ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о командном программном проекте на тему:

Сервис описания изображений для незрячих людей

Выполнили студенты:

группы БПМИ213, 3 курса	Рябков Игорь Дмитриевич
группы БПМИ213, 3 курса	Курдун Мария Андреевна
группы БПМИ213, 3 курса	Стамбеков Алмасбек Азатбекович
группы БПМИ211, 3 курса	Коротков Антон Сергеевич

Принял руководитель проекта:

Рогачев Александр Игоревич
Штатный преподаватель
Факультет компьютерных наук НИУ ВШЭ

Москва 2024

Содержание

Аннотация

В ходе данной курсовой работы был реализован сервис описания изображений для слабовидящих людей. Данный сервис предназначен для описания произвольного визуального контента, который может встретиться в повседневной жизни, в частности, это могут быть фотографии, рисунки, графические интерфейсы сайтов и мобильных приложений. Для генерации качественных описаний мы используем глубокие нейронные сети (DNNs) из областей компьютерного зрения (CV) и обработки естественного языка (NLP) и решаем такие вспомогательные задачи, как детекция элементов интерфейса, детекция и распознавание текста, определение возраста человека по фотографии и генерация описания сцен, представленных на изображении. Все результаты работы задействованных в сервисе моделей машинного обучения объединяются в текстовое описание и конвертируются в аудиозапись, предназначенную для пользователя. Данный проект способствует продвижению принципов цифровой инклюзивности, делая технологии доступнее для слабовидящих людей.

Ключевые слова

Машинное обучение, нейронные сети, компьютерное зрение, обработка естественного языка, распознавание текста, детекция объектов, описание изображений

1 Введение

Визуальная информация имеет определяющее значение во всех жизненных сферах. При этом достаточно существенная часть населения сталкивается со сложностями при потреблении визуального контента из-за проблем со зрением. Из-за этого таким людям может быть сложно адаптироваться к современной жизни. В связи с этим мы разработали сервис, способный генерировать по визуальной информации аудио-описания, которые бы позволили слабовидящим людям лучше взаимодействовать с медиаконтентом.

Для повышения практической ценности сервиса проводились консультации со слабовидящем экспертом. Его обратная связь была достаточно важна, так как трудно оценить, что будет важно для слабовидящих пользователей.

Сервис позволяет детектировать и классифицировать элементы интерфейса, описывать изображения, уточнять их тип (анимационное/реалистичное), определять возраст и пол людей, а также детектировать и распознавать текст.

1.1 Распределение задач

Модульность архитектуры сервиса позволила разделить обязанности следующим образом:

- Антон Коротков занимался первым этапом в работе сервиса: его работа заключалась в подготовке и развертывании модели, способной локализовывать и классифицировать элементы интерфейса на изображении.
- Мария Курдун занималась подготовкой и развертыванием модели детекции и распознавания текста. Мария также реализовала генерацию мультиязыковой озвучки текстовых сообщений.
- Игорь Рябков работал над задачей описания изображений с помощью нейронных сетей. Создал классификатор для разделения анимационных и реалистичных картинок.
- Алмасбек Стамбеков занимался подготовкой и развертыванием модели для определения возраста и пола людей по изображению.

Все участники команды также работали над созданием мастер-сервиса - "обертки объединяющего работу всех моделей, и над телеграм-ботом.

2 Анализ существующих решений

Перед началом разработки данного проекта был проведён анализ рынка с целью выяснения, существуют ли аналогичные решения поставленной задачи. В результате было установлено, что полноценных онлайн-сервисов для генерации описаний к изображениям с последующим их озвучиванием на данный момент мало. К тому же, консультант команды, который регулярно пользуется подобными сервисами, отметил, что большинство существующих решений, таких как VisionBot[28], остались на стадии экспериментальной разработки и давно не обновлялись.

Одним из очевидных недостатков VisionBot является его ограниченный функционал. Данный сервис не предоставляет полноценное описание изображений. Вместо этого, он ограничивается детекцией текста и людей на изображении, при этом анализируя их возраст и пол, и предоставляет эту информацию пользователю.

Кроме онлайн-сервисов, полностью направленных на решение рассматриваемой задачи, существуют решения на базе ОС или приложений. В частности, функционал, необходимый для решения описанной задачи, встречается у некоторых виртуальных ассистентов.

Анализ показал, что в различных приложениях подобные инклюзивные функции для слабовидящих людей встречаются достаточно редко. В частности, командой на наличие таких функций были исследованы популярные социальные сети, мессенджеры и браузеры, как приложения, генерирующие наибольший медиатрафик. Нами рассматривались приложения, наибольшим образом использующиеся в RU-сегменте - VK (соцсеть), Telegram (мессенджер), так и приложения, которые пользуются популярностью во всем мире, - WhatsApp (мессенджер), Google Chrome (браузер), Microsoft Edge (браузер). Как оказалось, во всех рассмотренных приложениях, за исключением браузера Microsoft Edge, нет встроенных функций, направленных на решение описанной задачи. В браузере Microsoft Edge представлено определённое решение задачи - функция “Прочитать эту страницу вслух”, но оно справляется с задачей лишь частично. В частности, данная функция ищет и проговаривает только текст, игнорируя изображения и виджеты, причём только тот текст, который помещён в соответствующие HTML-блоки.

На базе ОС существуют более эффективные и полные решения. Так, компания Apple реализовала в операционной системе IOS выпускаемых ей смартфонов IPhone и планшетов IPad функцию VoiceOver для помощи слабовидящим людям. VoiceOver не только способен читать вслух текст на web-страницах и в приложениях, но также умеет предварительно генерировать описания к изображениям, кнопкам и иконкам. Эта функция очень хорошо

умеет описывает объекты, которые могут появиться на экране мобильного устройства, причём детализацию описаний можно настраивать, что является несомненным преимуществом. Более того, VoiceOver можно очень гибко настроить: можно менять жесты для управления этим виртуальным ассистентом, голоса, которыми произносится текст и т.д. Также VoiceOver может воспринимать информацию на разных языках, а не только на каком-то одном фиксированном.

При всех сильных сторонах VoiceOver, у данной функции есть несколько ограничений:

- можно пользоваться только владельцам техники компании Apple
- ограниченность памяти, выделяемой на ML-модели, использующиеся в VoiceOver
- достаточно низкая скорость обработки изображений (после обработки изображения и выдачи описания к нему VoiceOver показывал большие задержки, вплоть до 30 секунд. Это очень долго)
- закрытая проприетарная платформа, пока не дающая возможность для дальнейшего развития функции

В данном проекте были учтены недостатки вышеупомянутых решений.

3 Обзор литературы

3.1 Детекция элементов интерфейса

3.1.1 Задача

Первым элементом задачи, которую решает сервис, является разбиение изображения на блоки интерфейса и классификация блоков по типу элемента интерфейса. Обе задачи — локализация объектов и их классификация — могут решаться одновременно. Такую объединённую задачу называют детекцией объектов. Очевидно, что в отличие от задачи классификации на подаваемом в модель изображении может быть больше одного объекта.

3.1.2 Наборы данных

Для классической постановки задачи - когда мы хотим распознавать какие-то объекты на фотографиях повседневной жизни - существует множество датасетов. Наиболее популярным в такой классической задаче детекции является набор данных Microsoft COCO Dataset Его зачастую используют как бенчмарк для данной задачи, а также на нём предобучают большинство современных нейронных сетей-детекторов.

Для задачи распознавания UI-элементов существует гораздо меньше подходящих наборов данных. Большинство из них либо содержат относительно малое количество объектов-изображений (около 1000 и меньше), что не позволяет использовать их для дообучения хороших с точки зрения метрик качества моделей, либо обладают разметкой низкого качества (см. следующий раздел, более детальным описывающий эти проблемы и их решение). Мы подобрали 3 набора данных, более или менее подходящих для качественного решения задачи детекции UI-элементов:

- WebUI [32]
- mrtoy/mobile-ui-design[20]
- VINS Dataset[2]

3.1.3 Метрики качества

В качестве основной метрики качества в задаче детекции используется mAP (mean average precision), то есть средний по всем непосредственно представленным в задаче классам

Average Precision:

$$\text{mAP} = \frac{1}{C}\sum_{i=1}^{C}\text{AP}_i \quad (C - \text{число классов}, \text{AP}_i - \text{average precision для класса i})$$

В принципе эта метрика качества достаточно интуитивна, переносит идеи из многоклассовой классификации в данную задачу: хотим определить в среднем, насколько модель точно предсказывает тот или иной класс - насколько мало она делает ошибок при предсказании класса.

При этом в силу особенности задачи - так как требуется не только классифицировать объекты, но и локализовывать их - в метрику вносятся изменения. Ниже представлен **метод вычисления mAP для детекции**:

- Рассмотрим некоторый класс i.
- Как известно, AP = AUC-PR (AUC = "Area under Curve PR = "Precision-Recall Curve"), следовательно, будем считать AP через AUC-PR для рассматриваемого на данный момент класса i.
- Для того, чтобы найти AUC-PR, надо отсортировать все предсказания модели по её “уверенности” (confidence) в результатах, чтобы затем варьировать пороги уверенности и считать в зависимости от них precision и recall для Precision-Recall Curve. Отсортируем предсказания по уверенности модели по убыванию.
- Нужно определиться с тем, что будет считаться True Positive (далее - TP - это такой объект, что предсказание модели для него совпало с истинным для него классом), а что - False Positive (далее - FP - объект, который модель отнесла к данному классу, но который на самом деле к нему не принадлежит) (True Negative (TN) и False Negative (FN) здесь не будет). Будем считать их по метрике **IoU** (Intersection over Union):

  $$\textbf{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

  Если IoU больше некоторого порога, то объект - TP, если IoU меньше порога, то объект - FP.
- Также стоит определить, как при детекции будут вычисляться precision и recall. Как и

в задаче классификации, здесь

$$\text{precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}$$

. С recall - немного сложнее, так как не определено, что такое здесь FN (в классификации recall $= \frac{\text{TP}}{(\text{TP+FN})}$). Несложно установить, что $\text{TP} + \text{FN} = L_i$, где L_i - кол-во объектов данного класса i в выборке, которое известно заранее, следовательно, считаем для каждого объекта recall как

$$\text{recall} = \frac{\text{TP}}{L_i}$$

.

- Далее строится precision-recall curve, считается AUC-PR.
- Так AP считается для каждого класса, а затем усредняется по всем классам.

У mAP, зависящей от некоторого фиксированного порога IoU, есть естественная модификация. Можно смотреть mAP сразу по нескольким порогам по IoU, а потом рассмотреть среднее результатов - тем самым получаем ещё более взвешенную оценку качества модели. Так, существует mAP@[0.5; 0.95] - то есть рассматриваются все пороги по IoU от 0.5 до 0.95 включительно с шагом 0.05.

3.1.4 Подходы для решения задачи

Существует два основных подхода к решению задачи детекции:

- **Двухэтапный подход:** сначала объекты локализуются, затем для каждого из локализованных объектов решается задача классификации. Данный подход, с одной стороны, интуитивно понятен, поэтому такие модели легче реализовывать, с другой стороны, такой подход может быть очень времязатратный.
- **Одноэтапный подход:** объекты одновременно локализуются и классифицируются. Данный подход может быть гораздо быстрее по сравнению с предыдущим.

Так как для конечного сервиса важно не только его общее качество, но и скорость его работы, далее будут рассмотрены только одноэтапные детекторы.

Среди множества детекторов на разных этапах работы над проектом рассматривались представители двух семейств моделей: **YOLO** и **Detr** (Detection Transformer). В частности,

рассматривались модели YOLOv5 [22], YOLOv8, YOLOv9 [29], Detr [3], Co-Detr [39]. Далее представлен обзор указанных выше детекторов.

YOLO (общая идея):

- Свёрточная нейронная сеть
- Подаваемое в модель изображение разбивается сеткой (обычно квадратной) некоторого размера
- Для каждой ячейки сетки предсказывается **B** bounding box'ов (ячейки, в которых потенциально должны быть локализованы объекты), где B - гиперпараметр, и то, насколько модель в данных bounding box'ах уверена. Под уверенностью модели понимается confidence score, показывающий, есть ли в данной ячейке некоторый объект или нет, а также - то, насколько хорошо определённый bounding box совпадает с ground-truth границами объекта. Таким образом, уверенность модели вычисляется следующим образом:

$$\text{confidence} = P(\text{obj}) \cdot \textbf{IoU}(\text{true}, \text{pred})$$

P(obj) - вероятность наличия в bounding box'е объекта, за который отвечает данная ячейка сетки, **IoU**(true, pred) - intersection-over-union, подсчитанный для найденного bounding box'а и для bounding box'а из разметки.

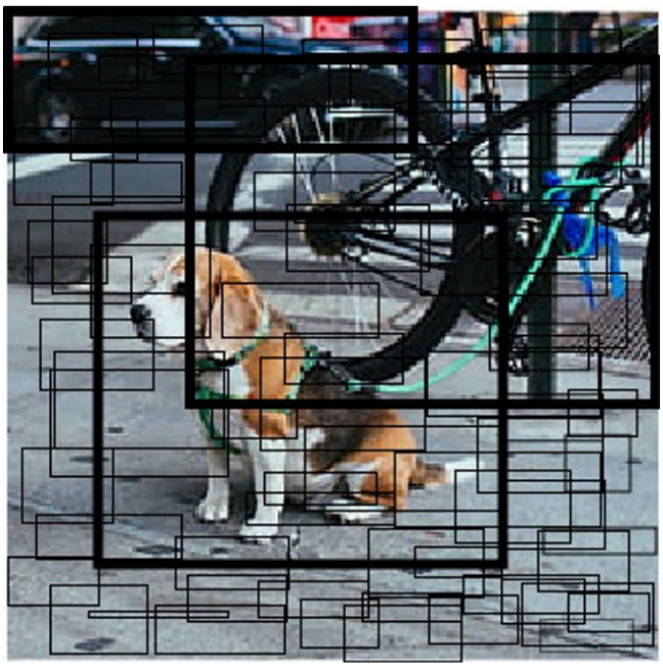

Рис. 3.1: Пример построения bounding box'ов для ячеек изображения

- Также для каждой ячейки предсказываются условные вероятности того, что представленный на ней объект относится к тому или иному классу, это делается для всех возможных классов. Таким образом, считаются условные вероятности $P(\text{class}_i \mid \text{obj}) \quad \forall i$.

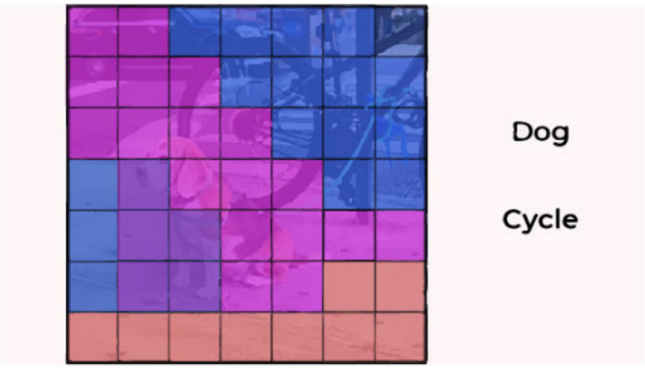

Рис. 3.2: На данном изображении показано распределение классов по ячейкам. Ячейке присвоен тот класс, условная вероятность для которого оказалась наибольшей.

- Найденные результаты объединяют:

$$\text{confidence} \cdot P(\text{class}_i \mid \text{obj}) = P(\text{obj}) \cdot \textbf{IoU}(\text{true}, \text{pred}) \cdot P(\text{class}_i \mid \text{obj}) =$$

$$= P(\text{class}_i) \cdot \textbf{IoU}(\text{true}, \text{pred}) = \{\text{class conf}\}$$

  - найденное значение показывает, насколько хорошо некоторый bounding box локализует объект, а также - то, правильный ли в нём указан класс или нет.

- В конце применяется алгоритм Non-Maximum Suppression (NMS) - для того, чтобы выбрать среди похожих bounding box'ов обладающие наибольшим значением {class conf}, и избавиться от всех остальных bounding box'ов.

- В дальнейших версиях YOLO была предложена концепция якорных рамок (anchor boxes) - это некоторые заранее заданные bounding box'ы, которые были предподсчитаны по набору данных и которые имеют определенные соотношения высоты и ширины. При детекции модель находит не сами bounding box'ы (их размеры), а ищет необходимое сжатие или растяжение наиболее близкого к данному объекту anchor box'а. Такой подход, например, помогает распознавать сразу несколько объектов в одной ячейке.

- Общая схема детекции с помощью YOLO описана ниже:

YOLOv5 - версия YOLO, основанная на YOLOv4, обладающая большей скоростью инференса (главное преимущество новой модели) и лучшим качеством (mAP) по сравне-

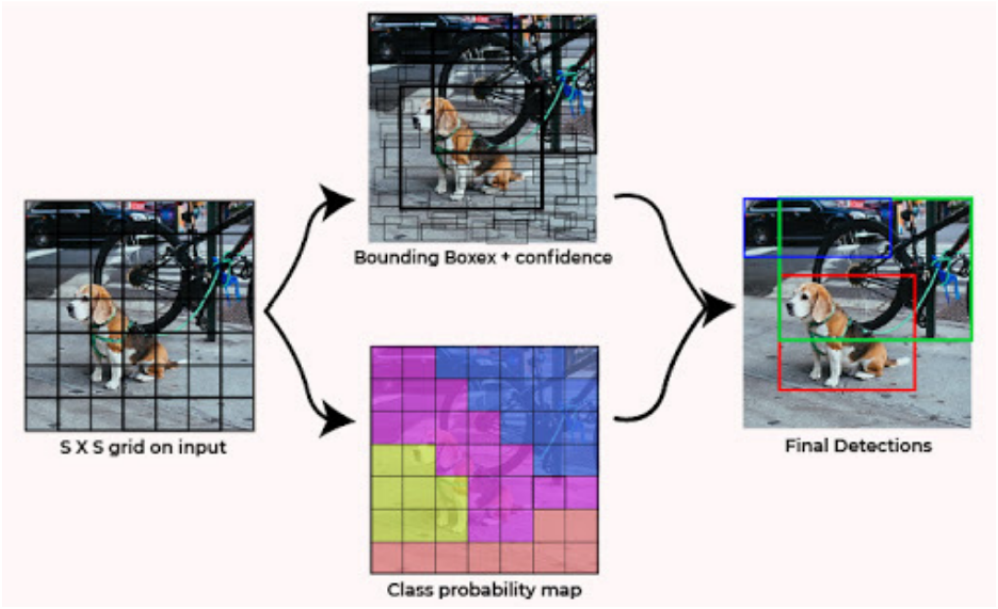

Рис. 3.3: Общая схема детекции изображения с помощью YOLO

нию с предшественниками. Также данная модель более удобна в обращении по сравнению с предыдущими в связи с переходом авторов с фреймворка Darknet на PyTorch. Модель отличается от предшествующих обновленным алгоритмом поиска якорных рамок AutoAnchor (далее называем якорные рамки anchor boxes) - теперь после применения алгоритма KMeans для поиска anchor boxes происходит корректировка плохо подходящих под данные anchor boxes с помощью генетического алгоритма.

У модели - новый backbone (основа) - CSPNet - модифицированный CSPDarknet53, начинающийся со Stem'a - свёрточного слоя с большим размером ядра свёртки, что позволяет уменьшить время вычислений и затрачиваемую на них память.

Ещё одним нововведением данной итерации YOLO стал пирамидальный Pooling-слой SPPF (spatial pyramid pooling fast), дающий некоторый выигрыш в скорости при использовании модели.

YOLOv8 - версия YOLO, выпущенная в 2023 году, основанная на YOLOv5 (выпущена той же компанией) и использующая нововведения других версий, что позволяет повысить скорость инференса и добиться лучшего качества (mAP) по сравнению с предшественниками (с той же YOLOv5).

Авторы модели, как и авторы других моделей-детекторов этого времени, отказались от использования anchor boxes, так как оказалось, что без них всё же можно добиваться лучшего качества.

Также авторы разделили голову (head) YOLOv5 на несколько голов, так, что каждая голова теперь концентрируется только на одной подзадаче: регрессия (для локализации объ-

ектов), objectness (наличие объекта в bounding box), классификация - это также позволило несколько улучшить качество.

Наконец, авторы модели обновили backbone модели, что также привело к повышению качества модели, несколько изменили композитный лосс, на который обучается модель, а также реализовали архитектуры YOLOv8 и для других задач, помимо детекции, - например, для сегментации.

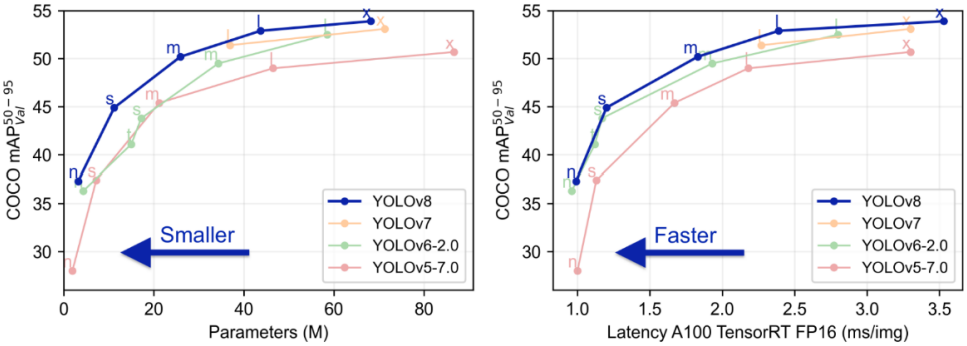

Рис. 3.4: Сравнение YOLOv8 с предшествующими моделями

YOLOv9 - версия YOLO, выпущенная в 2024 году, уже во время того, как мы работали над проектом. Данная версия основана на YOLOv7.

Авторы данной версии рассматривают проблему Information Bottleneck: при передаче информации между слоями нейросети часть информации о её входе при обычном подходе обязательно теряется. В связи с этим авторы обращаются к концепции обратимых функций - таких функций, что для них существует обратная. Для таких функций при их применении к аргументу вся информация об аргументе сохраняется. В связи с этим, а также в связи с тем, что слой нейросети - некоторая функция, авторы предлагают фреймворк PGI (Programmable Gradient Information), использующий идею с обратимыми функциями и позволяющий помимо информации с потерями передавать модели информацию о входе нейросети через слои без потерь.

Также авторы используют другой backbone модели - GELAN, гораздо более “легкий” с точки зрения параметров и в целом более гибкий.

Эти два нововведения позволяют добиться ещё лучшего качества модели при меньшем количестве параметров.

Detr (оригинальная архитектура, общая идея):

- Detr - модель, состоящая из свёрточного backbone’a и следующего за ним encoder-decoder трансформера, переделанного под object detection.

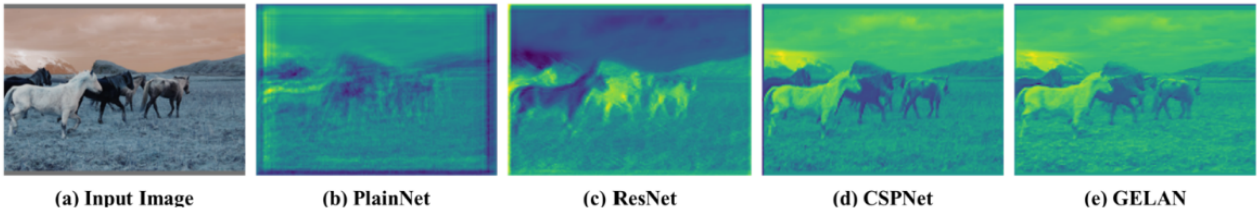

Рис. 3.5: Визуализация карт признаков (feature maps) для разных backbone'ов

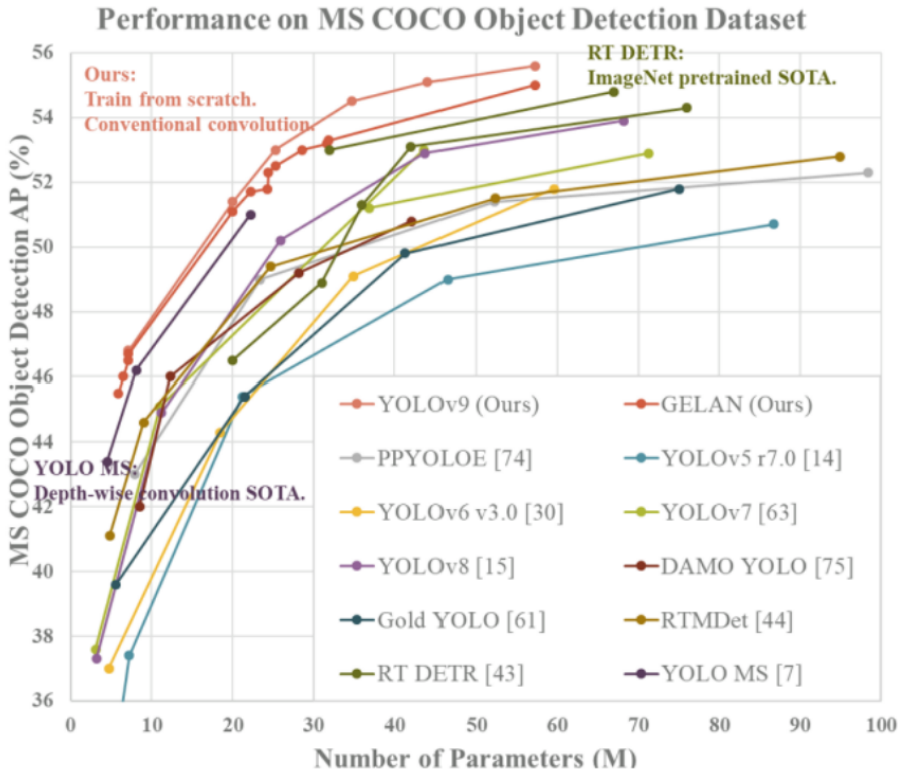

Рис. 3.6: Сравнение YOLOv9 с предшествующими моделями

- Свёрточный backbone выучивает полное 2D-представление поданного на вход модели изображения, затем полученный feature map растягивают в вектор и передают в трансформер. Выход трансформера передаётся двум головам (heads): одна - полносвязная сеть - отвечает за предсказание bounding box'ов, другая - просто линейный слой - за классификацию объектов. Важно отметить, что в классификации участвует объект NULL, обозначающий отсутствие объекта в bounding box'е - это позволяет избежать полноценного шага с проверкой bounding box'ов на наличие в них объектов.

- Модель не использует ни anchor boxes, ни NMS, что позволяет не тратить лишнее время на эвристики.

Co-Detr - SOTA-детектор на бенчмарке COCO на данный момент. Наилучшие версии Co-Detr используют в качестве backbone'ов Swin Transformer и Vision Transformer (ViT,

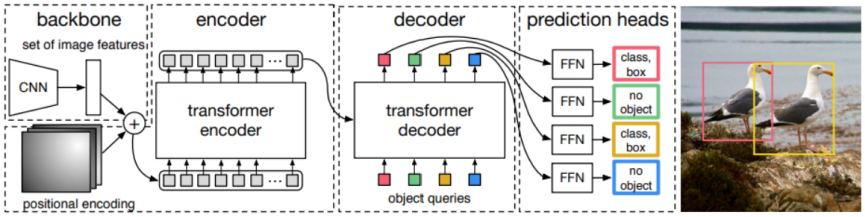

Рис. 3.7: Архитектура Detr

SOTA-результат достигается именно с использованием данного backbone'а). Основное нововведение Co-Detr - схема обучения с параллельными дополнительными головами (heads), прикрепленными к трансформеру, которые позволяют достичь лучших результатов при обучении модели. При инференсе эти головы отбрасывают. Таким образом, параметров у модели формально столько, сколько было без голов.

