# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc197811781)

[ВВЕДЕНИЕ 5](#_Toc197811782)

[Глава 1. Обзор проблемы машинного распознавания реквизитов банковских карт 9](#_Toc197811783)

[1.1 Ретроспективный анализ методов обнаружения и распознавания текста 9](#_Toc197811784)

[1.2 Существующие методы и готовые решения 12](#_Toc197811785)

[1.3 Методы, применяемые для решения схожих задач 15](#_Toc197811786)

[Глава 2. Теоретические основы детектирования объектов на изображении и оптического распознавания текста 17](#_Toc197811787)

[2.1 Описание исходных данных 17](#_Toc197811788)

[2.1.1 Наборы и форматы данных для обучения моделей компьютерного зрения 17](#_Toc197811789)

[2.2.2 Набор данных для обучения модели Ultralytics YOLO c текстовыми аннотациями 19](#_Toc197811790)

[2.2.3 Формат данных COCO с аннотациями в формате JSON 22](#_Toc197811791)

[2.3.4 Набор данных настоящего исследования 24](#_Toc197811792)

[2.2 Сверточные нейронные сети в задачах детекции объектов 27](#_Toc197811793)

[2.2.1 Общие сведения о нейронных сетях 27](#_Toc197811794)

[2.2.2 Детектирование объектов 29](#_Toc197811795)

[2.2.3 Обнаружение объектов в реальном времени с помощью Faster R-CNN 32](#_Toc197811796)

[2.2.4 Обнаружение объектов в реальном времени с помощью YOLOv10 38](#_Toc197811797)

[2.3 Доступные решения для оптического распознавания текста 42](#_Toc197811798)

[2.3.1 Технологии оптического распознавания текста 42](#_Toc197811799)

[2.3.2 OCR библиотека Tesseract 44](#_Toc197811800)

[2.3.3 OCR библиотека EasyOCR 44](#_Toc197811801)

[2.3.4 OCR библиотека KerasOCR 45](#_Toc197811802)

[2.3.5 OCR библиотека TrOCR 46](#_Toc197811803)

[2.4 Метрики и методы оценки качества 48](#_Toc197811804)

[2.4.1 Метрики применяемые для оценки качества модели 48](#_Toc197811805)

[2.4.2 Пересечение над объединением (Intersection over Union) 49](#_Toc197811806)

[2.4.3 Метрики Accuracy, Precision, Recall, F1-Score 50](#_Toc197811807)

[2.4.6 Средняя точность (AP) 52](#_Toc197811808)

[2.4.7 Среднее значение точности (mAP) 53](#_Toc197811809)

[Глава 3. Практическое исследование методов детектирования и распознавания реквизитов банковских карт 54](#_Toc197811810)

[3.1 Обучение моделей для детектирования объектов 54](#_Toc197811811)

[3.1.1 Обучение Faster R-CNN 54](#_Toc197811812)

[3.1.2 Обучение YOLOv12 59](#_Toc197811813)

[3.1.3 Сравнение метрик и точности моделей детектирования 63](#_Toc197811814)

[3.2 Распознавание текста с использованием популярных библиотек OCR 65](#_Toc197811815)

[3.2.1 Библиотека KerasOCR 65](#_Toc197811816)

[3.2.2 Библиотека EasyOCR 68](#_Toc197811817)

[3.2.3 Библиотека TrOCR 73](#_Toc197811818)

[3.2.4 Сравнение результатов распознавания 81](#_Toc197811819)

[Когда сравнивать слова, а когда символы? 92](#_Toc197811820)

[символьное сравнение 92](#_Toc197811821)

[Нормализация (удаление пробелов) 93](#_Toc197811822)

[3.3 Пользовательский интерфейс для тестирования исследуемых методов 95](#_Toc197811823)

[ЗАКЛЮЧЕНИЕ 95](#_Toc197811824)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 98](#_Toc197811825)

# ВВЕДЕНИЕ

Со стремительным развитием информационных технологий использование банковских пластиковых карт стало неотъемлемой частью жизни миллионов людей. Пластиковые карты используются для повседневных покупок и стали популярны с ростом электронной торговли. Они обеспечивают безопасный и быстрый способ оплаты товаров и услуг через Интернет, позволяя пользователям совершать покупки, не выходя из дома.

При онлайн покупках пользователи вводят данные своих карт на сайтах и в мобильных приложениях для завершения транзакций. Для упрощения этого процесса многие платформы предлагают сохранить реквизиты карт, что позволяет делать повторные покупки быстрее и удобнее.

Современные подходы к считыванию информации с пластиковых карт, включая использование оптического распознавания символов, часто сталкиваются с трудностями, связанными с различиями в форматах карт, физическим состоянием карт, условиями освещения, состоянием камеры устройства и т.п. Эти проблемы требует особого внимания к точности и скорости распознавания, и это вызывает необходимость углубленного исследования. Существующие методы и технологии не всегда эффективно решают задачи распознавания в разнообразных условиях, что особенно актуально с ростом популярности мобильных устройств. Рассмотрение этих вопросов на уровне научных исследований остается недостаточным, что создаёт пробел в знаниях и подтверждает необходимость дальнейшего изучения.

**Актуальность проблемы** заключается в необходимости разработки технологий, способных обеспечить точное и безопасное считывание реквизитов карт в условиях растущего использования мобильных платежей. Дальнейшее развитие и совершенствование технологий считывания реквизитов не только отвечает требованиям современного рынка, но и способствует формированию безопасной и эффективной финансовой среды [1].

**Гипотезой исследования** является предположение, что использование последних достижений в области нейронных сетей и машинного обучения позволит создать программный продукт, который ускорит процесс обработки платежей и снизит операционные расходы в банках и финансовых организациях, сократит время на ввод данных банковских карт и уменьшит риск ошибок клиентами при совершении платежей, предоставит более эффективный и удобный интерфейс для разработчиков платежных систем на участке обработки данных банковских карт.

**Целью** данной магистерской диссертации является разработка программного продукта для точного распознавания реквизитов банковских карт на изображениях.

Разрабатываемый программный продукт должен в себя включать алгоритмы поиска полей с текстовыми блоками (алгоритмы детекции), алгоритмы оптического распознавания символов, оптимизированные для работы с различными типами карт и различными условиями съемки, механизмы проверки корректности извлеченных данных и интерфейс пользователя.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. провести анализ существующих методов и технологий считывания реквизитов с банковских карт с целью выявления их преимуществ и недостатков. Проанализировать методы схожей по составу задачи по распознаванию автомобильных номеров;
2. рассмотреть виды и устройство сверточных нейронных сетей в задачах детектирования объектов на изображениях и решения для оптического распознавания текста;
3. сформировать исходные данные для обучения модели машинного обучения и выбрать методы расчета метрик оценки качества результата распознавания.
4. обучить три варианта нейронной сети для задачи детектирования полей реквизитов, произвести распознавание текстовых реквизитов в полях тремя разными библиотеками оптического распознавания текста, сравнить полученные результаты.

**Объектом исследования** является метод считывания реквизитов с банковских карт, являющийся одним из этапов процесса взаимодействия пользователя с финансовыми системами.

**Предметом исследования** выступают алгоритмы и технологии детектирования объектов на изображении и оптического распознавания текста, применяемые при считывании информации с пластиковых карт.

Исследование включает в себя изучение методов обработки изображений для повышения точности распознавания. Изучение объекта и предмета исследования позволит разработать усовершенствованные решения для оптимизации работы с банковскими картами. Это также способствует повышению безопасности финансовых операций и улучшению пользовательского опыта.

В данной работе используются разнообразные методы исследования, каждый из которых направлен на более глубокое понимание процессов, связанных с считыванием реквизитов с банковских пластиковых карт. **К числу основных методов относятся**:

* анализ и синтез. Анализ применяется для глубокого изучения существующих технологий детектирования объектов и оптического распознавания символов, а синтез используется для разработки новых подходов и алгоритмов, интегрирующих лучшие практики из разных областей;
* ранжирование. Данный метод применяется для оценки и выбора наиболее эффективных алгоритмов и технологий обработки изображений, а также для определения приоритетов в разработке новых решений.

**Научная новизна исследования** заключается в следующем:

* сравнение различных архитектур нейронных сетей для задачи детектирования полей реквизитов банковских карт;
* сравнение нескольких библиотек оптического распознавания текста для получения текстового содержимого полей реквизитов;
* разработка технологии, обеспечивающей распознавание различных типов карт: эмбоссированных, гравированных, с плоской печатью при различных условиях освещения и съемки;
* создание отечественного продукта способного работать без использования зарубежных сервисов.

Кроме того, техническая сложность разработки может включать:

* сбор и разметка датасета для обучения и тестирования нейронной сети;
* выбор наиболее подходящей архитектуры нейронной сети в условиях работы с ограниченными вычислительными ресурсами;
* реализация алгоритмов, устойчивых к различным условиям освещения и ракурсам съемки.

При написании данной работы использовался материал из 54 источников, ссылки на которые приведены в разделе Список использованных источников. Текст проиллюстрирован 15 рисунками и схемами, содержит 2 таблицы, 10 формул.

# Глава 1. Обзор проблемы машинного распознавания реквизитов банковских карт

## 1.1 Ретроспективный анализ методов обнаружения и распознавания текста

Проблема распознавания данных с банковских карт существует уже много лет и решалась разными способами. В общем случае, задача распознавания реквизитов включает два этапа: обнаружение областей, содержащий текстовые реквизиты на изображении, а затем распознавание символов на выделенной области [16].

До широкого распространения глубоких нейронных сетей для локализации объектов использовались традиционные методы, основанные на ручной настройке признаков и эвристик [3] . Так, широко применялись:

1. анализ границ и контуров [22]. Ранние подходы к обнаружению объектов часто включали поиск прямоугольных областей с высокой плотностью границ или контрастных переходов. Для этого использовались такие методы, как фильтрация по градиенту, контурный анализ и морфологическая обработка. Основным преимуществом этих методов была их относительно высокая скорость при низких вычислительных затратах. Однако они часто испытывали трудности с чувствительностью к условиям освещения, ориентации объекта и сложности фона, что приводило к большому количеству ложных срабатываний и снижению общей точности.
2. Цветовые модели и сегментация по цвету [26]. Другой распространенный подход заключался в использовании цветовых характеристик объектов. Переход в цветовые пространства, такие как HSI, позволял фильтровать неподходящие области и выбирать кандидаты с определенным цветом фона или символов. Однако эти методы были эффективны только в ограниченных условиях из-за зависимости от стабильных цветовых признаков и неустойчивости к изменениям освещения.
3. Текстурные признаки и традиционные классификаторы. Некоторые решения использовали анализ текстуры или локальных признаков для выделения областей, которые, вероятно, могли содержать интересующие объекты. Для этого применялись такие методы, как машины опорных векторов (SVM) или каскадные классификаторы [42, с. 576]. Хотя этот подход иногда обеспечивал более высокую точность по сравнению с простым анализом границ или цвета, он требовал тщательного подбора признаков и высококачественного набора данных для обучения.

С развитием вычислительных устройств, также развивается программное обеспечение, разрабатываются новые алгоритмы, появляются новые подходы к решению прежних задач. Распространение сверточных нейронных сетей (CNN) открыло возможности для более точной и устойчивой локализации объектов, в том числе и областей с текстовыми реквизитами банковских карт. Существует большое количество реализаций таких сетей, что требует исследования их особенностей и возможностей.

Первоначально для обнаружения объектов на изображениях использовались универсальные модели, такие как Faster R-CNN [51] и SSD [57], которые демонстрировали хорошую точность, но часто уступали в скорости и требовали значительных вычислительных ресурсов. Однако появление первой версии YOLO [35] стало переломным моментом: эта модель могла предсказывать ограничивающие рамки и классы объектов за один проход, что существенно увеличило скорость обнаружения объектов и снизило требования к вычислительным мощностям оборудования.

После успешного обнаружения возникает задача извлечения и распознавания символов. Изначально применялись классические OCR-решения, например, Tesseract [38] требующие тщательной̆ предобработки и ручного выделения признаков. Однако с развитием глубокого обучения появились модели, способные работать напрямую с текстовой областью, например, CRNN [37] или архитектуры на основе Transformers [24]. Трансформеры, первоначально разработанные для NLP, успешно применяются и в других областях, включая OCR. Они эффективно справляются с задачами, связанными с последовательностями, что делает их полезными для распознавания текста, особенно в сложных случаях, таких как рукописный текст или текст на сложных фонах.

EasyOCR [53] — это библиотека для распознавания текста, созданная с использованием современных методов глубокого обучения. Она основана на нейронных сетях для выполнения задач оптического распознавания символов (OCR). Хотя EasyOCR в основном использует архитектуру PyTorch и свои собственные модели для обработки изображений, идея интеграции с трансформерами вдохновлена исследованиями в области обработки естественного языка и компьютерного зрения. Хотя в EasyOCR трансформеры не используются в традиционном смысле, современные исследования в области OCR и глубокого обучения стимулируют интеграцию различных методов, включая экспериментальные подходы с трансформерами для повышения точности распознавания текста.

У каждой из двух библиотек Tesseract и EasyOCR есть свои сильные слабые стороны. У EasyOCR нет большого количества программных зависимостей, таких как Tesseract, поддерживает более 80 языков, таких как хинди, английский, китайский, русский и т.д. Библиотека Tesseract по сравнению с EasyOCR поддерживает меньше языков, однако лучше обнаруживает буквы на изображении [40].

Бесплатность, доступность и эффективность программных пакетов и библиотек для детектирования объектов и оптического распознавания текста подталкивают на исследование их возможностей и применения в практических задачах.

## 1.2 Существующие методы и готовые решения

Проанализировав рынок существующих программных продуктов и существующих методов решений поставленной проблемы за период начиная с 2016 года был составлен некоторый перечень. Ознакомимся с ним:

1. библиотека для распознавания банковских карт «Smart CardReader» [19]. Авторские права на библиотеку зарегистрированы в 2016 году. Сотрудники фирмы-разработчика преподают на кафедре когнитивных технологий в МФТИ и имеют многочисленные публикации в высокорейтинговых научных журналах [17]. В библиотеке используются собственные наработки авторов, последователей советской научной школы построения вычислительно эффективных интеллектуальных систем. Используя подходы этой школы, разработчики создают энергоэффективные решения без GPU. Язык разработки C++. Библиотека доступна на Windows, Linux, Mac OS, iOS, Android, Windows Phone;
2. система управления Online МФО.Распознавание карт [45]. Авторские права на программу зарегистрированы в 2019 году. Языки разработки: PHP, JavaScript, С++, Python;
3. в статье, опубликованной на habr.com [12] в конце 2015, для детекции областей данных автор использовал предварительную обработку изображений, а затем в различных разрезах исследовал интенсивность пикселей. После выделения на изображении блоков с данными, эти блоки передавались нейронной сети, основанной на cuda-convnet (высокопроизводительная реализация C++/CUDA для сверточных нейронных сетей) для распознавания текста;
4. в журнале «Программирование» за 2020 г. [4], авторы опираются на стандарт ISO/IEC 7811-5.4:2018, который определяет размеры карт и положение блоков с реквизитами. В статье предлагается находить на изображении границы карты, применяя алгоритм Виолы-Джонса [31] и метод OverFeat [36]. Алгоритм Виолы-Джонса – это набор простых, но довольно быстрых алгоритмов машинного обучения, способных детектировать объекты со скоростью 15 кадров в секунду). Далее для определения границ блоков символов используется адаптивная бинаризация, а для их уточнения используется скользящее окно реализованное через метод OverFeat, в основе которого лежит сверточная сеть ConvNet. Окончательное распознавание текста осуществляется при помощи библиотеки Tesseract.
5. API Google для распознавания платежных карт позволяет считывать данные, указанные на платежной карте, с помощью камеры. Считывание номера PAN и даты окончания срока действия кредитной или платежной карты выполняется с помощью функции оптического распознавания символов. Для сканирования карты используются сервисы Google Play. Результаты сканирования передаются в ваше приложение, поэтому ему не требуется доступ к камере. Изображения обрабатываются непосредственно на устройстве [14].
6. Microsoft Azure AI Document Intelligence. Модель кредитных карт Document Intelligence использует мощные возможности оптического распознавания символов (OCR) для анализа и извлечения ключевых полей из кредитных и дебетовых карт. Кредитные и дебетовые карты могут быть разных форматов и качества, включая изображения, снятые на телефон, отсканированные документы и цифровые PDF-файлы [7].

Указанная в перечне в п. 1 библиотека представляет собой готовое решение и позиционируется в нише для малого и среднего бизнеса. Продукт является платным и закрытым. Цена на библиотеку рассчитывается индивидуально, по заявке клиента. Необходимость покупки библиотеки может быть неприемлема для некоторых проектов, а закрытость библиотеки исключает возможность вмешаться в поведение программы.

Правообладателем программы (п. 2) является Общество с ограниченной ответственностью Микрофинансовая компания «ЧЕСТНОЕ СЛОВО», ведущее деятельность микро кредитного онлайн-сервиса. На сайте сервиса [43] ничего не говорится о продаже программы или предоставления доступа к ее возможностям.

Метод по детектированию текстовых блоков с реквизитами описанные в п. 3 представляют сейчас скорее исследовательский интерес и является устаревшим.

Метод из п. 4 использует бесплатные современные библиотеки и инструменты и демонстрирует неплохой результат, однако он довольно сложный и требует применения многочисленных методов обработки, что усложняет развертывание, настройку, эксплуатацию и возможную доработку.

API Google перечисленный в п.5 работает при наличии некоторых ограничений. К примеру, некоторые из них:

* для работы API распознавания платежных карт требуется доступ к Google Pay API для Android;
* находиться в регионе AU, BR, CA, CH, CZ, DK, ES, FI, FR, GB, HK, IE, JP, KR, NL, NO, NZ, PL, RU, SE, SG, TW, UA или US;

Если устройство не соответствует этим требованиям, API будет отключен. Включением и отключением API управляют сервисы Google Play. Этот сервис имеет существенные и ограничения и его использование может вызвать затруднения.

Облачный сервис Microsoft Azure AI Document Intelligence из п.6 предоставляет мощный инструмент для распознавания реквизитов банковских карт. Позволяет обучить пользовательскую модель на собственных данных. Для доступа к сервису необходима платная подписка, но можно оформить пробную бесплатную подписку. Ограничением использования данного сервиса для госкомпаний выступает запрет на использование иностранного программного обеспечения на объектах критической информационной инфраструктуры [46].

## 1.3 Методы, применяемые для решения схожих задач

Довольно часто исследуемой проблемой с близкой по составу решаемых задач является проблема распознавания номерных знаков на автомобилях. Здесь также, как и для распознавания текстовых реквизитов на банковских картах, необходимо осуществить поиск на изображении блока с номером (или нескольких блоков, если на изображении присутствуют несколько номерных знаков), а далее распознать текст в найденных блоках. Если обратиться к поисковым системам, то можно заметить гораздо больше статей и готовых разработок. Исходя из того, что эта тема широко и внимательно исследуется, то легко найти решение использующее самые современные методы и инструменты. К примеру, статья 2024 г. российских авторов Ревера В.С. и Шельминой Е.А. в рецензируемом журнале «Моделирование, Оптимизация И Информационные Технологии» [3] или документ 2024 г. конференции 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT) [27].

В статье [27] авторы исследуют одноэтапные детекторы YOLO (You Only Look Once, Смотришь один раз) и SSD (Single Shot Multibox Detector, Детектор – один снимок, много боксов). Результатом их исследования получены следующие данные представленные в таблице 1:

Таблица 1 − Оценка SSD, YOLOv8 и YOLOv9

|  |  |  |  |
| --- | --- | --- | --- |
| **Модель** | **Precision**  **(точность)** | **mAP**  **(средняя точность)** | **FPS,**  **кадры/сек** |
| SSD:Mobinet | 0.467 | 0.421 | 48 |
| YOLOv8 | 0.988 | 0.866 | 51 |
| YOLOv9 | 0.976 | 0.85 | 53 |

Выявлены заметные различия между SSD:Mobinet, YOLOv8 и YOLOv9. SSD:Mobinet продемонстрировал умеренные показатели: точность (0,467) и средняя точность (0,421), но отставал в обработке в реальном времени, достигнув частоты кадров (FPS) 48 кадров/сек. Напротив, YOLOv8 продемонстрировал превосходные показатели производительности, показав высокую точность (0,988) и mAP (0,866), а также эффективную скорость обработки в реальном времени - 51 кадр/сек. YOLOv9 последовал за ним, показав конкурентоспособные результаты с точностью 0,976 и mAP 0,85, а также немного улучшенным показателем FPS 53 кадра/сек. В полученных результатах YOLOv8 и YOLOv9 доминируют в точности и скорости обработки. Среди двух версий YOLO есть нюансы компромисса, что позволяет сделать выбор в пользу конкретных требований: точности или скорости обработки.

Авторы статьи считают, что используя современные модели обнаружения объектов, включающие варианты YOLO (v8, v9) и органично сочетая их с EasyOCR для распознавая текста, получили эффективное и перспективное решение для интеграции в практические системы безопасности.

# Глава 2. Теоретические основы детектирования объектов на изображении и оптического распознавания текста

## 2.1 Описание исходных данных

### 2.1.1 Наборы и форматы данных для обучения моделей компьютерного зрения

Одна из основных целей компьютерного зрения (Computer Vision, CV) понимание визуальных сцен [34]. Понимание сцены включает в себя множество задач, в том числе распознавание присутствующих объектов, локализацию объектов в 2D и 3D, определение атрибутов объектов. На протяжении всей истории исследования компьютерного зрения наборы данных играли важнейшую роль. Они не только служат средством для обучения и оценки алгоритмов, но и стимулируют исследования в новых и более сложных направлениях. Развитие наборов данных для распознавания объектов [54], [55], способствовало прямому сравнению сотен алгоритмов распознавания изображений благодаря стандартным метрикам оценки, таким как средняя точность (mAP). Набор данных ImageNet [56], содержащий миллионы изображений, позволил совершить прорыв в исследованиях по классификации и обнаружению объектов с помощью нового класса алгоритмов глубокого обучения [52].

С 2005 по 2012 год была проведена многолетняя работа по созданию и поддержке ряда эталонных наборов данных для обнаружения основных категорий объектов, которые получили широкое распространение. Например, набор данных PASCAL VOC включает в себя 20 категорий объектов, распределенных по более чем 11 000 изображений, с почти 27 000 экземпляров ограничивающих рамок, из которых около 7 000 имеют подробную сегментацию. Позднее была разработана задача обнаружения 200 категорий объектов, используя подмножество из 400 000 изображений из ImageNet, где было помечено впечатляющие 350 000 объектов с помощью ограничивающих рамок. С 2014 по 2015 год был выпущен еще один популярный набор данных - Microsoft Common Objects in COntext (MS COCO) с ограниченным подмножеством категорий до восьмидесяти. Если основное применение PASCAL VOC - обнаружение объектов на естественных изображениях, то MS COCO предназначен для обнаружения и сегментации объектов, встречающихся в естественном контексте.

