.**

REFERENCES

[1] AWS. AWS Free Tier. 2024. url: https://aws.amazon.com/ru/free/ (дата обр. 22.05.2024).
[2] Weights Biases. Weights Biases. 2024. url: https://wandb.ai/site (дата обр. 24.04.2024).
[3] Lukas Blecher. LaTeX OCR. 2024. url: https://github.com/lukas-blecher/LaTeX-OCR (дата обр. 22.05.2024).
[4] ChatGPT. ChatGPT: AI Language Model. url: https://chatgpt.com (дата обр. 18.05.2024).
[5] Yandex Cloud. Yandex Cloud. 2024. url: https://yandex.cloud/ru/?utm_referrer=https%3A%2F%2Fwww.google.com%2F (дата обр. 24.04.2024).
[6] CVAT. CVAT: Computer Vision Annotation Tool. 2024. url: https://www.cvat.ai (дата обр. 22.05.2024).
[7] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling и Alexander M. Rush. “Image-to-Markup Generation with Coarse-to-Fine Attention”. B: arXiv preprint (2016). url: https://arxiv.org/pdf/1609.04938.pdf.
[8] Docker. Compose: Define and Run Multi-Container Applications. 2024. url: https://docs.docker.com/compose/ (дата обр. 22.05.2024).
[9] Ankush Hafizov. CVAT2YOLO: Convert CVAT annotations to YOLO format. 2024. url:https://github.com/ankhafizov/CVAT2YOLO (дата обр. 22.05.2024).
[10] Yann LeCun, Leon Bottou, Yoshua Bengio и Patrick Haffner. “Gradient-Based Learning Applied to Document Recognition”. B: Proceedings of the IEEE 86.11 (1998), с. 2278—2324. doi: 10.1109/5.726791. url:http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf.
[11] Mathpix. Mathpix: Convert Images to LaTeX, AsciiMath, MathML, and more. url: https://mathpix.com (дата обр. 10.05.2024).
[12] Erik G. Miller и Paul A. Viola. “Ambiguity and Constraint in Mathematical Expression Recognition”. B: Proceedings of the AAAI Conference on Artificial Intelligence. 1998. url: https://people.cs.umass.edu/~elm/papers/AAAI-98.pdf.
[13] Norm. nougat-latex-base. 2024. url: https://huggingface.co/Norm/nougat-latex-base (дата обр. 22.05.2024).
[14] Faisal Shafait, Daniel Keysers и Thomas M. Breuel. “Performance Comparison of Six Algorithms for Page Segmentation”. B: Lecture Notes in Computer Science. T. 3872. Springer, 2006, с. 368—379. doi: 10.1007/11669487_33. url: https://link.springer.com/chapter/10.1007/11669487_33.
[15] Sumeet Sohan Singh. “Teaching Machines to Code: Neural Markup Generation with Visual Attention”. B: arXiv preprint (2018). url: https://arxiv.org/abs/1802.05415.
[16] Ultralytics. YOLOv8: Real-Time Object Detection and Segmentation. 2024. url: https://ultralytics.com/yolov8 (дата обр. 21.05.2024).
[17] Zenodo. Record 56198. 2024. url: https://zenodo.org/records/56198#.V2px0jXT6eA (дата обр. 22.05.2024).
[18] Zenodo. Record 7738969. 2024. url: https://zenodo.org/records/7738969 (дата обр. 20.05.2024)