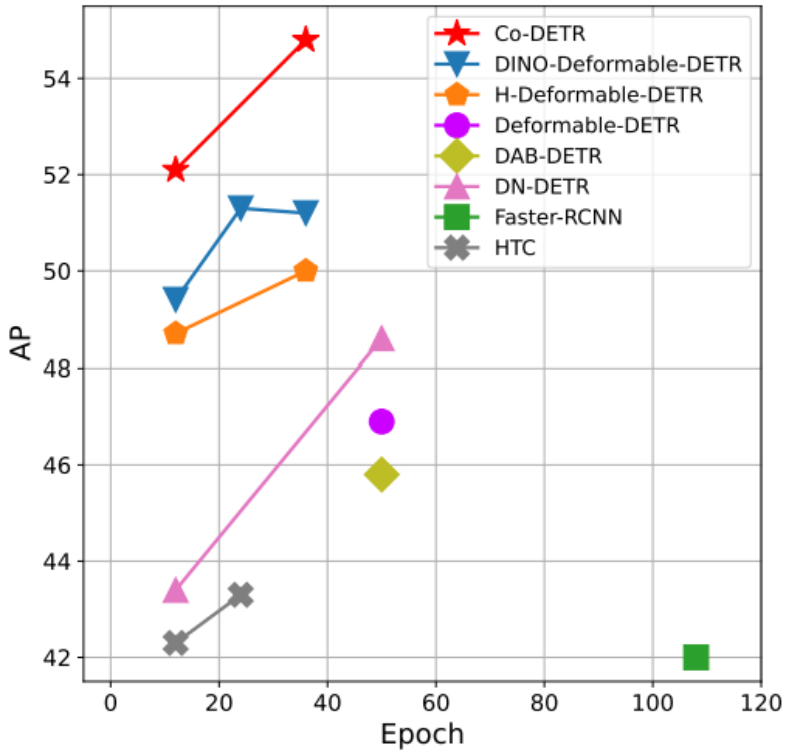

Рис. 3.8: Сравнение Co-Detr с предшествующими моделями

3.2 Описание изображений

После детекции разных блоков интерфейса, необходимо решить задачу описания изображений. Для решения данной задачи, было принято решение взять модели SOTA архитектуры трансформера. Для знакомства с этой архитектурой была изучена статья "Attention is all you need"[26], в которой данная архитектура была представлена в рамках задачи машинного

перевода. Его архитектура представлена на Рисунке 3.9, текст генерируется по следующему принципу:

- Берутся эмбеддинги (векторные представления) входных данных, например, текстов или картинок
- К данным эмбеддингам добавляется позиционная информация (Positional encoding):

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{dim}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{dim}})$$

  Данные значения просто прибавляются поэлементно к первоначальным эмбеддингам и кодируют информацию о взаимном расположении токенов в последовательности

- Данные проходят через несколько блоков энекодера, основная часть которого - multi-head attention блоки. Они предназначены, чтобы разные токены исходной последовательности влияли друг ну друга, изменяя их векторные представления. Они высчитываются следующим образом:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \ldots, head_h) * W^O$$

$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{where } Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_{dim}}})V$$

  На формуле выше Q, K, V - входные ключи(keys), запросы(queries) и значения(values) соответственно, $W_i^Q, W_i^K, W_i^V, W_i^O$ - веса

- После обработки энкодерами, последовательности передаются в декодеры. Каждый блок декодера содержит, в дополнение к multi-head attention, masked multi-head attention. Данный слой также предназначен, чтобы разные токены исходной последовательности влияли друг ну друга, однако в отличио от первого, он не даёт поздним токенам влиять на предшествуюшие. Это необходимо, чтобы модель не могла смотреть в будущее во время обучения и опираться на токены, которые она должна предсказать. Считается он следующим образом:

$$MaskedAttention(Q, K, V) = Softmax(M + \frac{QK^T}{\sqrt{d_{dim}}})V$$

$$\text{where } M = \begin{bmatrix} 0 & -\infty & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 \end{bmatrix}$$

- За слоями attention в декодерах и энкодерах, перед передачей в следующий блок, следуют feed-forward сети, которые дополнительно обрабатывают информацию:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- После прохождения через все блоки декодера, выходная последовательность подается на линейный слой (полносвязный), который затем пропускается через softmax-слой для получения вероятностей следующего токена в последовательности.

- Завершая цикл, модель генерирует выходные токены один за другим, основываясь на предыдущих выходах и входных данных, что позволяет формировать целевое предложение или последовательность в целом.

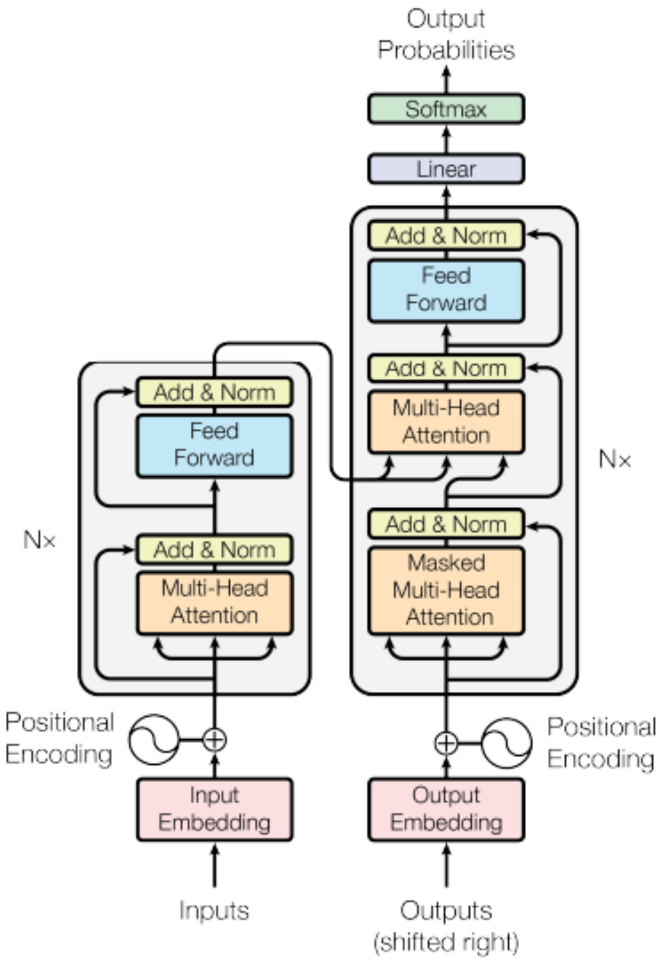

Рис. 3.9: Архитектура блока сети трансформер

3.2.1 Модели

Изучая существующие решения, для дальнейшего сравнения между собой были отобраны следующие модели (и их версии):

- Vit-GPT2 / Vit-RuGPT Данная модель состоит из двух других: VIT - модель обрабатывает разделённые на куски картинки (батчи) и преобразует в векторные представления. Потом они подаются на вход генеративной модели GPT-2 и преобразуются в предложение на английском/русском языке. (Рисунок 3.11)
  - VIT - классический энкодер трансформера, поверх которого написан многослойный перцептрон (несколько линейных слоёв). Изначально данную модель разрабатывали для решения задач классификации изображений.
  - GPT-2 - классический текстовый декодер трансформера
- GIT-Base / GIT-Large [30] Имеет схожую архитектуру, что и VIT-GPT, однако в качестве энкодера - энкодер изображений модели CLIP, а в качестве декодера - классический декодер трансформера со случайной инициализацией весов (Рисунок 3.10)
  - CLIP - модель, состоящая из двух кодировщиков: визуального (картинки) и текстового. CLIP обучается с использованием подхода контрастного обучения. В рамках этого подхода модель учится сопоставлять энкодированные представления изображений и их текстовые описания, опираясь на следующий принцип: текстовое описание должно быть ближе к соответствующему изображению и дальше от других изображений в пространстве эмбеддингов. Данное сопоставление позволяет кодировщику CLIP определять более информативные векторные представления изображений (обогащённые текстовой информацией)
- BLIP-Base / BLIP-Large [13] Архитектура BLIP (Bootstrapping Language-Image Pre-training) разработана для улучшения понимания и генерации связи между изображениями и языком. Эта модель включает в себя несколько ключевых компонентов:
  1. Image Encoder - классический энкодер трансформера, на вход которого приходит последовательность из эмбеддингов кусков итогового изображения.
  2. Text Encoder - классический энкодер трансформера, на вход которого приходит последовательность из эмбеддингов токенов текста.

3 Image-grounded Text Encoder - Текстовый энкодер, связывает между собой эмбеддинги текстов и изображений, объединяя информацию, это достигается за счёт нового слоя внимания: ”Cross-Attention”, который действует также, как и обычный, только для разных модальностей (текст/изображение)

4 Image-grounded Text Decoder - текстовый декодер с привязкой к изображению, используется для генерации текста, базирующегося на изображениях. Слой ”Causal Self-Attention” эквивалентен слою ”Masked Self-Attention”

Дополнительные компоненты:

– ITC loss (Image-Text Contrastive learning): Модуль, обучающий векторные представления текстов и картинок таким образом, чтобы они были близки только в том случае, если сгенерированное описание корректно характеризует картинку. (Идея модели CLIP)

– ITM loss (Image-Text Matching): Опирается на выход Image-grounded Text Encoder, который решает задачу бинарной классификации (1 - описание корректно, 0 - описание некорректно). Задача - найти наиболее информативные негативные примеры, чтобы на них более эффективно обучать модель

– LM loss (Language Modeling): представляет из себя CrossEntropyLoss, как и в обычном трансформере, для обучения модели максимизировать правдоподобие текста в авто регрессионной манере.

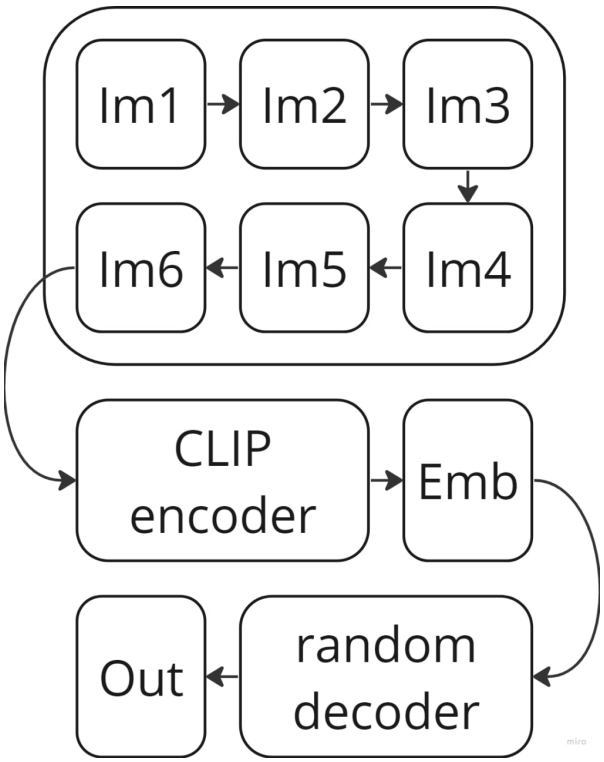

Рис. 3.10: Архитектура GIT

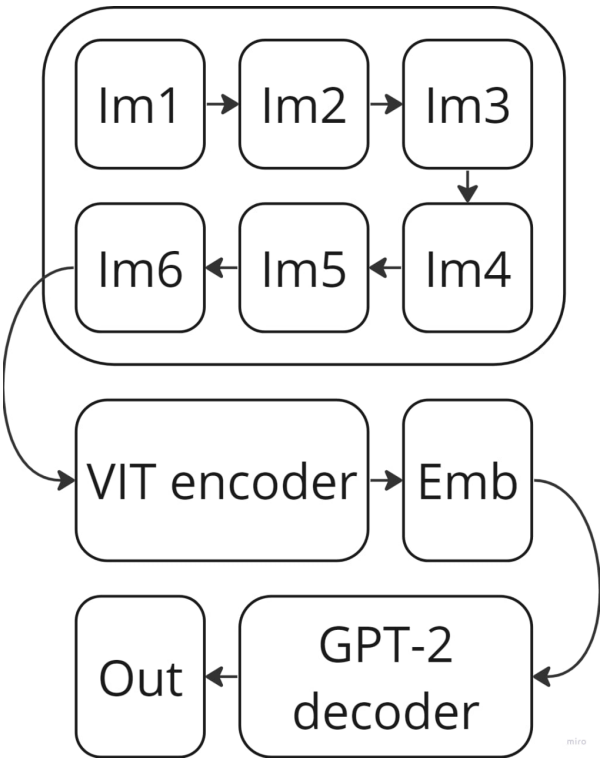

Рис. 3.11: Архитектура VIT-GPT

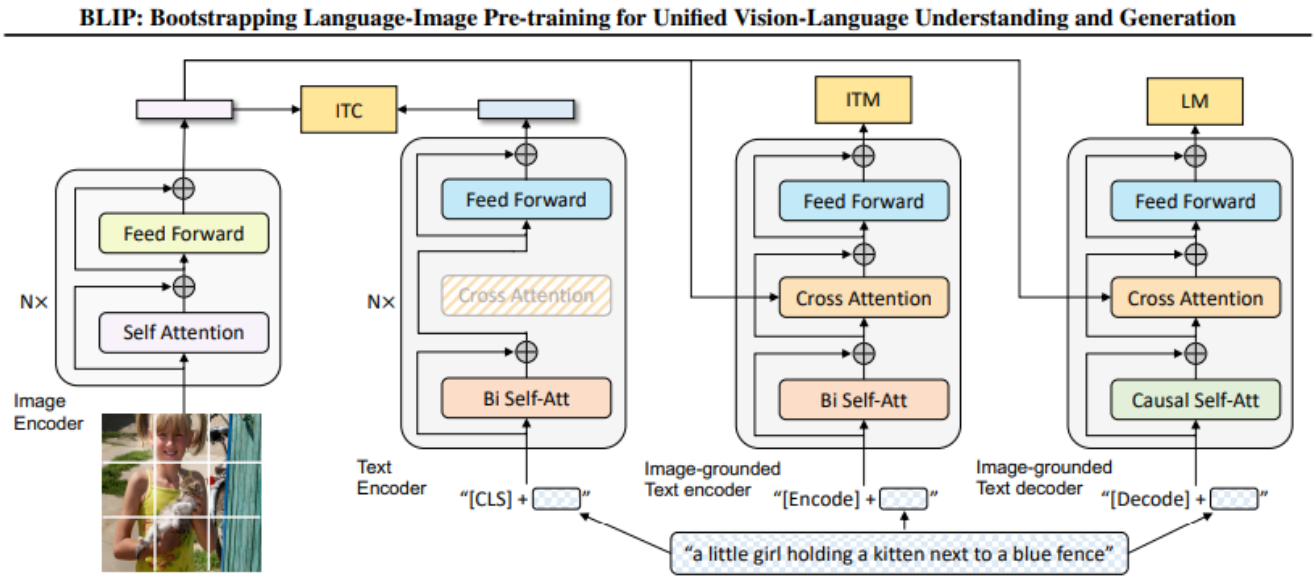

Рис. 3.12: Архитектура BLIP

3.2.2 Набор данных

При выборе набора данных, был поставлен акцент на его объём картинок и на их "разнообразие", а также кол-во примерных описаний. Более релевантным в нашем случае будет набор данных MS-COCO[17] (123,287 изображений, 5 описаний), который содержит большое кол-во картинок разного типа, кроме того, данный набор данных уже успел зарекомендо-

вать себя в задаче Image Captioning, так как его часто используют для обучения и проверки моделей.

Кроме того, чтобы покрыть домен картинок был взят датасет "Anime vs Cartoon"[4] (далее cartoons), Изначально он предназначался для обучения моделей отличать мультипликационные изображения от аниме, однако в контексте нашей задачи он предоставляет большое кол-во разнообразных картинок из различных источников (104 произведения)

В медиа чаще всего публикуются изображения людей: портреты, селфи, фото человека на фоне. На данном домене качество модели должно быть особенно хорошим. Для проверки был найден датасет "People Clothing Segmentation"[6] (далее clothes), он изначально был собран для сегментации элементов одежды, однако факт того, что в нём содержится большое кол-во людей в разной одежде идеально подходит для данной задачи

Был также задействован датасет "Portrait dataset for training GANs"[21] (далее portraits), его изображения имеют разное разрешение, стили, качество, структуру. Данный датасет поможет измерить отказоустоичивость модели

Рассматривались и другие наборы данных, например, Flickr8-30K[10] (8000/30000 изображений, 5 описаний) и SBU1M[7] (1000000 изображений, 1 описание), в которых собраны изрбражения с фотохостинга Flickr. Однако они содержат в себе слишком качественные изображения, с малым кол-вом шума. Наша задача подразумевает описывание изображений в медиа пространстве, где качество всегда ужимается в угоду производительности. Такие наборы данных показались нерелевантными, от них пришлось отказаться.

3.2.3 Метрики

Существует достаточно большое количество методов, каким образом можно измерить качество описаний [8]. Самые базовые — это использовать метрики наподобие BLEU. BLEU оценивает сгенерированный текст путем сравнения его с эталонными описаниями через совпадение n-грамм, где оценка варьируется от 0 до 100, указывая на точность сопоставления. Однако такие метрики в основном проверяют пары (предсказания, ответ) на грамматическое сходство, а не семантическое. Поэтому одинаковые по структуре, но разные по смыслу предложения будут рушить подобные метрики.

На данный момент наиболее актуальным решением будет использовать метрики, основанные на оценке восприятия моделями. Они используют для оценки представления обученных моделей, поэтому они лишены упомянутого ранее недостатка:

- BERTscore – это метод оценки качества машинного перевода с помощью языковой моде-

ли BERT. Этот метод сначала конвертирует тексты (исходный и переведенный) в векторные представления. Далее проводится попарное сравнение этих векторов для оценки семантического сходства между предложениями. Однако существует один критический недостаток - необходимость в разметке. Алгоритм вычисления данной метрики модно увидеть на Рисунке 3.13

- CLIPscore - является фаворитом, он обучен отображать картинки и описание в одно векторное пространство (Чем релевантнее описание, тем ближе его эмбеддинг будет к эмбеддингу картинки). хорошие стороны этой метрики заключаются в том, что она хорошо справляется с разными доменами данных, даже не имея для них разметки. Считается он следующим образом, сначала берутся эмбеддинги картинки и текста, а потом между ними смотрится косинусная близость, получившийся скаляр и есть значение метрики: $CLIPscore(I, C) = max(100 * cos(E_I, E_C), 0)$, где E_C, E_I - эмбеддинги текста и картинки соответственно

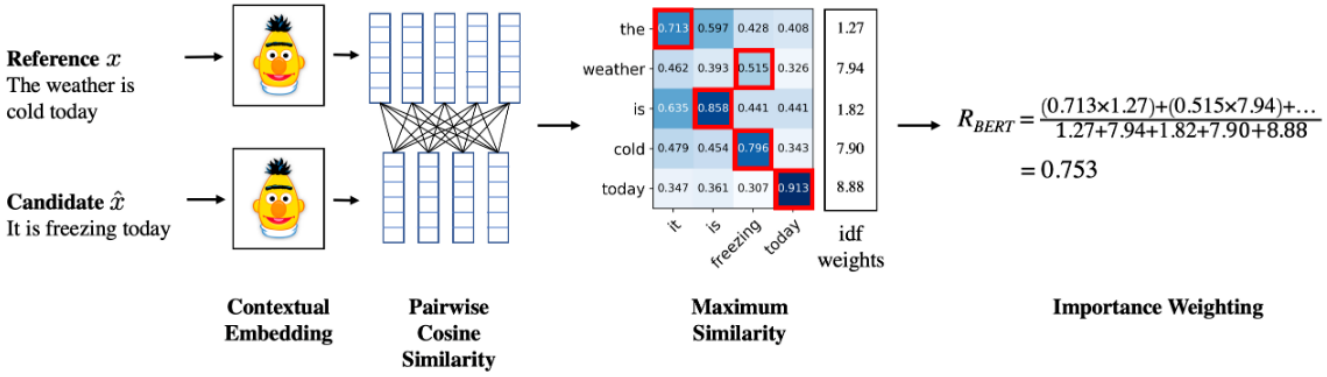

Рис. 3.13: Алгоритм вычисления метрики BERTscore

3.3 Определение возраста и пола человека по изображению

Задача данного раздела заключается в построении модели для распознавания возраста и пола человека и дальнейшем ее интегрировании. Данная модель состоит из двух частей: детекция объектов (таких, как лицо и туловище) и непосредственное определение возраста и пола человека по полученному изображению.

В целях систематического и эффективного подхода к решению данной задачи был разработан план действий:

1 Поиск и анализ наборов данных

- Исследование доступных наборов данных для задачи распознавания лиц и определения возраста и пола.
- Анализ качества и разнообразия данных в выбранных наборов данных.
- Выбор подходящих наборов данных на основе их релевантности задаче и качества.

2 Изучение литературы и существующих решений

- Чтение статей и исследований по теме распознавания лиц, определения возраста и пола.
- Анализ существующих алгоритмов и подходов, их преимуществ и недостатков.
- Изучение архитектур нейронных сетей, успешно применяемых в похожих задачах.

3 Предварительное сравнение базовых моделей

- Выбор базовых моделей на основе обзора литературы.
- Их построение с уже предобученными весами.
- Подсчет качества каждой модели на выбранном наборе данных.

4 Выбор оптимальных моделей

- Выбор итоговых моделей на основе результатов сравнения.
- Анализ ошибок и определение случаев, в которых модели работают недостаточно хорошо.

5 Улучшение качества выбранной модели и ее внедрение

- Дообучение модели.
- Подготовка функциональности модели для ее интеграции в сервис, включая разработку соответствующего API.

3.3.1 Наборы данных

Для задачи детекции возраста и пола на основе изображений широко используются несколько наборов данных, такие как IMDB-WIKI и UTKFace. Каждый из этих наборов имеет свои особенности, что делает их подходящими для различных исследований и приложений в данном разделе.

IMDB-WIKI[9] является одним из крупнейших доступных наборов данных для анализа лиц, содержащий более 500 тысяч изображений. Эти изображения собраны из интернет-базы данных фильмов (IMDB) и Википедии, включая метаданные с возрастом и полом, а также положением частей тела. Его преимущество заключается в большом количестве данных и разнообразие лиц, возрастных групп и этнических принадлежностей, что является идеальным вариантом для обучения и тестирования моделей для определения возраста и пола. Наглядное распределение возрастов можно наблюдать на Рисунке 3.14. По большей части набор состоит из снимков взятых из кино, телешоу и разного рода инагураций.

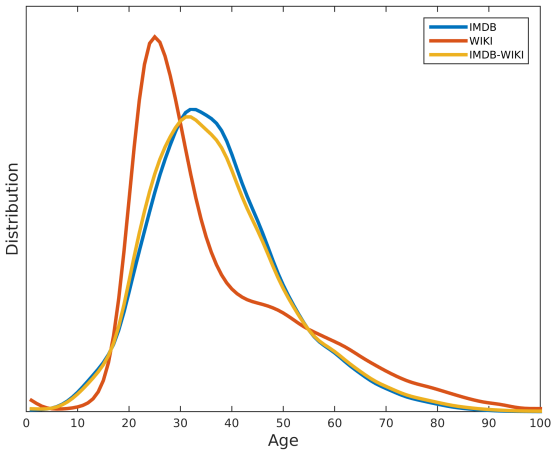

Рис. 3.14: Распределение возрастов в наборе IMBD+WIKI.

UTKFace[25] содержит около 20 тысяч изображений лиц с аннотациями возраста, пола и этнической принадлежности. Данный набор включает в себя широкий диапазон возрастов, начиная от детей до пожилых людей, и представителей разных этнических групп, что позволяет решать задачу распознавания демографических характеристик по лицам. Данные данного набора данных отличаются своей естественностью содержания, то есть они более приближены к реальным условиям использования, что делает его полезным для проверки работоспособности моделей.

На рисунках 3.15-3.16 ярко проявляются различия между этими двумя наборами данных: в первом преобладают фотографии с сценами, специально аранжированными для съемки, тогда как во втором - изображения повседневных ситуаций. Для обеспечения максимальной эффективности и универсальности модели распознавания лиц и определения возраста, в данном проекте будет использоваться комбинация вышеупомянутых наборов данных.

Рис. 3.15: Пример из IMDB+Wiki.

Рис. 3.16: Пример из UTKFace.

3.3.2 Детекция лица

Детекция лица — это процесс определения местоположения лиц на изображении. Это ключевой этап в системах распознавания лиц, который позволяет дальнейшему алгоритму фокусироваться на области изображения, содержащей лицо. Эффективность и точность детекции лиц напрямую влияет на производительность последующего этапа системы. Ниже будут рассматриваться подходы к решению задачи детекции лица.

В качестве метода для оценки качества моделей обнаружения объектов была применена метрика Intersection over Union (IoU), описанная в уравнении 1. Эта метрика использовалась для верификации наличия детекций путем сопоставления с аннотированными данными.

Haar Cascade[27] основан на признаках Хаара, которые представляют собой специфические структуры яркости на изображении. Алгоритм использует каскадные классификаторы, которые эффективно отсеивают области изображения, не содержащие лиц, на ранних этапах обработки. Данный алгоритм обеспечивает быстроту работы и низкие требования к вычислительным ресурсам.

MTCNN(Multi-task Cascaded Convolutional Networks)[36] - модель, использующая каскад (каскад представляет собой последовательность классификаторов, которые применяются к различным частям изображения с разной степенью детализации) из трех сверточных нейронных сетей для обнаружения лиц на разных масштабах изображения и одновременного определения их границ и ключевых точек (например, глаз, носа, рта). Его преимущества заключаются в точности детекции лиц и ключевых точек и способности работать с лицами разных размеров и в различных позах.

YOLO v5 (You Only Look Once)[22] — это алгоритм, основанный на сверточных ней-

ронных сетях, который разбивает изображение на сетку и предсказывает ограничивающие рамки и вероятности классов для каждой ячейки сетки одновременно. YOLO обеспечивает быстродействие в реальном времени с высокой точностью, способность обнаруживать объекты различных классов (включая не только лица, но и, например, тело) на одном изображении.

YOLO v8 - алгоритм, представляющий собой более актуальный вариант в серии You Only Look Once. Ожидается более эффективное выполнение задачи детекции лица с новой версией.

Следует отметить, что модели YOLO обладают способностью обнаруживать не только лица, но и объекты класса "тело".

Очень показательно и важно сравнивать именно эти модели, так как каждая из них представляет собой важный этап в развитии технологий обнаружения объектов на изображениях, в частности лиц. Начиная с метода Haar Cascade, который применяет классические методы компьютерного зрения, до более современных подходов, таких как MTCNN и YOLO, основанных на глубоком обучении нейронных сетей. Дальнейшее сравнение этих моделей позволяет нам лучше понять их особенности, преимущества и недостатки, а также выбрать наиболее подходящий вариант для конкретной задачи обработки изображений.

3.3.3 Модели оценки возраста и пола

Определение возраста и пола по изображениям лиц является ключевой в данной разделе. Эффективное решение этой задачи требует разработки алгоритмов, способных точно идентифицировать возраст и пол человека на разнообразных изображениях. Ниже представлены рассматриваемые модели.

Для оценки качества моделей, предназначенной для определения пола и возраста, использовались метрики точности (accuracy) и средней абсолютной ошибки (Mean Absolute Error, MAE). Точность классификации пола оценивалась с помощью метрики accuracy, определяемой как отношение числа правильно классифицированных индивидуумов к общему числу случаев, что представлено ниже:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Качество оценки возраста анализировалось через среднюю абсолютную ошибку, вычисляе-

мую по формуле

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

где где y_i - истинный возраст, а y_hat_i - предсказанный - предсказанный. Обе метрики сверялись с помеченными данными для верификации результатов модели.

DEX (Deep EXpectation)[23]использует глубокую сверточную нейронную сеть, основанную на архитектуре VGG-16, для прогнозирования возраста человека по его лицу на изображении. Основной идеей является использование регрессии и классификации одновременно для прогнозирования возраста, где возрастные группы используются как классы, и в конечном счете вычисляется ожидаемый возраст как взвешенная сумма по всем классам, как показано на Рисунке 3.17. Это позволяет модели точно и надежно определять возраст, учитывая разнообразие человеческих лиц. К главными преимуществамм данной модели можно отнести высокую точность за счет глубины и количества обучаемых параметров. Данная модель предлагается в паре с набором IMDB+WIKI, так как на момент составления она показала лучший на тот момент результат, что делает ее превосходным примером для сравнения с другими моделями.

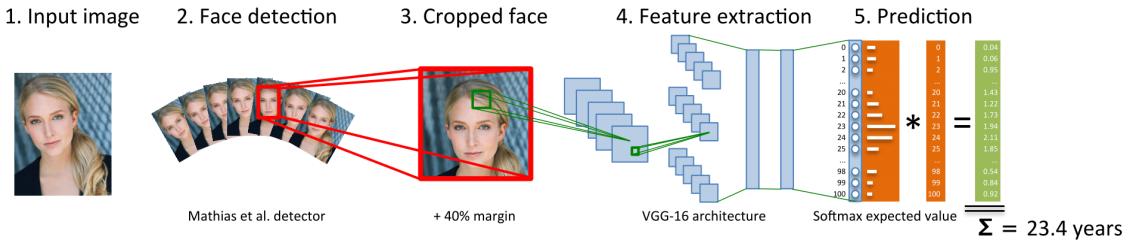

Рис. 3.17: Алгоритм работы DEX.

MiVOLO (Multi-input Transformer for Age and Gender Estimation) [11] - инновационная модель, направленная на одновременное решение задач определения возраста и пола по изображениям, включая те, которые не содержат лиц. Эта модель на данный момент занимает первое место на наборе данных UTKFace и второе место на наборе данных IMDb в соответствующем рейтинге. MiVOLO использует два сверточных стебля (conv-stem) для обработки изображений целиком и отдельных фрагментов (например, лица и тела) независимо друг от друга. Это позволяет модели эффективно извлекать и комбинировать признаки из разных областей изображения без потери детализации. Для объединения признаков на раннем этапе обработки был используется модуль, применяющий механизм cross-attention. Этот подход позволяет сначала обогащать признаки вниманием в одну сторону, затем в другую, и в результате сжимать их через многослойную сеть для достижения целевой размерности

признаков. Такая структура, изображенная на Рисунке 3.18, обеспечивает объединение информации из разных источников и способствует повышению точности предсказаний.