Наборы данных, связанные с распознаванием объектов, можно условно разделить на три группы: те, которые в первую очередь направлены на классификацию объектов, обнаружение объектов и семантическую маркировку сцен. Для задачи детектирования объектов текстовых реквизитов на банковских картах необходимо рассмотреть вторую категорию.

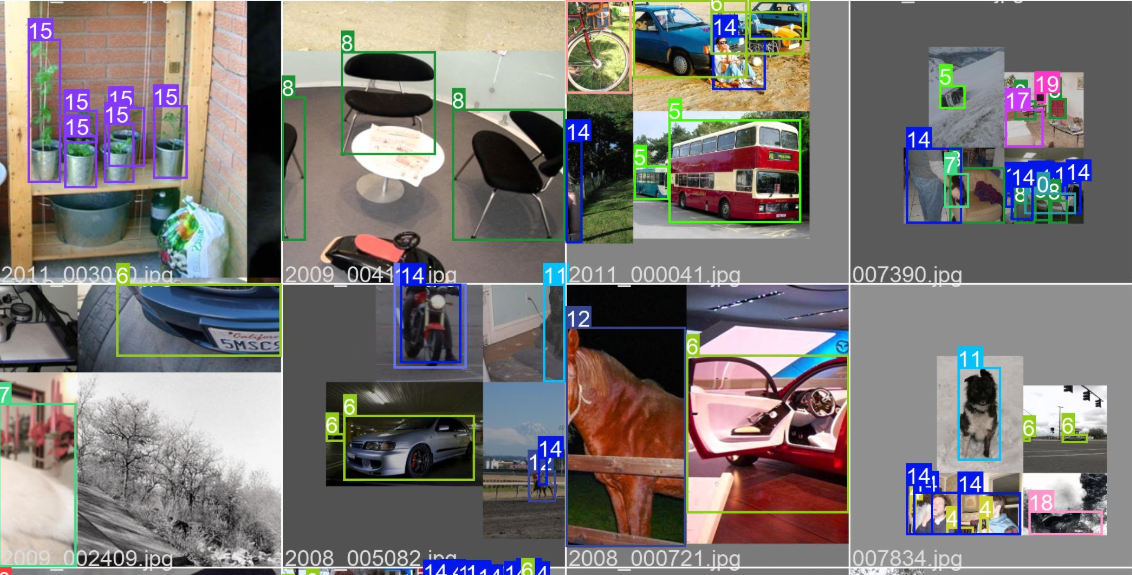
Обнаружение объектов включает в себя не только определение наличия объекта определенного класса, но и его точную локализацию на изображении. Обычно местоположение объекта обозначается ограничивающей рамкой. Для разных форматов датасетов могут быть использованы различные способы задания координат ограничительной рамки. Например, для PASCAL VOC координаты задаются так: верхнего левого угла и нижнего правого угла (xmin-верхний левый, ymin-верхний левый, xmax-нижний правый, ymax-нижний правый) [9].

Рисунок 1 − Мозаичное изображение набора данных VOC

Набор данных VOC содержит разнообразные изображения с различными категориями объектов и сложными сценами. На рисунке 1 приведены примеры изображений из этого набора, а также соответствующие аннотации к ним [10], а также демонстрируется обучающая партия, состоящая из мозаичных изображений набора данных. Мозаика − это техника, используемая в процессе обучения, которая объединяет несколько изображений в одно, чтобы увеличить разнообразие объектов и сцен в каждой обучающей партии. Это помогает улучшить способность модели к обобщению для различных размеров объектов, соотношения сторон и контекста.

### 2.2.2 Набор данных для обучения модели Ultralytics YOLO c текстовыми аннотациями

Более подробно остановимся на форматах данных COCO и совместимых с моделью Ultralytics YOLO.

Формат Ultralytics YOLO — это формат конфигурации набора данных, который позволяет определить корневой каталог набора данных, относительные пути к каталогам изображений для обучения/оценки/тестирования или \*.txt файлы, содержащие пути к изображениям, и словарь имен классов. Вот пример файла data.yaml:

path: ../datasets/coco8 # dataset root dir

train: images/train # train images (relative to 'path'

val: images/val # val images (relative to 'path')

test: # test images (optional)

# Classes (80 COCO classes)

names:

0: person

1: bicycle

2: car

# ...

78: hair drier

79: toothbrush

Этикетки для этого формата должны быть экспортированы в формат YOLO с одним \*.txt файлом на одно изображение. Если в изображении нет объектов, файл \*.txt не требуется. Файл должен быть отформатирован с одной строкой для каждого объекта в class x\_center y\_center width height формат. Координаты должны быть в нормализованном формате (от 0 до 1). Если поля в пикселях, то следует разделить x\_center и width по ширине изображения, и y\_center и height по высоте изображения [58]. Графически это показано на рисунке 2.

Рисунок 2 - Определение координат ограничительной рамки в формате Ultralytics YOLO

Номера классов должны иметь нулевую индексацию (начинаться с 0). Файл этикеток, соответствующий приведенному выше изображению, содержит двух человек (класс 0) и галстук (класс 27):

0 0.481719 0.634028 0.690625

0 0.741094 0.524306 0.933389

27 0.364844 0.975833 0.078125 0.400000

При использовании формата Ultralytics YOLO учебные и проверочные изображения и метки раскладываются по каталогам, как показано в примере набора данных COCO8 на рисунке 3:

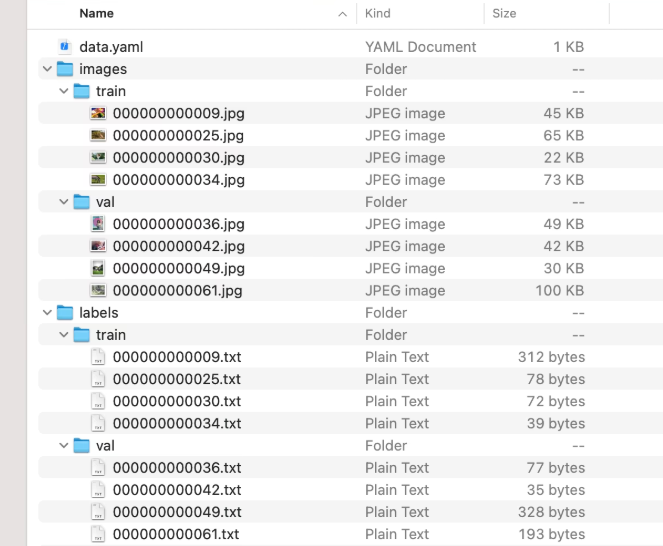


Рисунок 3 − Структура каталогов датасета в формате Ultralytics YOLO

### 2.2.3 Формат данных COCO с аннотациями в формате JSON

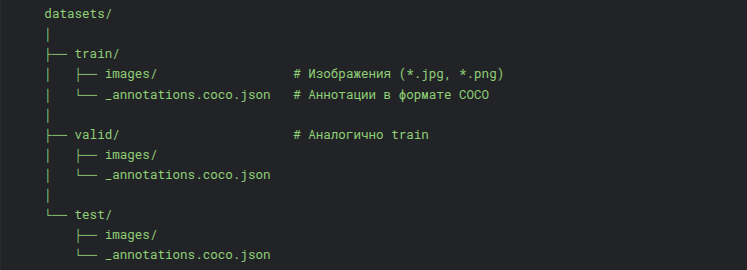
Еще один интересующий нас формат датасета – это COCO. Структура файлового каталога датасета приведена на рисунке 6.

Рисунок 6 − Структура каталога датасета в формате COCO

Файл аннотации annotations.coco.json имеет в своей структуре следующие основные разделы:

{ "info": {...},

"licenses": [...],

"categories": [...],

"images": [...],

"annotations": [...]}

В разделе info можно указать год, версию, описание, автора, веб-ссылку, дату создания датасета, а в разделе licenses указать название лицензии и веб-ссылку на нее.

Раздел categories содержит идентификаторы и названия классов объектов:

"categories": [{

"id": 0,

"name": "Card", // Основной класс: карта

"supercategory": "none" // Без родительской категории

}]

Раздел images содержит информацию об изображениях:

"images": [{

"id": 0, // Уникальный ID изображения

"license": 1, // ID лицензии

"file\_name": "g3\_png\_jpg.rf.xxxx.jpg", // Имя файла

"height": 640, // Высота в пикселях

"width": 640, // Ширина в пикселях

"date\_captured": "2025-04-04T...", // Дата создания

"extra": {"name": "g3\_png.jpg"} // Доп. информация

}]

Раздел annotations содержит информацию о разметке:

" annotations ": [{

"id": 0, // Уникальный ID аннотации

"image\_id": 0, // ID соответствующего изображения

"category\_id": 2, // ID класса (CardNumber)

"bbox": [42,352,562,63], // Bounding box [x,y,width,height]

"area": 35406, // Площадь в пикселях

"segmentation": [], // Пусто - нет полигональной разметки

"iscrowd": 0 // 0 - отдельный объект, 1 - группа

}]

Координаты ограничительной рамки (bounding box) следующим образом:

* x − горизонтальная координата левого верхнего угла прямоугольника (в пикселях от левого края изображения)
* y − вертикальная координата левого верхнего угла (в пикселях от верхнего края изображения)
* width − ширина ограничивающего прямоугольника (в пикселях)
* height − высота ограничивающего прямоугольника (в пикселях)

### 2.3.4 Набор данных настоящего исследования

Для обучения моделей данного исследования на ресурсе https://universe.roboflow.com/ [18] был найден датасет содержащий 764 изображения кредитных карт в формате \*.jpg. Размер изображений 640х640 пикселей. Над набором изображений была произведена аугментация: наложен фильтр с шумами, поэтому количество файлов изображений уникальной карты может составлять от одного до трех. Исходя из этого, можно сделать предположение, что количество уникальных изображений составляет не менее 764 / 3, т.е. более 254 файла. Достоинством данного датасета является наличие в нем изображений российских банковских карт. На рисунке 4 показана часть изображений данного датасета.

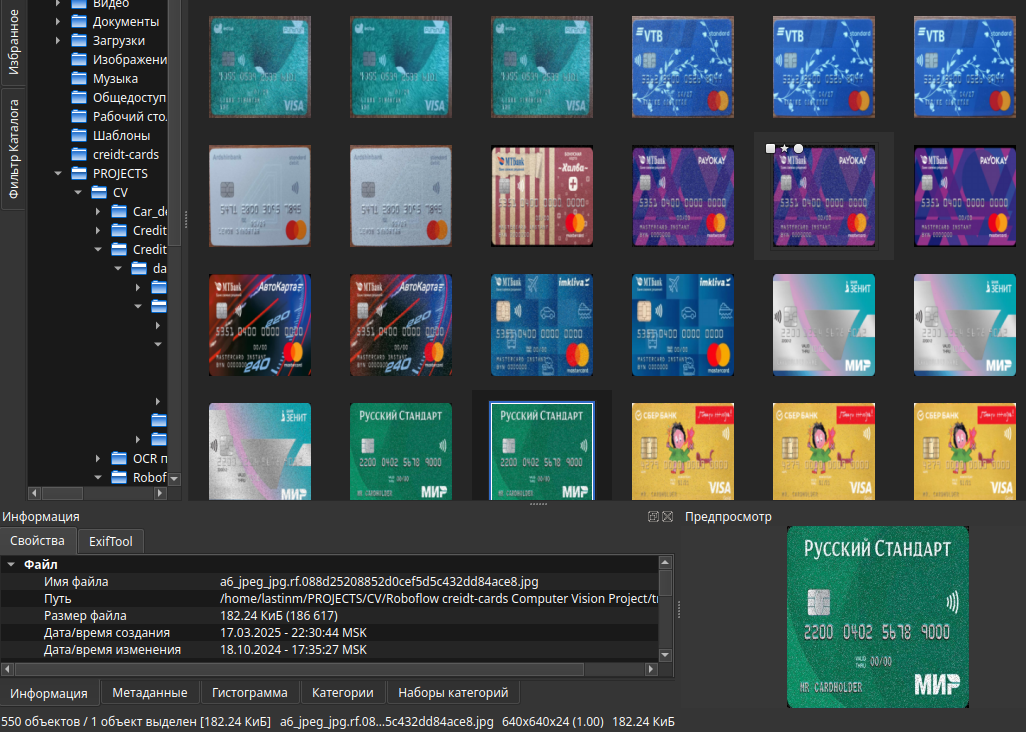


Рисунок 4 − Изображения банковских карт в датасете

Создатели датасета обучили на нем модель YOLO-NAS и получили метрики mAP = 100%, Precision = 100%, Recall = 100%. Протестировать модель можно прямо на сайте https://universe.roboflow.com/creditcards/creidt-cards/dataset/2, загрузив туда изображение своей карты. Результат детекции приведен на рисунке 5. Мы видим точное определение полей реквизитов карты с достаточно высокими показателями уверенности от 90% до 97%.

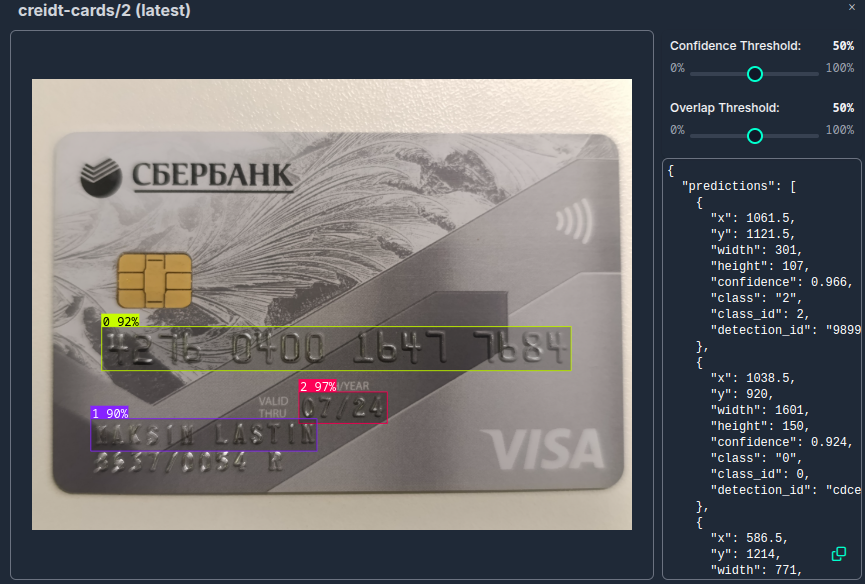


Рисунок 5 − Результат детектирования полей реквизитов банковской карты

Сервис http://roboflow.com предоставляет различные инструменты для автоматизации процесса создания своего собственного датасета:

1. произвести разметку изображений;
2. разбить датасет на тренировочную, валидационную и тестовую части;
3. применить предобработку изображений: изменить яркость, контрастность, цветность, размер, вырезку частей изображений и т.п.;
4. расширить датасет за счет аугментации. В бесплатной версии доступно увеличение тренировочного набора от двух до трёх раз;
5. скачать датасет в нужном формате. Поддерживаются многие популярные форматы: COCO, XML, TEXT, JSON, CSV и т.д. совместимые со многими архитектурами нейронных сетей.

Из найденного датасета создан собственный проект: https://universe.roboflow.com/lastinm/card-detection-knxly-om6ri. Датасет расширен за счет дополнительных изображений карт, выпущенных российскими банками. Добавлены восемь новых изображений. Над датасетом произведена аугментация. Для каждого изображения в обучающем наборе созданы две дополнительные версии. Вид примененной аугментации и её характеристики приведены в таблице 1.

Таблица 1 – Вид аугментации и значение

| Вид аугментации | Значение |
| --- | --- |
| Exposure (экспозиция) | ±30 % |
| Shear (сдвиг) | вертикальный: ±10 пикс.,  горизонтальный: ±10 писк. |
| Grayscale (оттенки серого) | до 15 % на всех изображениях |
| 90° Rotate (поворот на 90 град.) | против часовой стрелки, вверх ногами, по часовой стрелке |
| Rotation (вращение) | ±5 град. |
| Hue (оттенок) | ±15 град. |
| Crop (обрезать) | от 0 до 15 % |
| Brightness (яркость) | осветление и затемнение, ±30% |
| Blur (размытость) | до 2.1 пикс. |
| Saturation (насыщенность) | ±30 % |

После разбивки датасета на тренировочный, валидационный и тестовый наборы, а также применения аугментации к тренировочному набору получили следующий состав датасета:

* тренировочный набор – 1227 изображений;
* валидационный набор – 104 изображения;
* тестовый набор – 2 изображения.

## 2.2 Сверточные нейронные сети в задачах детекции объектов

### 2.2.1 Общие сведения о нейронных сетях

Глубокое обучение (Deep Learning, DL) стало инновацией в различных областях на фоне популярности различных методов машинного обучения (ML), таких как деревья принятия решений (DT), машины опорных векторов (SVM), KMeans, многослойные перцептроны (MLP) и искусственные нейронные сети (ANN) [5]. Глубокое обучение, подмножество ML и компонент искусственного интеллекта (AI), продемонстрировало значительный успех в различных областях. Одним из его приложений является компьютерное зрение (CV). Компьютерное зрение подразумевает обучение машин пониманию и интерпретации визуального контента на сложном уровне. Эта область включает в себя различные приложения, такие как обнаружение объектов, восстановление изображений, распознавание сцен и объектов, оценка позы и движения, сегментация объектов, отслеживание видео и другие. В отличие от традиционной обработки изображений, которая требует ручного извлечения признаков путем определения дескрипторов признаков, архитектуры глубокого обучения служат автоматическими экстракторами признаков. Это делает глубокое обучение выгодной альтернативой, позволяющей исследователям преодолеть традиционные ограничения в обработке изображений и сосредоточиться на повышении производительности конкретных приложений.

Модели глубокого обучения включают в себя различные методы, в том числе рекуррентные нейронные сети (RNN) для последовательной обработки данных и их архитектурные варианты, такие как длинная кратковременная память (LSTM) и рекуррентный блок с управлением (GRU) для сохранения памяти и контекста. Сверточные нейронные сети (CNN) специализируются на задачах визуального восприятия с использованием изображений, поскольку другие алгоритмы DL, такие как ANN, сталкиваются с неэффективностью масштабирования при работе с высокоразмерными входными данными, такими как изображения. Архитектурная схема сверточных нейронных сетей (CNN) на абстрактном уровне состоит из набора функций свертки, объединения и активации, которые преобразуют входные данные, чтобы достичь соответствующего выхода. Фундаментальным компонентом сверточных блоков CNN является определение количества ядер/фильтров и их соответствующих размеров. Они имеют решающее значение для извлечения признаков, предоставляя низкоуровневую пространственную информацию последующим слоям для развития семантических связей.

Схематичное представление абстрактной CNN представлено на рисунке 7 [2]. Ключевые компоненты CNN можно обозначить как набор сверточных слоев, состоящих из фильтров, которые могут быть оптимизированы, за которыми следует определенное количество полносвязанных слоев, ведущих к выходу.

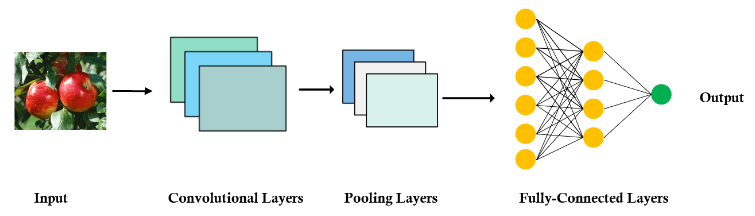


Рисунок 7 − Общая структура CNN с выделением сверточных слоев, пулинга и полносвязных слоев

### 2.2.2 Детектирование объектов

Разработка эффективных детекторов объектов ставит перед исследователями и практиками несколько задач. В первую очередь речь идет о работе с вариациями разрешения и соотношения сторон изображений, что становится еще более сложной задачей, когда целевые объекты имеют значительные различия в пространственных размерах. Несбалансированность классов, особенно в случаях, где сложно получить достаточное количество изображений для определенных классов, может негативно повлиять на производительность детекторов объектов, приводя к необъективным прогнозам. Другая существенная проблема заключается в вычислительной сложности архитектур обнаружения объектов, требующих значительных вычислительных ресурсов в плане мощности, памяти и времени. На рисунке 8 показано обнаружение объектов для одного и нескольких объектов на изображении, причем детекторы с глубокими внутренними сетями требуют значительных вычислительных ресурсов для обработки сложных наборов данных изображений и извлечения важнейших характеристик.



Рисунок 8 − Одиночные и множественные объекты на изображении: Классификация, локализация, сегментация

Методы обнаружения объектов можно разделить на двухэтапные и одноэтапные. Первые предлагают регионы-кандидаты на изображении, а затем классифицируют и локализуют их в пределах предложенных регионов. Примерами двухэтапных детекторов являются RCNN (Region-based Convolutional Neural Network), Fast R-CNN, Faster R-CNN и FPN (Feature Pyramid Network). RCNN, представленная в 2014 г., использовала селективный поиск предложений регионов-кандидатов. Затем использовалась сеть CNN для извлечения признаков, а затем SVM-классификатор для классификации и локализации. Несмотря на точность, RCNN была неэффективна с вычислительной точки зрения из-за двухэтапного процесса. Быстрая R-CNN решила проблему эффективности, внедрив объединение ROI. Этот подход использовал объединение ROI для извлечения карт признаков фиксированного размера для каждого региона из исходных карт признаков, что привело к значительному ускорению вычислений. Faster R-CNN усовершенствовал Fast R-CNN, введя сеть предложений регионов (RPN), которая напрямую генерировала предложения регионов из сверточных карт признаков, устраняя необходимость в отдельном этапе предложения. Интеграция RPN в Fast R-CNN повысила как скорость, так и точность. FPN (Feature Pyramid Network) усовершенствовала двухэтапные детекторы, решив проблему обнаружения целей разных масштабов. FPN генерирует пирамиду признаков путем включения карт признаков разного разрешения с разных этапов сети, что позволяет модели эффективно обнаруживать цели разного масштаба.

Хотя двухэтапные детекторы демонстрируют впечатляющую точность, высокая вычислительная потребность ограничивает их применение. Одноэтапные детекторы нацелены на обнаружение объектов за один проход, устраняя необходимость в отдельном шаге предложения области. Среди известных одноэтапных детекторов можно отметить SSD (Single Shot Multibox Detector), варианты YOLO (You Only Look Once), RefineDet++, DSSD (Deconvolution Single Shot Detector) и RetinaNet. SSD использует несколько сверточных карт признаков в разных масштабах для предсказания ограничительных рамок и оценки вероятности класса. Он эффективно обнаруживает объекты различных размеров и форм за один проход вперед. RefineDet++ расширяет оригинальную архитектуру RefineDet за счет итеративного уточнения целевых предложений на нескольких этапах. Улучшенные механизмы объединения признаков и уточненные границы целей способствуют повышению точности. DSSD (Deconvolution Single Shot Detector) включает в себя сверточные слои для сохранения пространственной информации, потерянной при объединении признаков. Это помогает сохранить пространственное разрешение, позволяя DSSD улавливать мелкие детали. RetinaNet решает проблему дисбаланса классов с помощью Focal Loss, присваивая большие веса тяжелым, неправильно классифицированным образцам, улучшая способность архитектуры справляться с дисбалансом классов и повышая эффективность обнаружения.

Одноэтапные детекторы имеют такие преимущества, как более высокая скорость вывода и меньшая требовательность к вычислительным ресурсам по сравнению с двухэтапными детекторами. YOLO стал сильным конкурентом среди одноэтапных детекторов, продемонстрировав впечатляющую точность и возможность делать выводы в режиме реального времени благодаря своей простой архитектуре. Он доказал свою эффективность в различных реальных приложениях, продемонстрировав свой потенциал для использования в производственных целях.

На рисунке 9 показан эволюционный путь развития компьютерных технологий детектирования объектов на изображениях [41].

Рассмотрим более внимательно практические реализации обоих подходов к детекции объектов. Возьмем по одному наиболее современному и эффективному представителю из каждого подхода. Для двухэтапной детекции рассмотрим Faster R-CNN, а для одноэтапной детекции – YOLOv10.

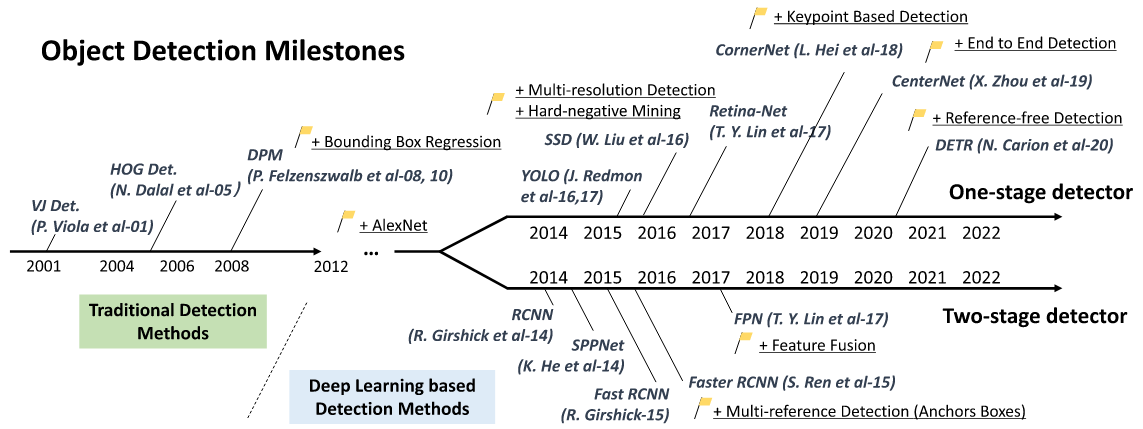


Рисунок 9 − Хронология развития технологии детектирования объектов

### 2.2.3 Обнаружение объектов в реальном времени с помощью Faster R-CNN

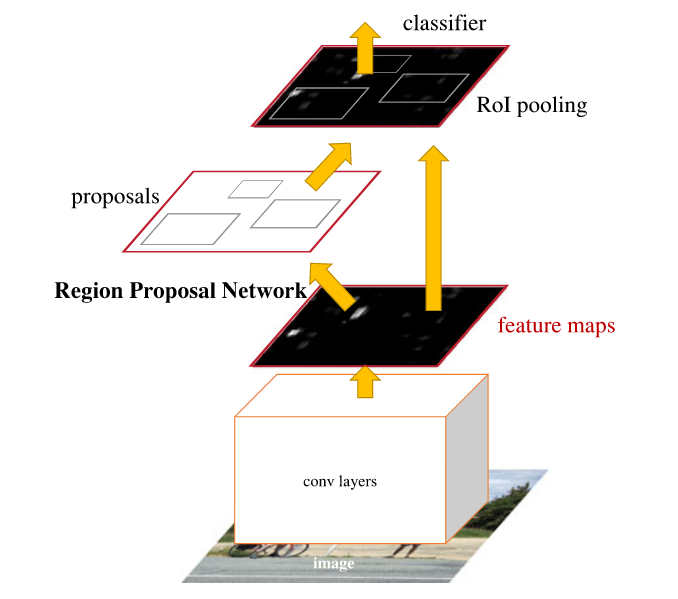
Современные сети обнаружения объектов зависят от алгоритмов предложения областей для гипотетического определения местоположения объектов. Предшественники сети Faster R-CNN, такие как SPPnet и Fast R-CNN, позволили сократить время работы сетей обнаружения, однако вычисление предложений по регионам все еще остается их узким местом. Региональная сеть предложений (Region Proposal Network, RPN) — это полносвязная сверточная сеть, которая одновременно предсказывает объектные границы и оценки вероятности обнаружения объекта в каждой позиции. Она берёт на вход изображение и выдаёт набор прямоугольных объектных предложений, каждое из которых имеет оценку вероятности обнаружения объекта [51]. Сгенерированные высококачественные предложения по регионам, затем передаются в Fast R-CNN для обнаружения. RPN и Fast R-CNN объединяются в единую сеть, в которой разделяются их сверточные функции и используется механизм «внимания». Следовательно, компонент RPN указывает объединенной сети, где искать. На рисунке 10 представлена структурная схема сети Faster R-CNN [9, с. 3].

Рисунок 10 − Сеть Faster R-CNN

RPN принимает на вход изображение (любого размера) и выдает набор прямоугольных предложений объектов, каждое из которых имеет оценку объектности. Этот процесс моделируется с помощью полносвязной сверточной сети. Поскольку конечной целью является совместная работа с сетью обнаружения объектов Fast R-CNN, предполагается, что обе сети имеют общий набор сверточных слоев. Исследуем модель Зейлера и Фергюса (ZF), которая имеет пять разделяемых сверточных слоев, и модель Симоняна и Зиссермана (VGG-16), которая имеет 13 разделяемых сверточных слоев. Для генерации предложений по регионам создается небольшая сеть на основе сверточной карты признаков, выведенной последним общим сверточным слоем. Эта малая сеть принимает на вход n×n пространственных окон входной карты признаков. Каждое скользящее окно сопоставляется с признаком более низкой размерности (256-d для ZF и 512-d для VGG, с последующим ReLU). Эти признаки поступают в два родственных полносвязных слоя - слой регрессии (reg) и слой классификации (cls).

На рисунке 11 изображены слева: сеть предложений регионов (RPN), справа: примеры обнаружения с использованием предложений RPN на датасете PASCAL VOC 2007.

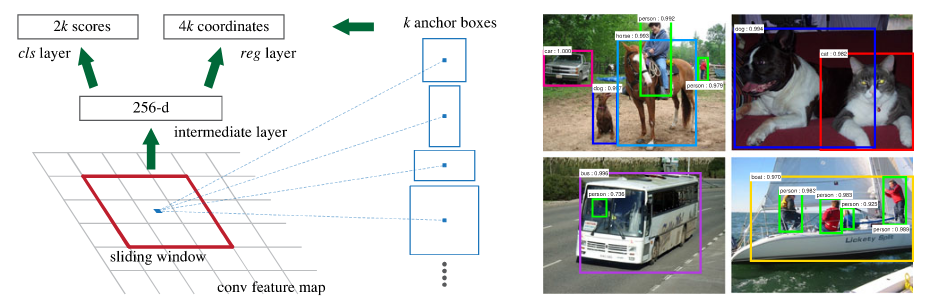


Рисунок 11 − Метод PRN обнаруживает объекты в широком диапазоне масштабов и соотношений сторон

Поскольку мини-сеть работает по принципу скользящего окна, полностью связанные слои являются общими для всех пространственных позиций. Эта архитектура естественным образом реализуется с помощью n×n сверточных слоев, за которыми следуют два родственных 1×1 сверточных слоя (для reg и cls, соответственно).

RPN генерируют несколько предложений в каждом месте скользящего окна, где количество максимально возможных предложений для каждого места обозначается как h. слой reg имеет 4 h выходов, кодирующих координаты h боксов, а слой cls выводит 2h оценок, оценивающих вероятность наличия или отсутствия объекта для каждого предложения. Каждое предложение параметризуется относительно якорей, которые представляют собой опорные боксы с разными масштабами и соотношениями сторон. По умолчанию используется три масштаба и три соотношения сторон, что дает h = 9 якорей в каждой позиции скольжения. Для сверточной карты признаков размером W×H (обычно~ 2400), всего имеется WHh якорей.

Трансляционная инвариантность - это свойство, которое означает, что при перемещении объекта на изображении предложение, сгенерированное алгоритмом, должно также переместиться. Это означает, что функция, предсказывающая предложение, должна работать одинаково хорошо в любом месте изображения. Преимущества инвариантности – универсальность и эффективность. Одна и та же функция может предсказывать предложения в любом месте изображения, что делает алгоритм более универсальным. Это свойство позволяет уменьшить размер модели, поскольку не требуется учесть все возможные положения объекта отдельно.

Для сравнения: метод MultiBox [39] использует k-средние для генерации 800 якорей, которые не являются трансляционно инвариантными. Это означает, что при перемещении объекта MultiBox не гарантирует, что предложение будет сгенерировано одинаково. MultiBox имеет гораздо больше параметров (6,1×10^6) по сравнению с предложенным методом (2,8×10^4). Это делает предложенный метод более компактным. На небольших наборах данных, таких как PASCAL VOC, меньший размер модели снижает риск переобучения, что делает алгоритм более надежным.

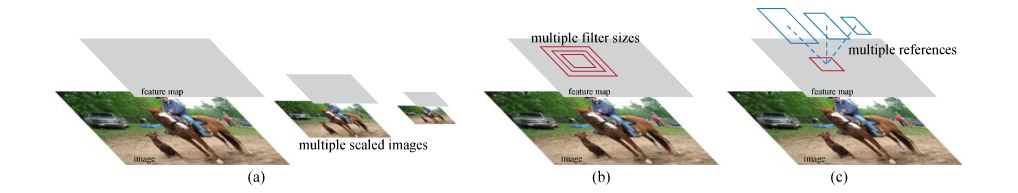
Многомасштабные якоря представляют собой новую схему для работы с несколькими масштабами и соотношениями сторон объектов на изображении. Они позволяют модели обнаруживать объекты разного размера и формы без использования пирамид изображений или фильтров, что делает процесс более эффективным. Как показано на рисунке 12, существует два популярных способа многомасштабного прогнозирования. Первый способ основан на пирамидах изображений/функций, например, в DPM и методы на основе CNN. Изображения изменяются в нескольких масштабах, и для каждого масштаба вычисляются карты признаков (рисунок 6а). Этот способ часто оказывается полезным, но требует много времени. Второй способ заключается в использовании скользящих окон с несколькими масштабами (и/или соотношениями сторон) на картах признаков. Например, в DPM модели с разным соотношением сторон обучаются отдельно с использованием фильтров разного размера (например, 5х7 и 7х5). Если этот способ используется для работы с несколькими масштабами, его можно представить как «пирамиду фильтров» (рисунок 6b). Второй способ обычно используется совместно с первым. Для сравнения метод основанный на якорях, построен на пирамиде якорей, что является более экономичным (рисунок 6c). Этот метод классифицирует и регрессирует ограничивающие рамки, ссылаясь на якорные рамки разных масштабов и соотношений сторон. Он опирается только на изображения и карты признаков одного масштаба и использует фильтры (скользящие окна на карте признаков) одного размера. Благодаря многомасштабному дизайну, основанному на якорях, можно просто использовать сверточные признаки, вычисленные на одномасштабном изображении.

Рисунок 12 − Различные схемы для работы с несколькими масштабами и размерами

Для обучения RPN мы присваиваем каждому якорю бинарную метку класса (быть объектом или нет). Положительную метку присваиваем двум типам якорей: (i) якорю/якорям с наибольшим пересечением Intersection-over-Union (IoU) с истинным боксом (ground-truth box), или (ii) якорю, у которого IoU перекрывается более чем на 0,7 с любым ground-truth box. Заметим, что один ground-truth box может присвоить положительные метки нескольким якорям. Мы присваиваем отрицательную метку неположительному якорю, если его коэффициент IoU ниже 0,3 для всех ground-truth box. Якоря, которые не являются ни положительными, ни отрицательными, не способствуют достижению цели обучения. Функция потрерь *L* рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

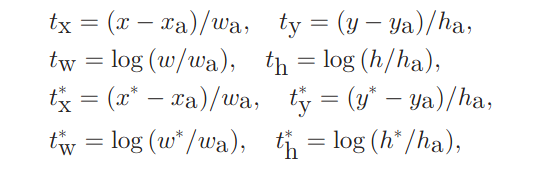
где *i -* это индекс якоря в мини-партии;

*pi* - предсказанная вероятность того, что якорь *i* является объектом.

Метка *pi* равна 1, если якорь положительный, и 0, если якорь отрицательный. *ti* - вектор, представляющий четыре параметризованные координаты предсказанного ограничительного бокса, а *ti* - координаты истинного бокса, связанного с положительным якорем. Потери при классификации *Lcls* - это логарифмические потери для двух классов (объект против не-объекта). Для регрессионных потерь мы используем *Lreg(ti, t\* ) = R(ti - t\*)*, где *R* - это устойчивая функция потерь (гладкая *L1*). Компонента *pi* *Lreg* означает, что регрессионные потери активируются только для положительных якорей (*pi*\*=1) и отключаются в противном случае (*pi*\*=0). Выходы слоев cls и reg состоят из { *pi*} и { *ti* } соответственно.

Обе компоненты нормируются на *Ncls* и *Nreg* и взвешиваются балансировочным параметром *λ*. В текущей реализации член cls в формуле (1) нормируется на размер минипартии (т.е. *Ncls* = 256), а член reg нормируется на количество якорных мест (т.е. *Nreg* ~ 2400). По умолчанию мы устанавливаем *λ*= 10, и таким образом обе компоненты cls и reg имеют примерно одинаковый вес.

Для регрессии ограничительного поля используется параметризация четырех координат:



где x, y, w и h обозначают координаты центра блока, его ширину и высоту. Переменные x, xa и x\* обозначают предсказанный бокс, якорный бокс и истинный бокс соответственно. Метод обнаруживает объекты в широком диапазоне масштабов и соотношений сторон. (Аналогично для y; w; h). Это можно представить как регрессию граничного бокса от якорного бокса к близлежащему истинному боксу.

Чтобы учесть различные размеры областей интереса, выучивается набор из h регрессоров для граничных боксов. Каждый регрессор отвечает за один масштаб и одно соотношение сторон, и h регрессоров не имеют общих весов. Таким образом, благодаря конструкции якорей можно предсказывать боксы разных размеров, даже если признаки имеют фиксированный размер/масштаб.

RPN может быть обучена из конца в конец методом обратного распространения и стохастического градиентного спуска (SGD). Для обучения этой сети используются изображения объектов. Формируются мини-батчи из одного изображения, содержащего множество положительных и отрицательных примеров-якорей. Случайным образом выбирается 256 якорей на изображении, чтобы вычислить функцию потерь для мини-батча, в котором соотношение положительных и отрицательных якорей может достигать 1:1. Если в изображении меньше 128 положительных образцов, мини-батч дополняется отрицательными.

### 2.2.4 Обнаружение объектов в реальном времени с помощью YOLOv10

Одной из популярных реализаций, использующих одноэтапное обнаружение объектов, является сеть YOLO (You Only Look Once, «ты смотришь только раз»). Используя одноэтапный подход приоритет отдается скорости при сохранении высокой точности, что делает YOLO подходящей для приложений реального времени.

Значимость YOLO заключается не только в ее одноэтапном подходе, но и в итерационном развитии, начиная с ее начальной версии, YOLOv1, и заканчивая современным вариантом, YOLOv10 [29]. В каждой̆ итерации проявлялось сочетание инновационных архитектурных достижений, обеспечивающих YOLO беспрецедентные результаты и универсальность во множестве промышленных применений. Этика YOLO подчеркивается внутренним равновесием между точностью и скоростью, что делает его привлекательным предложением как для исследовательского сообщества, так и для специалистов практиков.

В таблице 2 представлена эволюция YOLO, от YOLOv1 до YOLOv8, по широкому спектру показателей.

Значительный скачок в точности и эффективности произошел в пятой версии YOLO (2020 г.). YOLOv5 реализовала архитектуру Modified CSP v7 в PyTorch. Благодаря одноступенчатому механизму обнаружения и новым функциям потерь (CIoU, DFL, BCE) она достигла mAP 50,7 % и значительно увеличила FPS до 200, продемонстрировав свою эффективность в приложениях реального времени.

Таблица 2 − Эволюция YOLO



В период с 2022 по 2024 г.г. произошли итерационные улучшения с YOLOv6 по YOLOv11. YOLOv6, использующая архитектуру EfficientRep, повысила точность до 52,5 %, а YOLOv7, основанная на RepConvN, достигла mAP 56,8 %. YOLOv8, внедрив модель без якорей, сохранила высокую точность 53,9 % при впечатляющей скорости обработки 280 кадров в секунду.

YOLOv9 может похвастаться двумя ключевыми инновациями: фреймворком программируемой градиентной информации (PGI) и обобщенной эффективной сетью агрегации слоев (GELAN). Это очередной значительный прогресс в легком обнаружении объектов. YOLOv9 добился значительной конкурентоспособности в задачах обнаружения объектов, превзойдя YOLOv8 по уменьшению параметров и вычислительной эффективности, улучшив среднюю точность (AP) на 0,6% на наборе данных MS COCO.

YOLOv10 отличается тем, что полностью исключает зависимость от не-максимального подавления (non-maximum suppression - NMS) во время постобработки, что является значительным шагом вперед в повышении скорости вывода. В этой модели используется новый подход к обучению без NMS с использованием двойного присвоения меток, что позволяет гармонично сочетать точность и скорость, обеспечивая эффективность вычислений и сохраняя при этом важные особенности обнаружения. Более того, архитектурные усовершенствования в YOLOv10 включают реализацию облегченных классификационных головок, понижающую дискретизацию с развязкой по пространственным каналам и ранжирование блоков, каждое из которых способствует значительному снижению вычислительных требований и количества параметров. Эти инновации не только повышают эффективность модели, но и улучшают ее масштабируемость на различных устройствах, от мощных серверов до ограниченных в ресурсах конечных устройств. Обширное тестирование показало, что YOLOv10 устанавливает новую планку для компромисса между производительностью и эффективностью. По сравнению с YOLOv9 она достигает значительного снижения задержки и уменьшения размера модели, обеспечивая при этом конкурентоспособную или превосходящую точность обнаружения. Это особенно очевидно при использовании набора данных COCO, где YOLOv10 демонстрирует заметные улучшения в показателях обнаружения, укрепляя свои позиции в качестве лидера в области технологий обнаружения объектов в реальном времени. На рисунке 13 приводится сравнение различных версий Yolo с другими детекторами объектов с точки зрения компромиссов задержки-точности.

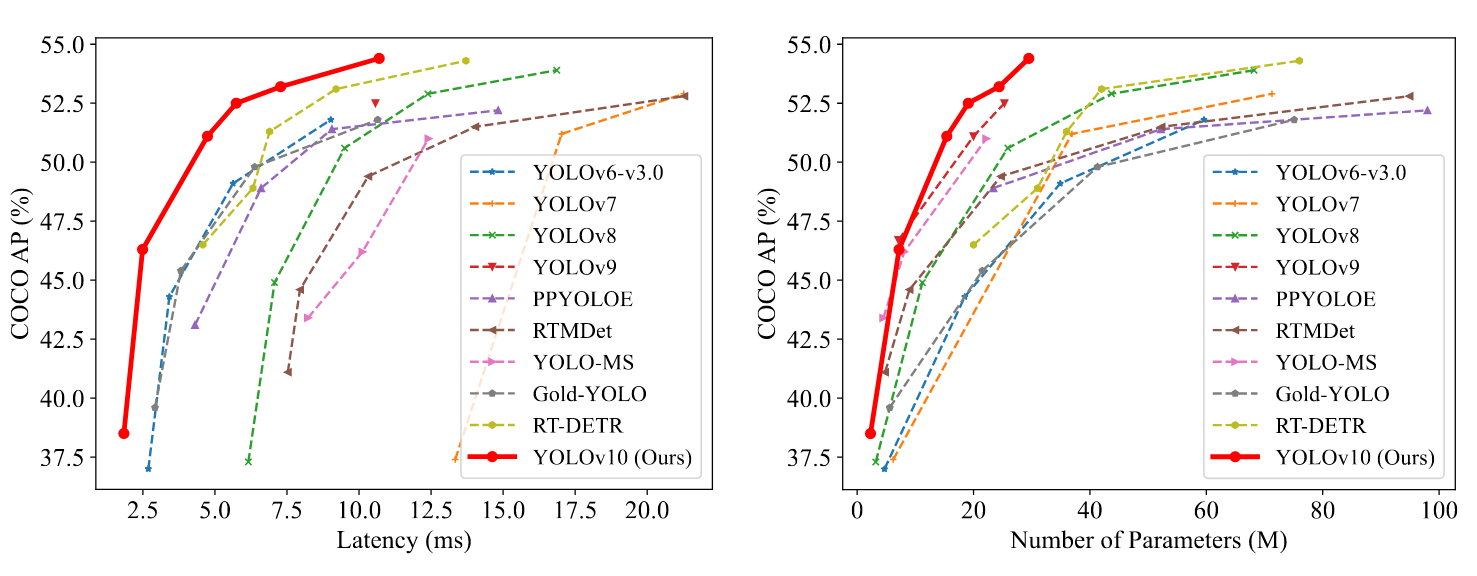


Рисунок 13 − Сравнение разных детекторов реального времени [12] (слева) и размера-точности (справа) [14]

На рисунке 14 показана архитектура YOLOv10. В целом YOLOv10 состоит из трех основных компонентов: позвоночника, шеи и головы [21].