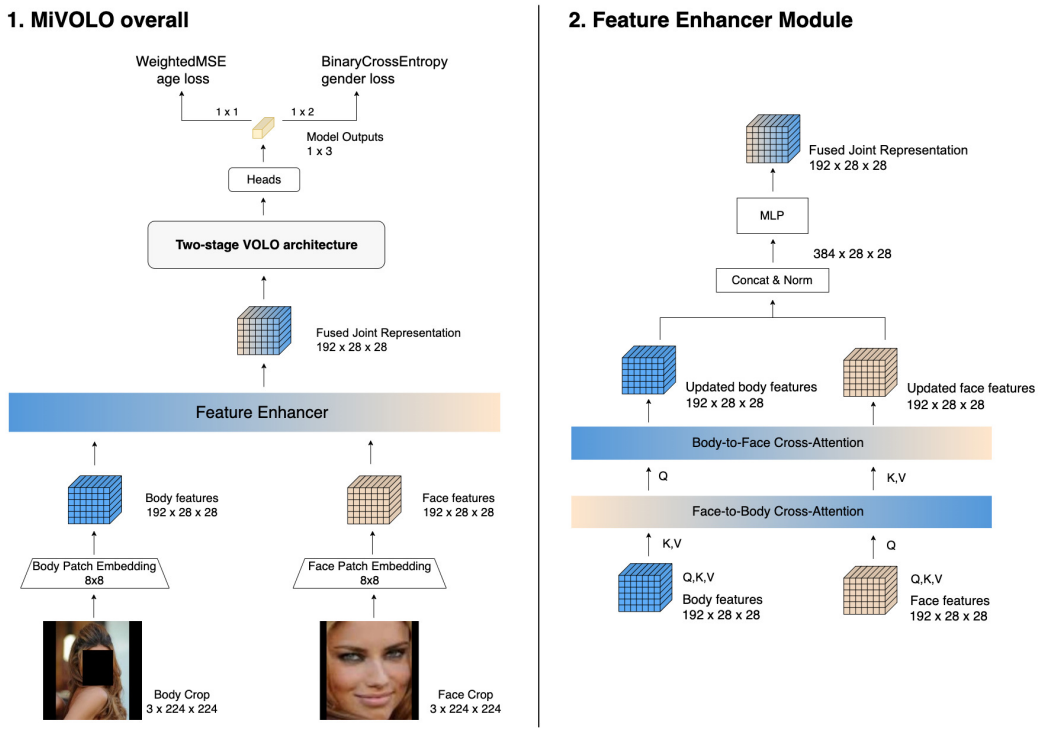

Рис. 3.18: Архитектура MiVOLO.

3.4 Детекция и распознавание текста

Задача — построить и интегрировать модель распознавания текста в пайплайн проекта, с фокусом на текст в естественной среде, например, уличные вывески и этикетки.

3.4.1 Подходы

Мы рассматривали две стратегии в построении таких моделей: end-to-end модели, прямо преобразующие изображение в текст, и двухступенчатый подход, где первая модель выделяет текст на картинке ограничивающими рамками (bounding boxes), а вторая — распознает текст в этих рамках. Из-за сложности обучения и настройки end-to-end моделей, а также для лучшей гибкости,был выбран подход с двумя моделями — для детекции и распознавания текста.

3.4.2 Набор данных

Для решения задачи требуется набор данных картинок с текстами. Так как наша цель — работать с русским и английским языком, то обычные наборы данных вроде ICDAR,

Total-Text и CTW не подходят, так как они используют английский (последний — китайский) языки, поэтому выбран менее известный RusTitW [19] , который включает как синтетические изображения, так и реальные. Синтетическая часть содержит около 600 тысяч картинок со случайными словами на случайных фонах. При генерации такого набора данных на картинке выделяются однородные области,текстом описывается форма произвольной фигуры, случайно выбирается шрифт и размер, настраивается перспектива и текст смешивается с исходной картинкой, чтобы получить как можно более натуральное изображение. Однако это создает риск ухудшения качества работы модели в реальных условиях и возможность неточности метрик. Поэтому реальные данные из другой части RusTitW, насчитывающей более 13 тысяч изображений с русским и английским текстом, будут использованы для точных метрик и потенциального дообучения моделей для повышения качества.

3.4.3 Метрики качества детекции

В качестве подходов для оценивания качества моделей детекции мы рассматривали метрику, использованную для оценки качества на соревновании ICDAR2015, где критерием успешности детекции является Intersection over Union (IoU):

$$\text{IoU}(box1, box2) = \frac{\text{area of overlap}(box1, box2)}{\text{area of union}(box1, box2)} \tag{1}$$

Также был происследован метод TedEval [12], который основывается на сопоставлении текстовых экземпляров и символьном уровне оценки. Этот подход включает создание пар между размеченными и предсказанными рамками, требуя, чтобы полнота и точность превышали установленные пороги, и исключает многострочные совпадения на основе анализа углов. Далее, расчеты recall и precision ведутся на символьном уровне, для каждого ground truth подсчитывается доля букв которые были покрыты хотя бы одним bounding box, и это значение усредняется по всем ground truth, а также для каждого bounding box подсчитывается доля букв,которую он покрыл в сопоставляющихся ground truth, и также усредняется:

$$precision = \frac{\sum_{i=1}^{n} precision_i}{n} \text{ где n - количество задетектированных рамок}$$

$$recall = \frac{\sum_{j=1}^{m} recall_j}{m} \text{ где m - количество ground truth рамок}$$

$$precision_i = \frac{\sum_{j=1}^{m} \sigma_{ji} \cdot \text{word match}_{ji}}{\sum_{j=1}^{m} \sigma_{ji} \cdot len_j}$$

$$recall_j = \frac{\sum_{k=1}^{len_j} I[(\sum_{i=1}^{n} \sigma_{ji} \cdot m_{ji}^k) > 0]}{len_j}$$

$$\sigma_{ji} = I[\text{ j-ый ground truth в паре с i-ым bounding box }]$$

$$\text{word match}_{ji} = \text{количество букв в j-ом ground truth, покрытые i-ым bounding box}$$

$$len_j = \text{количество букв в j-ом ground truth}$$

$$m_{ji}^k = I[\text{ k-ая буква в j-ом ground truth покрыта i-ым bounding box }]$$

3.4.4 Модели детекции

Среди изученных моделей детекции текста выделяются регрессионные подходы, которые напрямую предсказывают ограничивающие рамки текст и подходы, основанные на сегментации, архитектуры придерживающиеся данного подхода, обычно комбинируют предсказания на уровне пикселей и алгоритмы пост-процессинга чтобы получить рамки.

Примером регрессионной модели служит EAST [37], эта модель имеет U-net подобную архитектуру, анализируя карты признаков на разных уровнях для обнаружения мелкого и крупного текста, возвращая уровень уверенности и геометрические координаты текстовых областей, включая углы поворота.Для устранения перекрывающихся прямоугольников применяется NMS алгоритм.

Пример модели, придерживающаяся подхода, основанного на сегментации - PSENet[14]. Эта модель сегментирует текст на уровне пикселей с алгоритмом прогрессивного расширения масштаба, предсказывая маски текста разных масштабов, которые объединяются через BFS-подобный алгоритм. Это помогает различать текстовые экземпляры, расположенные близко друг к другу. Еще один пример модели основанной на сегментации - SAST [31]. Она использует point-to-quad метод для сегментации и внедряет два Context Attention Block, где матрица внимания вычисляется для каждого пикселя только по его столбцу и строке для учета глобального контекста. Далее формируются 4 матрицы - бинарная маска для центральной линии текста (TCL), расстояние от верхних и нижних границ текста (TBO), смещение пикселя от центра текста (TCO) и четырехугольники текста (TVO). Далее происходит кластеризация TCL карты – комбинируются знания из TVO и TCO карт и используется NMS алгоритм.

В случае сложной геометрии текста на картинке, у многих моделей может возникнуть проблемы с его детекцией. Эту проблему решает модель FCENet, которая предлагает новый подход в детекции текста, раскладывая контур текста как функцию в ряд Фурье, если пред-

сказывать какое-то количество коэффициентов ряда для низких частот, то с помощью обратного преобразования можно получить хорошую аппроксимацию контура текста. Помимо этой ветви регрессии параллельно идет ветвь классификации : прогнозируется попиксельная маска текстовых областей, а также маска центра текста (это помогает отфильтровать некачественные предсказания около границ) - для того чтобы получить итоговую карту мы перемножаем эти две матрицы. После используется NMS и обратное преобразование Фурье чтобы получить уже непосредственно границы.

Еще одной сложностью моделей построенных на сегментации, является трудный и долгий процесс постобработки для получения итоговых рамок. Эту проблему решает DBNet [15], внедряя адаптивную бинаризацию, которая упрощает перевод степени уверенности в бинарные метки. Используя аппроксимированную дифференцированную бинаризацию, модель обучается эффективнее, ускоряется постобработка за счет того что бинарные карты хорошо подобраны, улучшается отделение текстовых экземпляров и обработка текста сложной формы. DBNet++ [16] улучшает DBNet, добавляя модуль Adaptive Scale Fusion для адаптивного объединения карт признаков разных масштабов с помощью внимания, таким образом модель становится более устойчивой к масштабу текста, не теряя при этом в качестве.

3.4.5 Метрики для распознавания

Для оценки модели распознавания текста используется точность по словам, могут быть различные вариации данной метрики с учетом регистра, наличия цифр и знаков. В качестве второй метрики применяем NED (Normalized Edit Distance), отражающую качество с учетом сложности восстановления реального текста по выходу модели:

$$\text{NED(predicted text, real text)} = \frac{\text{levenshtein distance(predicted text,real text)}}{\text{max(len(predicted text),len(real text))}}.$$

Расстояние Левенштейна, используемое в формуле - это мера различий между двумя последовательностями символов. Оно определяется как минимальное количество операций, таких как вставка символа, удаление символа или замена символа необходимое для превращения одной последовательности в другую. В дальнейшем под NED мы будем иметь в виду 1-NED для того чтобы увеличение качества соответствовало росту метрики.

3.4.6 Модели распознавания

В процессе изучения различных методов и архитектур, применяемых для решения поставленной задачи, было рассмотрено несколько подходов. Одним из наиболее старых, но все

еще актуальных методов является использование архитектуры CRNN [1], которая включает в себя применение сверточных нейронных сетей для обработки изображений и извлечения карт признаков. Полученная таким образом карта признаков преобразуется в вектор, который затем обрабатывается с помощью двунаправленной сети долгой краткосрочной памяти (BiLSTM) для формирования текстовой последовательности. Несмотря на свою простоту, данный подход демонстрирует уязвимость к деформациям текста на изображении.

Альтернативный, более современный подход включает использование архитектур типа encoder-decoder, которые генерируют текстовую последовательность посимвольно, что позволяет улавливать более широкий контекст. Тем не менее, процесс инференса в таких моделях оказывается относительно медленным.

Дальнейшее развитие архитектур для распознавания текста заключается в интеграции лингвистических знаний непосредственно в модель, чтобы, опираясь на них, можно было корректировать визуальный шум или неточность на картинке. Однако такие модели обычно достаточно сложные по архитектуре и обучению, и требуют большей емкости, чтобы выучить полезные знания о языке.Это ограничивает используемость таких моделей.

В онлайн сервисах, использующих модели распознавания текста на картинке, достаточно важна скорость инференса, а также легковесность модели, для ее практического применения. Соответственно недавние тенденции предпочитают использование единой визуальной компоненты. Примером модели в такой парадигме является SVTR.

При выборе архитектуры для модели распознавания я анализировала следующие: CRNN, SVTR, ABINET, SPIN, Robust Scanner

SVTR [34] - Основной особенностью данной модели является её способность анализировать внутрисимвольные различия, а также улавливать взаимосвязи между символами и контекст их расположения. SVTR разбивает изображения на маленькие блоки,для каждого из которых через применение сверток формируются эмбеддинги. Далее следует три основных этапа обработки: global mixing, local mixing и процесс смешивания. На этапе global mixing применяется механизм self-attention с MLP, что позволяет оценить взаимную зависимость всех блоков в целом. Этап local mixing использует аналогичную архитектуру, однако с добавлением маски в механизме attention, что ограничивает внимание только окружающими блоками. Это нужно для того чтобы выучить информацию о конкретном символе и о связях между его элементами. Затем происходит этап смешивания, на котором карта признаков уменьшается по высоте с помощью сверток. Последний этап включает уменьшение высоты до одной строки через пулинг и последующее использование линейного слоя для каждого блока, что приводит к формированию логитов, из которых формируются вероятно-

сти символов. SVTR демонстрирует перспективные возможности в области распознавания текстов с изображений, сочетая глубокие особенности символов и их контекстуальную связь. Архитектура модели представлена на Рисунке 3.19

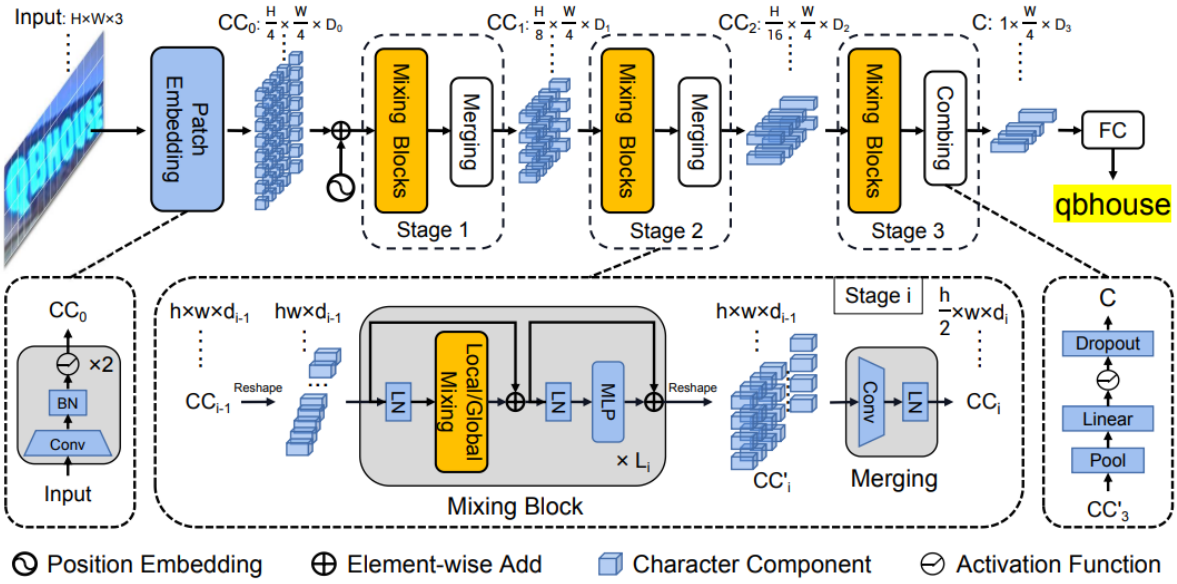

Рис. 3.19: Архитектура SVTR

ABINET [24] - Реализует подход с добавлением лингвистических знаний. Модель состоит из двух основных компонентов: визуального процессинга и двунаправленной языковой модели, между которыми применяется механизм блокировки градиента. Такая структура позволяет языковой модели обучаться независимо от визуального контента, фокусируясь на изучении языковых особенностей. Процесс распознавания начинается с обработки изображения с помощью ResNet, после чего изображение разделяется на патчи. Эти патчи последовательно проходят через слои внимания, где в качестве матрицы запросов используются позиционные эмбеддинги, соответствующие порядку символов. Далее используя линейные слои и softmax мы получаем для каждой позиции распределение вероятностей символов. Полученные данные подаются в языковую модель, которая состоит из нескольких слоев декодера трансформера, из которых исключены механизмы self-attention и маскирования. В качестве матрицы запросов в языковой модели выступают либо позиционные эмбеддинги порядка символов, либо результаты работы предыдущего слоя, в то время как матрицы key и value являются линейными преобразованиями над распределениями вероятностей символов. Языковая модель корректирует предсказания, полученные от визуальной части, используя при этом накопленные знания о языке. После каждой итерации коррекции, результаты визуальной модели и уже скорректированные версии смешиваются и повторно подаются в языковую модель. В итоге, такой итеративный процесс позволяет постепенно уточнять рас-

пределение вероятностей символов, обогащая их лингвистическими знаниями. Архитектура модели представлена на Рисунке 3.20

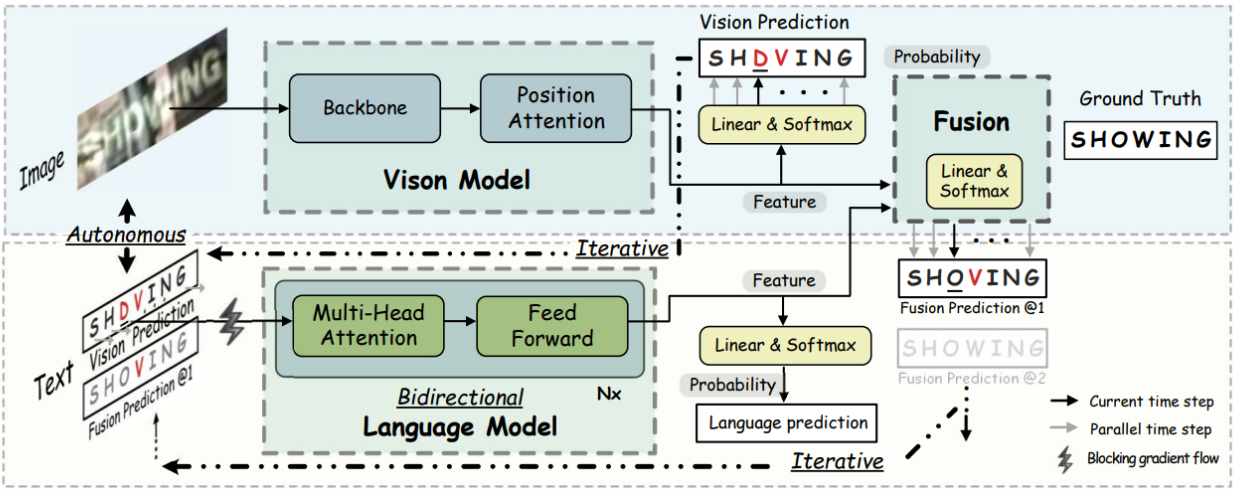

Рис. 3.20: Архитектура ABINet

SPIN [5] - Данная модель включает в себя модуль выпрямления, который настроен на изменение изображения для упрощения дальнейшего процесса распознавания текста. Несмотря на то, что искажения формы текста являются значимой проблемой в задаче распознавания текста, существуют и другие сложности, такие как плохая яркость, наличие теней, шума и так далее. Они относятся к категории хроматических искажений. SPIN предлагает архитектуру, способную справляться с подобными искажениями. Авторы классифицируют хроматические искажения на 2 типа: inter pattern, когда шумовые части близки к текстовым или когда текст рассеян, и intra pattern который включает различные окклюзии и тени на изображении. SPIN корректирует оба типа искажению, для этого вводится inter pattern модуль и intra pattern модуль. Модуль inter pattern работает путем изменения интенсивности каждого пикселя на основе сигмоидальной функции от линейной комбинации нескольких степенных функций примененных к исходной интенсивности. Веса для этого процесса генерируются из входного изображения через сверточные блоки. Пиксели, которые в изначальной картинке имели одинаковую интенсивность после преобразования также будут сохранять одинаковую интенсивность.Таким образом данный модуль позволяет отделять полезные структурные паттерны от шумовых путем выведения их на различные уровни интенсивности, что способствует повышению контрастности и яркости. Intra pattern модуль предсказывает смещение для каждого патча картинки и потом экстраполирует на все изображение. Измененное изображение является выпуклой комбинацией исходного изображения и предсказанного смещения. Модуль геометрического смещения: мы обучаем функцию f кото-

рая может представлять разнообразные формы текста. Для получения пикселя y скорректированного изображения проводится интерполяция соседних пикселей исходного изображения на позиции,определенной функцией f. Процесс полного исправления искажений выполняется последовательно: вначале мы корректируем intra pattern искажения, после inter pattern, затем искажения формы и после этого подаем в сеть распознавания CRNN. Архитектура модели представлена на Рисунке 3.21

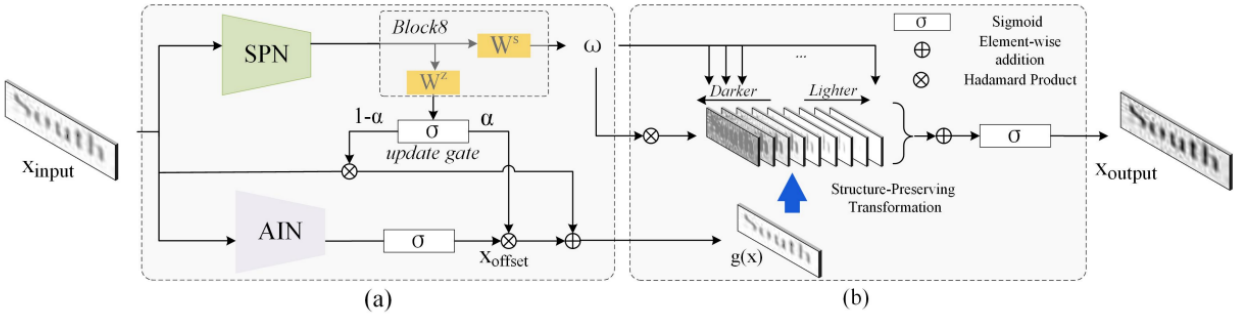

Рис. 3.21: Архитектура SPIN

Robust Scanner [33] - Данная модель модернизирует распространенную encoder-decoder архитектуру.Эта архитектура включает в себя первичную обработку изображения через сверточную сеть (ResNet), в результате которой формируется карта признаков. На этапе генерации текста используется LSTM для создания query векторов для каждого символа. Эти векторы затем применяются в механизме внимания, где key и value получаются из ранее сформированной карты признаков. После процесса внимания следует классификация символов с использованием линейных слоев. Авторы проанализировали эту модель и выяснили, что векторы, генерируемые LSTM, кодируют как позиционную, так и контекстную информацию о символе. Со временем процесса декодирования позиционная информация ослабевает, в то время как контекстная информация становится более выраженной. Это может привести к ухудшению качества распознавания в случае текста с непоследовательной структурой, а также к ошибкам в длине распознаваемого текста или в распознавании последних символов.Для того чтобы это исправить авторы вводят модуль с усиленной позиционной информацией и динамически объединяют его выходы с выходом обычного декодера, описанного ранее.

Новый модуль обучает эмбеддинги для каждой возможной позиции символа и преобразует карту признаков для включения информации о положении текста. Для этого используется LSTM, обрабатывающий каждую строку карты признаков, с последующим применением нескольких сверточных слоев. В процессе внимания в качестве query используется позиционный эмбеддинг, в качестве key - модифицированная карта признаков, а в качестве

value - оригинальная карта признаков. Далее, применяются линейные слой для получения логитов.Такая структура заставляет модуль изменять карту признаков так, чтобы она коррелировала с позициями символов. После генерации каждой из ветвей логитов для текущей позиции,они динамически смешиваются и получившиеся логиты используются для предсказания символа на текущей позиции. Архитектура модели представлена на Рисунке 3.22

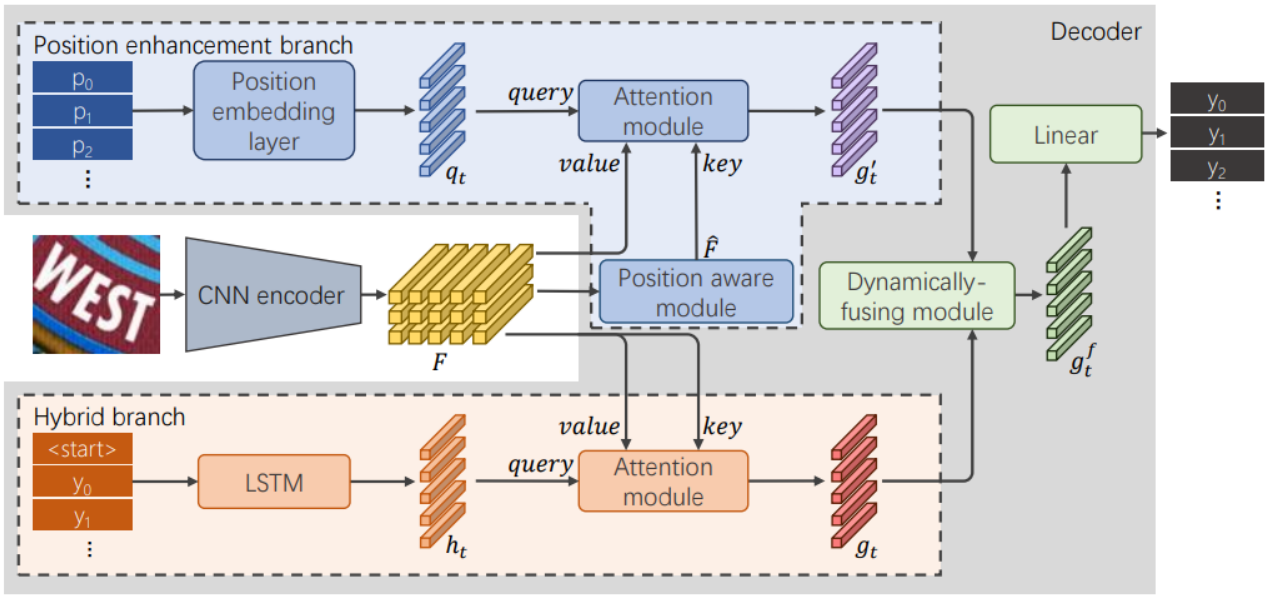

Рис. 3.22: Архитектура Robust Scanner

4 Полученные результаты

4.1 Детекция элементов интерфейса

Для того, чтобы успешно решить задачу детекции элементов интерфейса, необходимо было выбрать оптимальную для детекции модель машинного обучения и обучить её. Под оптимальностью имеется в виду не только качество инференса модели с точки зрения метрик, но и быстродействие модели при его генерации, чтобы обеспечить общую высокую скорость ответа сервиса.

Для обучения необходимо было найти или сгенерировать данные, представляющие различные скриншоты интерфейсов UI, которые пользователи могут видеть на своих электронных устройствах. Итоговый набор данных должен обладать следующими характеристиками:

- не менее 1000 объектов в наборе
- полнота разметки (отсутствие неразмеченных объектов интерфейса)
- аккуратная локализация объектов интерфейса (так, с одной стороны, границы рамок, в которые помещаются объекты, должны полностью покрывать их, с другой стороны, рамка не должна быть избыточна - она должна покрывать исключительно свой объект)
- оптимальное количество классов (не более 20 - чтобы не было таких ситуаций, что тот или иной объект встречается в наборе данных очень редко, в результате чего модели будет сложно “научиться” определять его)
- никакие два класса не должны пересекаться между собой, как по смыслу, так и по разметке

4.1.1 Выбор набора данных

В ходе работы над проектом была обнаружена и решена следующая проблема: в рассматриваемом домене компьютерного зрения - элементы интерфейса (далее будем называть их UI-элементами) - очень мало пригодных для решения задачи наборов данных. Среди всех рассмотренных нами наборов больше 50% содержали меньше 1000 объектов, что крайне мало для дообучения модели. Наборы данных более или менее удовлетворительного размера обладали очень низким качеством разметки из-за сложности разметки данных для задачи

детекции в целом - и в частности - для задачи детекции UI-элементов ввиду того, что деление на классы здесь может быть достаточно субъективно.

В итоге было отобрано 3 следующих набора, более или менее удовлетворяющих перечисленным выше критериям:

- WebUI [32]
- mrtoy/mobile-ui-design[20]
- VINS Dataset[2]

Более детальный дальнейший анализ набора данных **WebUI** (которым пришлось заниматься из-за отсутствия документации к нему) показал, что он не подойдёт под задачу распознавания UI-элементов, так как в нём в качестве классов рассматриваются html-объекты разных web-страниц, которых слишком много, из-за чего по ним будет сложно обучить хорошую модель.

Анализ набора данных **mrtoy/mobile-ui-design** показал, что он также не подойдёт для успешного решения задачи из-за достаточно “общего” выбора классов. В частности, - из-за обилия объектов классов ’group’ ("группа") и ’rectangle’ ("прямоугольник"), которые не очень содержательны (ими описывалось группы из нескольких объектов класса "текст"или "изображение"). Также в этом наборе очень много пустых с точки зрения авторов разметки зон, на которых на самом деле есть виджеты, которые также было бы полезно определять.