Архитектура YOLOv10 опирается на сильные стороны предыдущих моделей YOLO и в то же время представляет собой несколько ключевых инноваций [11].

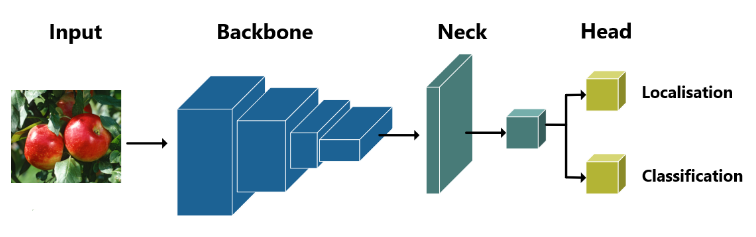


Рисунок 14 − Общая схема архитектуры сети YOLO [5]

Архитектура сети состоит из следующих компонентов:

* позвоничник/магистраль (Backbone). Отвечает за извлечение признаков, магистраль в YOLOv10 использует улучшенную версию CSPNet (Cross Stage Partial Network) для улучшения градиентного потока и сокращения вычислительной избыточности;
* шея (Neck). Шея предназначена для объединения признаков из разных масштабов и передачи их в голову. Она включает в себя слои PAN (Path Aggregation Network) для эффективного объединения разномасштабных признаков;
* одна-ко-многим голова (One-to-Many Head). Генерирует несколько прогнозов для каждого объекта во время обучения, чтобы обеспечить богатые сигналы супервизора и повысить точность обучения;
* одна-к-одной голова (One-to-One Head): генерирует одно наилучшее предсказание для каждого объекта во время вывода, чтобы устранить необходимость в NMS, тем самым уменьшая задержку и повышая эффективность.

## 2.3 Доступные решения для оптического распознавания текста

### 2.3.1 Технологии оптического распознавания текста

С развитием цифровых технологий и автоматизацией обработки данных оптическое распознавание текста (OCR, Optical Character Recognition) стало важной частью информационных систем, широко используемых в бизнесе, медицине, архивном деле и других областях. Эти технологии позволяют преобразовывать текст с изображений в машиночитаемый формат, обеспечивая быстрый доступ и высокую точность обработки данных, что особенно важно в задачах, связанных с массовым вводом документов, чеков и анкет [2].

Внедрение методов машинного обучения вывело OCR на новый уровень - модели на основе нейронных сетей, включая сверточные нейронные сети (CNN) и рекуррентные нейронные сети (RNN), позволили значительно повысить точность распознавания, особенно в условиях разнообразия шрифтов, сложных фонов и искажений изображений. Однако, несмотря на эти достижения, OCR по-прежнему сталкивается с рядом ограничений, таких как точное распознавание рукописного текста и малоконтрастных изображений, а также необходимость оптимизации моделей для устройств с ограниченной вычислительной мощностью.

Основу работы OCR-систем составляет последовательность процессов, включая предобработку изображения, выделение признаков и классификацию символов. Современные подходы к OCR можно разделить на традиционные методы, опирающиеся на правила и алгоритмы, и методы глубокого обучения, использующие нейронные сети.

Традиционные методы OCR основаны на детерминированных алгоритмах, таких как пороговое выделение, сегментация и выделение признаков. Эти методы успешно применяются для решения простых задач, таких как распознавание текста, напечатанного стандартным шрифтом на высококачественных изображениях. На этапе сегментации изображение разделяется на элементы, такие как строки, слова и символы, а затем анализируется. Несмотря на эффективность, традиционные подходы ограничены, когда текст содержит искажения, сложный фон или низкий контраст.

С появлением глубокого обучения системы OCR стали гораздо более точными. Сверточные нейронные сети (CNN) широко используются для извлечения визуальных признаков символов благодаря своей способности эффективно распознавать пространственные зависимости на изображениях. CNN автоматически извлекают значимые признаки текста, такие как формы и контуры, что повышает точность распознавания. Однако сверточные сети менее эффективны при работе с последовательными данными, такими как строки текста.

Кроме того, современные модели OCR всё чаще включают трансформеры — архитектуры, которые объединяют возможности CNN и RNN (рекуррентные нейронные сети) для эффективного анализа как пространственных, так и последовательных данных. Трансформеры применяет механизмы внимания для выделения важных элементов изображения, что позволяет решать задачи OCR более эффективно, обеспечивая высокую точность, но требуют значительных вычислительных ресурсов, что ограничивает их использование в некоторых практических приложениях.

### 2.3.2 OCR библиотека Tesseract

Tesseract − это OCR-движок с открытым исходным кодом, который̆ можно использовать для обучения моделей̆ или использовать существующие модели для чтения символов. Преимуществом Tesseract является возможность сегментации текста на уровне отдельных символов и использование механизма долговременной̆ памяти (LSTM) на основе рекуррентной̆ нейронной̆ сети, ориентированного на распознавание строк [6]. В настоящее время Tesseract доступен в версии 5.0. Tesseract поддерживает Unicode (UTF-8) и может распознавать более 100 языков.

Python предоставляет библиотеку Tesseract под названием pytesseract, которая может быть использована для реализации движка Tesseract-OCR на Python. Эта библиотека хороша для распознавания черных букв на белом фоне. Она хорошо подходит для классических книг, но плохо работает на изображениях со сложным фоном или с малоконтрастными изображениями.

### 2.3.3 OCR библиотека EasyOCR

EasyOCR - это готовый к использованию инструмент на языке Python, предназначенный для извлечения текста из изображений с помощью предварительно обученных моделей [27]. Поддерживается более 80 языков и все популярные письменности, включая: латиницу, китайский, арабский, деванагари, кириллицу и т. д. [53]

EasyOCR использует ResNet и VGG для извлечения признаков:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

где X - входное изображение;

Xres и Xvgg - признаки, извлеченные с помощью ResNet и VGG, соответственно.

Для маркировки последовательностей используются сети с долговременной памятью (Long Short-Ter Memory, LSTM):

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

где Y представляет собой помеченные последовательности, полученные от сети LSTM.

Алгоритм Classification (CTC) декодирует помеченные последовательности в реальный текст:

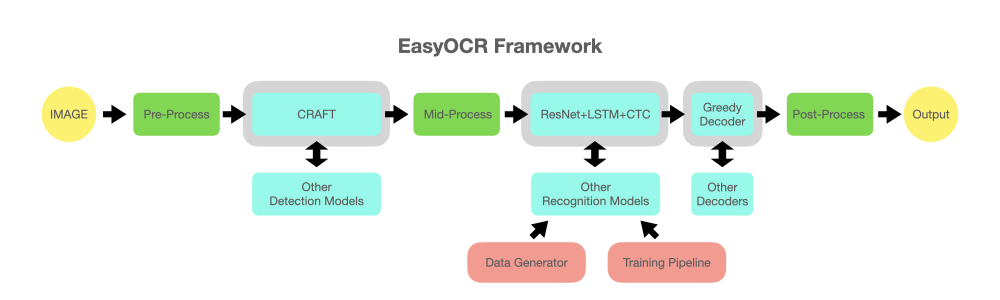


Рисунок 15 − Архитектура фреймворка EasyOCR [20]

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

### 2.3.4 OCR библиотека KerasOCR

Keras — это библиотека глубокого обучения, представляющая из себя высокоуровневый API, написанный на Python и способный работать поверх TensorFlow, Theano или CNTK. Он был разработан с расчетом на быстрого обучения. [13].

Keras поддерживает как сверточные и рекуррентные сети, так и их комбинации и работает как на процессоре (CPU), так и на графическом процессоре (GPU).

Библиотека Keras показывает высокую точность распознавания, но медленнее в работе с сравнении с Tesseract и EasyOSR [20]. Точность и скорость работы разных библиотек для распознавания реквизитов банковских карт следует выяснить в ходе дальнейшего практического исследования.

### 2.3.5 OCR библиотека TrOCR

TrOCR использует архитектуру трансформера для понимания изображений и для генерации текста на уровне слов [33]. Модель TrOCR проста, но эффективна, ее можно предварительно обучить на масштабных синтетических данных и отладить с помощью наборов данных, размеченных людьми. Эксперименты показывают, что модель TrOCR превосходит современные модели в задачах распознавания печатных, рукописных и сюжетных текстов.

TrOCR – это сквозная модель распознавания текста на основе трансформера с предварительно обученными моделями CV и NLP, архитектура которой показана на рисунке 16.

В отличие от существующих моделей распознавания текста, TrOCR — это простая, но эффективная модель, которая не использует CNN в качестве основы. Вместо этого, следуя [28], она сначала изменяет размер входного текстового изображения на 384×384, а затем изображение разбивается на последовательность из 16×16 патчей, которые используются в качестве входа для преобразователей изображений. Стандартная архитектура трансформеров с механизмом внимания используется как в кодирующей, так и в декодирующей части, где в качестве распознанного текста из входного изображения генерируются блоки слов. Для эффективного обучения модели TrOCR кодер может быть инициализирован предварительно обученными моделями в стиле ViT (Vision Transformer), а декодер - предварительно обученными моделями в стиле BERT (Bidirectional Encoder Representations from Transformers). Таким образом, преимущество TrOCR заключается в трех аспектах. Во-первых, TrOCR использует предварительно обученные модели преобразования изображений и преобразования текста, которые используют преимущества больших масштабов немаркированных данных для понимания изображений и моделирования языка, без необходимости использования внешней языковой модели. Во-вторых, TrOCR не требует конволюционной сети для основы и не вносит никаких специфических для изображения индуктивных смещений, что делает модель очень простой в имплементации и поддержке. Наконец, результаты экспериментов на эталонных наборах данных OCR показывают, что TrOCR может достичь передовых результатов на наборах данных печатных, рукописных и сюжетных текстовых изображений без каких-либо сложных шагов по предварительной и/или послеоперационной обработки. Более того, можно легко расширить TrOCR для распознавания многоязычных текстов с минимальными усилиями, просто используя многоязычные предварительно обученные модели на стороне декодера и расширяя словарь.

Рисунок 16 − Архитектура TrOCR [31]

## 2.4 Метрики и методы оценки качества

### 2.4.1 Метрики применяемые для оценки качества модели

Существует несколько метрик, которые можно использовать для оценки эффективности предсказательных моделей в общем случае, а также несколько метрик непосредственно относящихся к детекторам объектов.

Корректность/четкость/правильность (Accuracy), точность (Precision), полнота (Recall), F1-Score (гармоническое среднее), кривая точности и полноты (Precision-Recall curve, PR curve), средняя точность (Average Precision, AP). Средняя точность — наиболее часто используемая метрика, полученная с использованием полноты и точности. Все эти метрики используются для оценки результата большинства моделей машинного обучения.

Пересечение над объединением (Intersection over Union, IOU) – это метрика оценки качества определения ограничительных рамок объекта детекции [15] [8] [50] [23] [30] [49] [47]. Цель детекторов объектов — предсказать местоположение объекта, поместив ограничивающую рамку над объектом заданного класса на изображении/видео с высокой оценкой достоверности. Общее обнаружение можно рассматривать как совокупность трех элементов: класс объекта, ограничивающая рамка (bounding box, BB) вокруг этого объекта и оценка достоверности [8].

### 2.4.2 Пересечение над объединением (Intersection over Union)

Intersection over Union (IoU) - важнейшая метрика в области компьютерного зрения, в частности для оценки производительности моделей обнаружения объектов и сегментации изображений. Она измеряет точность детектора объектов путем количественной оценки степени перекрытия между предсказанными ограничительными рамками и истинными ограничительными рамками [44].

IoU - это отношение площади перекрытия между предсказанной ограничительной рамкой и истинной ограничительной рамкой к площади их объединения. На рисунке 17 приводится наглядное представление концепции IoU [32].

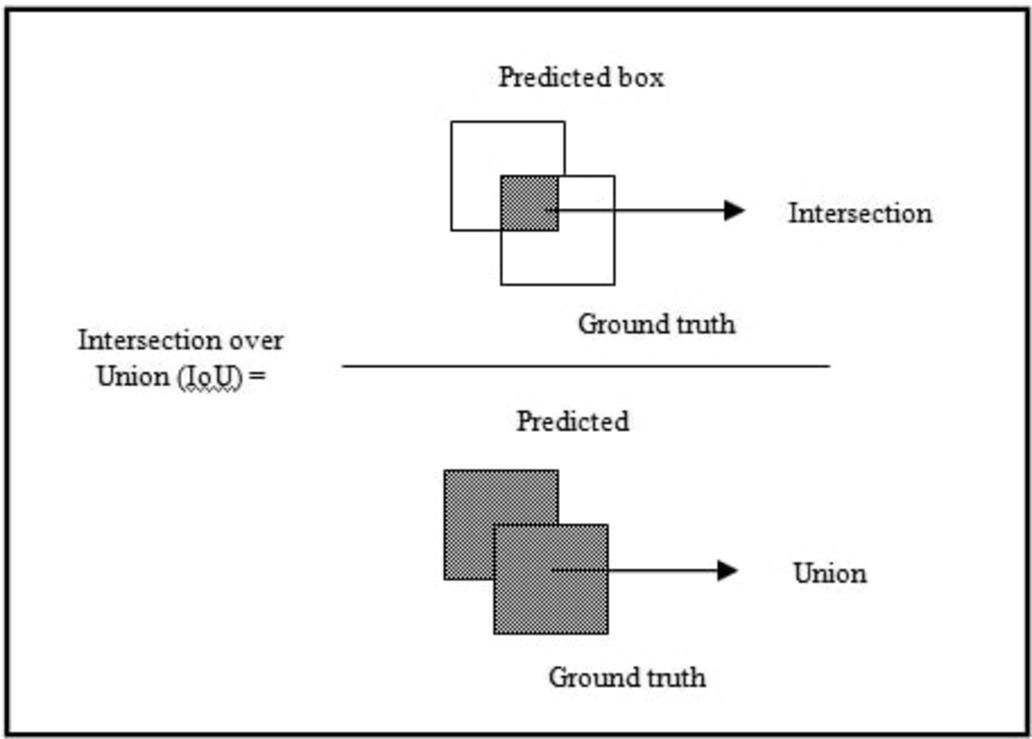


Рисунок 17 – концепция метрики IoU

Метрика Intersection over Union (IoU) принимает значения в пределах от 0 до 1, где более высокие значения указывают на большую точность обнаружения. При полном совпадении предсказанной и истинной ограничивающих рамок IoU достигает 1, а при отсутствии пересечения — 0. Понимание IoU важно для всех, кто работает с задачами обнаружения объектов, так как это стандартизированный способ оценки и сравнения производительности моделей. IoU помогает убедиться, что модели обнаружения объектов не только идентифицируют правильные объекты, но и точно определяют их местоположение. Такая точность очень важна в реальных приложениях, где точная локализация объектов имеет первостепенное значение.

Для определения корректности распознавания объекта обычно используется пороговое значение IoU (чаще 0,5). Если метрика превышает этот порог, обнаружение считается успешным. Формула расчета IoU представлена ниже:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

где BBpredict - предсказанная ограничительная рамка;

BBground - истинная ограничительная рамка.

### 2.4.3 Метрики Accuracy, Precision, Recall, F1-Score

Для вычисления значений данных метрик каждый объект должен быть классифицирован как:

* TP (True Positive) - модель предсказала положительный результат, и в действительности он оказался верным;
* TN (True Negative) - модель предсказала отрицательный результат, и в действительности он оказался верным;
* FP (False Positive) - ошибка "ложная тревога", модель предсказала положительный результат, а в действительности он оказался ложным;
* FN (False Negative) - ошибка "пропуск цели", модель предсказала отрицательный результат, а в действительности он оказался ложным [23], [50].

Для расчета метрик используются следующие переменные:



Accuracy (корректность) определяет производительность модели по всем классам. Она рассчитывается как отношение общего числа образцов, классифицированных правильно, к общему числу образцов. Формула записывается следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Precision (точность) означает, какое количество положительных идентификаций было действительно правильным. Другими словами, она рассчитывается как отношение числа точно идентифицированных положительных образцов к общему числу положительных образцов. Он определяется следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

Recall (полнота) − это мера того, сколько фактических положительных образцов было определено правильно. Она оценивается как отношение числа правильно идентифицированных положительных образцов к общему числу фактически положительных образцов. Recall также известен как чувствительность. Полнота определяется по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

F1-Score (гармоническое среднее) — это метрика, которая объединяет две важные характеристики качества precision и recall. Проблема зключается в том, что если оптимизировать только precision, модель может пропускать много правильных слов (низкий recall), и наоборот. F1-Score балансирует эти две метрики, вычисляя их гармоническое среднее:

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

Гармоническое среднее сильнее штрафует за разрыв между precision и recall, чем обычное среднее арифметическое. Если одна из метрик близка к нулю, то F1 тоже будет низким, что помогает выявлять несбалансированные решения. F1-Score — это удобный способ оценить модель, учитывающий и точность, и полноту. Чем ближе F1 к 1, тем лучше модель.

### 2.4.6 Средняя точность (AP)

Для расчета точности обнаружения наиболее часто используется метрика средней точности (Average Precision, AP). Она рассчитывается независимо для каждой категории объектов Cm. Она оценивается как:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

### 

В исследованиях часто можно встретить запись AP@50. Это означает:

* AP (Average Precision) — усреднённая точность для всех порогов классификации;
* @50 — вычисляется при IoU (Intersection over Union) = 0.50 (порог перекрытия боксов 50%).

Это ключевая метрика для задач обнаружения объектов (object detection), показывающая, насколько точно модель находит объекты с умеренно строгим критерием перекрытия.

Возможные значения показателя:

* AP@50 ≥ 0.7 — отличный результат (для сложных датасетов типа COCO);
* AP@50 ~ 0.5–0.7 — приемлемо, но есть куда улучшать;
* AP@50 < 0.3 — серьёзные проблемы с моделью или данными.

Встречается также более строгая метрика: AP@50:95 — усредненная точность AP при IoU принимающем значения от 0.5 до 0.95.

### 2.4.7 Среднее значение точности (mAP)

Показатель mAP рассчитывается путем усреднения по всем категориям объектов и таким образом оценивает эффективность работы детекторов объектов. Формула определяется следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

# Глава 3. Практическое исследование методов детектирования и распознавания реквизитов банковских карт

## 3.1 Обучение моделей для детектирования объектов

### 3.1.1 Обучение Faster R-CNN

Для обучения данной модели в качестве основы использован код с GitHub: https://github.com/GirinChutia/FasterRCNN-Torchvision-FineTuning. Доработанный программный код обучения модели приведен в Приложении 1.

Перед запуском процесса обучения необходимо задать параметры в файле train.py. Для начала указываем полный путь к датасету:

@dataclass

class DatasetPaths:

train\_image\_dir: str = r"/путь/train/images/"

val\_image\_dir: str = r"/путьvalid/images/"

train\_coco\_json: str = r"/путь/train/\_annotations.coco.json"

val\_coco\_json: str = r"/путь/valid/\_annotations.coco.json"

Затем устанавливаем количество эпох (итераций) обучения, размер батча (количество изображений, одновременно загружаемых в память видеокарты), частоту валидации (для большого количества эпох и медленного обучения можно устанавливать этот параметр больше двух):

@dataclass

class TrainingConfig:

epochs: int = 200

batch\_size: int = 6

val\_eval\_freq: int = 2

Запуск обучения производится командой:

python3 train.py

В результате обучения модели FasterRCNN ResNet50 FPN v.2 наилучшие метрики получили на 199 эпохе обучения. Средняя точность (AP) для IoU = 50% составила 0,9906, а для IoU ∈ [0.5;0.95] составила 0,9028. На рисунке 18 приведен график изменения средней точности в зависимости от эпохи обучения.

Рисунок 18 – Средняя точность (AP) модели Faster R-CNN

Метрика АР\_50 не представляет интереса в данном исследовании, т.к. смещение распознаваемого поля реквизита на половину не позволит получить все символы реквизита. Более строгий показатель AP\_50\_95 = 0,9028 далек от приближения к 100%. Необходимо проанализировать, как проходил процесс обучения. Для этого библиотека tensorflow и средство визуализации tensorboard формируют наглядные графики потерь (loss) на тренировочном наборе данных (Train/\*\_loss) и на валидационном наборе данных (Val/\*\_loss). Данные графики отражают, как модель обучалась на каждом этапе.

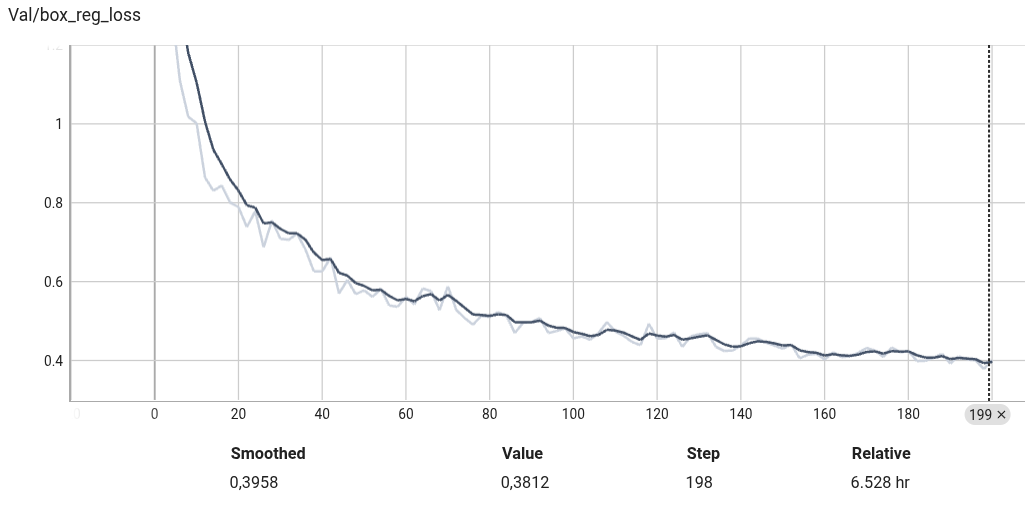
На рисунке 19 показан график Val/box\_reg\_loss, который демонстрирует потери для регрессии bounding box'ов (точность локализации объектов). Используемая формула: обычно Smooth L1-loss (Huber loss), которая штрафует модель за неправильные координаты центра (x, y), ошибки в размерах (width, height). Нормальные значения: 0.05–0.3 (чем меньше, тем точнее предсказанные bbox).