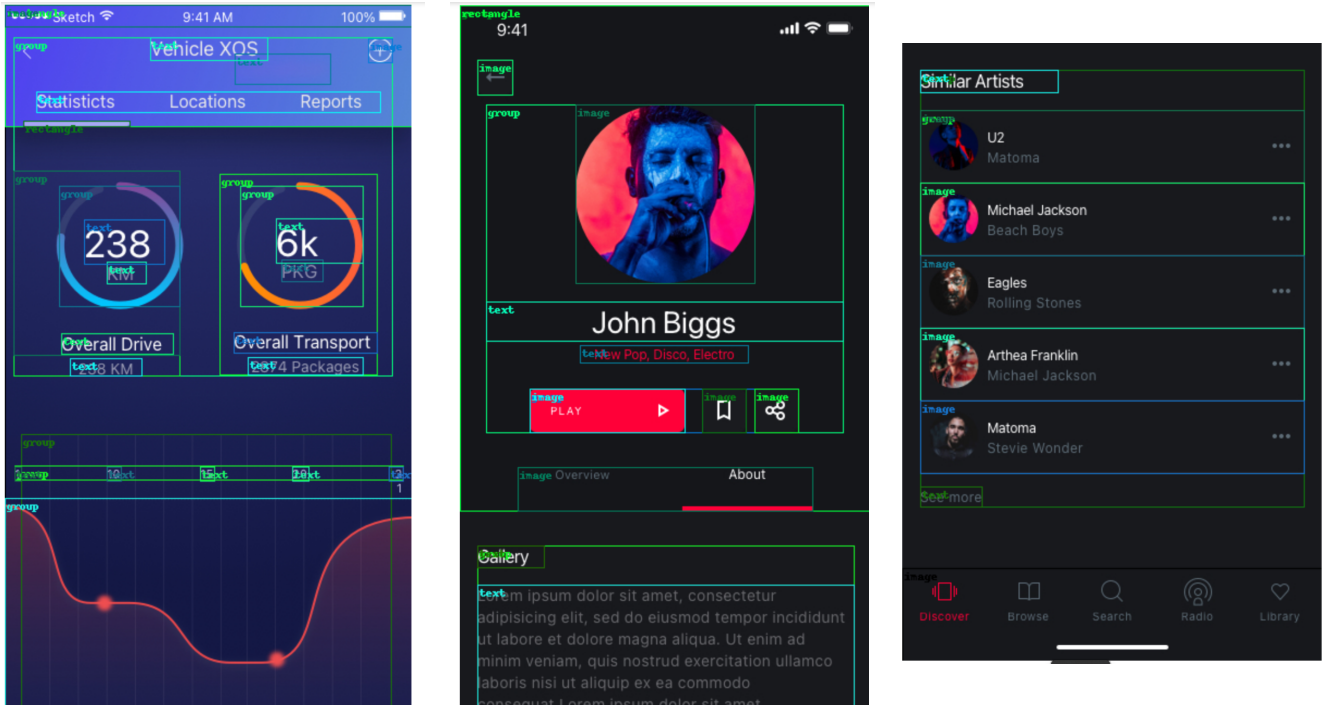

Рис. 4.1: Примеры разметки в наборе данных mrtoy/mobile-ui-design

Набор данных **VINS Dataset** удовлетворяет почти всем указанным выше критериям. В нём достаточно много изображений, разметка выполнена качественно, классы выбраны наиболее оптимально по сравнению со всеми остальными рассматривавшимися наборами.

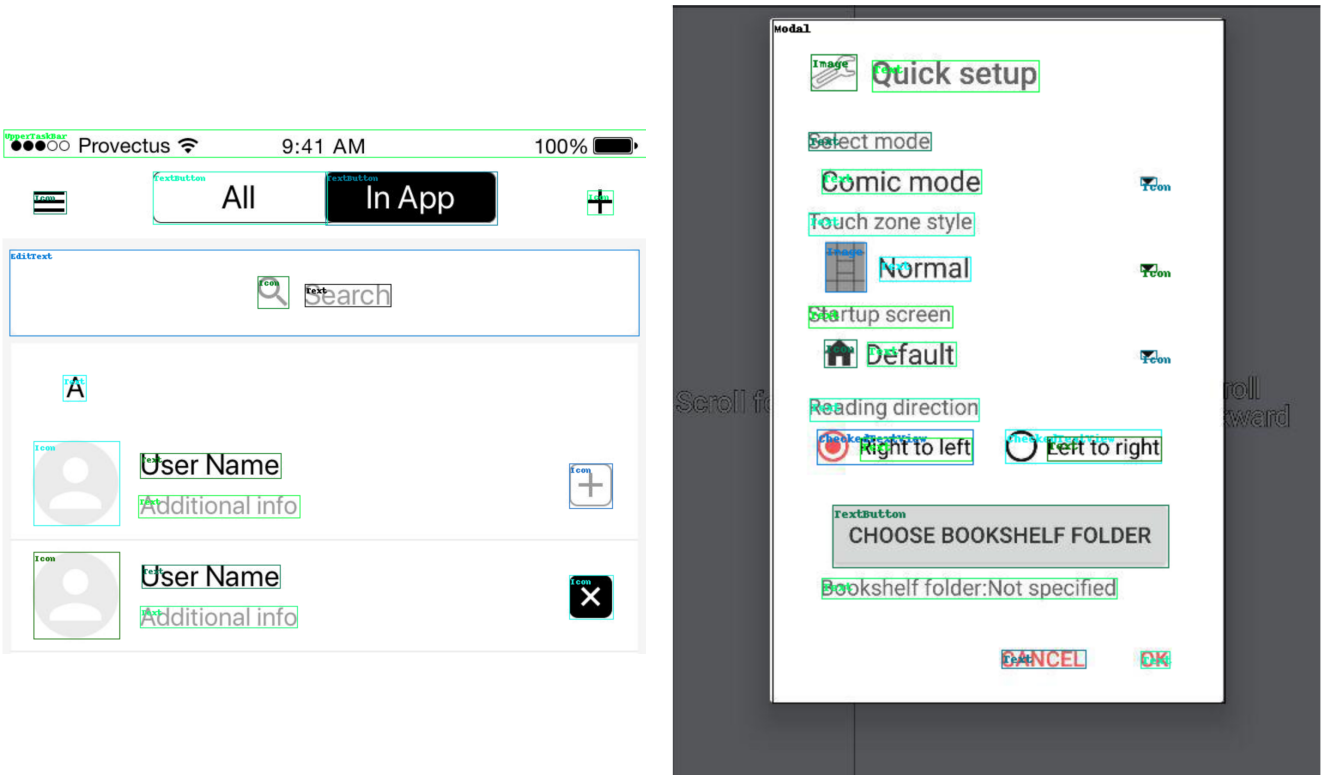

Рис. 4.2: Примеры разметки в наборе данных VINS Dataset

В итоге в качестве набора данных был выбран VINS Dataset, удовлетворяющий почти всем критериям качества набора данных, описанным выше. Тем не менее разметку пришлось переработать, чтобы она лучше подходила для решения задачи. В частности, было переработано разделение объектов на классы (см. дальнейший материал). Также в связи с наличием, хотя и небольшого, но обладающего высоким качеством - как самих данных, так и разметки - набора данных также было принято дополнительное решение отказаться от генерации дополнительных данных, так как скорее всего синтетические данные в этой задаче лишь сделают распределение объектов более шумным.

4.1.2 Бенчмарк для сравнения моделей

Помимо работы с данными, была изучена литература по задаче object detection и на основе изученного материала были приняты следующие решения для создания бенчмарка:

- были выбраны метрики качества модели. Будем рассматривать две вариации mAP - mAP50 и mAP@[50-95]: в первом варианте порог **IoU** модели равен 0.5, во втором

варианте оценка усредняется по всем порогам **IoU** в диапазоне от 0.5 до 0.95 с шагом 0.05. Также, как было указано выше, крайне важно обеспечить не только качество детектора, но и его быстродействие, так как детекция - лишь первый этап решения общей задачи. Поэтому, кроме mAP, важно было учитывать **average inference speed** (среднее время на генерацию предсказания для одного объекта или батча объектов, в миллисекундах).

- Для экспериментов нами были выбраны следующие модели: YOLOv5, YOLOv8, YOLOv9 (была добавлена в бенчмарк после выпуска модели), Co-Detr, Detr (обычный Detr имело смысл рассматривать только в контексте присутствия Co-Detr в бенчмарке для сравнения этих моделей).

- Сразу необходимо отметить, что у нас были некоторые сомнения в связи с использованием моделей семейства Detr, в частности, архитектуры Co-Detr, так как, несмотря на высокие значения метрик, большие трансформеры известны своей требовательностью к ресурсам при обучении (мы были ограничены в вычислительных возможностях). Также были риски того, что из-за особенностей окружения и отсутствия должной документации Co-Detr мог просто не запуститься. В данном случае планировалось перестать рассматривать модели семейства Detr.

4.1.3 Подготовка набора данных

Так как набор данных по умолчанию не подходил под формат моделей в бенчмарке, пришлось сделать ряд технических преобразований:

- Мы удалили фотографии-дубликаты

- Нами была полностью переработана разметка. В частности:

  – были убраны нерелевантные и редкие классы, не несущие дополнительной информации для пользователя, такие, как "CheckBox "Checkbox"(описывающие элементы интерфейса “чекбокс”) и т.п.

  – Некоторые классы были объединены в один для оптимизации работы модели

После этих преобразований требовалось разбить выборку на обучающую и валидационную. При этом, для того, чтобы дальнейшие эксперименты проводились корректно, важно было постараться сохранить соотношение классов в этих подвыборках. Вообще говоря, таких

алгоритмов на данный момент для задачи детекции нет, так как неочевидно, как сделать такое разбиение по фотографиям, чтобы объекты на них сохраняли пропорцию, так как самих объектов на фотографиях может быть разное количество.

Наша команда придумала для решения данной задачи следующий алгоритм.

Представим исходную выборку как корневую вершину дерева, а подвыборки - как её сыновей.

Из теории машинного обучения известно про функционал качества, использующийся при разбиении вершины в решающем дереве, основанный на так называемом **критерии информативности** $H(R_m)$:

$$Q(R_m) = H(R_m) - \frac{|R_l|}{|R_m|}H(R_l) - \frac{|R_r|}{|R_m|}H(R_r)$$

Для деревьев этот функционал максимизируют по множествам R_l и R_r (или - что то же самое - по одному из этих множеств, так как второе множество при объединении с первым даст исходное множество R_m).

Интуиция данного функционала заключается в следующем: критерий информативности $H(R)$ показывает качество распределения целевой переменной среди объектов множества - чем он меньше, тем меньше разнообразие значений целевой переменной среди объектов данного множества; максимизируя функционал Q по множествам R_l и R_r, мы добиваемся того, что распределения объектов в подмножествах, получаемых после разбиения некоторого множества R_m, становятся более вырожденными, чем распределение исходного.

Воспользуемся этой идеей в нашем случае. Хотим так разбить множество в корне дерева на два непересекающихся подмножества, чтобы распределение каждого из новых двух подмножеств было как можно ближе к распределению исходного. В терминах критерия информативности и функционала Q это означает, что Q должен стремиться к нулю.

Отсюда **идея алгоритма:**

Пусть $H(R)$ в нашем случае - значение энтропии, посчитанное по выборке R.

Пусть изначально имеется две подвыборки: левая - состоящая из всех элементов исходной выборки - и правая - пустая. Пусть N - это заданное количество объектов в валидации. Будем проходить N раз по левой выборке. Среди всех оставшихся фотографий из левой выборки будем искать ту, добавление которой наименьшим образом будет изменять распределения в левой и в правой выборке, то есть такую, для которой значение несколько обновленного функционала Q:

$$Q^*(R_l, R_r, \text{obj}) = \left| \frac{|R_l| \cdot H(R_l) + (|R_r| - 1) \cdot H(R_r) - (|R_l| - 1) \cdot H(R_l \setminus \{\text{obj}\}) - |R_r| \cdot H(R_r \cup obj)}{|R_l \cup R_r|} \right|$$

- будет наименьшим. Такую вершину будем перекладывать из левой подвыборки в правую.

Для того, чтобы данный алгоритм работал за адекватное время, требуется также предподсчитать все распределения объектов и работать уже с ними, а не с файлами в наборе данных.

В результате применение данного алгоритма позволило разбить выборку так, как нам требовалось.

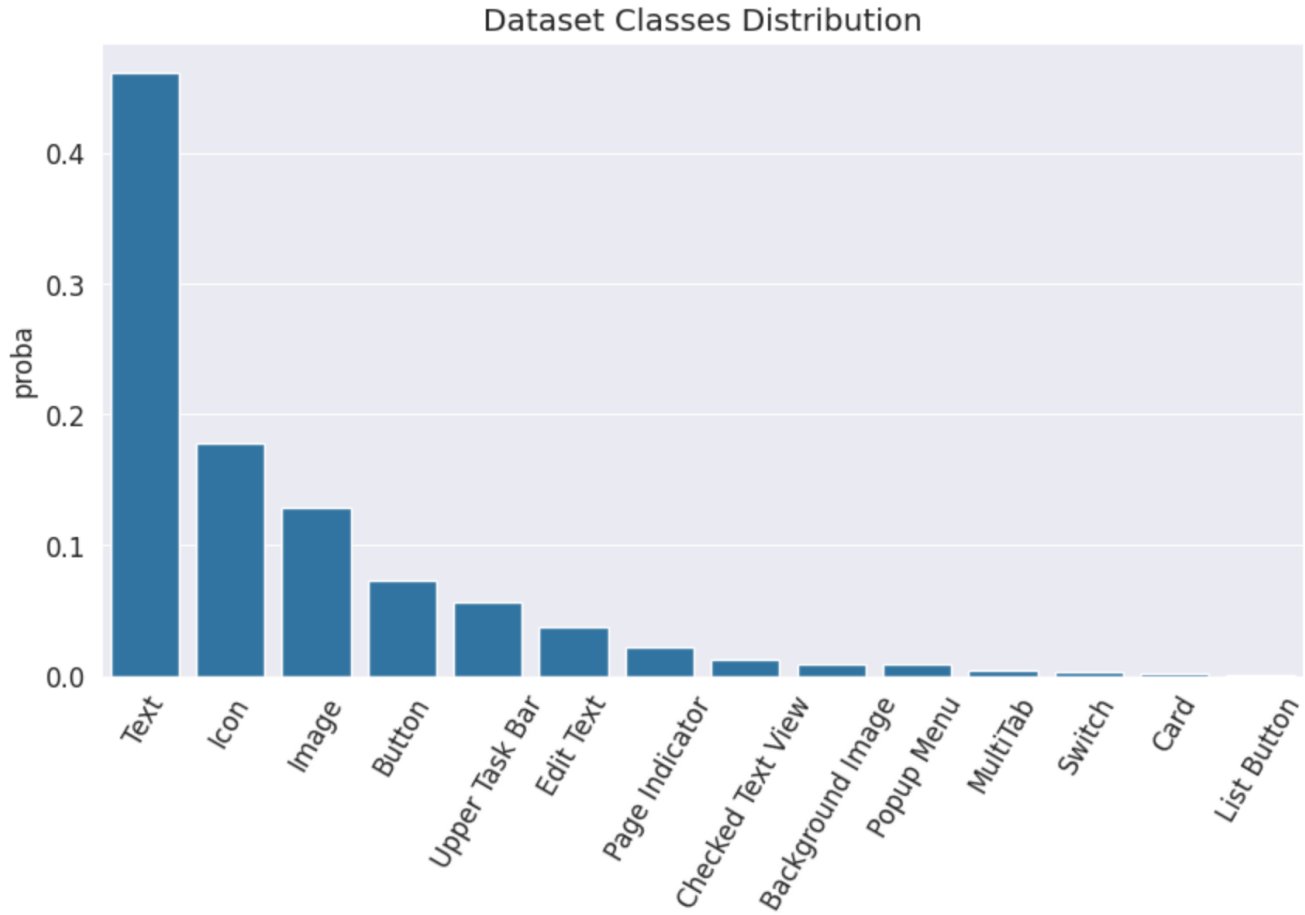

Рис. 4.3: Распределение классов в VINS Dataset

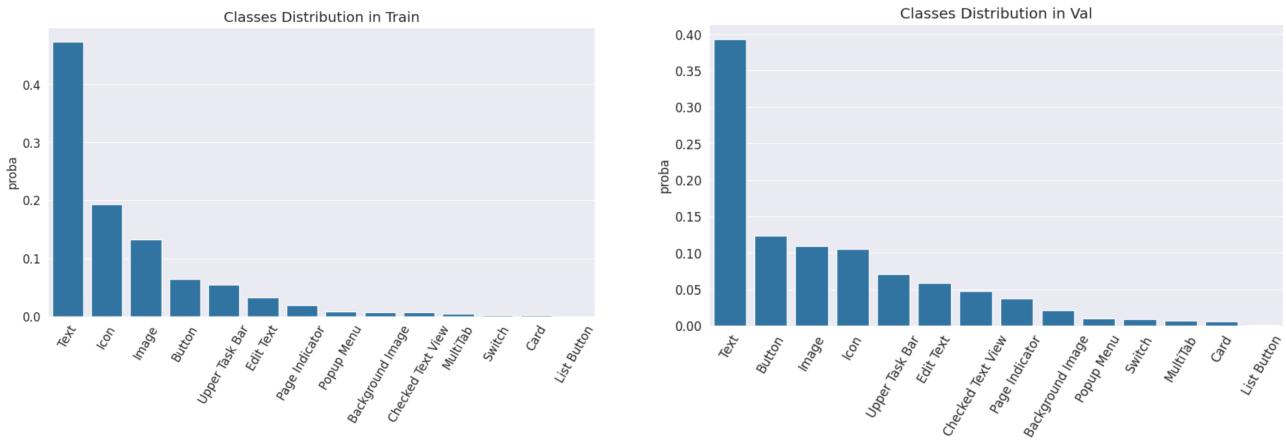

Рис. 4.4: Распределение классов в обучающей и валидационной выборках

После всех основных преобразований выборка переводилась в специфичные форматы, подходящие под различные модели.

4.1.4 Fine-Tuning моделей

Так как выбранные для бенчмарка модели машинного обучения - это не Zero-Shot-решения, то есть они не могут сходу работать с доменом, на котором не обучались, их необходимо либо обучать с нуля, либо дообучать (предобученные версии).

Так как дообучать полностью предобученные модели под конкретный домен вычислительно легче, чем обучать модель с нуля, а также - так как данный подход - дообучение целой модели - обычно демонстрирует лучшее качество, чем другие способы дообучения, например, Linear Probing (дообучение только нескольких последних линейных слоёв модели), будем делать именно так.

Также следует отметить, что в связи с малым количеством документации к Co–Detr и высокими техническими требованиями, не удалось дообучить Co-Detr. Поэтому от Co-Detr пришлось отказаться. Также, так как без Co-Detr задача дообучения классического Detr стала нерелевантной, пришлось отказаться и от него.

Было принято решение дообучать все модели по 30 эпох. YOLOv5 и YOLOv8 дообучались распределённо на трёх видеокартах Nvidia T4, YOLOv9 из-за тяжелого backpropagation не помещалась на них при обучении, поэтому дообучалась отдельно на одной видеокарте Nvidia P100.

Результаты дообучения YOLO представлены в таблице 4.1 и на изображениях 4.5, 4.6, 4.7, 4.8.

Model	mAP50	mAP@[50, 95]	# params (M)	Average Speed (ms)
YOLOv5	0.851	0.778	115.8	85.26
YOLOv8	0.866	0.824	**40.0**	68.14
YOLOv9	**0.878**	**0.835**	43.4	**57.39**

Таблица 4.1: Результаты различных детекторов при валидации на наборе данных VINS Dataset

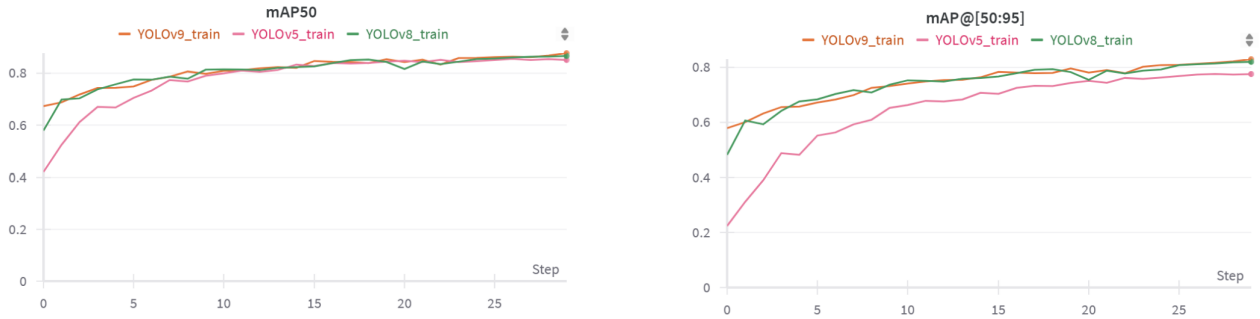

Рис. 4.5: Метрики качества разных версий YOLO по эпохам

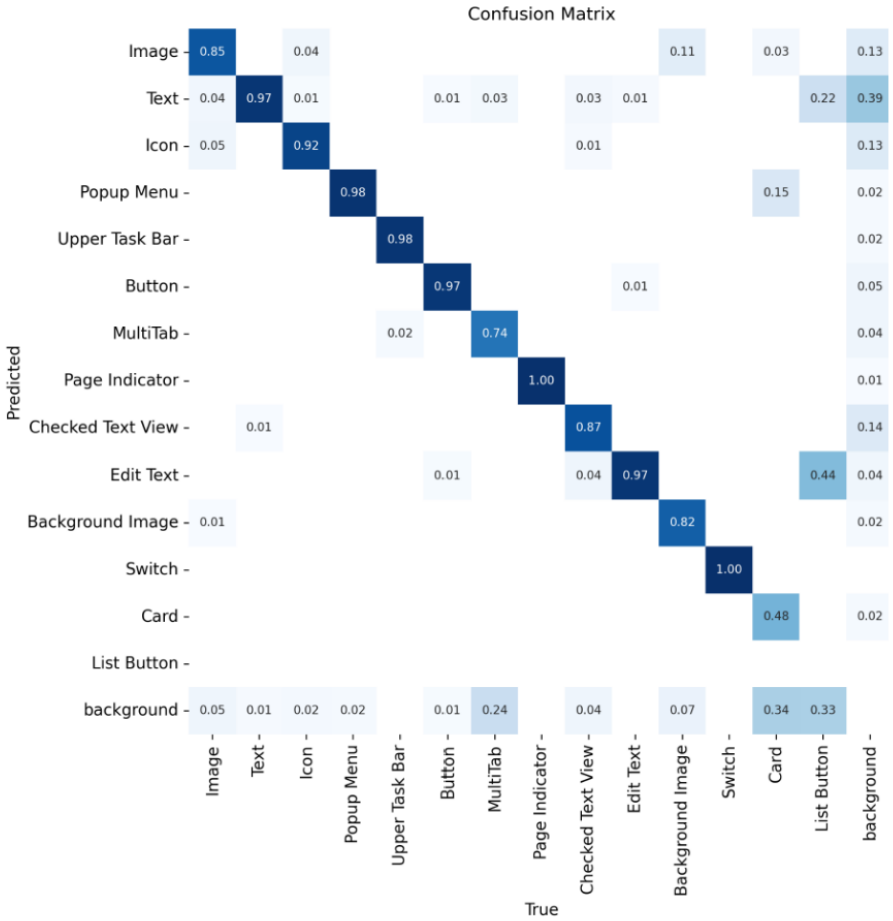

Рис. 4.6: Матрица ошибок, YOLOv5

Как можно видеть, все из рассмотренных моделей хорошо справляются с задачей

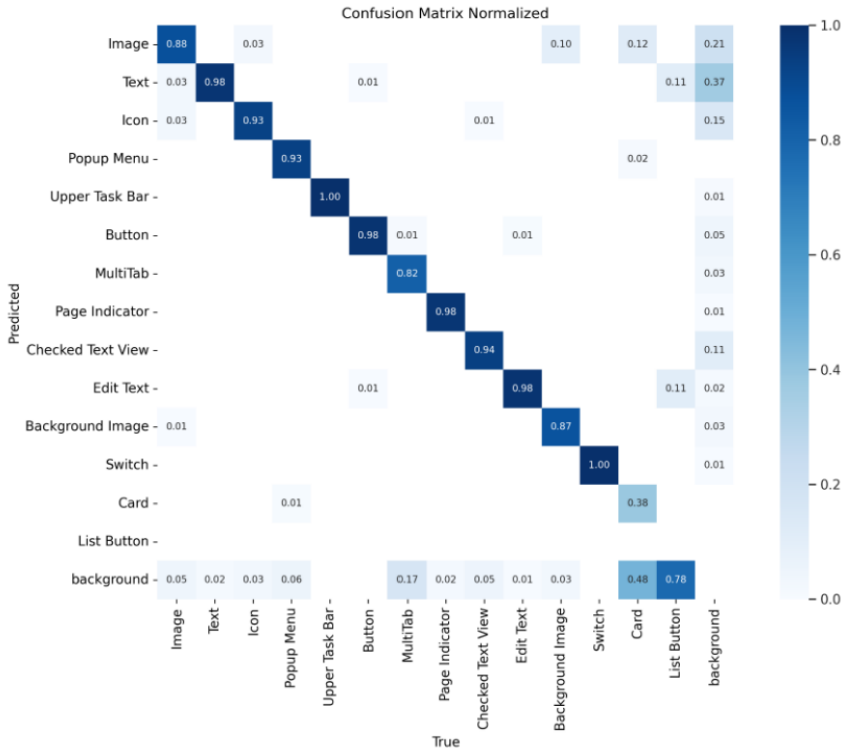

Рис. 4.7: Матрица ошибок, YOLOv8

детекции - доли правильных ответов велики практически по всем классам, за исключением лишь относительно редкого класса “ListButton” - из-за того, что он редкий, им можно будет пренебречь при промышленном использовании модели. Неожиданным результатом оказалось то, что модели умеют различать иконки (класс Icon) и изображения. Также обратил на себя внимание тот факт, что модели справляются с детекцией очень абстрактных и не всегда однозначно выделенных объектов, таких, как диалоговые окна, меню и т.п.

Из полученных результатов следует, что наилучшая с точки зрения метрик модель - YOLOv9 - при том условии, что у неё меньше всего параметров. Время её работы, хотя и не лучшее, но при этом удовлетворительно для решения рассматриваемой задачи распознавания GUI. В итоге Наша команда решила использовать YOLOv9 в конечной версии сервиса.

4.2 Описание изображений

4.2.1 Задача

После классификации элементов интерфейса на блоки, появляется задача описания тех из них, что представляют из себя фотографии и картинки. Данная проблема не нова, однако в контексте текущей задачи есть несколько существенных нюансов, которые нужно учесть:

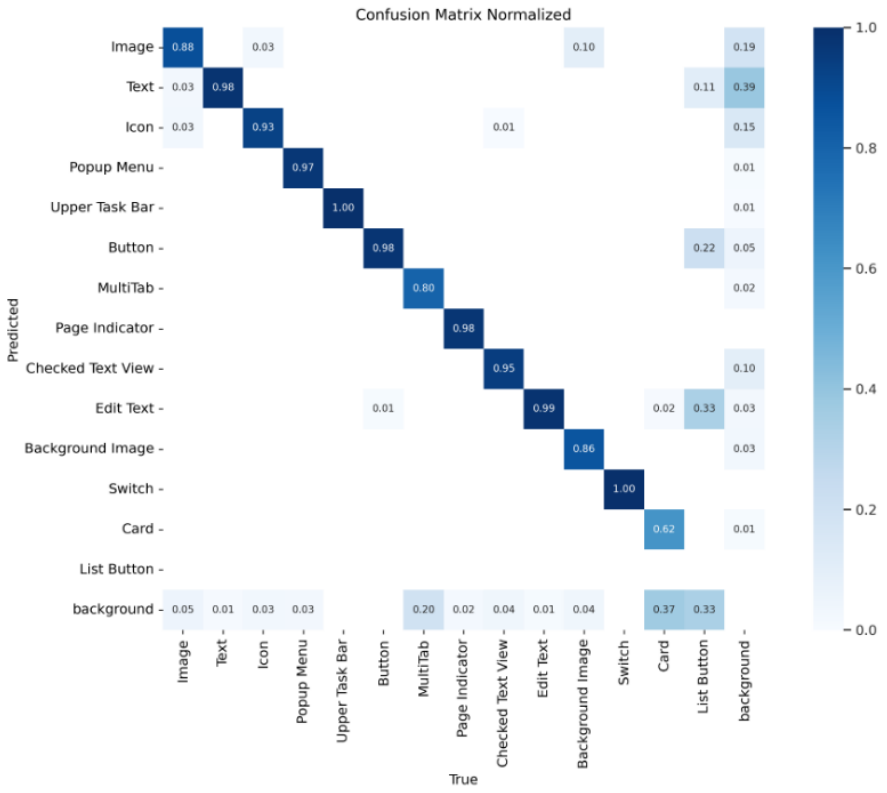

Рис. 4.8: Матрица ошибок, YOLOv9

- Сервис должен работать в реальном времени, а значит модель должна выдавать ответ за доли секунды для достижения положительного пользовательского опыта
- Разнообразие доменов изображений, которые необходимо охватить, огромно. Необходимо убедиться, что модель хорошо работает с разными типами контента, а также уметь обрабатывать материал, с которым модель справляется плохо.

Процесс изучения был разделён на этапы:

- Изучить домены, которые могут встретиться в медиа
- Подобрать набор данных, который содержит большое разнообразие картинок
- Выбрать метрики для оценки качества моделей
- Изучить какие существуют готовые модели для решения подобной задачи
- Выбрать модель(-и), которая(-ые) хорошо решает(-ют) поставленную задачу
- Найти слабые места модели
- Обработать данные уязвимости модели

Model	CLIPscore on COCO	Time (ms)	Parameters cnt
VIT-RuGPT	0.1831	755	1074584064
GIT-Base	0.2715	183	176619066
BLIP-Base	0.2898	230	247414076
VIT-GPT2	0.2917	161	239195904
GIT-Large	0.2723	1513	394196026
GIT-Large-coco	0.3085	1490	394196026
BLIP-Large	0.3054	370	469732924

Таблица 4.2: Сравнение моделей

4.2.2 Сравнение моделей

В Таблице 4.2 представлены результаты моделей, которые были посчитаны на тестовой выборке набора данных MS COCO, за основную метрику был взят CLIPscore.

Русскоязычная модель VIT-RuGPT точно не подходит под поставленную задачу, она показала плохое значение метрики, а также имеет просто недопустимо большое кол-во параметров

Изначально предполагалось сделать акцент на модели базового типа, так как они не только имеют достаточно хорошее значение метрик, но и более высокую производительность. Однако, исследовав их качество на доменах, которые не встречаются в наборе данных MS-COCO, было выявлено, что они работают на них достаточно нестабильно. Примеры можно увидеть в блоке "Домены", где показаны некоторые описания модели BLIP-Base

GIT-Large показала достаточно неплохое качество на тестовой выборке, однако имеет очень высокое время одного предсказания. В контексте поставленные задачи данный факт является критичным

Наиболее подходящий для нас результат показала модель BLIP-Large, было принято решение взять её, как **основную модель** (не только из-за значения метрики, но и из-за большей стабильности на различных доменах)

Было бы неправильно не упомянуть более современную версию BLIP: BLIP-2, однако она имеют слишком большое число параметров, даже у самой легковесной её версии их больше 3 млрд, что не допустимо в контексте нашей задачи

4.2.3 Домены

Изображения в медиа можно разбить на следующие домены:

1 Фотографии, пейзажи

2 Мультяшные картинки, аниме

3 Текст

4 Портреты, фотографии человека в полный рост

5 Мемы

Изучая первый домен, можно прийти к выводу, что модель справляется с ним достаточно хорошо, имеется в виду, что она описывает основную суть картинки, сильно не концентрируясь на мелких деталях (Примеры на рис 4.9, 4.10).

Рис. 4.9

BLIP-Large: there is a red and white bus driving down the street.

Рис. 4.10

BLIP-Large: there is a plate of food with broccoli and rice on it.

Для того чтобы окончательно в этом убедиться, было построено распределение CLIPscore на картинках датасета COCO. Оно изображено на Рисунке 4.11. Можно заметить, что данное распределение похоже на нормальное с мат ожиданием 0.3.

Для анализа был создан pdf файл с описаниями всех картинок, которые имели низкий CLIPscore. Можно заметить, что даже изображения с маленьким значением метрики имеют достаточно валидные описания, однако почти в каждом втором сообщении находятся странные, не имеющее смысла слова **arafed, arafee и тд**. Обработка данных артефактов и причина их возникновения будет описана ниже

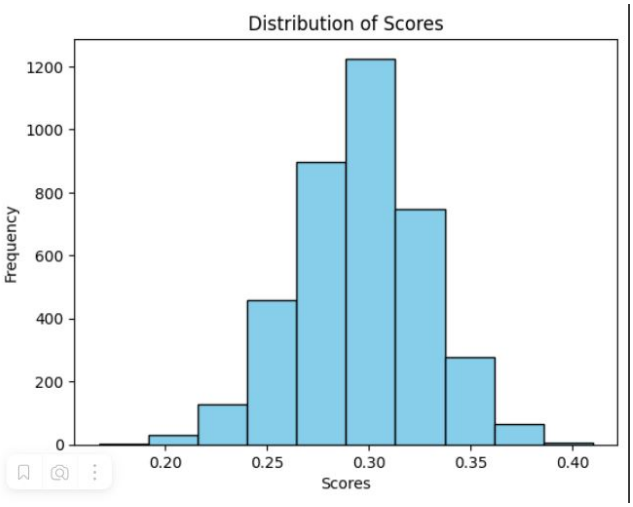

Рис. 4.11: Распределение CLIPscore на наборе данных COCO

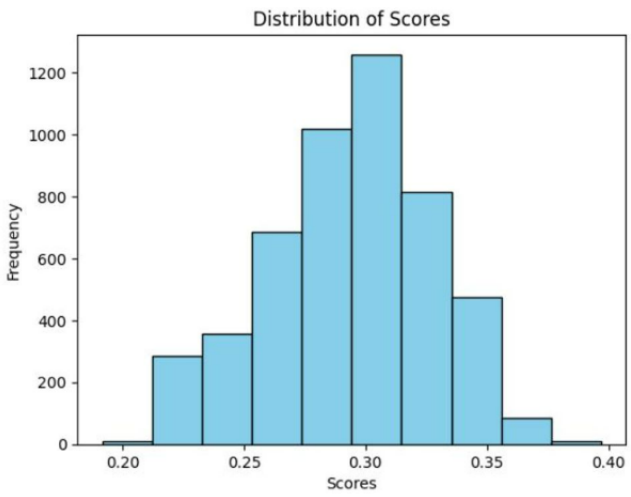

Рис. 4.12: Распределение CLIPscore на наборе данных cartoons

Теперь рассмотрим домен анимационных изображений. Описания, выдаваемые моделью выглядят более чем приемлемо, более того, часто в ответах фигурируют известные анимационные персонажи (однако только при условии, что персонаж, во первых, не появился совсем недавно и, во вторых, не является забытым)

Примеры подобных описаний можно увидеть на Рисунках 4.13-4.15. Кроме того, для большей наглядности, почему была выбрана именно модель BLIP, были приведены примеры ответов других моделей для сравнения. Можно заметить, что базовые модели работают нестабильно, а также часто упираются в самоповторы. Русунок 4.14 яркий тому пример; GIT-Large показал достаточно хорошее качество, однако в его предсказаниях встречаются артефакты + заметна тавтология. BLIP-Large напротив показал не только высокое качество описаний, но и лучшую стабильность.

Для этого набора данных было также построено распределение метрики CLIPscore (Рисунок 4.12), оно достаточно похоже на предыдущее за исключением того, что кол-во картинок с худшим значением метрики немного больше. Был также проведён повторный анализ изображений, которые имеют маленький CLIPscore (pdf файл). Можно заметить некоторые неточности, однако в рамках нашей задачи подобные описания приемлемы.

Однако стоит сделать несколько замечаний. Модель не всегда указывает, что перед пользователем именно анимационное изображение. Из-за этого теряется достаточно большое кол-во информации. Например, трагедия в мультике и в жизни - далеко не одно и тоже, важно данные понятия различать. Кроме того, модель по определению возраста и пола у людей не в состоянии корректно оценить мультяшных персонажей. Поэтому было принято решение обучить или найти отдельный **классификатор Анимационных изображений**

для решения данной подзадачи.

Рис. 4.13

BLIP-Large: a close up of a person holding a gun in a room.

BLIP-Base: a man in a cloak holding a gun

GIT-Large: 0] is a japanese manga and anime character from the anime anime series. he is a japanese

Рис. 4.14

BLIP-Large: venom is a character in the venomverse series.

BLIP-Base: venom venom venom ... (15 раз слово venom)

GIT-Large: venom art print featuring the digital art venom by [unused0]

Рис. 4.15

BLIP-Large: sonic the hedge running in a game.

BLIP-Base: sonic running through the jungle

GIT-Large: sonic the hedgehog game wallpapers

Третий домен покрывает отдельная модель OCR.

С четвёртым пунктом всё не так однозначно, смотря на портрет, люди в первую очередь хотят видеть эмоции человека, вырожение его лица, детали. Модели описывают их в слишком общем ключе, хотелось бы больше конкретики с фокусом на эмоции, на Рисунках 4.16-4.18 предоставлены некоторые примеры. В данном домене BLIP-Large снова показал себя лучше остальных моделей: У легковесных моделей есть самоповторы; У GIT-Large неудобные специальные символы (и достаточно скучные описания)

Рис. 4.16

BLIP-Large: there is a man in a black coat standing on a city street.

BLIP-Base: a man with a black coat and a black coat

GIT-Large:0]is a spanish model who is best known for his role as [unused0] in

Рис. 4.17

BLIP-Large: there is a woman with a white dress posing in a field.

BLIP-Base: a woman with long hair standing in a field

GIT-Large: portrait of a woman in a field

Рис. 4.18

BLIP-Large: a painting of a woman with long hair and a white top.

BLIP-Base: a painting of a woman with long hair

GIT-Large: portrait of a young woman

По аналогии, посмотрим на распределение CLIPscore на двух датасетах: ”People Clothing Segmentation” и ”Portrait dataset for training GANs”. (Рисунки 4.19, 4.20)

- People Clothing Segmentation (pdf файл картинок с худшим значением метрики) - содержит все недостатки домена портретов, которые были упомянуты выше.
- Portrait dataset for training GANs (pdf файл картинок с худшим значением метрики) - содержит картины и достаточно абстрактные изображения. Такой набор данных показал, что модель справляется с обрезанными объектами (часть не вошла в кадр), объектами неправильной формы (картины художников) и тд. Даже зашумлённые изображения в большинстве своём имеют валидные описания. Периодически ошибки происходят, но часто и человек в них может допустить оплошность. Однако есть изображения, на которых модель выдаёт плохое описание - **картинки, которые почти полностью состоят из шума**.

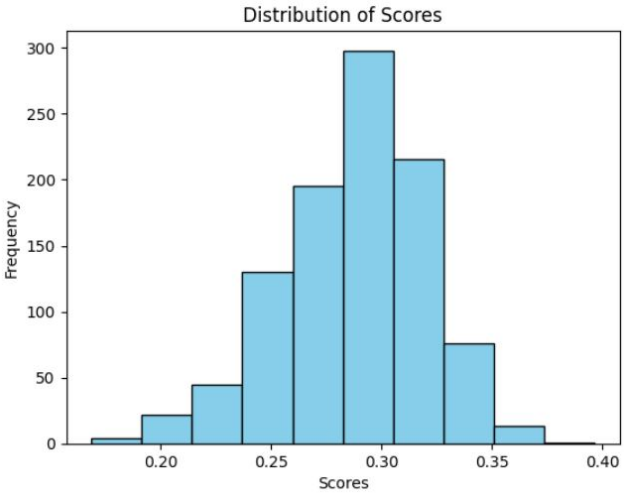

Рис. 4.19: Распределение CLIPscore на наборе данных ”People Clothing Segmentation”

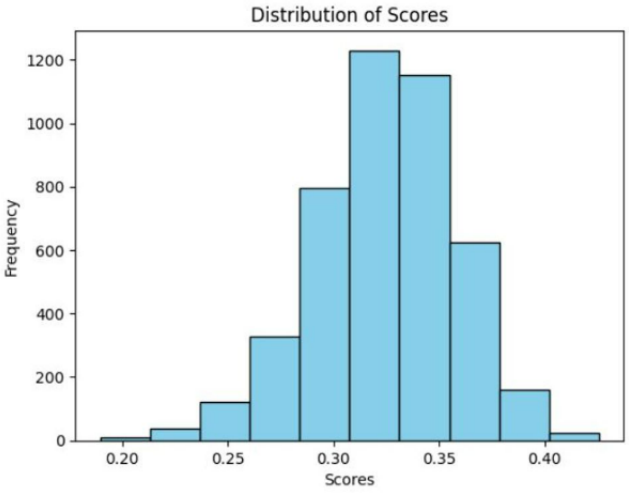

Рис. 4.20: Распределение CLIPscore на наборе данных ”Portrait dataset for training GANs”

Отдельной задачей ялвяется последний класс: для описания мемов необходимо понимание большого кол-ва контекста, которого у модели нет, даже оценить качество не так просто. CLIPscore использовать на данном домене некорректно, так как он тоже не в состоянии уловить разные уровни иронии данного класса картинок. Решение проблемы - создать отдельный классификатор для разделения картинки на классы ”мем” и ”не мем”

4.2.4 Работа над недостатками модели BLIP

Проанализировав домены выше, мы обнаружили следующие уязвимости модели BLIP:

1. Возникают артефакты arafed, arafee и тд
2. Теряется информация о типе изображения (анимационное изображение /реалистичное изображение)
4. Уязвимость к полностью зашумлённым изображениям
5. Неспособность адекватно описывать мемы

4.2.5 Проблема: arafed, arafee

Данные артефакти очень частов возникают в ответах модели. Причём на данных двух словах (вернее набора токенов ##ra-##fed и ##ra-##fee) данные артефакты не заканчиваются.

После запуска модели на датасете COCO были выявлены слова, которые можно отнести к данному артефакту: araffle, arafflent, arafiance, arafiancence, araffe, araffatured, araffiction,

araflane, arafal, araful, araffite, arafoasta, arafy, araffler, arafitoo, arafect, araffaged, araffasher, arafficial, arafft, araflanes, araffature, arafflex, araffion, araffled, arafiguard, arafak, araffes, arafalaia, arafly, arafexor, araffy, araffactor, arafovs, arafor, araffa, arafiane, araffair, arafied, araffed, arafectible, arafecth, araff, araffactors, arafson

Если посмотреть на частоту их встречаемости в наборе данных coco (Рисунок 4.21), можно сделать вывод, что кол-во слов-артефактов, которые регулярно встречаются не так много.

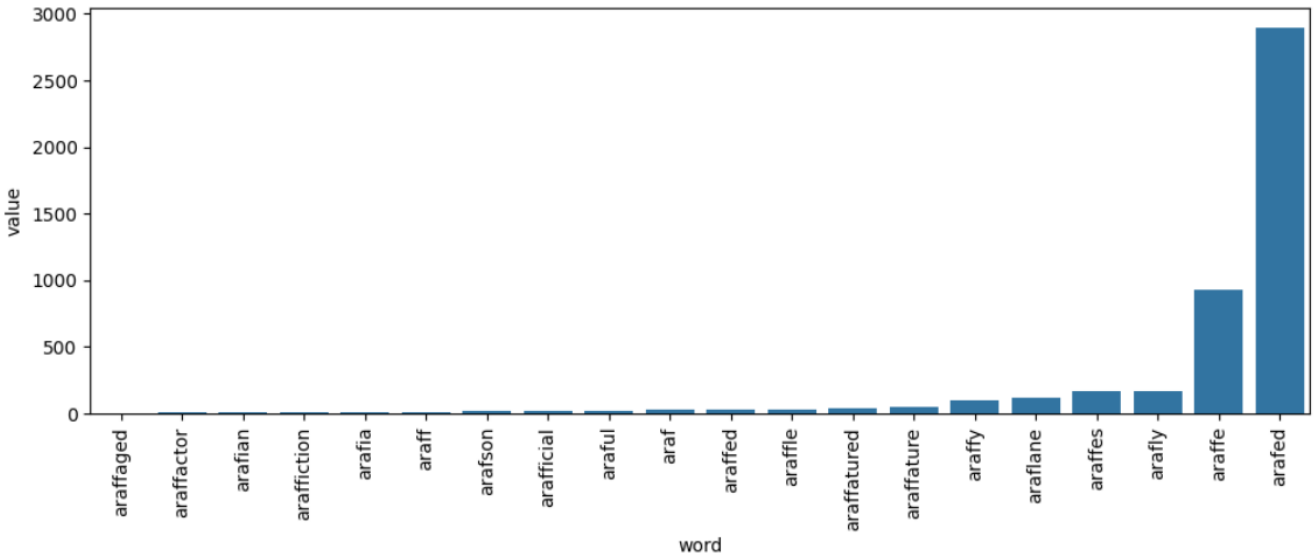

Рис. 4.21: Распределение артефактов araf*

Можно попробовать убрать данные слова из описаний и потом сравнить CLIPscore для очищенных и не очищенных данных. Результаты можно увидеть на Рисунках 4.22, 4.23. Можно заметить, что среднее значение метрики осталось тем же, тоже самое можно сказать и про правую часть графика. Однако после фильтрации левая часть графика стала сильнее прижиматься к среднему значению. Подобные наблюдения говорят о том, что результат на хороших изображениях не изменился, при этом кол-во изображений с плохим значением метрики уменьшилось.

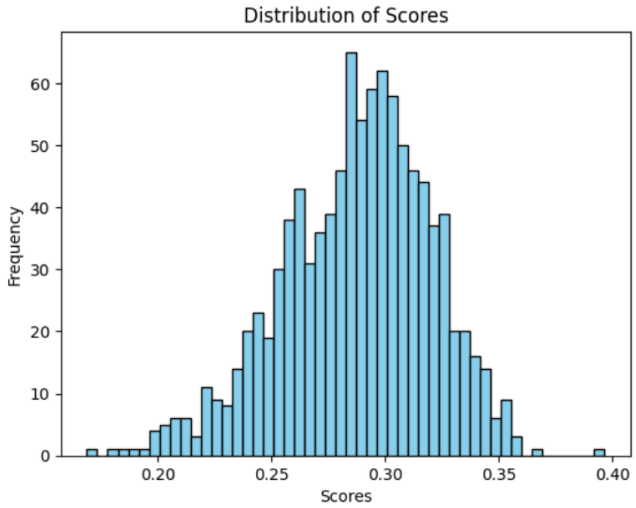

Рис. 4.22: До фильтрации

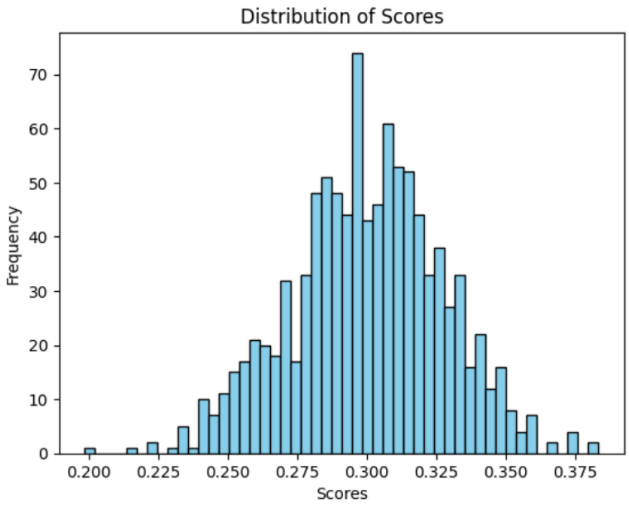

Рис. 4.23: После фильтрации

Однако данное решение проблемы не является удовлетворительным в контексте нашей задачи, так как, убрав одно слово, мы часто рушим грамматическую и логическую целостность предложений:

- araffe on a boat with a mountain in the background
- araffe walking in front of a green door with a red umbrella
- araffes of people are walking around a train station with a clock

Хочется подметить, что модель BLIP умеет работать с условной генерацией, когда на вход подаётся не только изображение, но и текстовый промпт. Выяснилось, что данные артефакты полностью исчезают, если предоставить модели любой текстовый промпт.

Было решено каждый раз подавать модели фразу: ”a photography of”

4.2.6 Проблема: анимационное изображение/реалистичное изображение

Данная задача не выглядит сильно трудной для задачи классификации, так как мультяшные и реалистичные картинки имеют явно разную структуру.

Было решено попробовать zero shot решение, основанное на CLIP: считаем 3 значения:

- photo_clip = CLIP(image, “It is a photo”)
- image_clip = CLIP(image, “It is a cartoon”)
- class = sign(image_clip - photo_clip)

По итогу, мы делим картинки на 3 класса:

- class = 1: мультяшная картинка
- class = -1: реалистичное фото
- class = 0: не удалось классифицировать

Однако возникает вопрос, как подобрать такие фразы: “It is a photo” и “It is a cartoon”, чтобы классификация была хорошая. За метрику был взят accuracy, классы по размерам равны. train/val = 0.75/0.25, где train - выборка на которой мы подбираем оптимальную пару, а val - выборка для проверки на переобучение. Вдальнейшем будут отображаться результаты только валидации.

Для поиска подходящих фраз применялась большая языковая модель YandexGPT. Ей сначала подавался промпт с просьбой выдать все слова, которые ассоциируются с мультяшными картинками, а потом с реалистичными изображениями. После фильтрации ответа, собранный набор фраз выглядел, как в Таблице 4.3

Cartoons	**Photos**
It is an animation	It is a photo
It is a comic book	It is a photograph
It is a toon	It is a photo, it is a photograph
It is an animated series	It is a picture
It is a cartoon strip	It is an image
It is an anime	It is a snapshot
They are cartoon characters	It is a portrait
It is cartoony image	It is a print
It is a graphic novel	It is a photography
It is a cartoonish image	It is a visual photo
It is an anime image	It is a shot
It is a cartoon or anime image	It is a picturesque photo
Cartoon image	It is a photo not a cartoon image
It is a cartoon image, not a photo	It is a realistic photo
	It is a realistic image
	Realistic photo

Таблица 4.3: Comparison of Cartoons and Photos

В качестве датасетов были взяты: MSCOCO - данный набор данных содержит большое кол-во повседневных изображений; cartoons - идеально подходит для текущей задачи, так как содержит большое кол-во кадров из большого кол-ва различных произведений

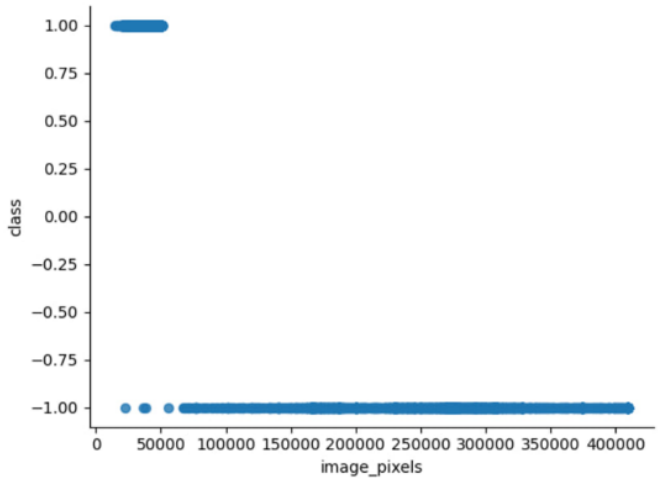

Рис. 4.24: Распределение кол-ва пикселей в изображении: 1: мультик, -1: фото

Так как картинки сильно различались по разрешению (Рисунок 4.24), было решено свести все изображения к формату (3, 225, 225). Это было сделано, чтобы свести любое стороннее влияние на ответ к минимуму.

Дальнейшим шагом было перебрать все пары промптов (это мультик / это фото) и найти самую оптимальную с точки зрения accuracy. Подробные результаты можно увидеть по ссылке. Легко заметить, что от выбора промпта результат модели кординально меняется. Худшие результаты показали пары:

- с наличием отрицания: (It's a cartoonish image/it's a photo, not a cartoonish image)
- с неоднозначными словами: (It's a cartoonish image/it's a shot)

У данных пар accuracy ≈ 0.5. Лучшая же пара из найденных: (It's an animated series/It's a photo, It's a photograph) показали accuracy > 0.98. Данный эксперимент проводился на версии CLIP по умолчанию: ViT-B/32

Было решено повторить эксперимент но на различных моделях, лучший accuracy, а также комбинацию промптов можно посмотреть в таблице

Наилучший результат показала модель RN50 с accuracy ≈ 0.994888, найденная пара промптов: (It is a cartoon or anime image/It is a photo, it is a photograph)

На результаты работы данной модели можно посмотреть в следующих pdf:

- Примеры классификации
- Примеры неправильной классификации (только 160 изображений из 17000). Хотелось бы подметить, что часть этих изображений очень спорно размечена, поэтому ошибочных классификаций ещё меньше

Таблица 4.4: Comparison of Model Performances and Sizes

Model	Image Phrase - Photo Phrase	Accuracy	Model Size
RN50x64	“It is an anime - It is a picturesque photo”	0.95944	1.26G
ViT-L/14	“It is an animation - It is a photo”	0.985332	890M
ViT-L/14@336px	“It is an animation - It is a photo”	0.989332	891M
RN50x4	“mean cartoons - mean real”	0.990555	402M
ViT-B/32	“mean cartoons - mean real”	0.990666	338M
RN101	“It is a cartoon or anime image - Realistic photo”	0.992888	278M
RN50x16	“It is an animation - It is a visual photo”	0.993444	630M
ViT-B/16	“It is a cartoon or anime image - It is a photo”	0.993444	335M
RN50	“It is a cartoon or anime image - It is a photo...”	0.994888	244M

4.2.7 Проблема: шумные изображения

Иследуя набор данных portraits, было замечено, что некоторые очень шумные изображения выдают описания вводящие в заблуждение (Рисунки 4.25, 4.26).

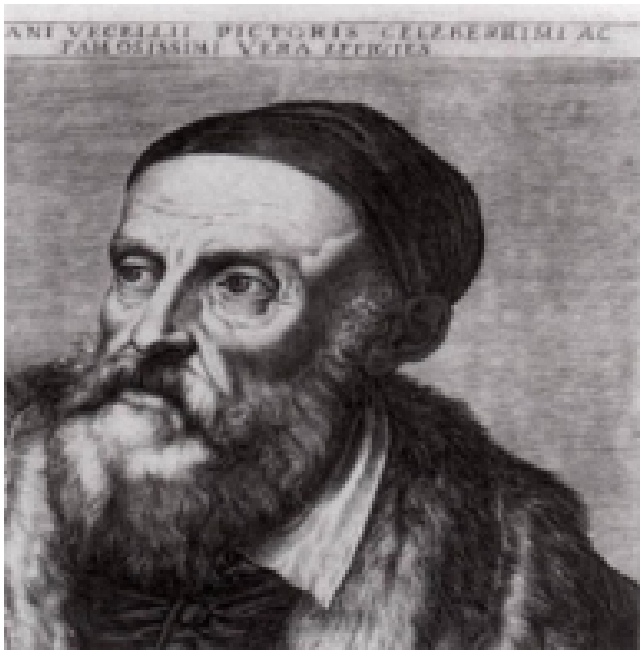

Рис. 4.25: Слегка шумное изображение

BLIP-Large: an old black and white photo of a man with a beard.

Рис. 4.26: Сильно зашумлённое изображение

BLIP-Large: there is a man, riding a skateboard on a ramp.

Первое предположение было связано с тем, что модель плохо работает с затемнёнными изображениями, так как они содержат мало информации, однако после нескольких экспериментов выяснилось, что модель справляется с подобного рода изображениями иногда даже лучше, чем человек (Рисунки 4.27, 4.28)

Рис. 4.27: Тёмное изображение

BLIP-Large: dimly lit room with a person standing in the dark.

Рис. 4.28: Осветлённое изображение

BLIP-Large: dimly lit room with a person standing in the dark.

Прблема оказалась в другом, так как в датасете нет изображений, на которых не было ничего содержательного, модель стабильно выдаёт на них странные описания. Было выдвинуто предположение, что данные изображения можно отлавливать, обращая внимание на уверенность модели. Если на них нет содержательной информации, модель вероятнее всего не будет уверена в своём ответе, а значит распределения токенов во время генерации, вероятно, будет ближе к равномерному. Для оценки уверенности модели было решено воспользоваться перплексией:

1. считается кросс энтропию на распределении выходных токенов
2. берётся экспонента от полученного результата

Чем ближе данная метрика к 1, тем более уверена модель в своём ответе.

Было решено построить распределения данной метрики на датасетах: cartoons, coco и noise (новый). Датасет noise содержит:

- Гауссовский шум
- Сплошной цвет (немного зашумлённый)
- Выборочные фотографии из датасета portraits, на которых ничего нет, кроме шума

Итоговое распределение изображено на Рисунке 4.29

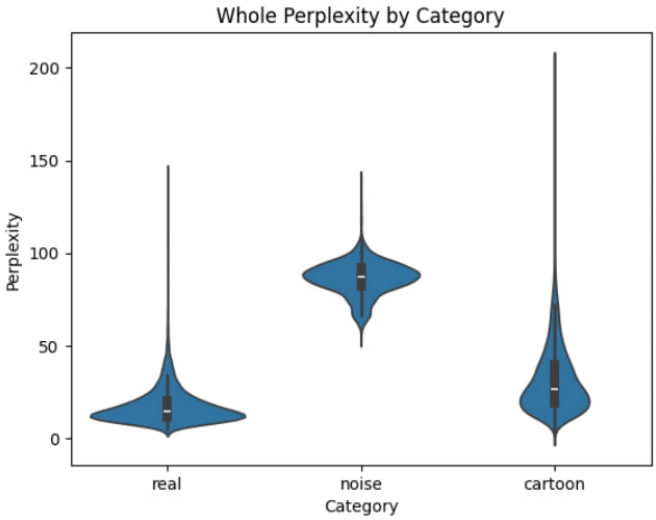

Рис. 4.29: Распределение уверенности модели

Было принято решение взять perplexity=80 за границу, прочие изображение для большей стабильности описываться не будут.
Конечно данный порог зависит от задачи, поэтому у пользователя будет возможность поменять его в случае необходимости.