Рисунок 19 – потери для регрессии ограничительных рамок

Возможные проблемы − нестабильные градиенты, что на графике может выглядеть в виде резких скачков. На полученном графике проблем с градиентами не заметно. Высокие значениями считаются значения более 0.5. Полученное значение 0,3812, а сглаженное 0,3958, что говорит о плохой способности модели локализовать объекты (можно попробовать уменьшить learning\_rate).

На рисунке 20 приведен график потерь классификации Val/classifier\_loss (правильность определения категории объекта). Используемая формула Cross-Entropy loss для multi-class классификации. Нормальные значения: 0.1–0.5 (зависит от числа классов). Высокие значения сообщают о путанице между классами (необходимо проверить разметку). Полученное значение 0,3044 указывает на достаточно точное определение классов. Если график растёт вместе с эпохами, то это свидетельствует о переобучении модели. На данном графике такая проблема отсутствует.

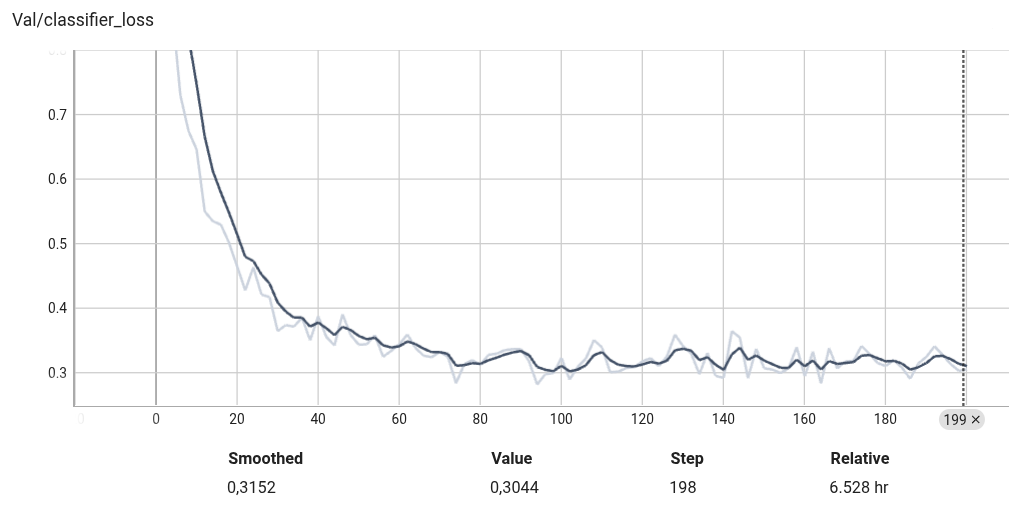
На рисунке 21 приведен график потерь для предсказания "объектности" Val/objectness\_loss (есть ли объект в предложенной области). Это характерно для RPN (Region Proposal Network) в Faster R-CNN. Формула: Binary Cross-Entropy (объект / фон). Нормальные значения: 0.01–0.2.

Рисунок 20 – потери для классификации

Полученное значение равняется 0,0111, а сглаженное 0,0138, следовательно RPN достаточно хорошо выделяет регионы (anchor boxes не требуют настройки).

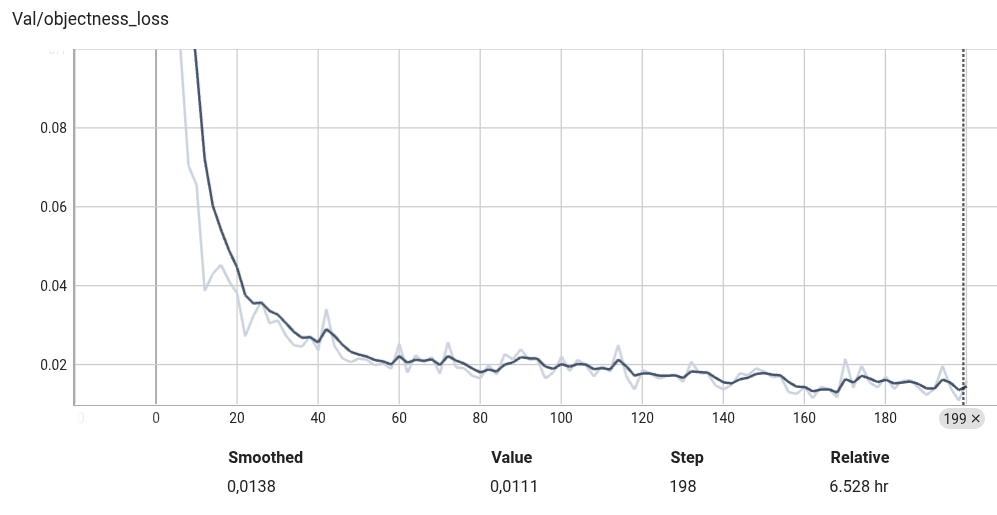


Рисунок 21 – потери для предсказания "объектности"

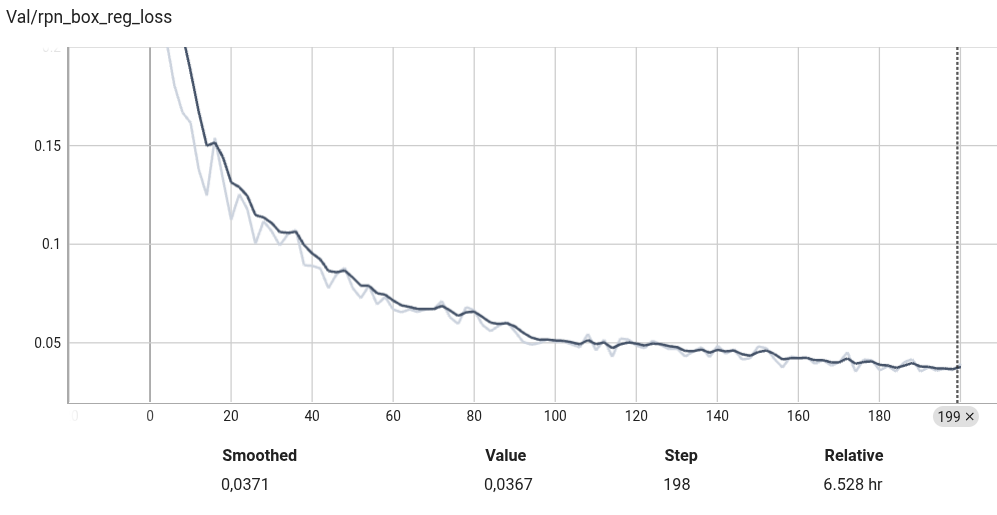
 На рисунке 22 приведен график потерь регрессии bbox в RPN Val/rpn\_box\_reg\_loss (точность anchor boxes).

Рисунок 22 – потери регрессии bbox в RPN

Рассчитывается аналогично box\_reg\_loss, но для предложенных регионов. Нормальные значения: 0.05–0.25. Полученное значение близко к нижней границе допуска и составляет 0,0367. Если значение выше верхнего предела, то это говорит о том, что Anchor boxes не подходят под используемые данные.

На рисунке 23 приведен график суммарных потерь Val/total\_loss (все компоненты с весами). Нормальные значения: 0.3–2.0 (зависит от архитектуры). Поученное значение составляет 0,7333. Срединное значение допустимого диапазона, что указывает на то, что в обученной модели есть узкие места, над которыми можно поработать.

В нормальной ситуации график должен монотонно уменьшаться (с колебаниями), что и происходит на представленном изображении. Наличие резких скачков может указывать на проблемы с LR или данными (у нас не наблюдается).

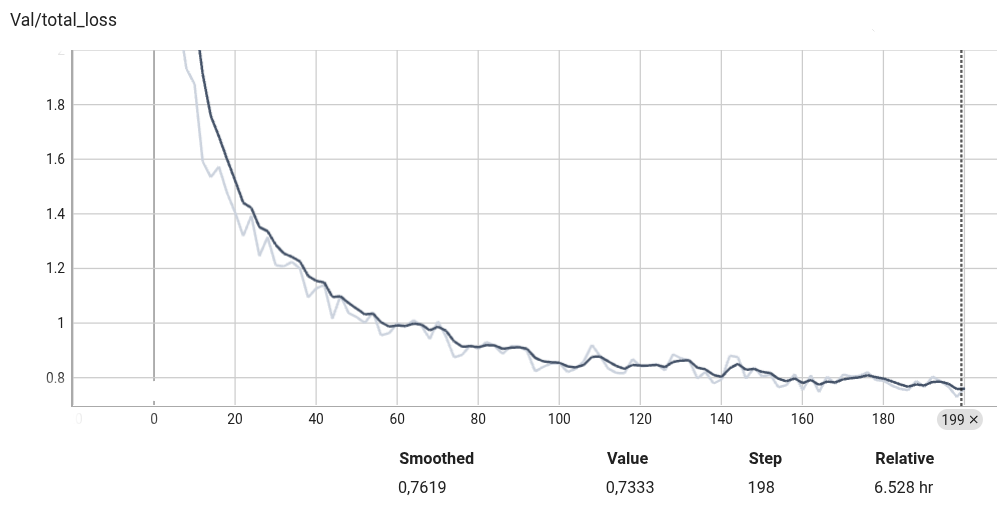


Рисунок 23 – суммарные потери

### 3.1.2 Обучение YOLOv12

Обучить модель Yolo намного проще, чем Faster R-CNN. Код выглядит очень компактно:

from ultralytics import YOLO # Импорт библиотеки YOLO

model = YOLO('./yolo12n.pt') # Используем предобученные веса

results = model.train(data='указать путь/data.yaml',

epochs=200,

patience=0,

batch=16,

device='cuda',

imgsz=640)

В качестве параметров передаваемых перед обучением указывается путь к конфигурационному файлу датасета data.yaml, количество эпох обучения, размер батча, устройство обучения cuda или cpu, размер изображений.

В результате обучения в каталоге, откуда было запущено обучение, появится подкаталог с результатами обучения: run/detect/train. Во вложенном каталоге weights/ сохраняется лучшая и последняя модель.

Если передать путь с результатами обучения приложению tensorboard, то получим наглядные графики аналогично рассмотренным в предыдущей главе для модели Faster R-CNN.

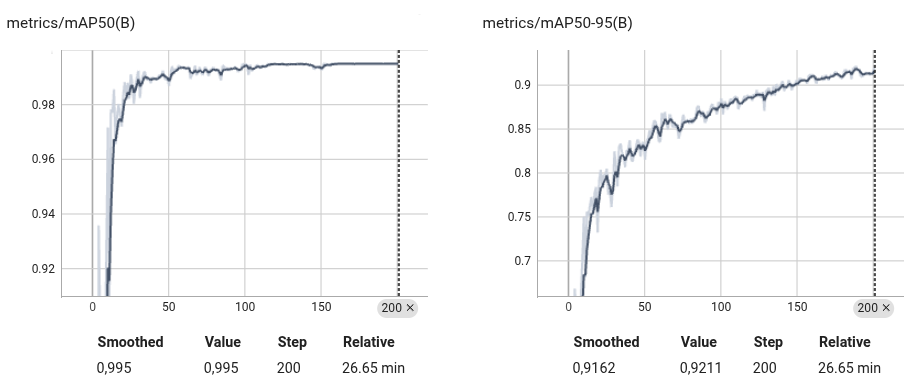
Среднее значение точности (mAP) для IoU = 50% составило 0,995, а для IoU ∈ [0.5;0.95] также составило 0,9211. В обоих случаях значения близки к единице, что означает очень хороший результат. На рисунке 24 приведен график изменения среднего значения точности в зависимости от эпохи обучения.

Рисунок 24 – метрики mAP (bounding boxes) для модели YOLOv12

Дополнительно модель YOLO посчитала метрики precision и recall. График приведен на рисунке 25. Precision равняется 0,9956, что очень близко к единице, а метрика recall равняется 0,999. Это отличные показатели.

Далее посмотрим на графики функции потерь. На рисунке 26 показан график val/box\_loss, который демонстрирует потери для регрессии bounding boxes. Полученное значение составило 0,4345, что чуть выше нормального диапазона значений: 0.05–0.3. Это свидетельствует о недостаточной точности предсказанния bbox.

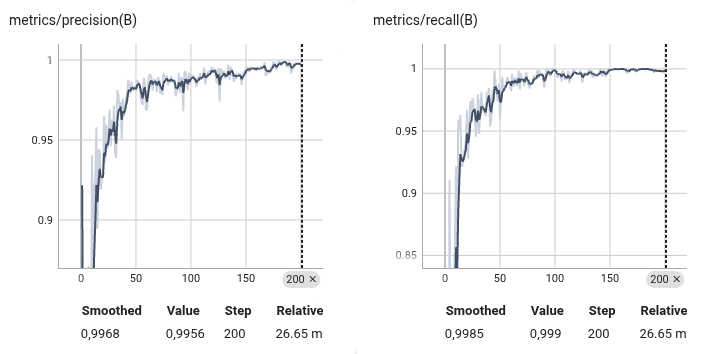
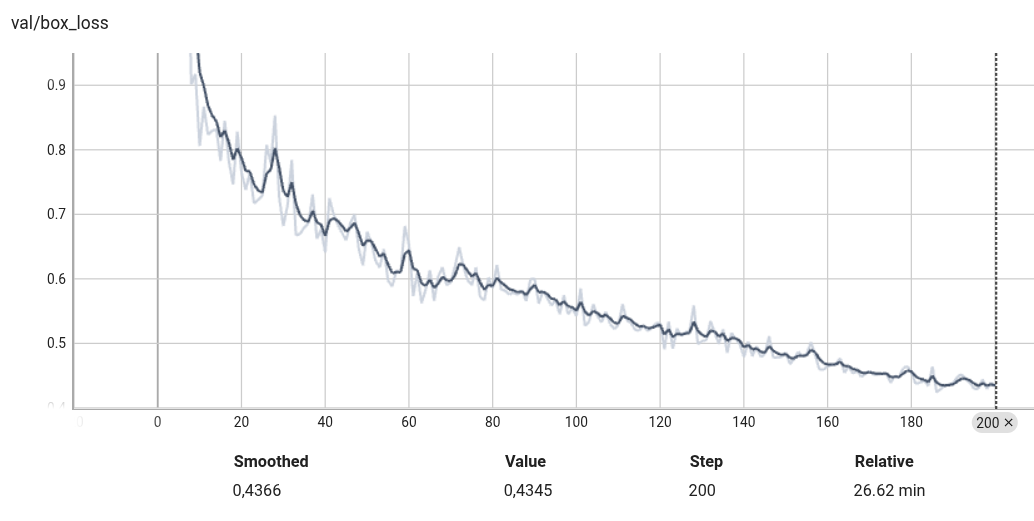


Рисунок 26 – потери для регрессии ограничительных рамок

Рисунок 25 – метрики precision и recall для модели YOLOv12

На рисунке 27 приведен график потерь классификации val/cls\_loss. Достигнутое значение равняется 0,251, что посредине диапазона нормальных значений: 0.1–0.5. Это указывает на точное определение классов.

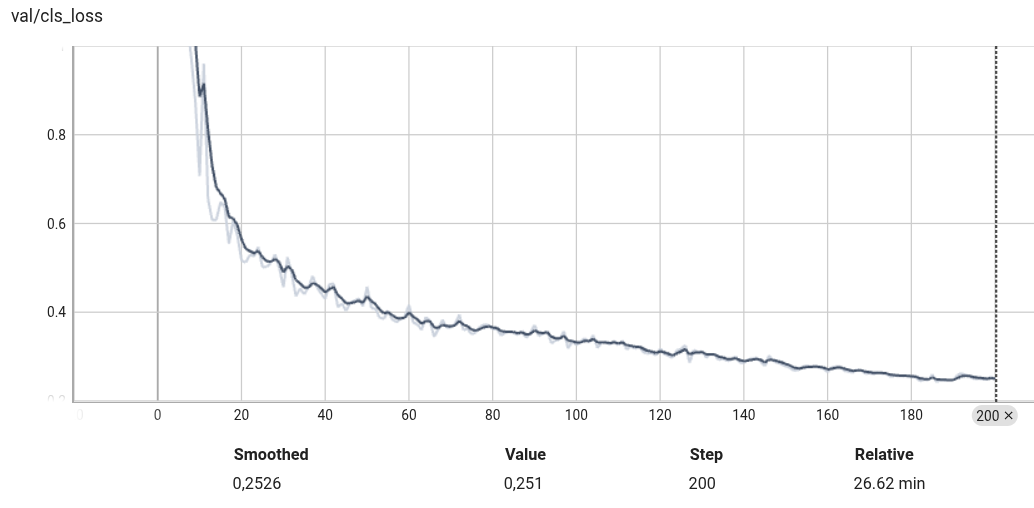
DFL (Distribution Focal Loss) – это функция потерь, используемая в YOLOv8 и более новых версиях для предсказания координат bounding boxes. В отличие от обычного регрессионного подхода (например, MSE), DFL:

Рисунок 27 – потери для классификации

* моделирует распределение вероятностей возможных позиций границ бокса;
* оптимизирует форму этого распределения, фокусируясь на наиболее вероятных значениях;
* позволяет модели точнее предсказывать координаты, особенно для небольших объектов.

Чем ниже dfl\_loss, тем лучше боксы соответствуют объектам. Высокие значения могут указывать на плохую детекцию мелких объектов, ошибки в предсказании размеров боксов, недообучение или переобучение модели. На рисунке 28 показан график val/dfl\_loss.

В интервале от 100 до 180 эпох график скакал между плохими и хорошими решениями (возможно, слишком высокий learning\_rate). После 180 эпох обучения можно видеть выход на плато. Значение показателя в конце обучения составило 0,7889. Есть вероятность улучшить этот показатель за счет настройки анкеров (anchors) под датасет.

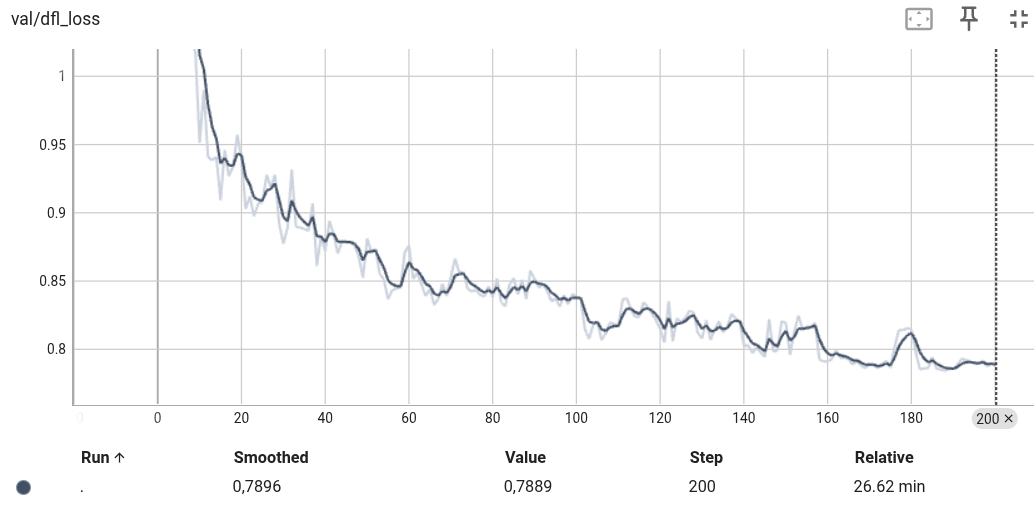


Рисунок 28 – функция потерь Distribution Focal Loss

### 3.1.3 Сравнение метрик и точности моделей детектирования

Сравним модели Faster R-CNN v.2 и YOLO v.12 обученные для детектирования на одном и том же датасете изображений банковских карт.

Сеть Faster R-CNN v.2 рассчитывает метрику Val/AP50-95 – это средняя точность (AP), усреднённая по порогам IoU от 0.50 до 0.95 (с шагом 0.05) на валидационном наборе. YOLO v.12 рассчитывает метрику mAP50-95 – это mean Average Precision (mAP), также усреднённое по IoU от 0.50 до 0.95. Фактически, это одна и та же метрика, но в разных фреймворках её могут называть [48]:

1. AP50-95 (в COCO-стиле, используется для Faster R-CNN, Mask R-CNN и др.);
2. mAP50-95 (в YOLO, но иногда просто mAP@.5:.95).

Сравнение этих метрик корректно, когда выполняются условия:

1. одинаковый датасет;
2. одинаковая методика расчёта AP (COCO-style или Pascal VOC-style);
3. одинаковый диапазон IoU (0.50–0.95).

В таблице 2 приводятся полученные метрики. Сеть YOLO показывает незначительно лучший результат в средней точности распознавания, в сравнении с Faster R-CNN.

Таблица 2 – метрики обученных сетей Faster R-CNN v.2 и YOLO v.12

|  |  |
| --- | --- |
| Faster R-CNN v.2 | YOLO v.12 |
| Val/AP50-95 = 0,9028 | mAP50-95 = 0,9211 |

Передадим на вход каждой модели по два тестовых изображения банковской карты. Выходные результаты приведены в таблице 3.

Таблица 3 – Визуализация выходного результата обученных моделей

| Faster R-CNN v.2 | YOLO v.12 |
| --- | --- |
|  |  |
|  |  |

Сеть Faster R-CNN показывает уверенность в обнаруженных областях равную единице, YOLO показывает чуть хуже значение колеблющееся от 0,88 до 0,92. Faster R-CNN даёт более точные боксы, однако иногда промахивается и не находит некоторые важные области, а иногда принимает похожие области реквизитов за искомые. YOLO менее уверенно детектирует объекты, хотя показатель практически равен единице, но зато находит точно все классы объектов. В скорости обучения и в инференсе YOLO гораздо быстрее, что, несомненно, имеет преимущество в реальных задачах.

## Распознавание текста с использованием популярных библиотек OCR

### Библиотека KerasOCR

Код для работы с фреймворками OCR достаточно прост:

# Импорт необходимых библиотек

import matplotlib.pyplot as plt

import keras\_ocr

directory = 'output' # директория с изображениями

# Получаем список файлов

image\_files = [f for f in os.listdir(directory)

if f.lower().endswith(

('.png', '.jpg', '.jpeg', '.bmp', '.tiff'))]

if not image\_files:

print(f"В директории {directory} не найдено изображений")

else:

print(f"Найдено {len(image\_files)} изображений.")

# Обрабатываем каждое изображение

for img\_file in image\_files:

img\_path = os.path.join(directory, img\_file)

print(f"\nОбработка файла: {img\_file}")

pipeline = keras\_ocr.pipeline.Pipeline()

image = keras\_ocr.tools.read(img\_path)

predictions = pipeline.recognize([image])

keras\_ocr.tools.drawAnnotations(image, predictions[0])

plt.imshow(image)

plt.show()

В каталоге output/ находятся изображения продетектированных областей с банковской карты. В результате работы программы получаем четыре изображения с указанием на распознанные символы представленные в таблице 4.