4.3 Определение возраста и пола человека по изображению

4.3.1 Анализ наборов данных

Перед началом сравнения и последующей работой с определенными моделями необходимо изучить набор данных, на котором эта модель будет обучаться и применяться. Изучение набора данных позволяет оценить его качество, разнообразие объектов, содержащихся в нем, и достаточность для поставленных целей задачи. Это также помогает выявить возможные проблемы, связанные с качеством или разметкой данных, которые могут повлиять на работу модели. Анализ набора данных обеспечивает более точную и объективную оценку производительности модели, а также позволяет предпринять необходимые шаги по его предварительной обработке или аугментации для повышения эффективности обучения и обобщения.

IMDB-clean

В дальнейших исследованиях планируется использовать улучшенный набор данных **IMDB-clean** [18], который был получен путем очистки шумных данных из оригинального набора IMDB-WIKI. Для очистки авторы применяли метод ограниченной кластеризации,

что позволило создать новый эталон для оценки возраста в условиях повседневной жизни. Кроме того, аннотации в IMDB-clean расширяют возможности его использования для других задач, таких как классификация по полу и распознавание/верификация лиц.

Данный набор состоит из 286000 изображений с размеченными данными. Выборка была разделена на обучающий, валидационный и тестовый наборы данных в соотношении 65%, 15% и 20% соответственно.

На Рисунках 4.30-4.31 изображены данные о содержании набора IMDB-clean.

Так, на Рисунке 4.30 график возрастного распределения напоминает форму колокола, но с более резким уменьшением количества наблюдений после определенного возраста, что указывает на нормальное распределение с длинным левым хвостом. Большая часть данных сосредоточена в возрастной группе от примерно 20 до 50 лет. Вершина графика находится около 30-40 лет, что может отражать большую доступность фотографий людей в этом возрастном диапазоне. Стоит сразу же отметить, что присутствует резкое снижение количества данных для возрастов старше 50 лет, а также относительно небольшое количество данных доступно для возрастов менее 20 лет.

На рисунке 4.31 график показывает, что количество мужчин (обозначено как "M") примерно равно количеству женщин (обозначено как "F") в наборе данных. Это свидетельствует о хорошем балансе пеолов в выборке.

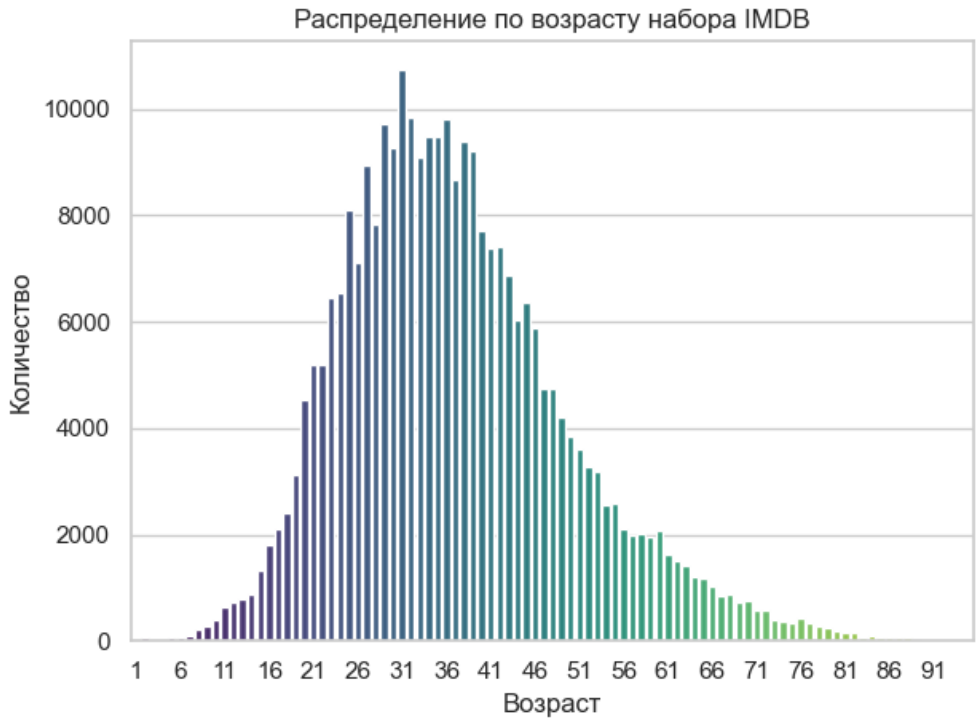

Рис. 4.30: Распределение возрастов в наборе IMDB-clean.

Итак, набор данных IMDB-clean характеризуется хорошим балансом по половому при-

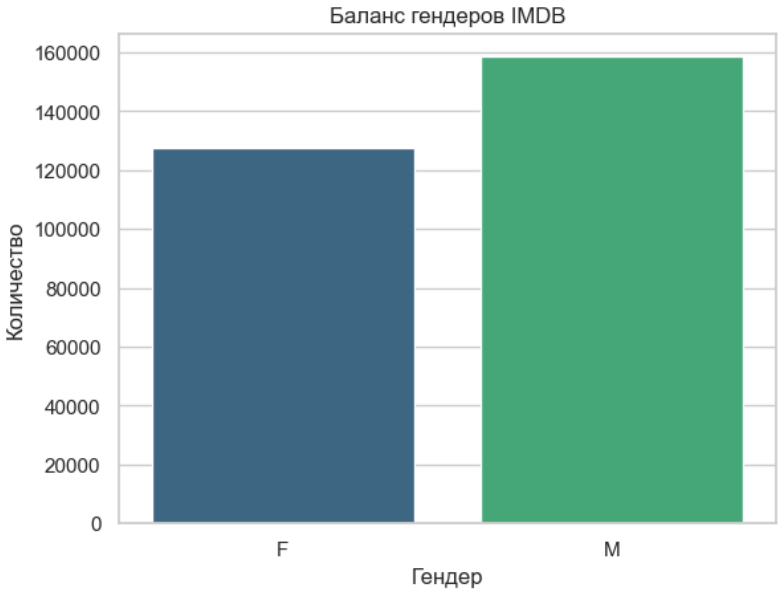

Рис. 4.31: Распределение полов в наборе IMDB-clean.

знаку, что делает его подходящим для задач, связанных с классификацией полов, а также распознаванием и верификацией лиц. Возрастное распределение представлено достаточно широко, хотя и с уклоном в более молодую аудиторию. Небольшое представление очень молодых и пожилых возрастов потребует дополнительной коррекции и дополнения данных для улучшения производительности моделей в этих возрастных группах.

UTKFace

В UTKFace всего содержится 24106 размеченных изображений.

Распределение по возрасту в наборе UTKFace на Рисунке 4.32 отличается значительной концентрацией данных в младшей возрастной группе, с несколькими пиками (около 1 года и 20-30 лет). Это отличается от распределения IMDB, где пик находился в возрастной группе 30-40 лет. Данное распределение включает широкий диапазон возрастов от младенцев до старших людей. Наличие значительного количества данных для возрастной группы младше 10 лет отличает UTKFace от IMDB.

На рисунке 4.33 график показывает близкое к равному количество мужчин и женщин, что является хорошим показателем баланса полов и схоже с наблюдаемым в IMDB.

UTKFace представляет более широкий и разнообразный диапазон возрастов, особенно включая очень молодые возраста, что делает его полезным для приложений. В то время как IMDB лучше справляется с взрослой аудиторией, что может быть связано с характером источника данных, большинство которых составляют профессиональные актёры. Оба набора данных могут дополнять друг друга в задаче для оценки возраста и классификации пола за

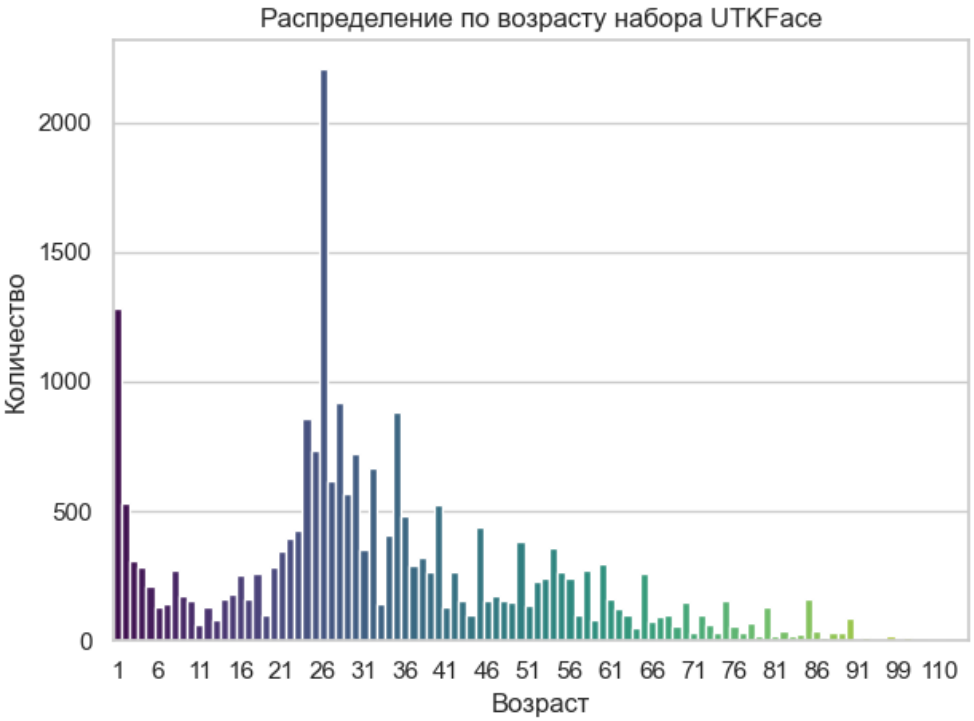

Рис. 4.32: Распределение возрастов в наборе UTKFace.

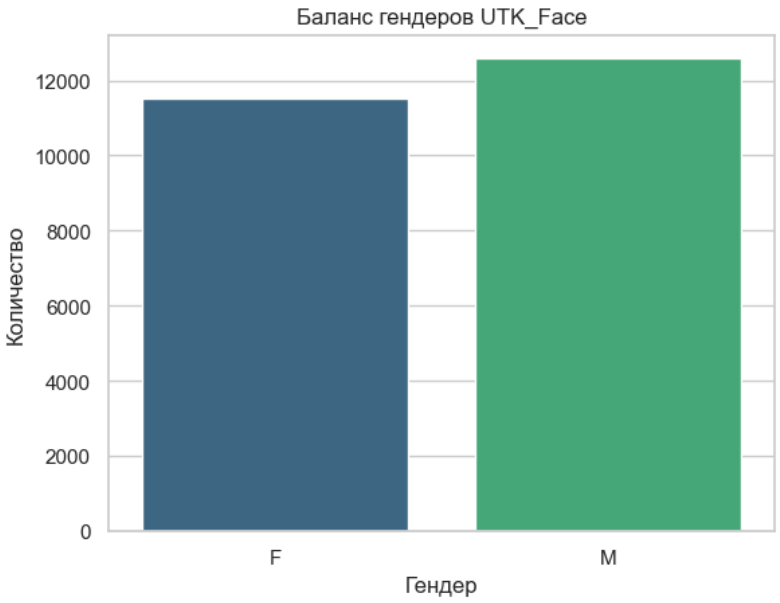

Рис. 4.33: Распределение полов в наборе IMDB-clean.

счёт хорошего баланса между полами.

Сравнение моделей

В рамках исследования ключевым этапом является анализ и сравнение различных моделей. Выбор наиболее подходящей модели представляет собой важную задачу, от решения которой зависит не только точность и надежность предсказаний, но и эффективность внедрения решения в реальные приложения и системы. В данном разделе представлен первоначальный обзор выбранных моделей, оценка их производительности и анализ ключевых метрик.

4.3.2 Модели детекции

Для объективного сравнения моделей детекции объектов используется метрика Intersection over Union (IoU) 1. Согласно данной метрике, объект считается успешно обнаруженным, если IoU между предсказанными координатами и реальными аннотациями не менее 0.5. Эффективность и точность моделей тестировались на подвыборке из датасета IMDB-cleaned, содержащей 56,087 изображений. На этих изображениях аннотированы около 180,000 лиц, что указывает на присутствие нескольких лиц на одних и тех же изображениях.

Таблица 4.5: Сопоставление предобученных моделей детекции лица на тестовой подвыборке IMDB-clean (56087 изображений).

Model	Faces detected	Labels missed	#params(M)	Average time (ms)
Haar Cascade	179664	2992	-	38.2
MTCNN	128431	105	2.31	155.6
YOLO v5	184318	0	1.73	40.94
YOLO v8	178415	23	3.01	13.52

Анализ данных в Таблице 4.5 показывает следующие результаты:

- модель **Haar Cascade** обнаружила примерно нужное количество объектов, но при этом упустила почти три тысячи нужных лиц (из 56 тыс.), что указывает на неточность в определении. Иными словами, в детекции присутствует большое количество "ложных срабатываний".
- **MTCNN** показала более качественный результат по детекции необходимых объектов - всего 105 пропущенных аннотированных лиц, но при этом имеет около 50000 пропусков в обнаружении лиц. Плюс ко всему, данная модель имеет самое большое время обрабатывания.

- по сравнению с моделями выше **YOLO v5** показала отличный результат: нет пропусков необходимых объектов, при этом сохраняется высокая скорость вычислений. Все же стоит отметить небольшой излишек в детекции, то есть у данной модели, хоть и не в большом количестве, но имеются ложные срабатывания.

- **YOLO v8** выделяется среди остальных благодаря высокой точности и скорости, что делает её предпочтительным выбором для задач детекции лиц в условиях наличия большого количества объектов на изображении. В дальнейшем будет применяться данная модель детекции.

4.3.3 Модели определения возраста и пола

Перед внедрением модели в сервис важно выбрать наиболее подходящий вариант, основываясь на сравнительном анализе. Рассмотрим модели DEX и MiVOLO, результаты тестирования которых на подвыборках IMDB-clean (56087 изображений) и UTKFace (3553 изображений) представлены в Таблицах 4.6-4.7. Стоит подчеркнуть, что модель MiVOLO демонстрирует высокую универсальность, так как способна анализировать как только лицо, так и в комбинации с телом.

Таблица 4.6: Сопоставление предобученных моделей на тестовой подвыборке IMDB-clean (56087 изображений).

Model	Age (MAE)	Gender (Acc)	#params (M)	Average time (ms)
DEX	9.18	85.2	134.7 + 134.2	10 + 9.5
MiVOLO (face-only)	6.46	99.3	25.86	5.63
MiVOLO (face-and-body)	5.5	99.52	27.43	10.3

Таблица 4.7: Сопоставление предобученных моделей на тестовой подвыборке UTKFace (3553 изображений).

Model	Age (MAE)	Gender (Acc)	#params (M)	Average time (ms)
DEX	7.16	78.9	134.7 + 134.2	7.4 + 5.93
MiVOLO (face-only)	5.02	97.3	25.86	3.28
MiVOLO (face-and-body)	4.6	97.63	27.43	6.2

DEX показывает средний абсолютный ошибку (MAE) по возрасту на уровне 9.18 и точность определения пола в 85.2% при значительном количестве параметров (268.9 миллионов) и времени обработки 19.5 мс. В то время как MiVOLO в режиме "только лицо"достигает MAE в 6.46 и точности 99.3% с меньшим числом параметров (25.86 миллионов) и более быстрой обработкой (5.63 мс). Расширенная версия MiVOLO, работающая с лицом и телом, улучшает показатели до MAE 5.5 и точности 99.52%, при этом время обработки возрастает до 10.3 мс и количество параметров составляет 27.43 миллиона.

Исходя из этих данных, MiVOLO является более предпочтительной моделью для интеграции в сервис, где требуется высокая точность определения возраста и пола. Так как вариация с лицом-телом в качестве входных данных несильно больше, но в то же время потенциально качественнее, далее будет рассматриваться именно она. Особенно примечательной является способность MiVOLO работать как с лицом, так и с телом, что позволяет учитывать больше контекста при анализе изображений, что может быть критично для задач, требующих повышенной точности. DEX, несмотря на хорошие показатели, уступает по всем параметрам и требует более значительных вычислительных ресурсов.

4.3.4 Индивидуальная реализация.

Для обеспечения всестороннего анализа предложенной методики, необходимо также рассмотреть и реализовать собственный подход к решению поставленной задачи. Это позволит не только подтвердить эффективность текущей модели, но и выявить потенциальные направления для дальнейшего улучшения и адаптации метода.

Цель данной задачи состоит в оценке возраста на основе изображений лиц. Предлагаемый подход включает в себя использование предобученной модели CLIP (Contrastive Language–Image Pre-training) для извлечения векторных представлений из изображений. Эти эмбеддинги предоставляют набор визуальных признаков, которые отражают такие признаки, как физические характеристики лиц.

После получения эмбеддингов будет использоваться многослойный перцептрон (MLP), который представляет собой архитектуру нейронной сети, специализированную для решения регрессионных задач. Относительно небольшой MLP, в котором несколько линейных слоев и функций активация, будет обучен на основе эмбеддингов для предсказания возраста, что позволит модели выявлять зависимости между визуальными признаками и возрастом человека.

После обучения вышеуказанной модели на наборе данных UTKFace получились результаты, приведенные в Таблице 4.8

Таблица 4.8: Результаты CLIP-based модели.

Train Dataset	Test Datset	Age (MAE)	Average time (ms)
UTKFace	UTKFace	4.2	11.06
UTKFace	IMDB-clean	14.23	15.17

Исходя из представленных результатов CLIP-based модели, можно сделать выводов: для обучающего и тестирующего набора данных UTKFace модель показала относительно низкую среднюю абсолютную ошибку (MAE) в 4.2 года и время ответа 11.06 мс, что говорит

о хорошей производительности на этом конкретном датасете, но при тестировании на другом наборе данных, IMDB-clean, ошибка возрастает до 14.23 года, а время ответа увеличивается до 15.17 мс, указывая на необхоимость в дополнительном обучении модели под IMDB-clean.

В контексте сравнения с предыдущей моделью MiVOLO, которая показала лучшие результаты по времени на аналогичных задачах и качеству на IMDB-clean, увеличение сложности MLP в текущей модели не кажется целесообразным. MiVOLO не только справляется с задачами быстрее, но и демонстрирует хорошую точность, что делает её более предпочтительным выбором.

Таким образом, вместо дальнейшего усложнения архитектуры CLIP-based MLP, были сосредоточены усилия на улучшении и оптимизации модели MiVOLO.

4.3.5 MiVOLO

Перед подробным изучением модели MiVOLO требуется тщательное рассмотрение основных аспектов функционирования ее архитектуры.

На Рисунке 3.18 показана архитектура модели MiVOLO, которая использует двухступенчатую структуру для обработки и улучшения признаков как лица, так и тела. Рассмотрим каждый элемент и его функции в этой архитектуре:

1 **Two-stage VOLO [35] architecture** - это двухступенчатая архитектура представляет собой основу модели, в которой данные проходят через два этапа обработки для улучшения признаков и классификации. Сначала исходные данные обрабатываются для извлечения базовых признаков, после чего следует дополнительная обработка для уточнения и объединения признаков:

- На первом этапе архитектуры VOLO основная цель - захватить спектр признаков из входных данных (изображений лиц и тел в контексте MiVOLO). На этом этапе обычно используются плотно связанные слои или серия сверточных слоев, которые помогают обнаружить общие признаки, такие как края, цвета и базовые формы. Здесь ключевым моментом является подготовка основы признаков, которые можно далее уточнить.
- На втором этапе архитектура VOLO отличается. После начального извлечения признаков архитектура использует более продвинутый набор механизмов для их уточнения. Это включает механизм внимания Outlook и Слияние признаков.

  В отличие от традиционных механизмов внимания, которые работают в пределах ограниченного локального окрестности, внимание Outlook расширяет рецептивное

поле более глобально и адаптивно. Этот механизм позволяет модели сосредоточиться на более релевантных признаках на большей площади изображения, что имеет решающее значение для интеграции контекстуальной информации из различных частей изображения.

После улучшения признаков лица и тела архитектура VOLO объединяет эти признаки. Это слияние — не просто конкатенация, а интеграция, учитывающая взаимозависимости и дополнительный характер извлеченных из разных регионов признаков. Это особенно важно в задачах, таких как определение возраста и пола, где признаки лица и тела могут предоставлять дополнительную информацию.

Все это отображено на Рисунке 4.34

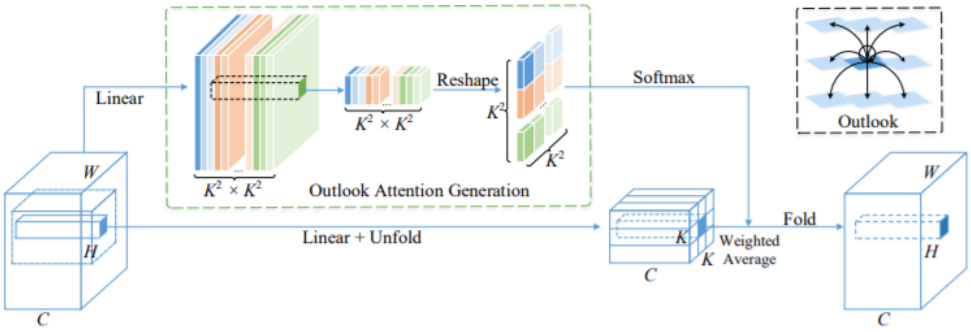

Рис. 4.34: Архитектура VOLO.

2 **Fused Joint Representation** - слияние в единую представленность. Это позволяет модели учитывать информацию как о лице, так и о теле одновременно, улучшая точность задач определения возраста и пола.

3 **Heads (Model Outputs)** - Конечные "головы"модели, где одна используется для расчета потерь по возрасту (Weighted MSE age loss), а другая — для определения пола (Binary Cross-Entropy gender loss). Эти выходы позволяют модели выполнять задачи регрессии и классификации соответственно.

4 **Feature Enhancer Module** - модуль, служащий для дополнительного улучшения признаков, извлеченных из изображений лица и тела. Он обеспечивает более глубокую обработку данных, что повышает их релевантность и полезность для конечной задачи. Данный модуль в свою очередь включает в себя такие компоненты, как:

- Body-to-Face и Face-to-Body Cross-Attention - модули кросс-аттенции позволяют признакам лица и тела взаимодействовать друг с другом. Это помогает модели по-

нять взаимосвязи между различными областями изображения, улучшая точность предсказаний по возрасту и полу.

- Concat & Norm - этап, на котором признаки лица и тела объединяются и нормализуются, что обеспечивает стабильность и улучшает обучение модели.
- MLP (Multi-Layer Perceptron) используется для дальнейшего анализа и улучшения признаков, подготавливая их к этапу конкатенации и нормализации. MLP может захватывать нелинейные зависимости между признаками.

5 **Body Patch Embedding и Face Patch Embedding**: перед тем как признаки лица и тела подаются в модуль улучшения, они разбиваются на патчи размером 8x8. Это улучшает возможности модели по локальному анализу и обработке информации.

MiVOLO использует продвинутую архитектуру с кросс-аттенцией и улучшением признаков, чтобы достичь высокой точности в задачах, связанных с анализом лиц и тел. Плюсы включают возможность модели адаптироваться к сложным и разнообразным типам данных, тогда как минусы заключаются в потенциально высокой вычислительной стоимости и сложности обучения модели, что потенциально не является проблемой в силу имеющихся предобученных весов.

4.3.6 Качество MiVOLO

Для более глубокого анализа качества модели на доступных наборах данных необходимо начать с изучения ошибок по каждой возрастной категории, учитывая имеющееся распределение данных из Рисунков 4.30 и 4.32, а также результатов из Таблиц 4.6 и 4.7.

Далее будет рассматриваться предобученная модель MiVOLO с весами, предобученными на IMDB-cleaned для объектов лицо-тело.

Рассмотрим отображенные на Рисунке 4.35 ошибки текущей модели.

На графике видно, что MAE высока для возрастной группы от 0 до 10 лет, достигая пиковых значений около 7-8. Сами по себе черты лица младенцев и маленьких детей могут значительно различаться в короткие временные промежутки, что усложняет точное предсказание возраста.

С возрастом 10 лет и до 45 лет средняя ошибка MAE уменьшается, достигая самых низких значений около 4-5 в возрастной группе 30-45 лет. Стоит вспомнить, что именно в этом промежутке лежит пик распределения возрастов.

После 45 лет наблюдается тенденция к повышению MAE, которая становится более выраженной после 60 лет. Это повышение может быть связано с увеличением вариабельности

внешности людей в более старшем возрасте, а также с меньшим количеством данных в этом возрастном промежутке.

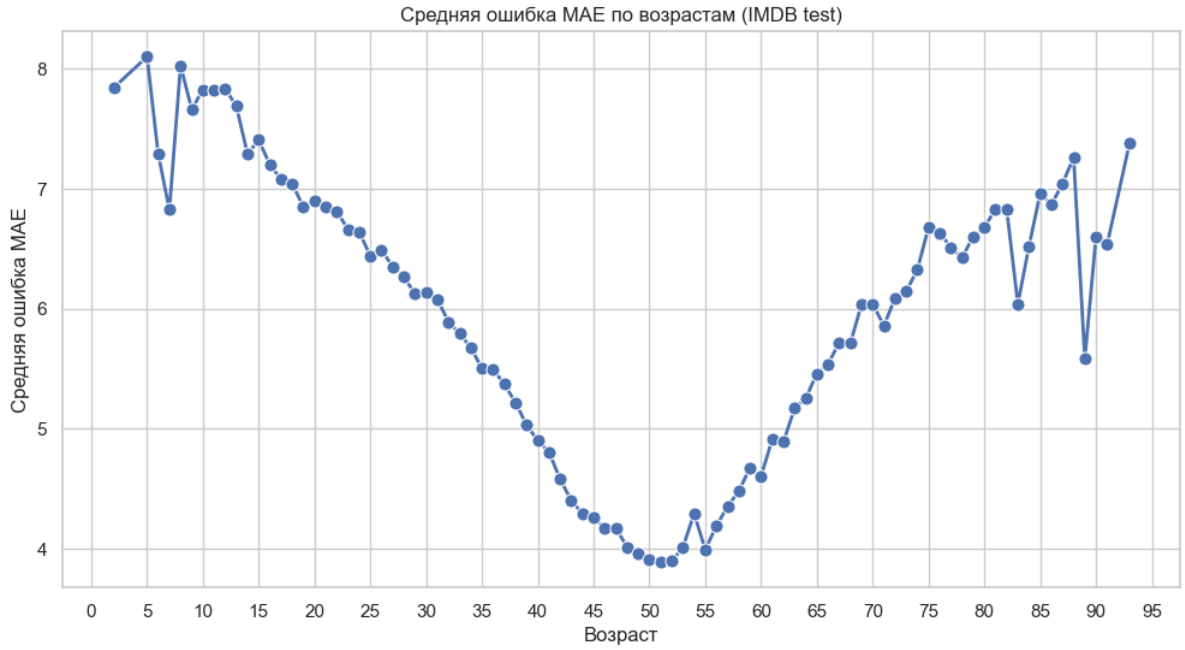

Рис. 4.35: Средняя ошибка по возрастам IMDB-clean.

Тем же способом проверим и на другом тестовом наборе - UTKFace:

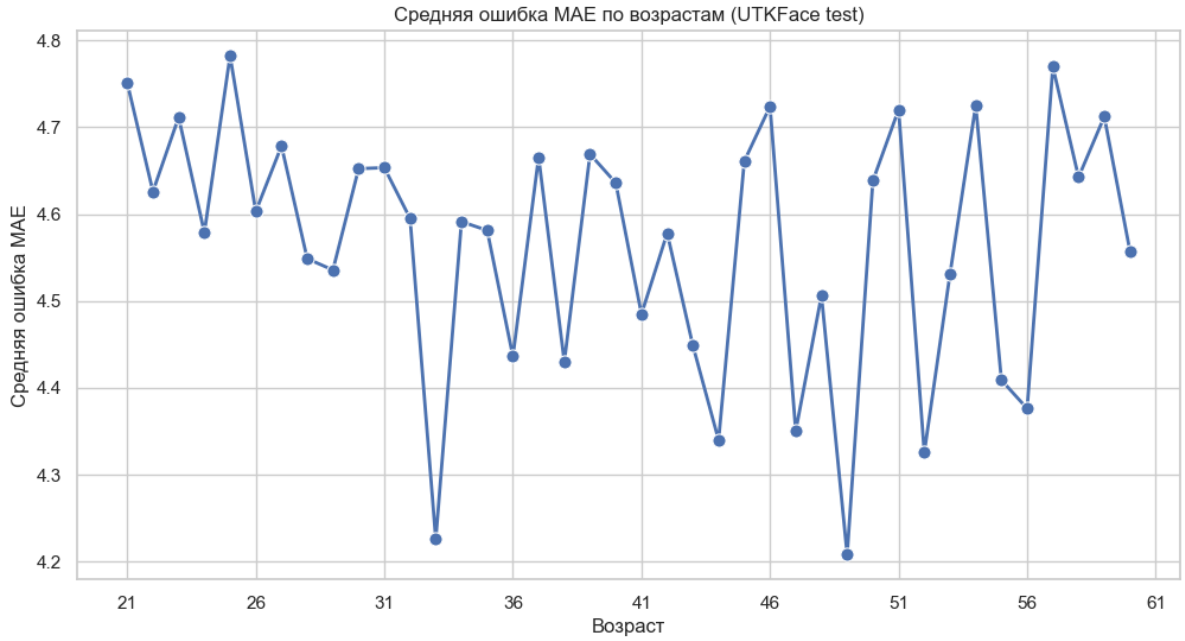

Рис. 4.36: Средняя ошибка по возрастам UTKFace.

На графике 4.36 же ошибка MAE колеблется между примерно 4.3 и 4.8, что указывает на относительно стабильное качество модели по возрастам в данном диапазоне. Хоть качество кажется неплохим, это скорее указывает на нехватку данных в данной подвыборке. И потому стоит опереться на подвыборку IMDB-clean.

График на Рисунке 4.35 очевидным образом подводит к мысли о необходимости дообучить модель на "хвостах"распределения.

4.3.7 Дообучение MiVOLO

Итак, мы пришли к необходимости дообучения модели, особенно для данных, расположенных на краях возрастного распределения.

Была использована стандартная техника дообучения (fine-tuning), обучая модель на расширенном наборе данных, полученном путем слияния подвыборок IMDB-clean и UTKFace, с особым упором на крайние возрастные группы. Это позволит улучшить способность модели к обобщению и точности в "трудных"случаях.

После проведения корректировок и дообучения модели, изменения в характеристиках кривой средней абсолютной ошибки (MAE) можно наблюдать на графиках, изображенных на Рисунках 4.37 и 4.38.

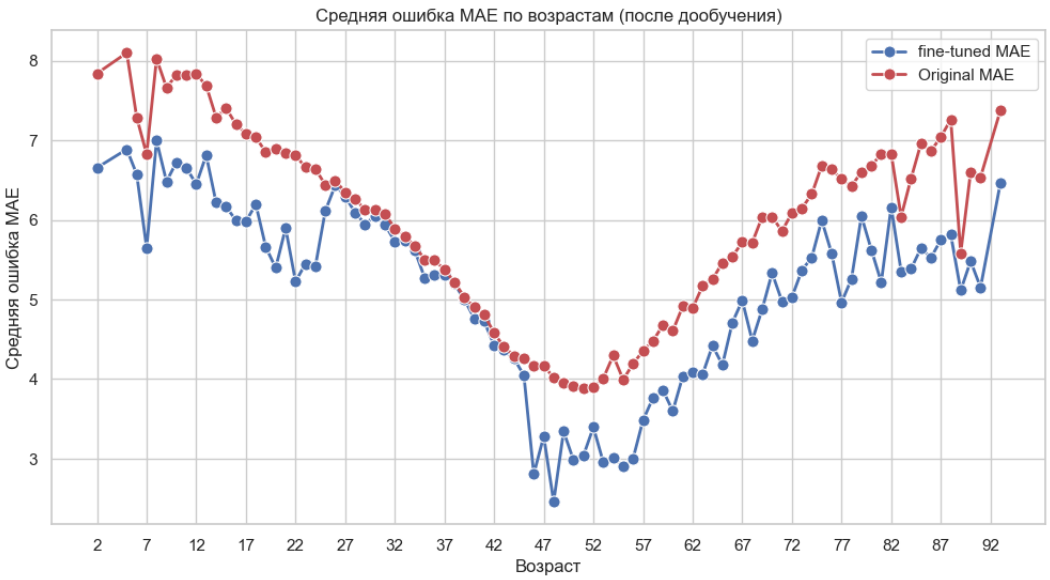

Рис. 4.37: Средняя ошибка MAE IMDB-clean после дообучения.

4.3.8 Результат

Итак, итоговый результат выбранной модели можно наблюдать в таблице 4.9. А также пример взаимодействия модели MiVOLO с моделью детекции объектов YOLO v8 представлен на Рисунке 4.39.

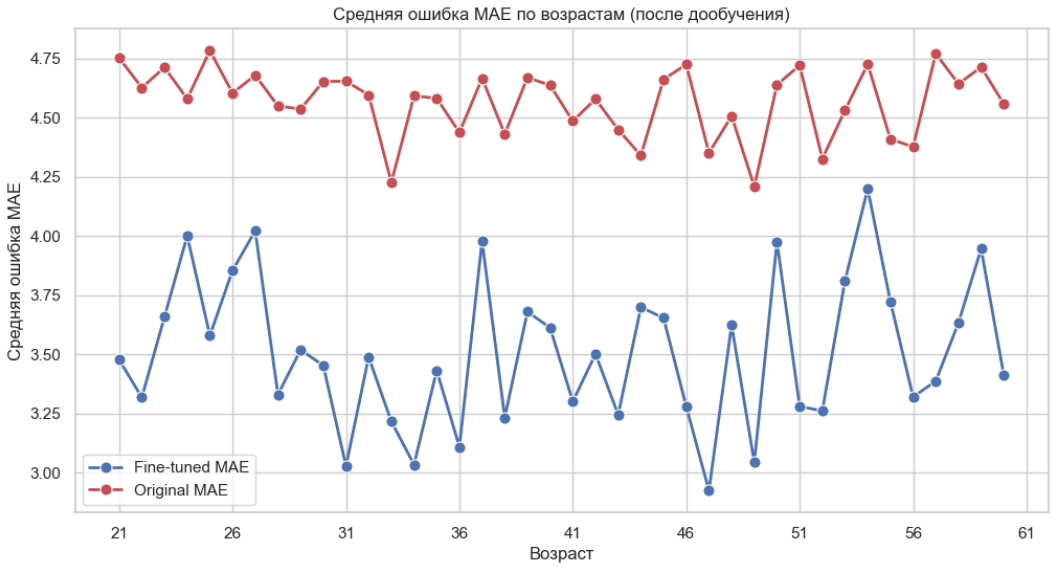

Рис. 4.38: Средняя ошибка MAE UTKFace после дообучения.

Таблица 4.9: Результаты модели MiVOLO (face-and-body) на различных тестовых выборках.

Test Set	Age (MAE)	Gender (Acc)	#params (M)	Average time (ms)
UTKFace	3.77	97.52	27.43	6.2
IMDB-clean	4.69	99.46	27.43	10.5

4.4 Детекция и распознавание текста

Данная задача заключается в том чтобы построить модель детекции и распознавания текста на картинках.

4.4.1 Метрики детекции

Вначале мы немного модифицировали метрику, использованную для оценки качества на соревновании ICDAR2015, основанной на метрике Intersection over Union (IoU).Считается, что предсказанный bounding box соответствует ground truth box, если их коэффициент IoU больше определенного порога (в нашем случае 0.5). Однако из-за того что набор с реальными данными имеет очень завышенные рамки, которые могут охватывать по несколько строк текста и имеют большие зазоры между границами и самим текстом, часть IoU, даже при хороших предсказаниях, не превышает этот порог, поэтому было добавлено еще одно условие для покрытия как можно большего числа положительных случаев. Предсказанный bounding

Рис. 4.39: Пример работы MiVOLO совместно с YOLO v8.

box теперь считается соответствующим ground truth, если происходят следующие условия:

$$\begin{cases} IoU(\text{bounding box, ground truth}) > 0.5 \\ \begin{cases} IoU(\text{bounding box, ground truth}) > 0.2 \\ \frac{\text{area of overlap (bounding box, ground truth)}}{\text{area (bounding box)}} > 0.9 \end{cases} \end{cases}$$

Такой подход (далее буду обозначать как asis) позволяет учитывать больше правильных совпадений. Исходя из этого, рассчитываются точность (precision), полнота (recall) и f1 для оценки производительности модели:

$$\text{Precision} = \frac{\text{количество детекций, имеющих соответствие с ground truth}}{\text{количество всех детекций}}$$

$$\text{Recall} = \frac{\text{количество ground truth, имеющих соответствие с предсказанными рамками}}{\text{количество ground truth рамок}}$$

$$F1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Также был происследован метод TedEval [12] для оценки качества детекции текста. В результате анализа выяснилось, что TedEval неэффективен для наших данных с многострочными и завышенными ground truth рамками и эта метрика сильно занижалась из-за преждевременного исключения хороших детекций. (Рисунок 4.40) В ответ на это была разработана дополнительная метрика (далее буду обозначать как join), объединяющая пересекающиеся

bounding boxes, чтобы лучше адаптироваться к неточностям разметки и предоставить более реалистичную оценку.

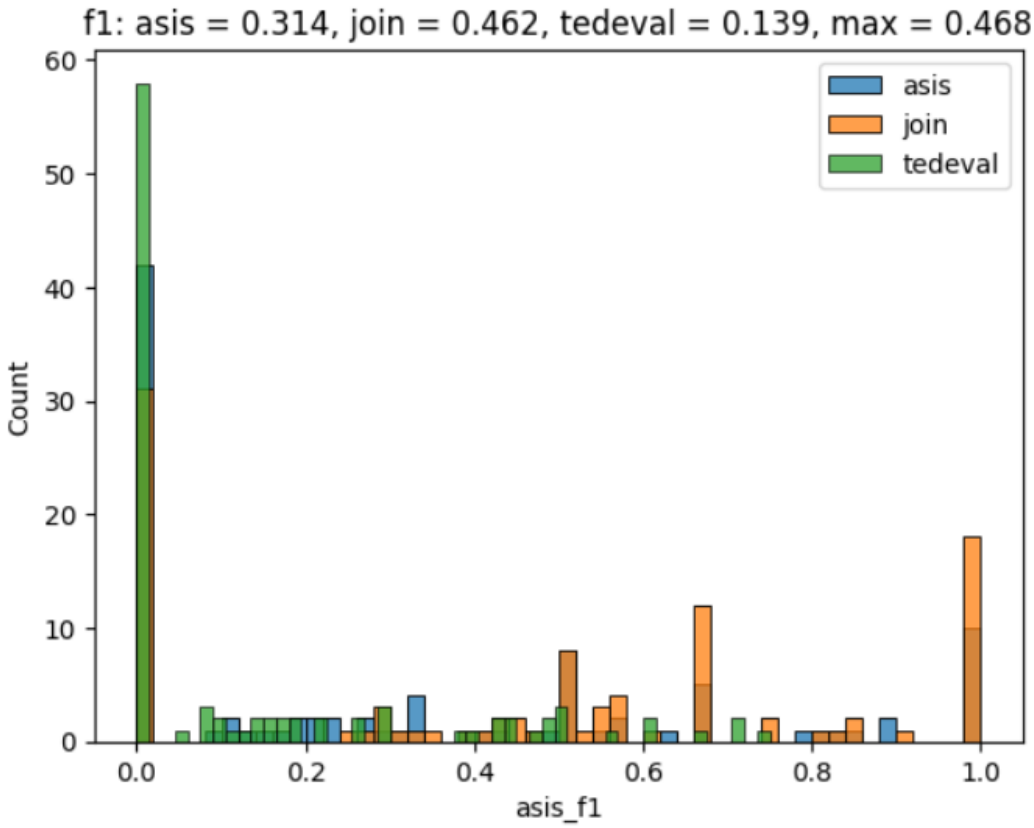

Рис. 4.40: Распределение оценок качества детекции модели DBNet++ на части картинок из набора с реальными данными. На графике представлена метрика f1 score, посчитанная с помощью 3 подходов к оценке качества детекции : asis, join, tedeval. Max соответствует максимальной метрике среди этих 3 подходов для каждой картинки. В заголовке графика отмечено среднее значение f1 score для каждого из подходов

4.4.2 Фреймворки

Были изучены фреймворки MMOCR и PaddleOCR, которые содержат инструменты для обучения моделей детекции и распознавания текста, а также содержат уже обученные модели (в основном на популярных англоязычных наборах данных и на синтетически сгенерированных наборах данных). Были рассмотрены разные модели детекции для определения их эффективности на реальных данных через описанные метрики, визуальное качество по доменам и время инференса. Домены включали декоративный шрифт, дорожные знаки, светящиеся вывески, простые и размытые тексты, этикетки, обложки книг, инструкции, текст на всю картинку, изогнутый и вертикальный текст.

Рассмотренные модели детекции : EAST, SAST, PSENet, FCENet, DBNet, DBNet++

4.4.3 Модели детекции

В процессе работы были происследованы различные архитектуры и backbones, такие как ResNet50, ResNet18, MobileNet, ResNet50 DCN, и ResNet50 OCLIP. Для каждой модели были выделены проблемные домены, посчитаны и проанализированы метрики качества и время инференса.

EAST [37]: несмотря на быстрое время инференса, генерирует много лишних прямоугольников и показывает слабые результаты на сложных текстах, включая декоративный шрифт, размытый, кривой текст и обложки книг.

PSENet [14]: сталкивается с трудностями при обработке многих доменов и при этом имеет инференс в 15 раз дольше, чем EAST.

SAST [31]: модель эффективна в детекции, но имеет большое время инференса, причем испытывает небольшие сложности с изогнутым и крупным текстом.

DBNet [15]: модель показывает быстрый инференс, хотя результаты ухудшаются с тяжелыми backbone, например, ResNet50.В целом общие проблемные домены это кривой, большой и крупный тексты.

FCENet [38]: несмотря на уникальность идеи, модель показала низкую эффективность по всем учтенным доменам.

DBNet++ [16] : стала лучшей моделью по балансу качества и времени инференса, при resnet50 dcn в качестве backbone работает неплохо во всех доменах, имея лишь небольшие неточности при кривом тексте. Примеры детекции этой модели представлены на Рисунке 4.41

В Таблице 4.10 отражены метрики качества и время инференса для каждой из протестированных моделей

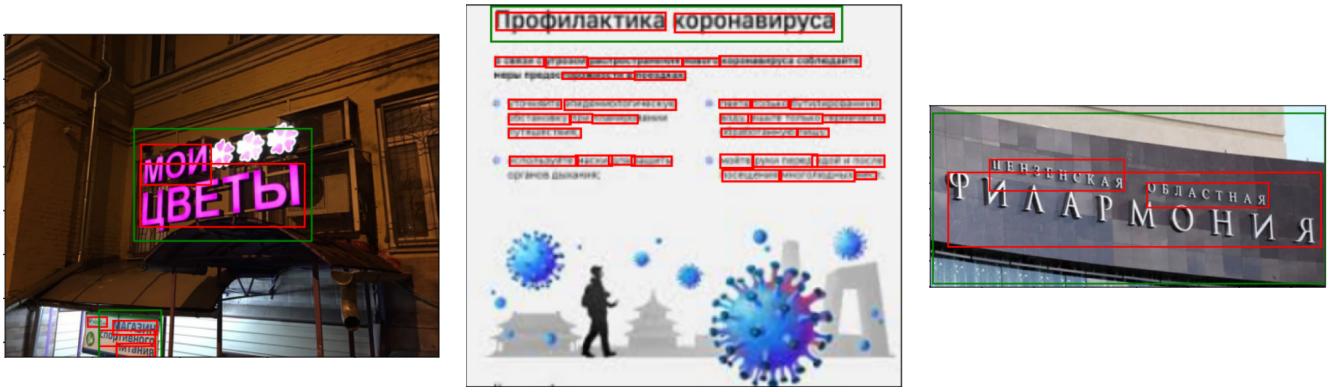

Рис. 4.41: Примеры детекции DBNet++ с backbone resnet50 dcn. Зеленые рамки - ground truth, красные рамки - детекция

Таблица 4.10: Качество и время инференса для каждой из протестированных моделей. Качество оценивается с помощью 3 метрик: precision, recall и f1 score - по каждому из 3 подходов: asis, join, tedeval.

Model	asisPrec	asisRec	asisF1	joinPrec	joinRec	joinF1	tedPrec	tedRec	tedF1	time
dbnet resnet18	0.242	0.427	0.286	0.270	0.429	0.310	0.113	0.203	0.135	76
dbnet dcn	0.227	0.397	0.267	0.264	0.411	0.297	0.108	0.191	0.129	193
dbnet oclip	0.233	0.437	0.284	0.305	0.485	0.348	0.103	0.200	0.128	182
dbnetpp dcn	0.234	0.428	0.279	0.280	0.454	0.321	0.119	0.211	0.142	213
dbnetpp oclip	0.224	0.439	0.275	0.277	0.482	0.330	0.097	0.196	0.122	280
psenet	0.084	0.253	0.112	0.175	0.291	0.200	0.035	0.106	0.048	625
psenet oclip	0.155	0.363	0.194	0.229	0.414	0.266	0.071	0.155	0.088	689
fcenet	0.185	0.275	0.199	0.261	0.344	0.275	0.060	0.113	0.070	883
fcenet oclip	0.177	0.319	0.207	0.279	0.409	0.309	0.073	0.132	0.085	913
sast	0.235	0.462	0.289	0.319	0.519	0.368	0.111	0.217	0.137	1538
east	0.223	0.366	0.253	0.298	0.429	0.326	0.102	0.159	0.114	41
east mobilenet	0.245	0.370	0.261	0.332	0.423	0.342	0.111	0.165	0.122	27

4.4.4 Выбор модели распознавания

Для тестирования и обучения моделей распознавания был сформирован синтетический датасет путем вырезания текстовых частей из изображений, предназначенных для детекции. С каждого изображения были получены два новых образца данных. Созданный датасет был разделен на тренировочную и валидационную выборки.

В качестве бейзлайна была взята модель,обученная на русскоязычных и англоязычных синтетических данных из фреймворка PaddleOCR. Архитектура данной модели включала в себя модуль извлекающий карты признаков из изображений на основе MobileNet и рекурентную сеть BiLSTM для генерации текстовых последовательностей. Однако, результаты тестирования показали, что в 20% случаев модель корректно распознавала текст на синтетических изображениях. Примеры работы модели представлены на Рисунке 4.42.Это подчеркнуло необходимость дополнительного обучения моделей на изображениях, содержащих как русский, так и английский тексты.

Учитывая, что используемые метрики выдавали одно усредненное значение по всей валидационной выборке, было затруднительно оценить производительность модели в худших сценариях. По этой причине, в анализе также учитывалось распределение значений метрики NED и уровень уверенности модели. Пример распределений в случае CRNN модели

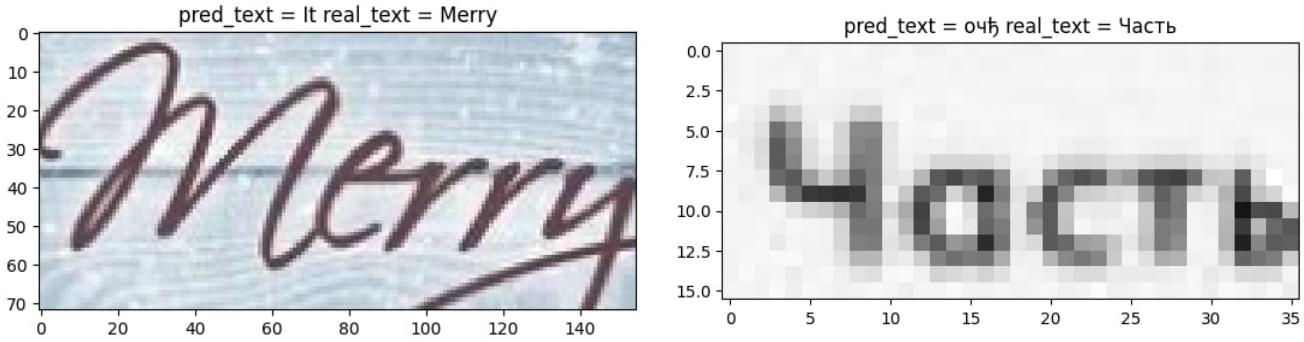

Рис. 4.42: Примеры распознавания предобученной CRNN

представлен на Рисунке 4.43. Дополнительно проводилась визуализация случаев с самым низким качеством распознавания, что помогало выявить потенциальные направления для дальнейших улучшений модели. В ходе анализа также уделялось внимание времени инференса модели, так как это критически важный параметр для практического применения модели.

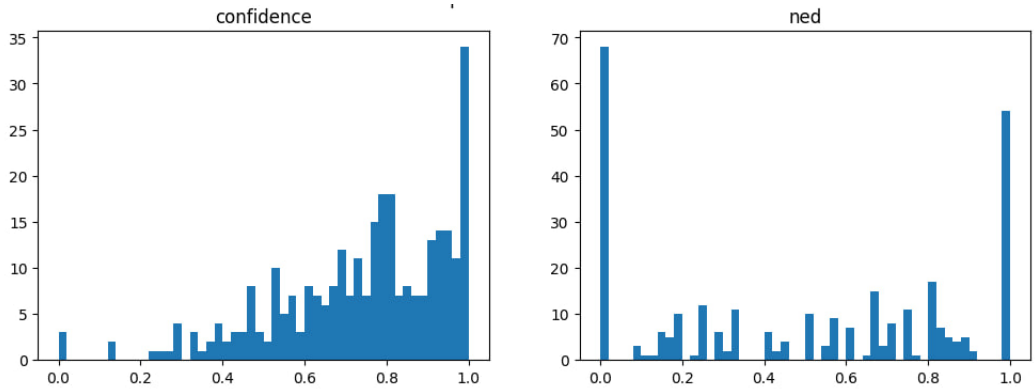

Рис. 4.43: Распределение уверенности и ned метрики предобученной CRNN модели

В начале мы решили дообучить существующую модель CRNN, адаптированную для распознавания русского языка. Исходный объём синтетической обучающей выборки составлял 1.4 тысячи изображений. В процессе обучения здесь и далее использовалась стратегия ранней остановки для определения оптимальной модели по метрике NED на валидационной выборке. В результате дообучения качество распознавания улучшилось, достигнув значений 0.8 по метрике NED и 0.58 точности на синтетическом датасете.

Затем был сформирован увеличенный тренировочный датасет, содержащий 14 тысяч синтетических изображений, после чего модель была дообучена вновь. Это позволило добиться дальнейшего улучшения результатов, где точность составила 0.67, а NED – 0.85. Графики точности двух моделей представлены на Рисунке 4.44.

В дополнение к этому, я провела эксперименты с дообучением других архитектур. Процесс обучения оказался более длительным, поскольку происходила значительная смена

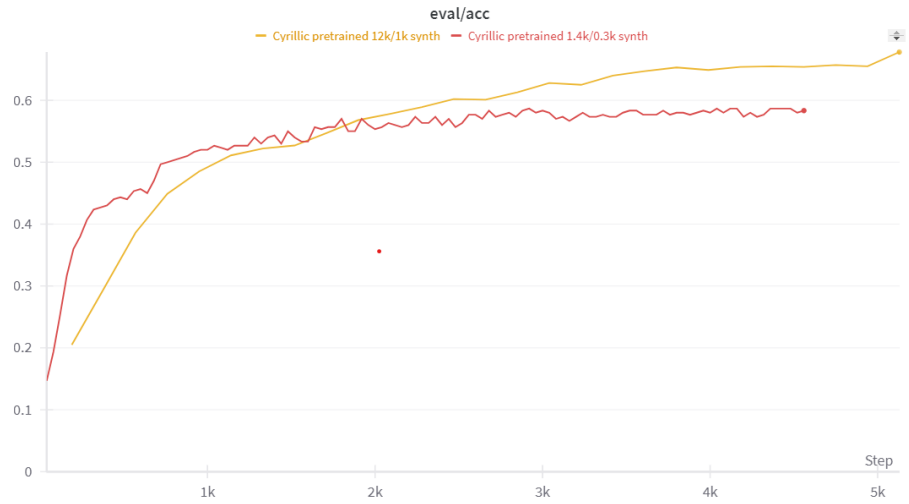

Рис. 4.44: График изменения точности на валидационной выборке в процессе обучения модели CRNN

домена исходной модели, адаптация к новому языку. В частности, модель SVTR-Tiny была дообучена на 12 тысячах изображений, показав значительно лучшие результаты по сравнению с моделью CRNN. Несмотря на то, что изначально SVTR-Tiny была предназначена только для распознавания английского языка, модель демонстрировала высокое качество на синтетической валидационной выборке с показателями 0.98 NED и 0.94 точности. Распределение ned метрики и уверенности модели на синтетической валидации представлено на Рисунке 4.45. Также данная модель достаточно быстро обучается и имеет малое время инференса в 15 миллисекунд за счет небольшого количества параметров и архитектуре, которая легко распараллеливается.

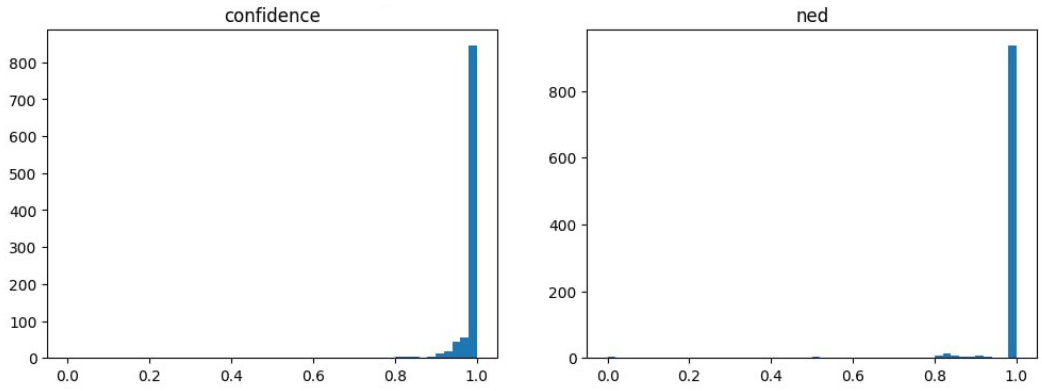

Рис. 4.45: Распределение уверенности и ned метрики на синтетической валидации дообученной модели SVTR

Однако домен синтетических данных все-таки отличается от изображений, которые можно встретить на практике. Примеры синтетических и реальных изображений представлены на Рисунке 4.46. Для оценки качества моделей на реальных данных была размечена

выборка из 300 изображений, полученных в результате применения лучшей модели детекции к датасету реальных изображений из RusTitW. Задетектированные области с коэффициентом уверенности больше 90% были вырезаны из изображения, были использованы не более 3 семплов с одной картинки для того чтобы сделать данные разнообразными. Для организации удобной разметки мы использовали инструмент LabelStudio.

Рис. 4.46: Примеры синтетических и реальных изображений

Результаты валидации на этих реальных данных у модели SVTR показали значительное снижение качества (точность 0.6, NED 0.93), что подтвердило гипотезу о значительном отличии распределения синтетических и реальных изображений.Качество модели представлено на Рисунке 4.47.Модель слишком сильно подстроилась под синтетические данные. В дальнейшем чтобы лучше понимать реальное качество модели мы опирались на качество на валидации из реальных данных в том числе и для ранней остановки.

Мы экспериментировали также с дообучение модели ABINet, архитектура которой включает компоненты, учитывающие знания о языке. Модель оказалась сложной в обучении и имела относительно продолжительное время инференса, составляющее 45 миллисекунд.Проблема смещения синтетических данных относительно реальных данных, особенно в контексте обучения языковым особенностям, привела к тому, что качество модели на реальных валидационных данных было крайне низким и практически не улучшалось в процессе обучения, оставаясь на уровне 0.42 по NED и 0.26 по точности. Примеры распознаваний

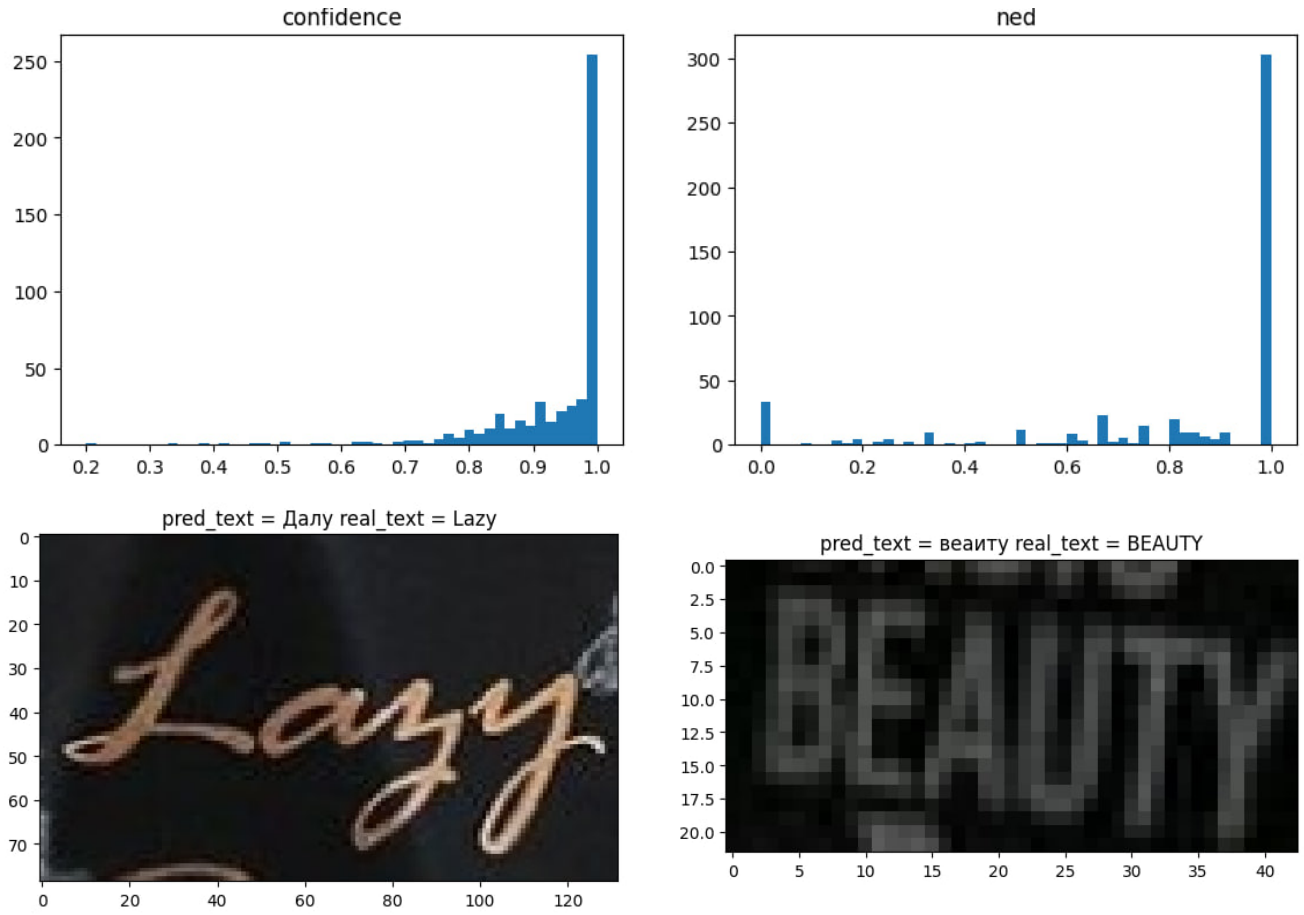

Рис. 4.47: Качество на реальных данных SVTR модели, обученной на синтетических данных

модели представлены на Рисунке 4.48.