Таблица 4 – результаты распознавания текста с помощью KerasOCR

|  |  |
| --- | --- |
| Карта «Сбербанка» | Карта банка «Авангард» |
|  |  |
|  |  |
|  |  |
|  |  |

На основании изображений из таблицы 4 можно сделать вывод, что фреймворк KerasOCR хорошо справляется с распознаванием текстов содержащих буквы, но совершенно непригоден для полей с цифровыми данными или полей со смешанным набором символов. Для проверки набора символов стандартной модели выполним код:

recognizer = keras\_ocr.recognition.Recognizer()

print(recognizer.alphabet)

На выходе получаем набор символов:

Looking for /home/lastinm/.keras-ocr/crnn\_kurapan.h5

0123456789abcdefghijklmnopqrstuvwxyz

Следовательно символы цифр присутствуют в наборе, но по какой-то причине модель их практически не узнает. Обучение модели KerasOCR выходит за пределы данного исследования, поэтому исключим этот фреймворк из дальнейшего исследования.

### Библиотека EasyOCR

Программный код для распознавания текста при помощи библиотеки EasyOCR (файлы изображений продетектированных областей находятся в каталоге output/):

# Импорт библиотек

import easyocr

def recognize\_images\_in\_directory(directory, languages=['en', 'ru'], gpu=False):

# Инициализация EasyOCR

reader = easyocr.Reader(languages, gpu=gpu)

# Получаем список файлов

image\_files = [f for f in os.listdir(directory)

if f.lower().endswith(

('.png', '.jpg', '.jpeg', '.bmp', '.tiff'))]

if not image\_files:

print(f"В директории {directory} не найдено изображений")

return

print(f"Найдено {len(image\_files)} изображений.")

# Обрабатываем каждое изображение

for img\_file in image\_files:

img\_path = os.path.join(directory, img\_file)

print(f"\nОбработка файла: {img\_file}")

try:

# Извлекаем класс из имени файла

class\_id = img\_file.split('\_')[0]

# Настройки распознавания в зависимости от класса

allowlist = None

if class\_id == '2': # Только цифры

allowlist = '0123456789'

print("Режим распознавания: только цифры")

elif class\_id == '3': # Цифры и символ '/'

allowlist = '0123456789/'

print("Режим распознавания: цифры и символ /")

# Загрузка и предобработка изображения

img = cv2.imread(img\_path)

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

# Увеличьте маленький текст:

img = cv2.resize(img, None, fx=2, fy=1,

interpolation=cv2.INTER\_CUBIC)

# Распознавание

results = reader.readtext(img,

allowlist=allowlist,

text\_threshold=0.3,

link\_threshold=0.3,

width\_ths=0.3,

slope\_ths=0.2,

ycenter\_ths=0.3)

# Визуализация с указанием класса

visualize\_enhanced\_results(img\_path, results,

class\_id, img)

except Exception as e:

print(f"Ошибка обработки {img\_file}: {str(e)}")

def visualize\_enhanced\_results(orig\_path, results,

class\_id, processed\_img):

orig\_img = cv2.cvtColor(cv2.imread(orig\_path),

cv2.COLOR\_BGR2RGB)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))

# Оригинальное изображение

ax1.imshow(orig\_img)

ax1.set\_title(f"Оригинал (Класс {class\_id})")

# Обработанное изображение

ax2.imshow(processed\_img, cmap='gray')

ax2.set\_title("После обработки")

# Рисуем bounding boxes на оригинале

for (bbox, text, prob) in results:

(tl, tr, br, bl) = bbox

tl = (int(tl[0]), int(tl[1]))

br = (int(br[0]), int(br[1]))

rect = patches.Rectangle(tl, br[0]-tl[0], br[1]-tl[1],

linewidth=2, edgecolor='r',

facecolor='none')

ax2.add\_patch(rect)

ax2.text(tl[0], tl[1]-10, f"{text} ({prob:.2f})",

color='red', fontsize=10,

bbox=dict(facecolor='white', alpha=0.8))

plt.tight\_layout()

plt.show()

# Вывод результатов

print("Распознанный текст:")

for i, (\_, text, prob) in enumerate(results):

print(f"{i+1}. {text} (точность: {prob:.2f})")

В данном программном коде производится расширение полученного изображения, в 2 раза, т.к. изображения в датасете были приведены к размеру 640х640 пикселей, что приводило к сжатию букв. Растяжение изображения иногда помогает библиотеке EasyOCR лучше узнавать такие буквы, как, например, «M» или «H». Далее, для повышения точности распознавания полей содержащих только цифры, или цифры и символ «/», сформированы списки символов allowlist, которые используются при распознавании поля соответствующего класса.

Функция visualize\_enhanced\_results выводит графическое отображение полученного результата, а также результат дублируется текстовым выводом в консоль.

Пример вывода программы для двух банковских карт приведен в таблице 5.

Таблица 5 – результаты распознавания текста с помощью EasyOCR

|  |
| --- |
| Карта «Сбербанка» |
| Распознанный текст:  «Jiksi LASTM»  Ошибки в 7 из 12 букв. |
| Распознанный текст:  «0400 1641 4684»  Ошибочно распознаны шесть цифр из шестнадцати. |
| Распознанный текст:  «01/24»  Ошибка в одной цифре. |
| Распознанный текст:  «0 7005k R»  Множественные ошибки. |
| Карта банка «Авангард» |
| Распознанный текст:  «K X)M LSTUR»  Множественные ошибки. |
| Распознанный текст:  «5332 0602 3118 10»  Пять цифр ошибочны. |
| Распознанный текст:  «10/2»  Ошибка в одной цифре. |

По данным, представленным, в таблице 5 можно заметить, что качество распознавания текста у библиотеки EasyOCR уже значительно лучше, чем у библиотеки KerasOCR. В поле номера карты мы видим много ошибок, но все же точность уже выше 50%. В поле имени держателя карты ошибок значительно больше, точность явно менее 50%. Поле срока действия всего четыре цифры и разделитель «/», однако и тут в двух случаях произошла ошибка в одной цифре.

Библиотеку EasyOCR так же нельзя использовать без дополнительного обучения для распознавания текстовой информации с банковских карт, однако мы возьмем ее результаты распознавания EasyOCR для сравнения с еще одной библиотекой - TrOCR.

### Библиотека TrOCR

Программный код для распознавания текста на изображении при помощи библиотеки TrOCR следующий (предполагается, что изображения полей находятся в каталоге output/):

# импорт необходимых библиотек

from transformers import (TrOCRProcessor,

VisionEncoderDecoderModel)

import matplotlib.pyplot as plt

import torch

import cv2

import os

from PIL import Image

import numpy as np

# Загрузка модели и процессора

processor = TrOCRProcessor.from\_pretrained(

"microsoft/trocr-base-printed")

model = VisionEncoderDecoderModel.from\_pretrained(

"microsoft/trocr-base-printed")

def visualize\_enhanced\_results\_one\_img(orig\_path, confidences,

class\_id, processed\_img, full\_text):

"""Визуализация обработанного изображения"""

# Создаем фигуру только для обработанного изображения

plt.figure(figsize=(12, 8))

# Преобразование PIL.Image в numpy array

if isinstance(processed\_img, Image.Image):

processed\_img\_array = np.array(processed\_img)

# Отображаем обработанное изображение

plt.imshow(processed\_img\_array, cmap='gray')

plt.title(f"Результаты распознавания (Класс {class\_id})")

plt.axis('off') # Скрываем оси

plt.tight\_layout()

plt.show()

# Вывод результатов

print("Соответствие символов и уверенностей:")

for char, conf in zip(full\_text, confidences):

print(f"'{char}': {conf:.1%}")

# Извлечение уверенности

def get\_text\_with\_confidence(outputs):

# Получаем все токены (игнорируя <s> и </s>)

tokens = outputs.sequences[0][1:-1]

# Собираем результаты

result\_text = []

confidences = []

# Обрабатываем каждый токен

for i, token in enumerate(tokens):

# Получаем текстовое представление токена

token\_text = processor.decode([token],

skip\_special\_tokens=True)

# Получаем уверенность для этого токена

confidence = torch.softmax(outputs.scores[i],

dim=-1)[0, token].item()

# Для каждого символа в декодированном токене

# добавляем уверенность

for char in token\_text:

result\_text.append(char)

confidences.append(confidence)

# Собираем полный текст

full\_text = ''.join(result\_text)

return full\_text, confidences

# Распознает текст на изображениях

def recognize\_images\_in\_directory(directory):

# Получаем список файлов

image\_files = [f for f in os.listdir(directory)

if f.lower().endswith(

('.png', '.jpg', '.jpeg', '.bmp', '.tiff'))]

if not image\_files:

print(f"В {directory} не найдено изображений")

return

print(f"Найдено {len(image\_files)} изображений.")

# Обрабатываем каждое изображение

for img\_file in image\_files:

img\_path = os.path.join(directory, img\_file)

print(f"\nОбработка файла: {img\_file}")

try:

# Извлекаем класс из имени

class\_id = img\_file.split('\_')[0]

# Загрузка и предобработка изображения

image = Image.open(img\_path).convert("RGB")

pixel\_values = processor(image,

return\_tensors="pt").pixel\_values

# Генерация текста

outputs = model.generate(

pixel\_values,

output\_scores=True,

return\_dict\_in\_generate=True,

max\_length=50

)

# Результат

full\_text, confidences = get\_text\_with\_confidence(

outputs)

# Визуализация с указанием класса

visualize\_enhanced\_results\_one\_img(img\_path,

confidences, class\_id, image, full\_text)

except Exception as e:

print(f"Ошибка обработки {img\_file}: {str(e)}")

output\_dir = 'output' # директория с изображениями

recognize\_images\_in\_directory(output\_dir)

Библиотека TrOCR возвращает результат в токенах. Токены обрабатываем функцией get\_text\_with\_confidence. Из токинов собираем полный текст и извлекаем информацию об уверенности для каждого символа. Функция visualize\_enhanced\_results\_one\_img отображает каждое изображение для класса и выводит результат распознавания. В таблице 6 приведен вывод программы для двух банковских карт.

Таблица 6 – результаты распознавания текста с помощью TrOCR

|  |  |
| --- | --- |
| Карта «Сбербанка» | Карта банка «Авангард» |
| Распознанный текст:  'MAK3LN LASTIN'  Три ошибки. | Распознанный текст:  'MAXIM LASTIN'  Без ошибок. |
| Распознанный текст:  '12 15-0400 1641 1884'  Шесть ошибок из шестнадцати цифр. | Распознанный текст:  '5392 0602 3179 2005'  Без ошибок. |
| Распознанный текст:  '07/24'  Без ошибок. | Распознанный текст:  '10/23'  Без ошибок. |
| Распознанный текст:  '863770054 R'  Одна ошибка: обратный слэш «/» распознан, как цифра «7». |  |

Анализ полученных результатов распознавания двух банковских карт показывает, что на изображении карты банка «Авангард» библиотека TrOCR совершенно точно распознала все символы. На карте «Сбербанка» точно распознано поле даты действия карты, во втором поле держателя карты допущена одна ошибка: символ «/» воспринят, как «7». В полях держателя карты и номер карты допущено неприемлемо много ошибок.

Сравнивая два изображения банковских карт, можно предположить, что негативное влияние на распознавание вносит фон изображения. Попробуем его удалить при помощи адаптивной бинаризации. Обработаем распознаваемые изображения с помощью следующей функции:

def remove\_background(img\_path):

img = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)

# Получаем текущие размеры (высота, ширина)

height, width = img.shape[:2]

# Увеличиваем в 2 раза

new\_width = width \* 2

new\_height = height \* 1

# Масштабируем с интерполяцией

# (лучшее качество для увеличения)

img = cv2.resize(img, (new\_width, new\_height),

interpolation=cv2.INTER\_LANCZOS4)

# Адаптивная бинаризация

binary = cv2.adaptiveThreshold(img, 255,

cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,

cv2.THRESH\_BINARY\_INV, 19, 6)

# Удаление шумов

kernel = np.ones((2,2), np.uint8)

cleaned = 255 - cv2.morphologyEx(binary,

cv2.MORPH\_OPEN, kernel) # Инвертируем обратно

# Автоматическая бинаризация (Otsu)

\_, thresh = cv2.threshold(cleaned, 0, 255,

cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

# Инвертирование обратно

binary = cv2.bitwise\_not(thresh)

# Конвертируем обратно в RGB (чтобы TrOCR не ругался)

result = cv2.cvtColor(binary, cv2.COLOR\_GRAY2RGB)

return Image.fromarray(result)

Полученные результаты распознавания на изображениях с удаленным фоном приведены в таблице 7.

Таблица 7 – результаты распознавания TrOCR на изображениях с удаленным фоном

|  |  |
| --- | --- |
| Карта «Сбербанка» | Карта банка «Авангард» |
| Распознанный текст:  'TAX8.00 LAST ID'  Более 50% ошибки. | Распознанный текст:  'MAXIN LASTIN'  Ошибка в одной букве. |
| Распознанный текст:  '421E-0400 16471584'  Четыре цифры ошибочны. | Распознанный текст:  '5332 0602 3139 2005'  Ошибка в одной цифре. |
| Распознанный текст: '07/24'  Без ошибок. | Распознанный текст: '10/22'  Ошибка в одной цифре. |
| Распознанный текст: '3637/0054 R'  Без ошибок. |  |

Результаты распознавания символов на изображении для карты «Сбербанка» представлены в таблице 8:

Таблица 8 – результат распознавания текста карты «Сбарбанка» с изображения с фоном и удаленным фоном

|  |  |  |
| --- | --- | --- |
| Реквизит | Изображение с фоном | Изображение без фона |
| Держатель карты | Три ошибки. | Более 50% ошибки. |
| Номер карты | Шесть ошибок из шестнадцати цифр. | Четыре цифры ошибочны. |
| Срок действия | Без ошибок. | Без ошибок. |
| Держатель карты 2 | Одна ошибка. | Без ошибок. |

Результаты распознавания символов на изображении для карты банка «Авангард» представлены в таблице 8:

Таблица 9 – результат распознавания текста карты банка «Авангард» с изображения с фоном и удаленным фоном

|  |  |  |
| --- | --- | --- |
| Реквизит | Изображение с фоном | Изображение без фона |
| Держатель карты | Без ошибок. | Ошибка в одной букве. |
| Номер карты | Без ошибок. | Ошибка в одной цифре. |
| Срок действия | Без ошибок. | Ошибка в одной цифре. |

На основании данных из таблиц 8 и 9 можно сделать следующие выводы: удаление фона не увеличивает качество распознавания символов фреймворком TrOCR, скорее наоборот, алгоритм вырезания фона мешает точному распознаванию. Далее в сравнении результатов распознавания фреймворками EasyOCR и TrOCR будем проводить эксперименты на оригинальных изображениях.

### Сравнение результатов распознавания

Для того, чтобы оценить качество распознавания и сравнить между собой разные фреймворки необходимо иметь эталонное описание полей. Для удобного и быстрого формирования такого описания разработано приложение с веб-формой на FastAPI. Программный код приведен в приложении 2. Форма для заполнения аннотаций приведена на рисунке 29.

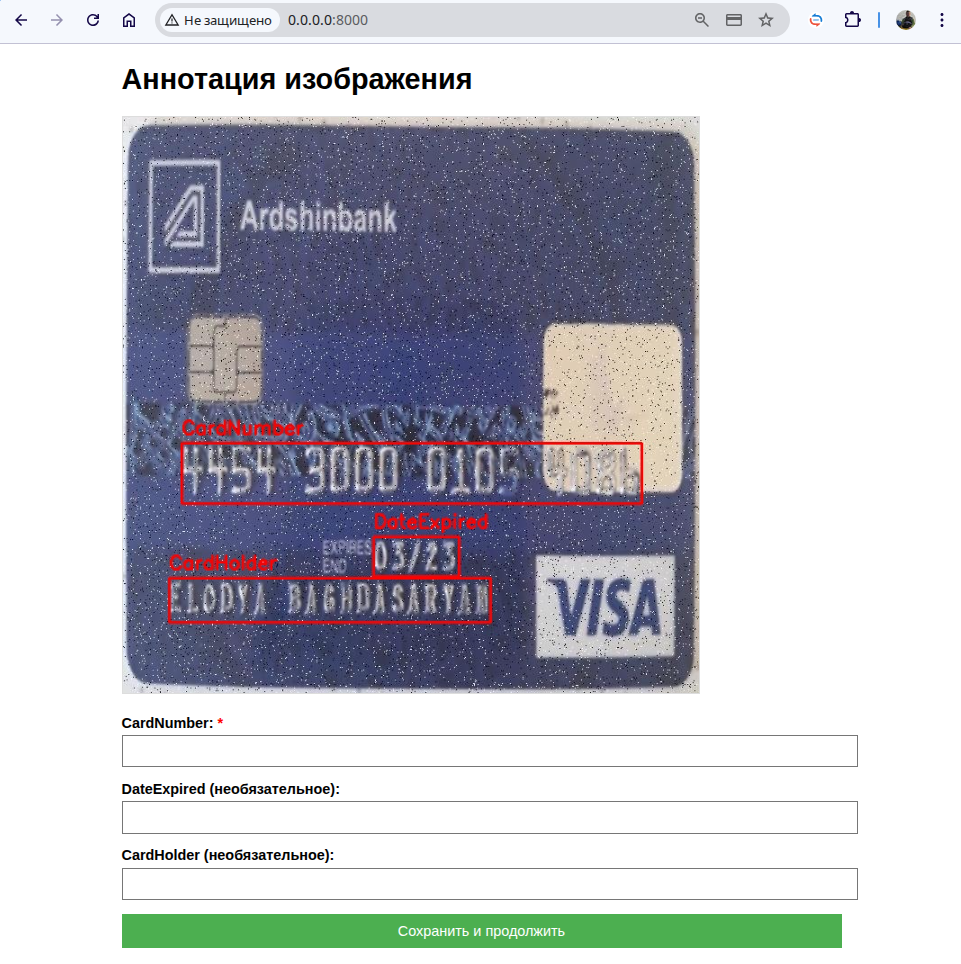


Рисунок 29 – форма приложения для составления аннотаций к изображениям

На выходе получаем файл image\_data.cvs содержащий следующие поля: image, CardHolder, CardNumber, DateExpired, где image – имя файла изображения. Для целей сравнения результатов работы библиотек EasyOCR и TrOCR были составлены эталонные аннотации к 155 изображениям карт, которые содержат информацию о 461 поле банковских карт.

Далее передаем полученный файл с эталонными полями на вход скрипта 1\_OCR\_recognition.py (программный код приведен в приложении 2), который выполняет следующие действия:

1. осуществляет детектирование полей в каждом изображении с помощью предобученной модели YOLOv12;
2. выделенные области передаются для распознавания текста в EasyOCR и TrOCR;
3. осуществляется подсчет метрики точности распознавания для каждого поля;

На выходе имеем файл results.csv со следующими полями:

* image – наименование файла изображения банковской карты;
* field\_type – наименование реквизита;
* true\_text – эталонный текст;
* easyocr\_text – текст распознанный EasyOCR;
* easyocr\_similarity – сходство с эталоном для EasyOCR;
* easyocr\_exact\_match – точно распознано да/нет для EasyOCR;
* trocr\_text – текст распознанный TrOCR;
* trocr\_similarity – сходство с эталоном для TrOCR;
* trocr\_exact\_match – точно распознано да/нет для TtOCR;
* processing\_time – время обработки;
* confidence – уверенность детектирования;
* bbox – координаты ограничительной рамки.

Подсчет точных совпадений заносится в поля easyocr\_exact\_match/ trocr\_exact\_match.

Сравнение распознанной строки с эталонной easyocr\_similarity/ trocr\_similarity определяем путем расчета относительного расстояния Левинштейна [25].

Расстояние Левенштейна (другое название редакционное расстояние) определяется как минимальное количество односимвольных операций (вставки, удаления, замены), необходимых для превращения одной последовательности символов в другую. Расстояние Левенштейна, нормированное в диапазон от 0 до 1, называется нормированным расстоянием Левенштейна или коэффициентом схожести Левенштейна, определяемым по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

где str1 и str2 – сравниваемы строки.

Значения коэффициента схожести Левенштейна могут принимать значения: 0 — полное совпадение строк и 1 — полностью разные строки (максимальное расстояние). Существует вариант для меры схожести (а не расстояния). Например, если нужно получить показатель схожести от 0 до 1 (где 1 — полное совпадение), используют формулу:

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

Проанализируем собранные данные в файле результатов с помощью скрипта 2\_Recognition\_data\_analysis.py (программный код приведен в приложении 3). Скрипт выполняет следующие задачи:

* вычисляет общую статистику по ошибкам распознавания;
* строит график распределения схожести распознавания текста;
* топ-10 ошибок для каждого класса;
* анализ условно точных совпадений;
* сохранение изображений для топ-10 ошибочных распознаваний по каждому фреймворку и классу;

На рисунке 30 приведена гистограмма распределения схожести распознания текста. Наглядно видно, что точность распознавания текстовых реквизитов банковских карт фреймворком TrOCR заметно выше, чем точность у EasyOCR. Фрейворк TrOCR не только делает больше точных распознований, но также выдает намного меньше полностью неверных ответов.

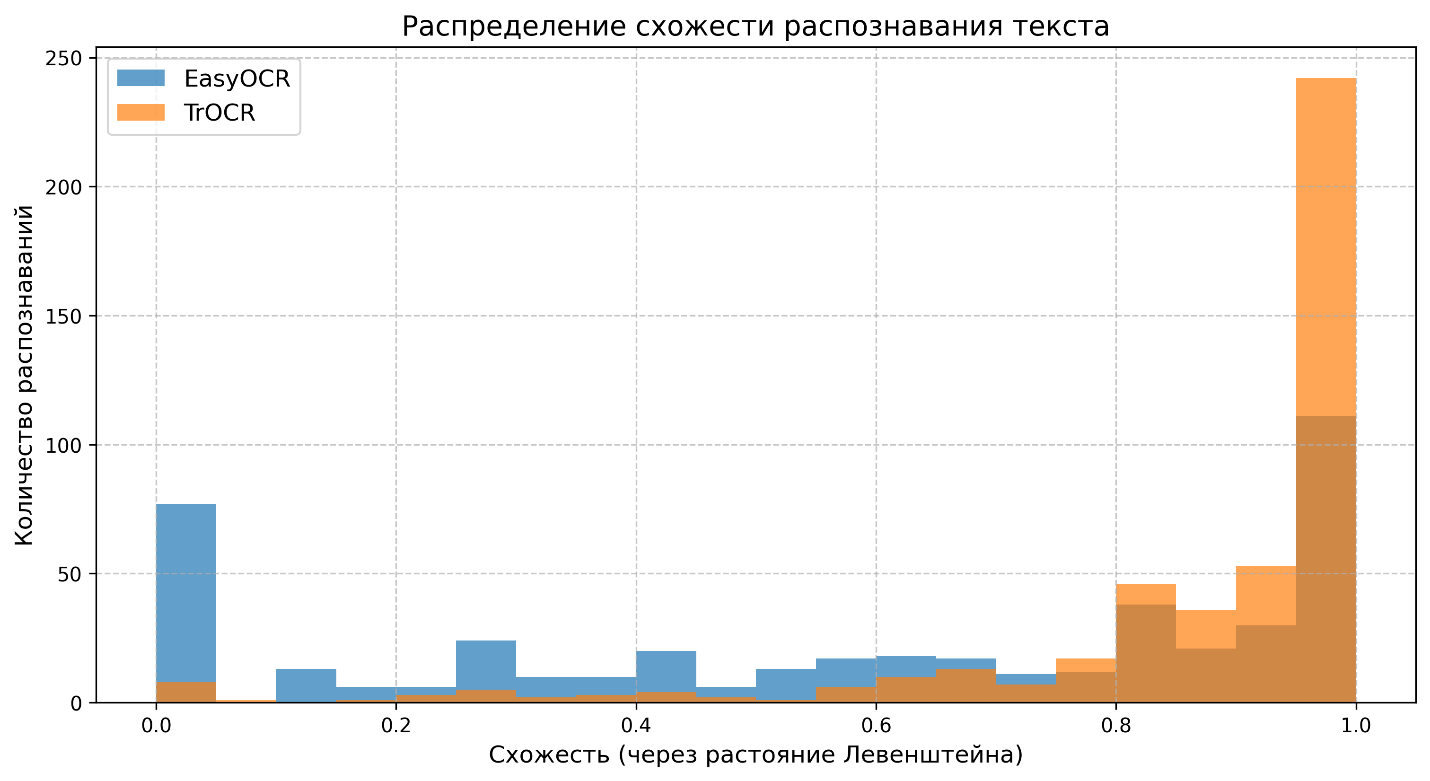


Рисунок 30 – распределение схожести распознавания текста для библиотек EasyOCR и TrOCR

Общая статистика по ошибкам распознавания приведена в таблице 10. Под ошибками здесь подразумеваются случаи, когда в поле допущена хоть одна ошибка. В таком случае результат полностью отвергается. TrOCR допустил 48,7% таких ошибок, а EasyOCR – 76,52%.

Таблица 10 - общая статистика по ошибкам распознавания

|  |  |
| --- | --- |
| Библиотека | Показатели |
| EASYOCR | Всего ошибочных распознаваний: 352  Процент ошибок: 76.52%  Точность распознавания: 23.48% |
| TROCR | Всего ошибочных распознаваний: 224  Процент ошибок: 48.70%  Точность распознавания: 51.30% |
|  | Всего записей в данных: 460 |

Под условно точными совпадениями будем принимать случаи, когда строки совпали после приведения их нормализованному виду, т.е. убраны все пробелы, символы приведены в верхний регистр, например:

Примеры (EasyOCR):

Изображение: Media1\_jpeg\_jpg.rf.9863f37fb8e4d11c5d2bb482286b7ad1.jpg

Тип: CardNumber

Эталон: '4454 3000 0357 2853'

Распознано: '44543000 0357 2853'

Примеры (TrOCR):

Изображение: card22\_jpg.rf.15f0a57a40498b78119314e51a509e5d.jpg

Тип: DateExpired

Эталон: '05/27'

Распознано: '05 / 27'

Величина условно точных совпадений в числе точных совпадений приведена в таблице 11:

Таблица 11 – условно точные совпадения в числе точных совпадений

|  |  |
| --- | --- |
| Библиотека | Показатели |
| EasyOCR | Процент условно точных совпадений: 12.04%  Всего точных совпадений: 108  Из них условно точных: 13 |
| TrOCR | Процент условно точных совпадений: 1.69%  Всего точных совпадений: 236  Из них условно точных: 4 |

Опять можно видеть лучшую работу по распознаванию у TrOCR: этот фреймворк лучше определяет пробелы и регистр символов, чем EasyOCR.

В каталоге top\_fails сохранены топ-10 изображений ошибочных распознаваний. Пример изображений приведен в таблице 12:

Таблица 12 – пример изображений с максимальным количеством ошибок распознавания

|  |  |
| --- | --- |
| Фреймворк: EasyOCR | |
|  |  |
| Фреймворк: TrOCR | |
|  |  |

Из изображений таблицы 12 можно сделать вывод, что случаи встречаются иногда действительно сложные, даже человеческим взглядом сложно точно определить, что написано. Однако непонятно, почему для EasyOCR посчитана схожесть 0.0 на первом изображении, хотя большинство символов распознано верно. Анализ проблемы привел к неверной разметке датасета. Для номера карты указан класс CardHolder. Следовательно, анализ данных и сбор статистики помимо выявления лучшего фремворка OCR помогают также выявить проблемы и ошибки в датасете и обучении моделей.

В таблице 13 приведен рейтинг Топ-10 проблемных символов для обоих фремворков. Эти данные получены в результате работы скрипта 3\_Top10\_fail\_simbols.py (программный код приведен в приложении 4).

Таблица 13 – топ-10 проблемных символов для распознавания

|  |  |  |  |
| --- | --- | --- | --- |
| Фреймворк: EASYOCR | | | |
| Рейтинг | Эталонный | Распознанный | Количество |
| 1  2  3  4  5  6  7  8  9  10 | '0'  ' '  '2'  'A'  '1'  'N'  '4'  '0'  '7'  'E' | None  None  None  None  None  None  None  ' '  None  None | 254  134  94  85  75  74  63  63  54  48 |
| Фреймворк: TROCR | | | |
| Рейтинг | Эталонный | Распознанный | Количество |
| 1  2  3  4  5  6  7  8  9  10 | '6'  '9'  '0'  '1'  ' '  'N'  '7'  '3'  '1'  '2' | '5'  '4'  None  '0'  None  None  '1'  '9'  None  None | 28  27  26  25  23  22  21  20  18  17 |

Из данных таблицы 13 можно сделать ввод, что фреймворк EasyOCR при неуверенности в символе ничего не подставляет, в отличие от TrOCR, который что-то предлагает. Видно, что EasyOCR плохо распознает пробелы между символами. В таких случаях этот фреймворк ошибается в шесть раз чаще. TrOCR иногда делает ошибки заменяя «6» на «5», «9» на «4», «7» на «1», «3» на «9», что вполне логично при сложном фоне и плохой читаемости.

Распределение схожести распознавания разными библиотеками для наглядности удобно изобразить на графике называемом «Ящик с усами». Для номера карты такой график изображен на рисунке 31.

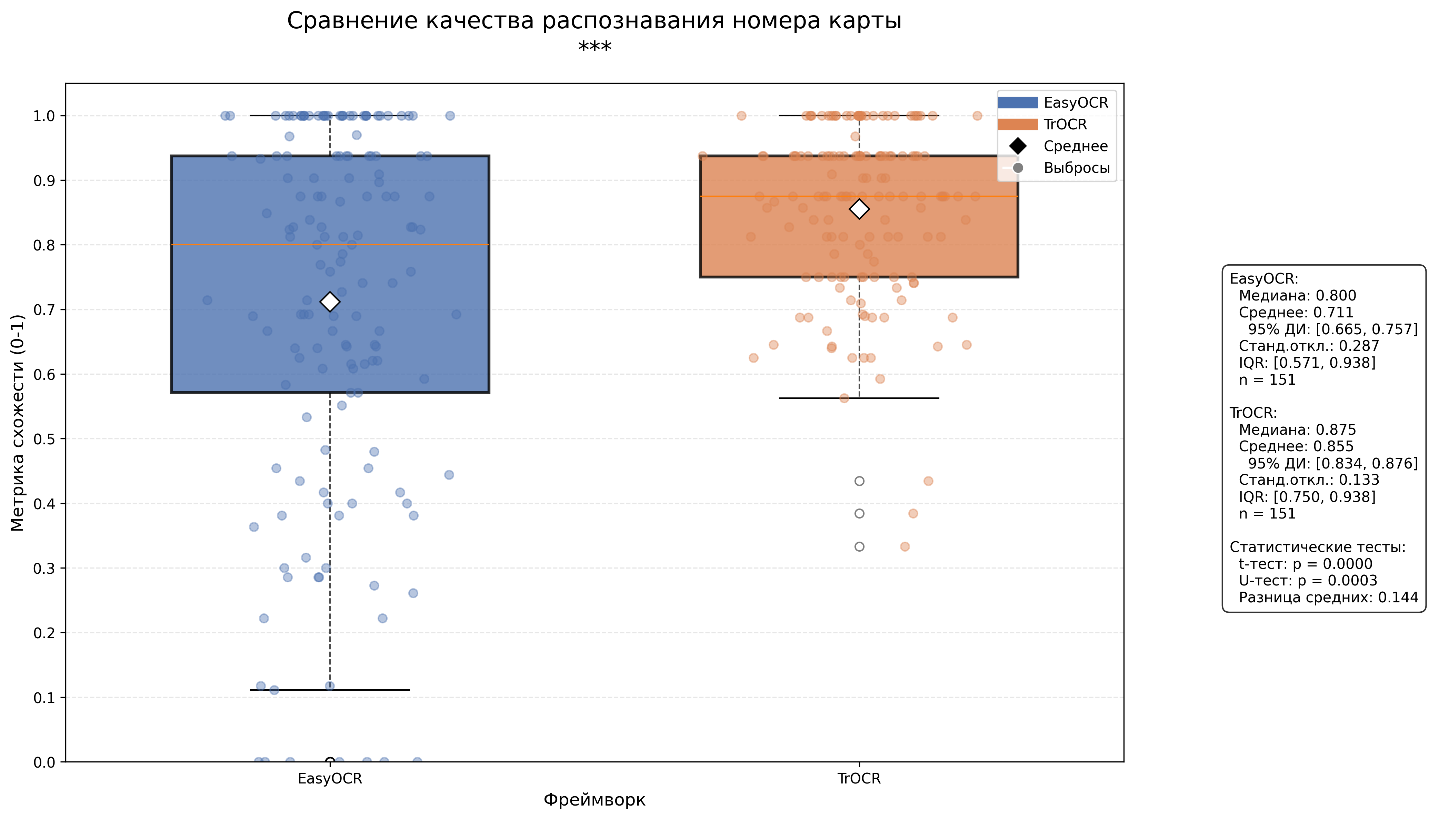


Рисунок 31 – распределение схожести распознавания текста реквизита номера карты

Результаты статистических тестов указывают на статистически значимые различия между качеством распознавания EasyOCR и TrOCR:

Значения p-value:

* t-тест: p = 0.0000. Сильно меньше 0.05, следовательно различия в средних весьма значимы. Подходит, если данные нормально распределены;
* U-тест Манна-Уитни: p = 0.0003. Также < 0.05, следовательно подтверждает значимость даже без предположения о нормальности. Более надежен при асимметричных распределениях.

Разница средних: 0.144:

* TrOCR в среднем показывает на 0.144 балла выше по метрике схожести;
* Для шкалы 0-1 это существенное улучшение (примерно 14.4%).

Статистическая значимость результата:

* оба теста единогласно отвергают нулевую гипотезу (что разницы нет);
* вероятность, что такие различия возникли случайно: < 0.03% (по U-тесту).

На рисунке 32 приведен график «Ящик с усами» для реквизита держателя карты.

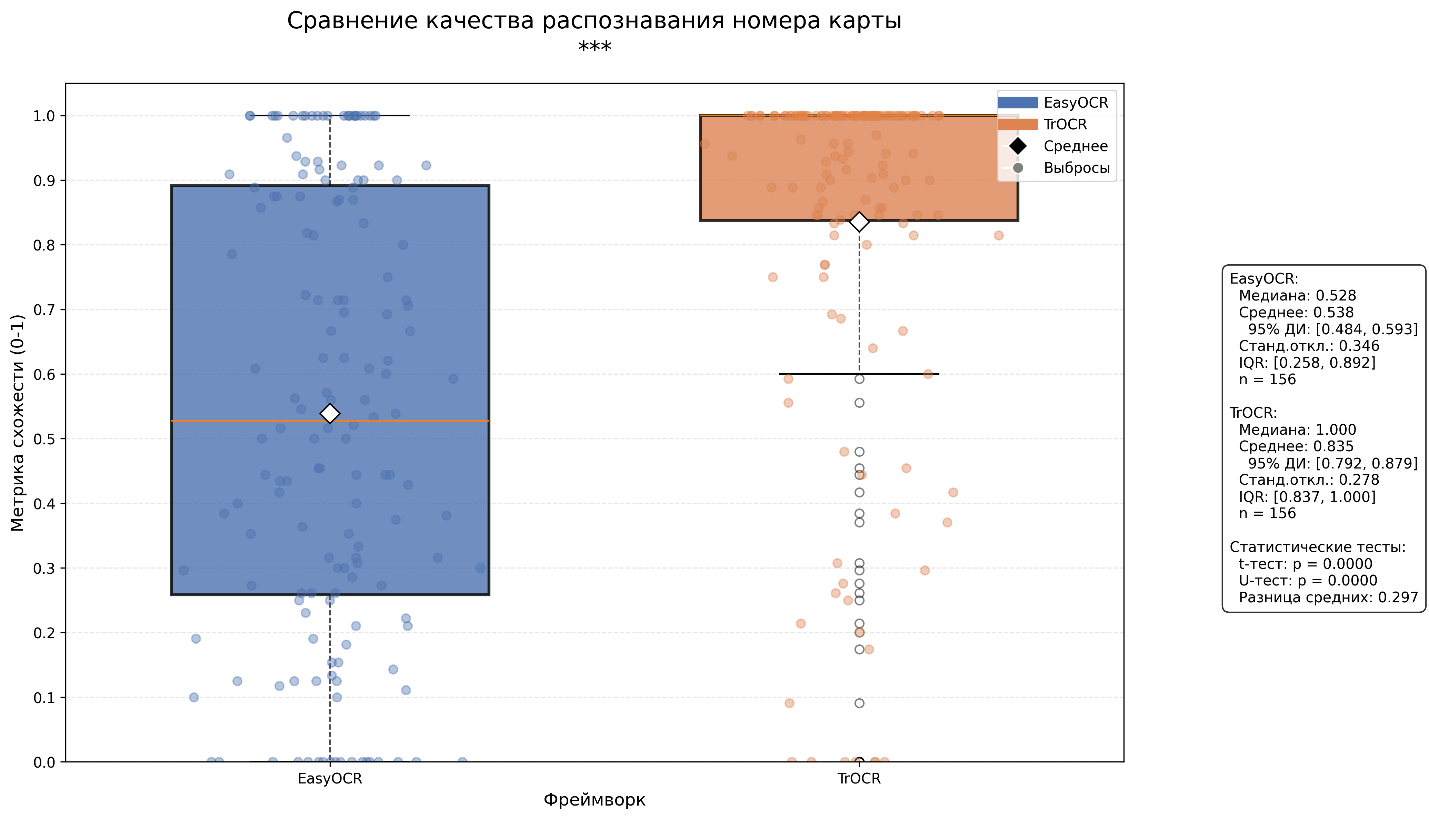


Рисунок 32 – распределение схожести распознавания текста реквизита держателя карты

Изучая график на рисунке 32 и рассуждая аналогично результатам поля номера карты можно сделать вывод о статистически значимой разнице между двумя библиотеками OCR и более высоком результате у TrOCR на 0,297.

На рисунке 33 показан график «ящик с усами» для реквизита срока действия карты.

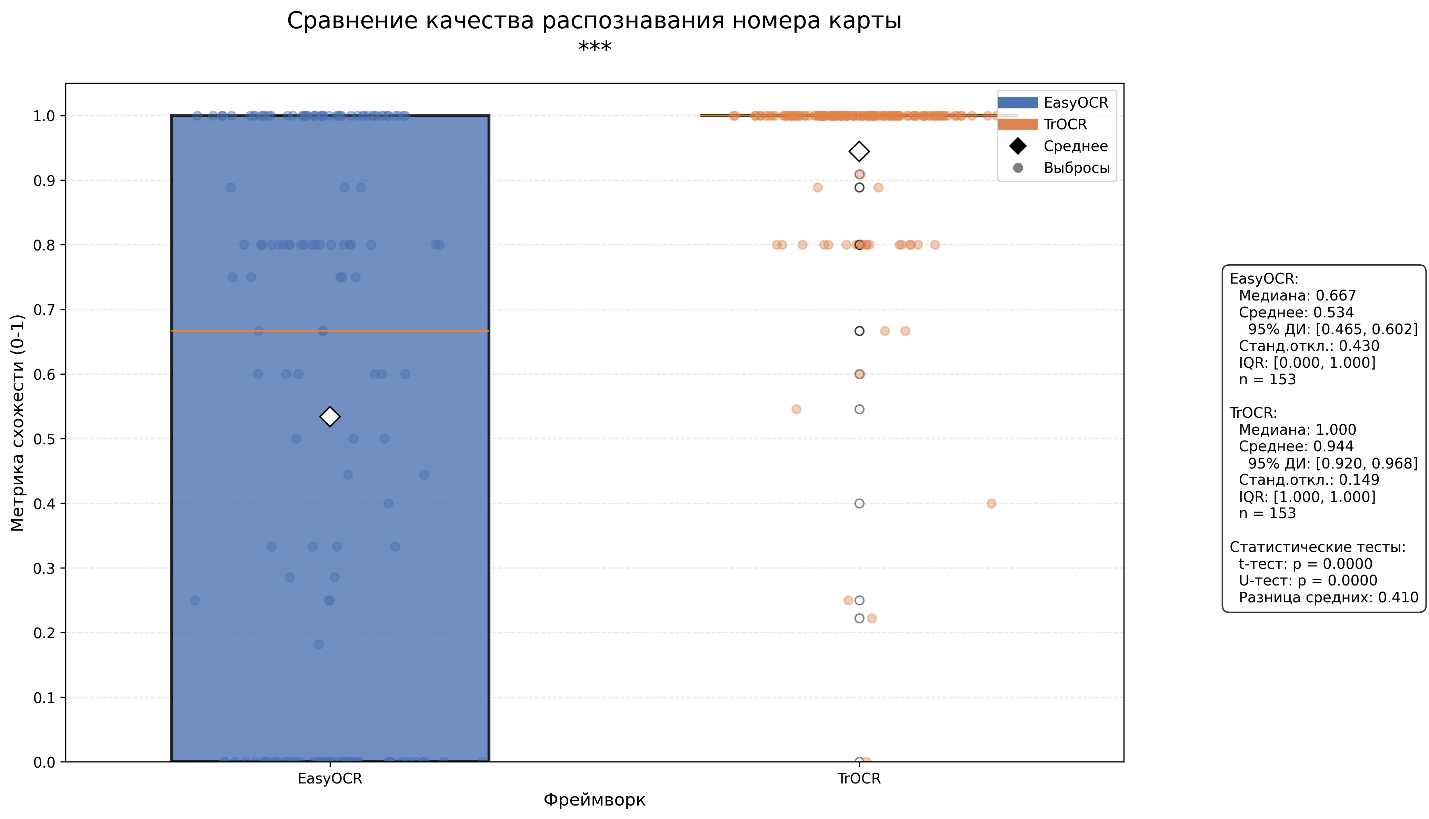


Рисунок 33 – распределение схожести распознавания текста реквизита срока действия карты

На рисунке 33 мы видим, что синий ящик имеет ширину от 0 до 1. Это указывает на большую вариативность распознавания EasyOCR. Напротив, у TrOCR ящик сужен до узкой полоски, что говорит о достаточно точных результатах. Статистические тесты показывают на значимость результата. Разница средних составила 0.41.

Далее посчитаем точеные оценки для обоих фреймворков. Для этого определимся, когда будем сравнивать слова, а когда символы. Оформим методы сравнения для каждого поля реквизита в таблице 14.

Таблица 14 – методика сравнения результатов для каждого класса

|  |  |  |
| --- | --- | --- |
| Тип поля (класс) | Уровень сравнения | Почему? |
| **CardHolder** | Слова | Имена и фамилии сравниваем отдельно (например, "IVAN PETROV" и "IVAN PETROFF" – имя совпало, фамилия нет) |
| **CardNumber** | Символы | Даже одна неверная цифра критична (например, "1234 5678" и "1234 5679")  Добавлена нормализация (игнорирование пробелов/регистра для номеров карт) |
| **DateExpired** | Символы | Формат жёсткий (MM/YY), ошибка в одном символе значима ("05/25" и "06/25") |

Под метрикой Word accuracy будем подразумевать точное совпадение на уровне всего поля (только приводим все буквенные символы к одному регистру: нижнему).

Для расчета точечных оценок разработан скрипт 4\_Calculate\_all\_metrics.py (программный код приведен в приложении 5).

Для расчета Word Accuracy применен метод точный биномиальный интервал (Clopper-Pearson):

* даёт гарантированное покрытие (не менее 95% для 95% ДИ);
* консервативен (шире, чем другие методы);
* лучше подходит для малых выборок (<30).

Реализация:

from scipy import stats

word\_acc\_ci = stats.binomtest(correct,n).proportion\_ci(

confidence\_level=confidence)

Для расчета интервальных оценок Precision и Recall применен интервал Вильсона (Wilson score interval):

* точно работает для крайних значений (~0 или ~1);
* подходит для любых размеров выборки;
* учитывает асимметрию распределения;
* лучше нормального приближения Вальда при малых выборках.

Реализация в коде:

from statsmodels.stats.proportion import proportion\_confint

precision\_ci = proportion\_confint(tp, tp+fp, alpha=1-confidence,

method='wilson') if (tp+fp) > 0 else (0, 1)

recall\_ci = proportion\_confint(tp, tp+fn, alpha=1-confidence,

method='wilson') if (tp+fn) > 0 else (0, 1)

Для посчтета интервальных оценок F1-Score применен метод Бутстреп (percentile bootstrap):

* не требует предположений о форме распределения;
* точно отражает вариабельность данных;
* особенно важен для несбалансированных классов.

Реализация в коде:

from sklearn.utils import resample

f1\_scores = []

for \_ in range(n\_bootstrap):

# Генерация бутстреп-выборки

sample\_true, sample\_pred = resample(true\_texts, pred\_texts)

# Расчет метрик для выборки

bt\_tp, bt\_fp, bt\_fn = 0, 0, 0

for t, p in zip(sample\_true, sample\_pred):

tpfpfn = calculate\_tp\_fp\_fn(t, p)

bt\_tp += tpfpfn[0]

bt\_fp += tpfpfn[1]

bt\_fn += tpfpfn[2]

bt\_precision = bt\_tp / (bt\_tp + bt\_fp) if (bt\_tp +

bt\_fp) > 0 else 0

bt\_recall = bt\_tp / (bt\_tp + bt\_fn) if (bt\_tp +

bt\_fn) > 0 else 0

bt\_f1 = 2 \* (bt\_precision \* bt\_recall) / (bt\_precision +

bt\_recall) if (bt\_precision + bt\_recall) > 0 else 0

f1\_scores.append(bt\_f1)

f1\_ci = np.percentile(f1\_scores, [2.5, 97.5])

Данные расчетов точечных оценок представлены в таблице 15. Из данных таблицы видим, что наибольшая точность распознавания у реквизита DateExpired. Лучший показатель F1-Score здесь у TrOCR и составляет 93,8%. Это логично, так как поле содержит всего пять символов.

Таблица 15 – точечные оценки распознавания фреймворками для каждого реквизита

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип поля (класс) | Фреймворк | Word Accuracy | Precision | Recall | F1-Score |
| CardHolder | EasyOCR | 0.128205 | 0.190635 | 0.182692 | 0.186579 |
| CardHolder | TrOCR | 0.49359 | 0.583851 | 0.602564 | 0.59306 |
| CardNumber | EasyOCR | 0.18543 | 0.691721 | 0.525445 | 0.597225 |
| CardNumber | TrOCR | 0.211921 | 0.806522 | 0.76748 | 0.786517 |
| DateExpired | EasyOCR | 0.313725 | 0.625 | 0.490196 | 0.549451 |
| DateExpired | TrOCR | 0.803922 | 0.943272 | 0.934641 | 0.938936 |

Интервальные оценки для доверительного интервала с точностью 95% представлены в таблице 16 (где CH - CardHolder, CN – CardNumber, DE - DateExpired).

Таблица 16 – доверительные интервалы (95%) для результатов распознавания

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип поля (класс)/ библиотека | Word Accuracy | Precision | Recall | F1-Score |
| CH/ EasyOCR | [0.080, 0.191] | [0.150, 0.239] | [0.144, 0.229] | [0.132, 0.245] |
| CH/ TrOCR | [0.413, 0.575] | [0.529, 0.636] | [0.547, 0.655] | [0.526, 0.658] |
| CN/ EasyOCR | [0.127, 0.257] | [0.670, 0.712] | [0.506, 0.545] | [0.538, 0.653] |
| CN/ TrOCR | [0.150, 0.286] | [0.790, 0.822] | [0.750, 0.784] | [0.749, 0.823] |
| DE/ EasyOCR | [0.241, 0.394] | [0.586, 0.663] | [0.455, 0.526] | [0.482, 0.619] |
| DE/ TrOCR | [0.732, 0.864] | [0.924, 0.958] | [0.915, 0.950] | [0.914, 0.963] |

Хуже всего дело обстоит с распознаванием реквизита CardHolder. Лучший показатель F1-Score принадлежит TrOCR и составляет 59,3%, а у EasyOCR составляет всего 18,6%. Худшие показатели для этого реквизита объясняются преобладанием количества буквенных символов над количеством цифр в поле CardNumber.

Примечательно, что интервальная оценка вероятности распознавания целиком реквизита держателя карты библиотекой EasyOCR составляет всего лишь от 8% до 19,1%.

Графически точечные и интервальные оценки изображены в виде гистограмм на рисунке 31.

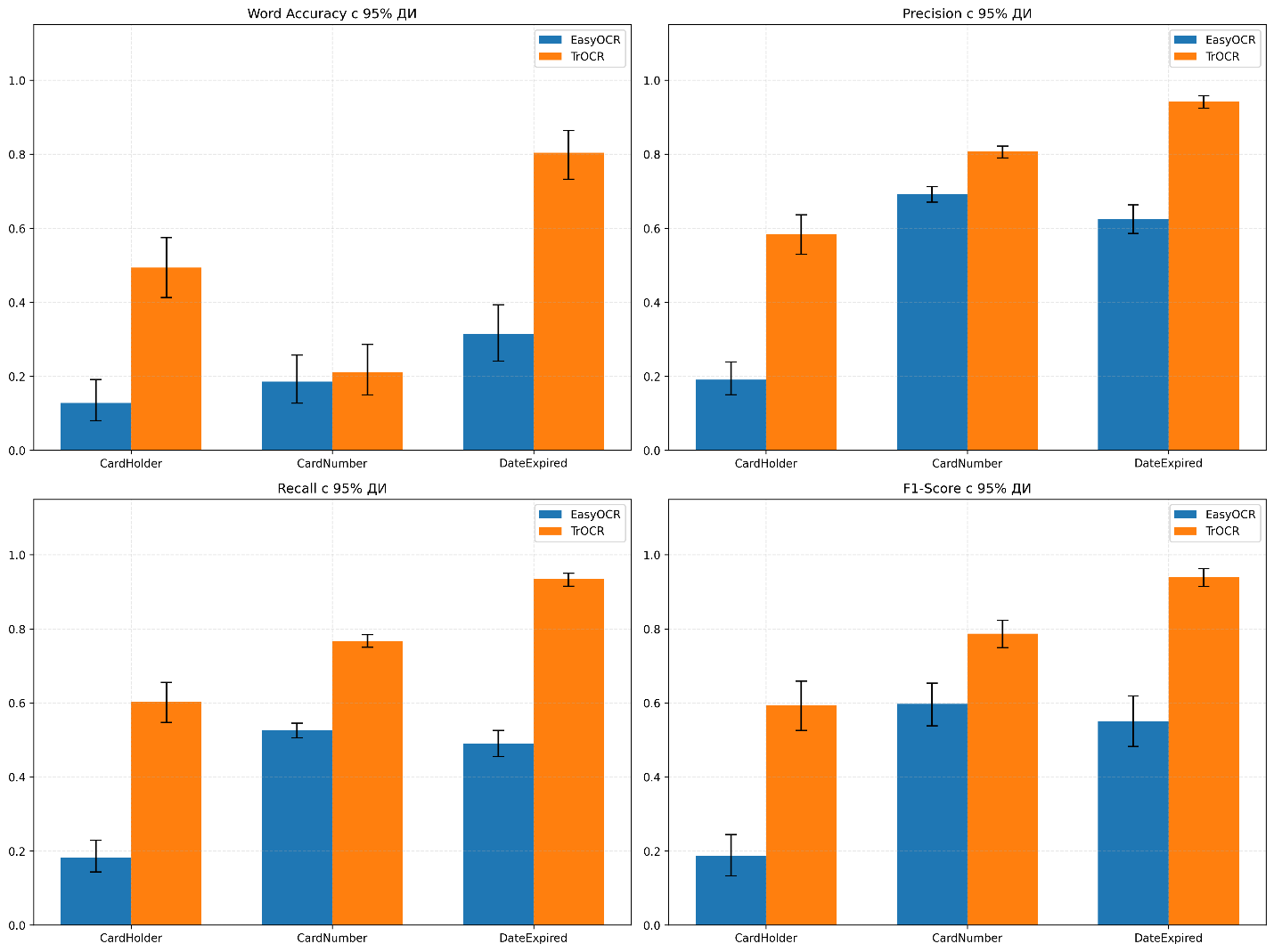


Рисунок 31 – точечные оценки распознавая текста с 95% доверительными интервалами

## 3.3 Пользовательский интерфейс для тестирования исследуемых методов

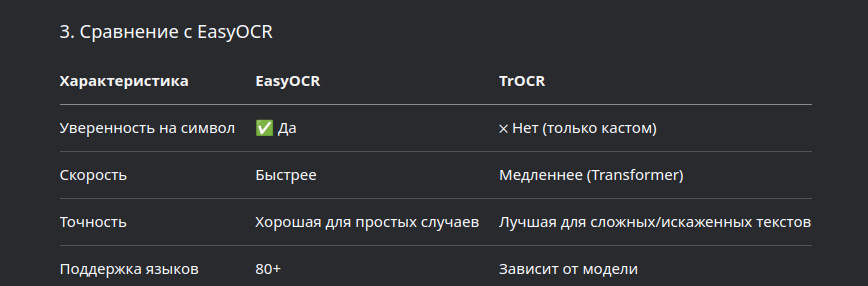
# ЗАКЛЮЧЕНИЕ

В ходе прохождения учебной практики была проведена работа с доступными источниками научной литературы и выполнены следующие задачи:

* проанализированы существующие методы и готовые решения для задачи по распознаванию реквизитов банковских карт, а также проанализирован метод решения схожей по составу задачи по распознаванию автомобильных номеров;
* изучено устройство сверточных нейронных сетей в задачах детектирования объектов на изображениях. Внимательно рассмотрены архитектуры Faster R-CNN и Yolo;
* проанализированы решения для оптического распознавания текста, произведено детальное рассмотрение и сравнение библиотек Tesseract и EasyOCR;
* сформирован обучающий датасет и установлены методы расчета метрик оценки качества результата распознавания.

Все поставленные цели прохождения учебной практики достигнуты. Полученные результаты и выводы будут учтены и использованы для дальнейшей работы при разработке приложения для распознавания реквизитов банковских карт.

Раздел 3. Разработка приложения



Модели обнаружения YOLO12 - это модели yolo11n.pt предварительно обученные на COCO.

<https://docs.ultralytics.com/ru/models/yolo12/#usage-examples>

<https://docs.ultralytics.com/ru/tasks/detect/>

<https://docs.ultralytics.com/ru/modes/predict/#key-features-of-predict-mode>

from ultralytics import YOLO

# Load a model

model = YOLO("yolo12n.yaml") # build a new model from YAML

model = YOLO("yolo12n.pt") # load a pretrained model (recommended for training)

model = YOLO("yolo12n.yaml").load("yolo12n.pt") # build from YAML and transfer weights

# Train the model

results = model.train(data="coco8.yaml", epochs=100, imgsz=640)

# Run inference with the YOLO12n model on the 'bus.jpg' image

results = model("path/to/bus.jpg")

# Process results list

for result in results:

boxes = result.boxes # Boxes object for bounding box outputs

masks = result.masks # Masks object for segmentation masks outputs

keypoints = result.keypoints # Keypoints object for pose outputs

probs = result.probs # Probs object for classification outputs

obb = result.obb # Oriented boxes object for OBB outputs

result.show() # display to screen

result.save(filename="result.jpg") # save to disk

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Луговая, М. В. Искусственный Интеллект В Маркетинговой Аналитике Банков / М.В. Луговая. Общество с ограниченной ответственностью «Наука и Просвещение», 2021. – С. 53-55.

2. Набиева, Г. Современные Модели И Методы Распознавания Текста: Технологии, Вызовы И Перспективы / Г. Набиева, Г. Нуржанов – Текст : непосредственный. // Norwegian Journal of Development of the International Science. 2024. Современные Модели И Методы Распознавания Текста. № 146. – С. 144-150.

3. Ревера, В. С. Особенности Применения Глубокого Обучения Для Обнаружения Номерных Знаков На Изображении И Их Последующая Классификация Методами Компьютерного Зрения / В.С. Ревера, Е.А. Шельмина – Текст : непосредственный. // Моделирование, Оптимизация И Информационные Технологии. 2024. Т. 12. № 4 (47).

4. Чен, Х. Распознавание Реквизитов Банковских Карт В Мобильных Устройствах По Видеопоследовательностям // Программирование. 2020. № 6. – С. 3-15.

5. Hussain, M. In-Depth Review of YOLOv1 to YOLOv10 Variants for Enhanced Photovoltaic Defect Detection / M. Hussain, R. Khanam – Text : direct // Solar. 2024. Vol. 4. № 3. – P. 351-386.

6. Ircham, A. N. The Design of a C1 Document Data Extraction Application Using a Tesseract-Optical Character Recognition Engine // Jurnal Resti (rekayasa Sistem Dan Teknologi Informasi). 2024. Т. 8. № 1. – С. 42-53.

7. Document Intelligence credit debit card model - Azure AI services laujan – Use Document Intelligence credit/debit card model extract key fields from credit and debit cards. – URL: https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/prebuilt/credit-card?view=doc-intel-4.0.0 (дата обращения: 15.03.2025) – Текст: электронный.

8. Padilla, R. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit // Electronics. 2021. Vol. 10. № 3. – P. 279.

9. Team, T. A. Understanding Pascal VOC and COCO Annotations for Object Detection | Towards AI / T. A. Team. – 2020.

10. VOC Ultralytics – Откройте для себя набор данных PASCAL VOC, необходимый для обнаружения, сегментации и классификации объектов. Узнайте о ключевых особенностях, областях применения и советах по использованию. – URL: https://docs.ultralytics.com/ru/datasets/detect/voc (дата обращения: 11.03.2025) – Текст: электронный.

11. YOLOv10 Ultralytics – Откройте для себя YOLOv10, новейшее решение для обнаружения объектов в реальном времени, устраняющее NMS и повышающее эффективность. Достигайте высочайшей производительности при низких вычислительных затратах. – URL: https://docs.ultralytics.com/ru/models/yolov10 (дата обращения: 02.03.2025) – Текст: электронный.

12. Легко ли распознать информацию на банковской карточке? – Когда мы общаемся с нашими заказчиками, то, будучи специалистами в этой области, активно используем соответствующую терминологию, в частности слово «распознавание». При этом слушающая аудитория,... – URL: https://habr.com/ru/companies/smartengines/articles/272607/ (дата обращения: 23.02.2025) – Текст: электронный.

13. Модели Keras.

14. Распознавание дебетовых и кредитных карт | Google Pay API для Android. – URL: https://developers.google.com/pay/payment-card-recognition/debit-credit-card-recognition?hl=ru (дата обращения: 15.03.2025) – Текст: электронный.

15. Evaluating Object Detection Models Using Mean Average Precision – In this article we will see see how precision and recall are used to calculate the Mean Average Precision (mAP). – URL: https://www.kdnuggets.com/evaluating-object-detection-models-using-mean-average-precision (дата обращения: 22.02.2025) – Текст: электронный.

16. Optical Character Recognition (OCR): Definition & How To Guide – Optical character recognition (OCR) is the process of extracting text from images. Learn about the different types of OCR and how to do it efficiently. – URL: https://www.v7labs.com/blog/ocr-guide (дата обращения: 15.03.2025) – Текст: электронный.

17. Smart Engines – научная IT-компания в области искусственного интеллекта – Компания Smart Engines основана учеными в области распознавания и обработки изображений. Наша специализация – технологии ИИ, оптическое распознавание и решения на базе компьютерного зрения для бизнеса. – URL: https://smartengines.ru/about/ (дата обращения: 26.02.2025) – Текст: электронный.

18. Version – 764 open source num-holder-exprie\_date images plus a pre-trained creidt-cards model and API. Created by creditcards. – URL: https://universe.roboflow.com/creditcards/creidt-cards (дата обращения: 17.03.2025) – Текст: электронный.

19. Шешкус, А. В. Библиотека для распознавания банковских карт «Smart CardReader».

20. AKARSU, M. mftnakrsu/Comparison-of-OCR / M. AKARSU. – 2025.

21. Alif, M. A. R. YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain / M.A.R. Alif, M. Hussain – Текст : непосредственный. 2024. YOLOv1 to YOLOv10.

22. Anagnostopoulos, C. N. E. A License Plate-Recognition Algorithm for Intelligent Transportation System Applications // IEEE Transactions on Intelligent Transportation Systems. 2006. Т. 7. № 3. – С. 377-392.

23. Arulprakash, E. A study on generic object detection with emphasis on future research directions / E. Arulprakash, M. Aruldoss – Текст : непосредственный. // Journal of King Saud University - Computer and Information Sciences. 2022. Т. 34. № 9. – С. 7347-7365.

24. Baek, Y. Character Region Awareness for Text Detection // 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019. – С. 9357-9366.

25. Christopher D. Manning. Introduction to Information Retrieval / Christopher D. Manning, Hinrich Schütze, Prabhakar Raghavan Cambridge University Press, 2008.

26. Deb, K. Automatic Vehicle Identification by Plate Recognition for Intelligent Transportation System Applications // Modern Approaches in Applied Intelligence : Lecture Notes in Computer Science / ред. K.G. Mehrotra [и др.]. – Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. Т. 6704. – С. 163-172.

27. Devisurya, V. Enhanced Number Plate Recognition for Restricted Area Access Control Using Deep Learning Models and EasyOCR Integration // 2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT) 2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT). – Vellore, India: IEEE, 2024. – С. 1-6.

28. Dosovitskiy, A. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2021. An Image is Worth 16x16 Words.

29. Hussain, M. YOLOv1 to v8: Unveiling Each Variant–A Comprehensive Review of YOLO / M. Hussain – Текст : непосредственный. // IEEE Access. 2024. Т. 12. YOLOv1 to v8. – С. 42816-42833.

30. Jiao, L. A Survey of Deep Learning-Based Object Detection // IEEE Access. 2019. Т. 7. – С. 128837-128868.

31. Kaur, R. A comprehensive review of object detection with deep learning / R. Kaur, S. Singh – Текст : непосредственный. // Digital Signal Processing. 2023. Т. 132. – С. 103812.

32. Kaur, R. A comprehensive review of object detection with deep learning / R. Kaur, S. Singh – Текст : непосредственный. // Digital Signal Processing. 2023. Т. 132. – С. 103812.

33. Li, M. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. 2022. TrOCR.

34. Lin, T.-Y. Microsoft COCO: Common Objects in Context. 2015. Microsoft COCO.

35. Redmon, J. You Only Look Once: Unified, Real-Time Object Detection // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. You Only Look Once. – С. 779-788.

36. Sermanet, P. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. 2014. OverFeat.

37. Shi, B. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition / B. Shi, X. Bai, C. Yao – Текст : непосредственный. // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017. Т. 39. № 11. – С. 2298-2304.

38. Smith, R. An Overview of the Tesseract OCR Engine / R. Smith // Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Ninth International Conference on Document Analysis and Recognition (ICDAR 2007). 2007. Т. 2. – С. 629-633.

39. Szegedy, C. Scalable, High-Quality Object Detection. 2015.

40. Vedhaviyassh, D. R. Comparative Analysis of EasyOCR and TesseractOCR for Automatic License Plate Recognition using Deep Learning Algorithm // 2022 6th International Conference on Electronics, Communication and Aerospace Technology 2022 6th International Conference on Electronics, Communication and Aerospace Technology. 2022. – С. 966-971.

41. Zou, Z. Object Detection in 20 Years: A Survey // Proceedings of the IEEE. 2023. Т. 111. Object Detection in 20 Years. № 3. – С. 257-276.

42. Глубокое обучение на Python. 2-е межд. издание. – URL: https://www.piter.com/product/glubokoe-obuchenie-na-python-2-e-mezhd-izdanie (дата обращения: 08.03.2025) – Текст: электронный.

43. МКК Честное слово — срочно деньги в долг на карту онлайн на сайте 4slovo.ru. – URL: https://4slovo.ru/about (дата обращения: 26.02.2025) – Текст: электронный.

44. Пересечение через Союз (IoU). – URL: https://www.ultralytics.com/ru/glossary/intersection-over-union-iou (дата обращения: 12.03.2025) – Текст: электронный.

45. Система управления Online МФО.Распознавание карт.

46. Указ Президента Российской Федерации от 30.03.2022 № 166 ∙ Официальное опубликование правовых актов. – URL: http://publication.pravo.gov.ru/Document/View/0001202203300001?index=1 (дата обращения: 15.03.2025) – Текст: электронный.

47. A review of object detection based on deep learning | Multimedia Tools and Applications. – URL: https://link.springer.com/article/10.1007/s11042-020-08976-6?utm\_source=getftr&utm\_medium=getftr&utm\_campaign=getftr\_pilot&getft\_integrator=sciencedirect\_contenthosting (дата обращения: 12.03.2025) – Текст: электронный.

48. COCO - Common Objects in Context. – URL: https://cocodataset.org/#detection-eval (дата обращения: 03.05.2025) – Текст: электронный.

49. Deep Learning for Generic Object Detection: A Survey | International Journal of Computer Vision. – URL: https://link.springer.com/article/10.1007/s11263-019-01247-4?utm\_source=getftr&utm\_medium=getftr&utm\_campaign=getftr\_pilot&getft\_integrator=sciencedirect\_contenthosting (дата обращения: 12.03.2025) – Текст: электронный.

50. Evaluating Deep Learning Models: The Confusion Matrix, Accuracy, Precision, and Recall - KDnuggets. – URL: https://www.kdnuggets.com/2021/02/evaluating-deep-learning-models-confusion-matrix-accuracy-precision-recall.html (дата обращения: 12.03.2025) – Текст: электронный.

51. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. – URL: https://www.computer.org/csdl/journal/tp/2017/06/07485869/13rRUx0gera (дата обращения: 25.02.2025) – Текст: электронный.

52. ImageNet Classification with Deep Convolutional Neural Networks. – URL: https://www.researchgate.net/publication/267960550\_ImageNet\_Classification\_with\_Deep\_Convolutional\_Neural\_Networks (дата обращения: 10.03.2025) – Текст: электронный.

53. JaidedAI/EasyOCR. – Jaided AI, 2025.

54. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories | Request PDF. – URL: https://www.researchgate.net/publication/222817667\_Learning\_Generative\_Visual\_Models\_from\_Few\_Training\_Examples\_An\_Incremental\_Bayesian\_Approach\_Tested\_on\_101\_Object\_Categories (дата обращения: 10.03.2025) – Текст: электронный.

55. (PDF) Caltech-256 Object Category Dataset. – URL: https://www.researchgate.net/publication/30766223\_Caltech-256\_Object\_Category\_Dataset (дата обращения: 10.03.2025) – Текст: электронный.

56. (PDF) ImageNet: a Large-Scale Hierarchical Image Database. – URL: https://www.researchgate.net/publication/221361415\_ImageNet\_a\_Large-Scale\_Hierarchical\_Image\_Database (дата обращения: 10.03.2025) – Текст: электронный.

57. SSD: Single Shot MultiBox Detector | SpringerLink. – URL: https://link.springer.com/chapter/10.1007/978-3-319-46448-0\_2 (дата обращения: 08.03.2025) – Текст: электронный.

58. ultralytics/JSON2YOLO. – Ultralytics, 2025.