Было проведено дообучение модели SPIN, которая быстро достигала высокого качества на синтетической обучающей выборке, однако показывала стагнацию качества на реальных данных. Модель демонстрировала относительно низкое качество на синтетических данных и невысокую точность на реальных данных, достигнув значения 0.46 по NED и 0.71 по точности, и обладала самым большим временем инференса в 60 миллисекунд. Примеры распознавания модели представлены на Рисунке 4.49.

Была также исследована архитектура Robust Scanner, которая не продемонстрировала значительного улучшения качества на реальных данных, быстро достигнув плато,и идеально выучивала тренировочную выборку достигая на ней 100% точности. Эта модель также имела самое большое время инференса, что недопустимо для модели распознавания которая потенциально на практике должна обрабатывать много частей текста. Итоговое качество на валидации - 0.80 ned, 0.745 accuracy. Несмотря на недостатки эта модель стала одной из лучших по качеству из моделей которые изначально дообучались с английского языка (у SVTR лучше ned метрика но хуже accuracy что значит что в целом svtr допускает меньше ошибок в словах однако Robust scanner покрывает больше слов).Таким образом, из рассмот-

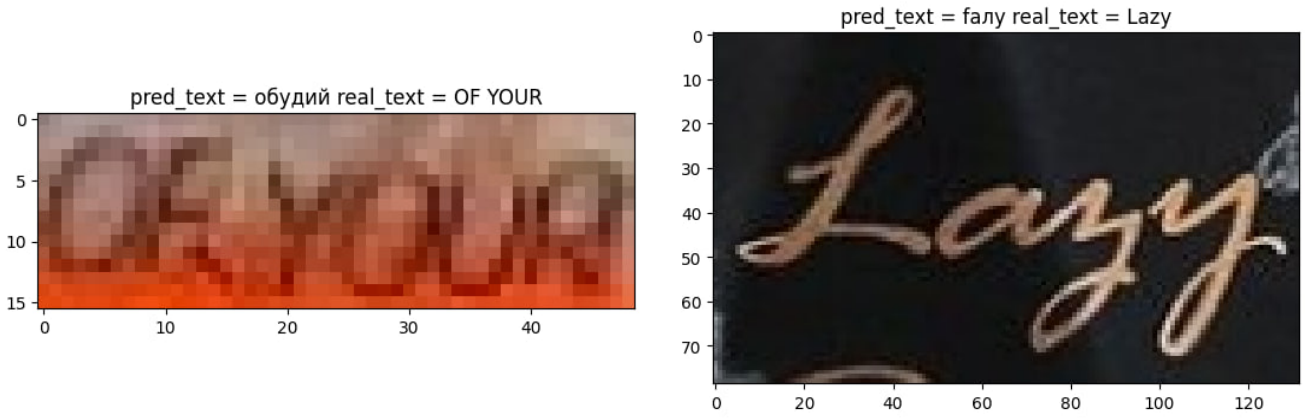

Рис. 4.48: Примеры распознаваний ABINet

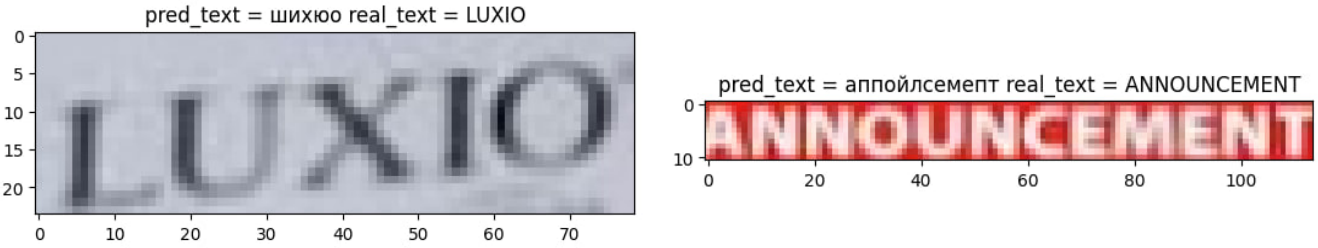

Рис. 4.49: Примеры распознаваний SPIN

ренных моделей была выбрана SVTR, так как она поддерживает идеальный баланс скорости и качества.

4.4.5 Проблема смещения синтетических данных

Для корректировки смещения было решено обогатить обучающий набор данных за счет включения дополнительных реальных изображений. В частности, мы разметили 2800 изображений, используя методику, описанную ранее.Кроме того мы использовали в дальнейшем больше синтетических данных - около 30 тысяч. Анализ словаря, используемого для обучения готовой модели CRNN, выявил наличие крайне редко встречающихся символов,например: љ,њ,ћ,ў а также отсутствие некоторых специальных символов, актуальных для рассматриваемых текстов, например:) ; “. В ответ на выявленные недостатки были созданы два новых словаря: advance, поддерживающий русские и английские буквы, цифры и основные спецсимволы ASCII (всего 161 символ), и light, включающий только русские и английские буквы и цифры (129 символов).

Мы экспериментировали с удалением заглавных букв из словарей, так как в контексте использования модели для слабовидящих пользователей и дальнейшего озвучивания текста значимость регистра сильно снижается. А это изменение сокращает значительно количество классов и потенциально может упростить задачу модели, улучшив ее обобщающую способ-

ность, так как ей больше не придется выучивать разницу между заглавными и строчными буквами.

Для оценки влияния внесенных изменений были введены новые вариации метрик: accuracy filtered и ned filtered. Эти метрики считают accuracy и ned соответственно после приведения текста к нижнему регистру, удаления пробелов и специальных символов. При обучении и валидации мы приводим текст к нужному словарю.

Мы дообучили модель SVTR-tiny на смешанном наборе синтетических и реальных данных с использованием словаря advance, включающего заглавные буквы. Результаты на синтетическом валидационном наборе снова показали высокое качество(accuracy 0,95, ned 0,988) в то время как качество на реалистической валидации также улучшилось значительно (accuracy 0.786 ned 0.932 accuracy filtered 0.83 ned filtered 0.94). Примеры распознавания представлены на Рисунке 4.50.

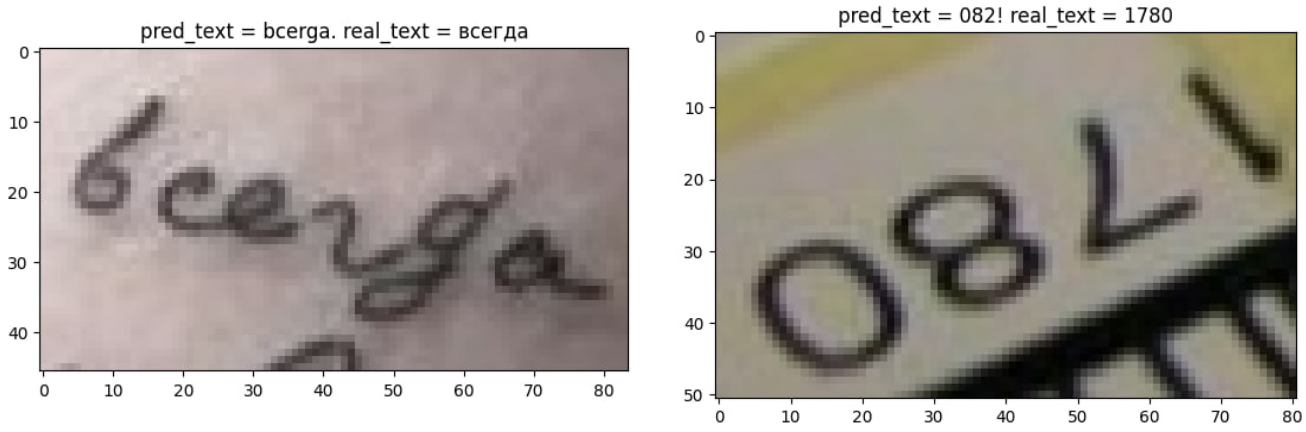

Рис. 4.50: Примеры распознаваний SVTR обученной на смеси синтетических и реальных данных

4.4.6 Проблема балансировки данных

Проанализировав результаты прошлой модели, было замечено однако что достаточно часто английские и русские буквы модель путала, могла отдельные буквы в английском слове распознавать как русские и наоборот,причем чаще английские буквы заменялись русскими. Примеры такой проблемы представлены на Рисунке 4.51. В 15% английских слов модель распознала русские буквы. После анализа данных был выявлен дисбаланс в распределении английского текста между реальными и синтетическими данными (в синтетических данных английский текст составляет лишь 3,5% против 20% в реальных данных), что предполагает необходимость корректировки. Так как в целом английский текст мы встречаем достаточно часто, следовательно обучаясь на таких данных у нас заведомо есть смещение в сторону

русского текста.

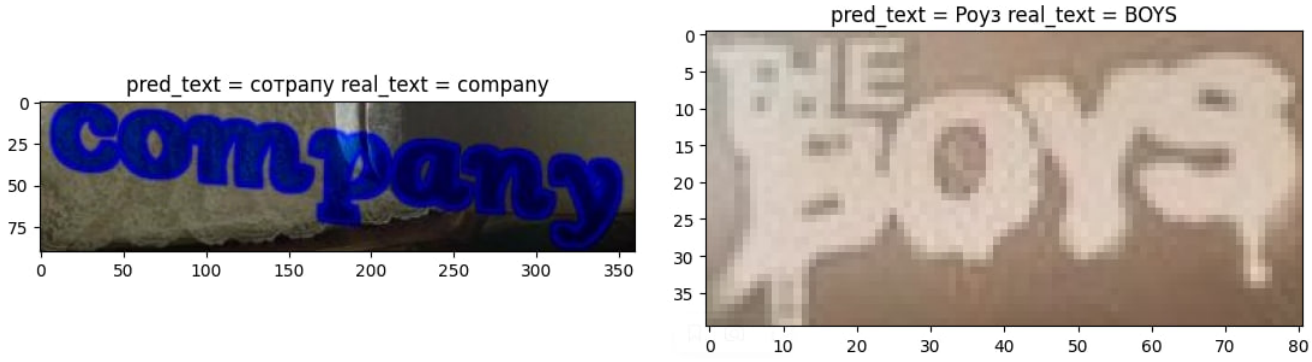

Рис. 4.51: Примеры предсказаний модели где присутствует проблема с распознаванием английских символов

Поэтому были выделены англоязычная и русскоязычная части из прошлого обучающего датасета. И был проведен следующий эксперимент: в процессе обучения на каждой эпохе доля английского и русского текста поддерживалась 1:1. Качество стало немного лучше (accuracy 0.846, ned 0.939), особенно сильно улучшилась точность. Однако модель все равно делала ошибки в английский и русских буквах, возможно потому что она подстроилась под сравнительно малый объем английского текста и видела недостаточно много примеров. Плюс ко всему обучение было достаточно нестабильным поэтому мы отказались от использования такой модели.

4.4.7 Эксперименты с разными словарями

Был проведен эксперимент с дообучением модели SVTR на словаре advance, исключав из него заглавные буквы. Основная гипотеза заключалась в том, что сокращение алфавита позволит увеличить количество обучающих примеров для каждой буквы, тем самым улучшится способность модели к их распознаванию. Результаты подтвердили предположения: точность модели улучшилась в плане распознавания именно буквенных символов (accuracy filtered 0.876 ned filtered 0.952), а также модель стала лучше справляться с распознаванием отсутствия текста на изображении в сравнении с другими моделями, которые часто выдавали случайный набор символов в таких случаях.

Улучшилась также ситуация с перепутыванием английских и русских символов. В случаях, когда в распознанном тексте все же оставались ошибочные символы, их количество было небольшим — в среднем менее трех.

После этого было решено провести эксперимент с использованием упрощенного словаря light, не включающего спецсимволы, чтобы оценить способность модели к распознаванию

буквенно-числовых символов без необходимости выучивать спецсимволы, также хотелось проверить не начнет ли модель принимать спецсимволы за другие буквы. Эксперименты проводились как со словарем содержащим заглавные буквы, так и без них. Причем прирост точности был менее заметен по сравнению с моделями, использующими словарь advance. В частности, переход с advance на light привел к улучшению accuracy filtered на 3 процента и ned filtered на 0.2 процента. При переходе от advanced little к light little разница составила всего 0.1 процента для ned filtered. Графики сравнения качества моделей обученных на разных словарях представлено на Рисунке 4.52.

Рис. 4.52: Графики изменения качества моделей,обученных на разных словарях, на реальной валидационной выборке

Анализ результатов подтвердил, что модель достаточно часто ошибочно принимает спецсимволы за другие символы, выдавая ложный текст. В 10 из 19 случаев на синтетических данных где были лишь спецсимволы модель обученная на light словаре выдавала какой-то текст.

4.4.8 Эксперименты с данными

Были рассмотрены два способа упрощения процесса обучения и распознавания текста на изображениях. Первый подход касается обработки многострочного повернутого текста. В случаях, когда из изображения вырезается прямоугольник, соответствующий определенному повернутому слову, в этот прямоугольник может попасть дополнительная строка текста. Это может сбивать модель распознавания, предназначенную для работы с однострочным текстом. Решением данной проблемы стало использование техники экстраполяции, при которой по пикселям внутри обозначенной рамки текста заполняется оставшееся пространство прямоугольника, с целью “размытия” и удаления лишних строк. Это требует дополнительного дообучения модели на таких экстраполированных изображениях, поскольку они заметно отличаются от типичных вырезанных частей с текстом. При этом, для улучшения устойчи-

вости модели, 30% изображений оставляются без экстраполяции.

Второй подход связан с корректировкой угла поворота текста на изображениях. Имея информацию о прямоугольнике, который ограничивает текст, можно рассчитать угол его поворота и, соответственно, повернуть изображение так, чтобы текст располагался горизонтально. Реализованный алгоритм поворачивает изображение так, чтобы длинная сторона текста была выровнена вдоль оси Ox. В тех случаях, когда описанный угол поворота превышает 89 градусов, мы доворачиваем так чтобы короткая сторона была выровнена по оси Ox, в таком случае мы предполагаем что текст располагается вертикально. С целью поддержания робастности модели также сохраняется 30% изображений без коррекции поворота. Пример экстраполяции и поворота представлены на Рисунке 4.53

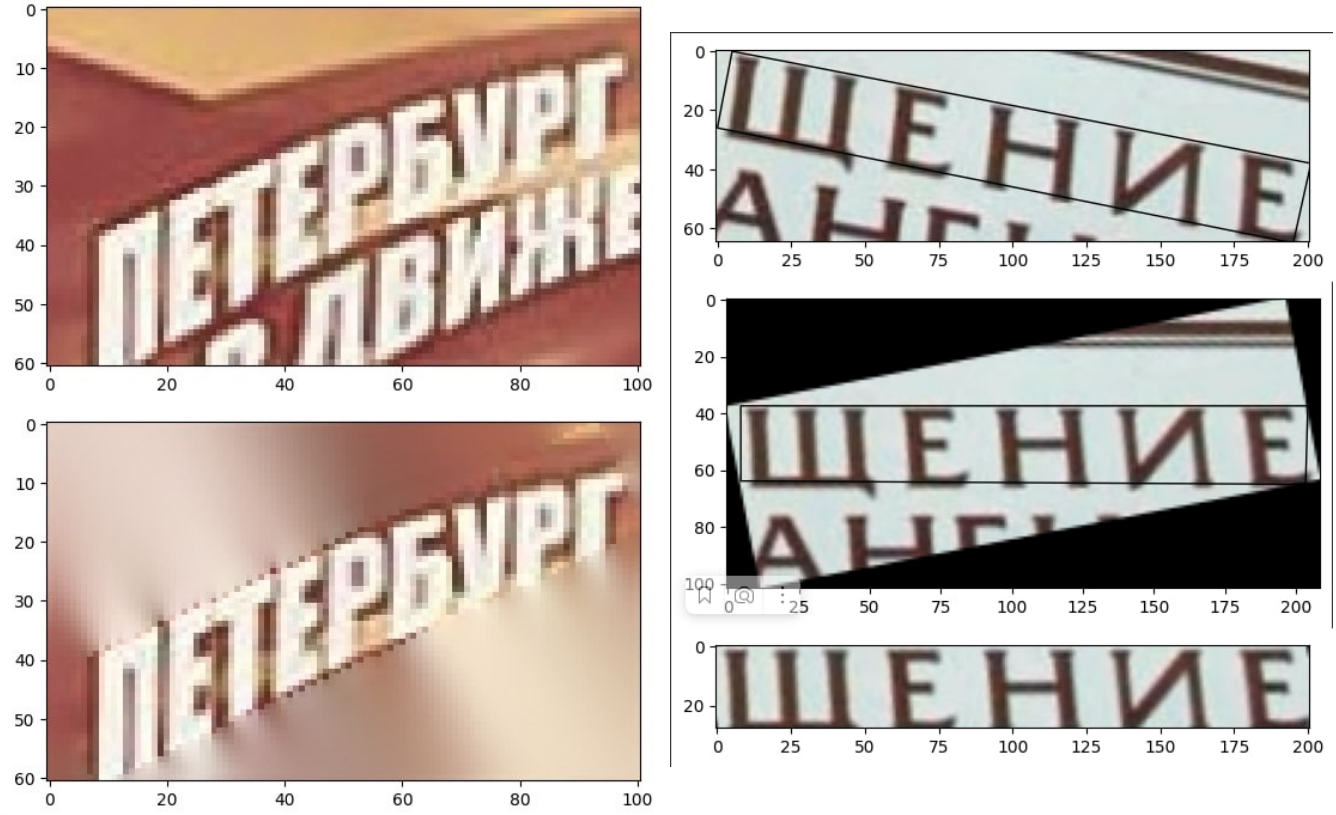

Рис. 4.53: Примеры экстраполяции и поворота

Результаты эксперимента с экстраполяцией показали ухудшение метрик относительно прошлых экспериментов, сильнее всего упала точность, что может быть обусловлено затруднениями в распознавании символов с размытыми контурами, например в буквах "щ "д "й". Пример данной проблемы представлен на Рисунке 4.54. Анализ результатов показал, что в 9 из 21 случая обработанные таким образом изображения распознавались моделью лучше, чем на стандартных данных, и в 6 случаях результаты были сопоставимы. В то же время следует отметить потенциальные проблемы с распознаванием определенного подмножества

символов, что является значительным недостатком. Также важно подчеркнуть, что наличие подобных обработанных изображений в обучающей выборке позволяет распознавать текст достаточно хорошо даже на не экстраполированных картинках, разница в точности менее 2 процентов.

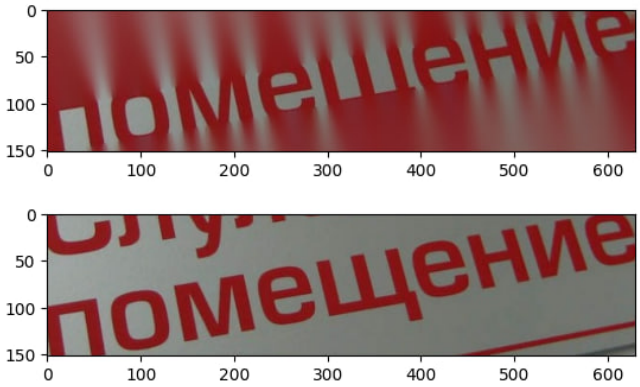

Рис. 4.54: Пример затруднения распознавания текста при экстраполяции

Касательно экспериментов по повороту текста, результаты оказались неоднозначными. Качество стало лучше относительно экстраполяции, однако оказалось хуже некоторых предыдущих удачных экспериментов.Одной из проблемой стало не всегда корректное определение угла поворота, особенно в случаях, когда текст на изображении достаточно короткий и рамка, относительно которой вычислялся угол, имеет форму, близкую к квадратной. В таких ситуациях текст часто оказывался в нестандартном положении, что мешало корректному распознаванию. Примеры такой ситуации представлены на Рисунке 4.55. Однако проблемы с определением угла можно было выявить, анализируя уверенность модели в своих предсказаниях, которая, как правило, была низка в случаях некорректного распознавания.

Качество на не повернутых текстах как и в случае с экстраполяцией упало не сильно, соответственно регуляризация модели путем добавления в обучающую выборку 30 процентов не повернутых примеров позволяет достаточно неплохо распознавать повернутые тексты.

Первоначально для улучшения качества распознавания была использована техника, при которой первоначально из исходного изображения вырезался прямой прямоугольник в который была вписана рамка повернутого текста, затем производился поворот текста внутри данного сегмента. Однако такой метод часто приводил к возникновению на итоговом изображении артефактов в виде черных полос, которые влияли на результаты распознавания моделью. В результате дальнейших экспериментов для того чтобы избавиться от данных артефактов было решено сначала поворачивать всё изображение на нужный угол, и только после этого вырезать прямоугольник с текстом. Этот подход позволил немного стабилизи-

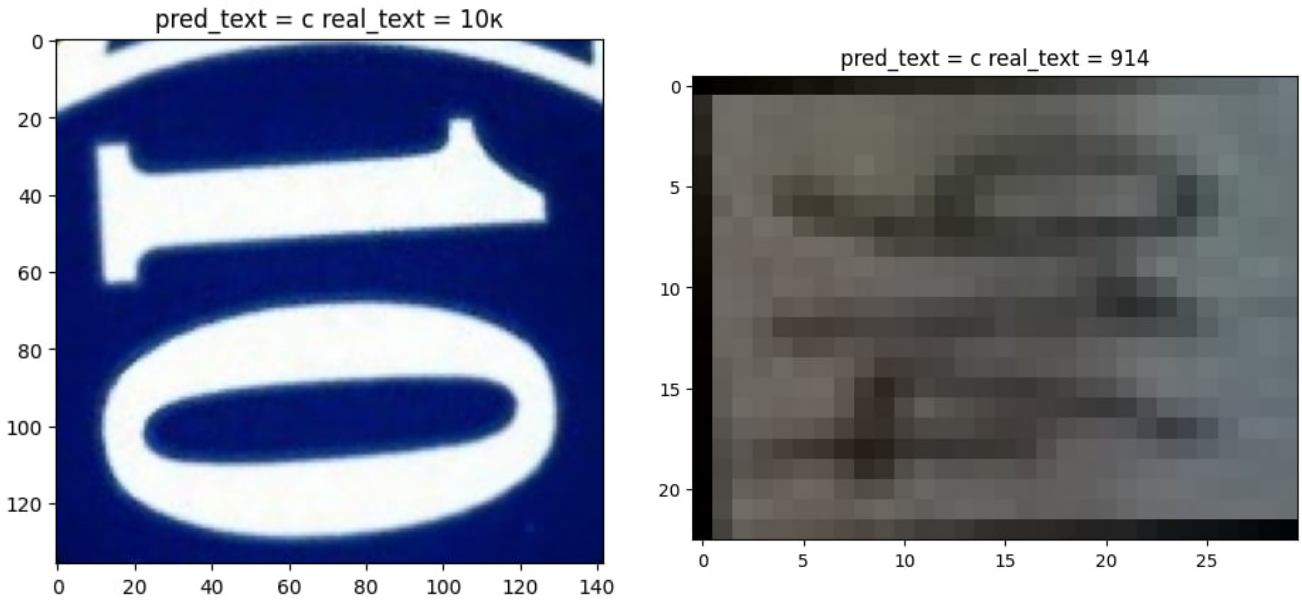

Рис. 4.55: Пример затруднения распознавания текста при неправильном повороте изображения

ровать качество распознавания текста моделью.

4.4.9 Эксперименты с улучшением синтетического набора данных

В процессе анализа использованной синтетической выборки было выявлено, что пропорция английских текстов и специальных символов значительно отличается от их доли в реальных данных, что негативно сказывалось на качестве обучения модели. В связи с этим было принято решение построить собственный пайплайн для генерации синтетических данных, основываясь на статистиках, полученных по реальным данным, и учитывая все символы которые есть в словаре чтобы модель хорошо выучила каждый из них.

Для генерации качественных синтетических данных был сформирован текстовый корпус, который использовался фреймворком SynthText для размещения текста на изображениях. Пайплайн генерации текстовых данных был разработан на основе анализа статистик по размеченной выборке данных и включал следующие этапы:

1. Выбор типа первого слова: 65% - на русском языке, 29% - на английском языке, 1% - отдельный спецсимвол, 5% - числовое значение.

2. Определение регистра первого слова: 40% - все буквы заглавные, 40% - все буквы строчные, 20% - только первая буква заглавная.

3. Решение о добавлении в начало спецсимвола или числа: 2.5% - да, число; 2.5% - да, спецсимвол; 95% - не добавлять.

4. Решение о добавлении в конец спецсимвола или числа: 2.5% - да, число; 7.5% - да,

спецсимвол; 90% - не добавлять.

5. Определение необходимости добавления дополнительного слова: 10% - да, 90% - нет.

- Если не добавлять, процесс завершается.

- Если добавлять, удаляется добавленный в конец спецсимвол или число если добавили на предыдущем этапе и процесс продолжается.

6. Метод соединения слов: 45% - пробелом, 45% - с помощью соединительных спецсимволов, 5% - без соединения, 5% - числами.

7. Выбор языка второго слова: 90% - тот же язык, 10% - другой.

8. Регистр второго слова: 80% - такой же, как у первого, 10% - все буквы строчные, 5% - только первая буква заглавная, 5% - все буквы заглавные.

9. Решение о добавлении в конец спецсимвола или числа: 2.5% - число, 7.5% - спецсимвол, 90% - не добавлять.

Дополнительно для каждой ситуации, где мы добавляем спецсимволы мы строим вероятности выбора конкретного спецсимвола, так как, к примеру, вопросительный знак более характерен для конца строки нежели для начала. Таким образом был разработан новый корпус для генерации синтетических данных с использованием фреймворка SynthText. Для обогащения данных, было добавлено 20 различных шрифтов, включая популярные рукописные и печатные варианты, что позволило расширить исходные возможности фреймворка, поддерживавшего всего 3 шрифта. Примеры получившихся синтетических изображений представлены на Рисунке 4.56.

Рис. 4.56: Примеры новых синтетических изображений

После создания новой синтетической выборки из 20 тысяч изображений (далее - новая синтетика) я провела эксперимент с обучением модели на этой новой синтетике с поворотами

текста. Обучение происходило дольше и модель уже не так быстро выучивала тренировочную выборку, что говорит о том что данные стали более информативными и разнообразными. Качество стало хуже по точности чем в случае с экспериментом когда мы обучали модель на старой синтетике с поворотами, однако ned метрика возросла. Особенно интересно, что неповернутые изображения демонстрировали лучшее качество распознавания, несмотря на то, что большинство тренировочных примеров было с поворотами. Также разница в качестве между повернутыми и неповернутыми изображениями уменьшилась по сравнению с предыдущим экспериментом, это скорее всего связано с избавлением от артефактов. Если проанализировать результаты такой модели то можно заметить что она в целом менее устойчива, чем модели обученные без поворотов текста,возможно это связано с тем что мы подаем много простых примеров в процессе обучения.Следовательно логичная идея - использовать модель обученную на исходной новой синтетике без поворотов и использовать повороты уже во время инференса. Однако было замечено, что использование поворотов в процессе инференса может быть не всегда эффективным. Например, определенные аспекты, такие как обрезка характеристических элементов букв (к примеру у букв у,д,й,щ) или влияние перспективы, могут приводить к схожести символов с другими, что снижает точность распознавания. Примеры когда повороты эффективны и когда неэффективны представлены на Рисунке 4.57.

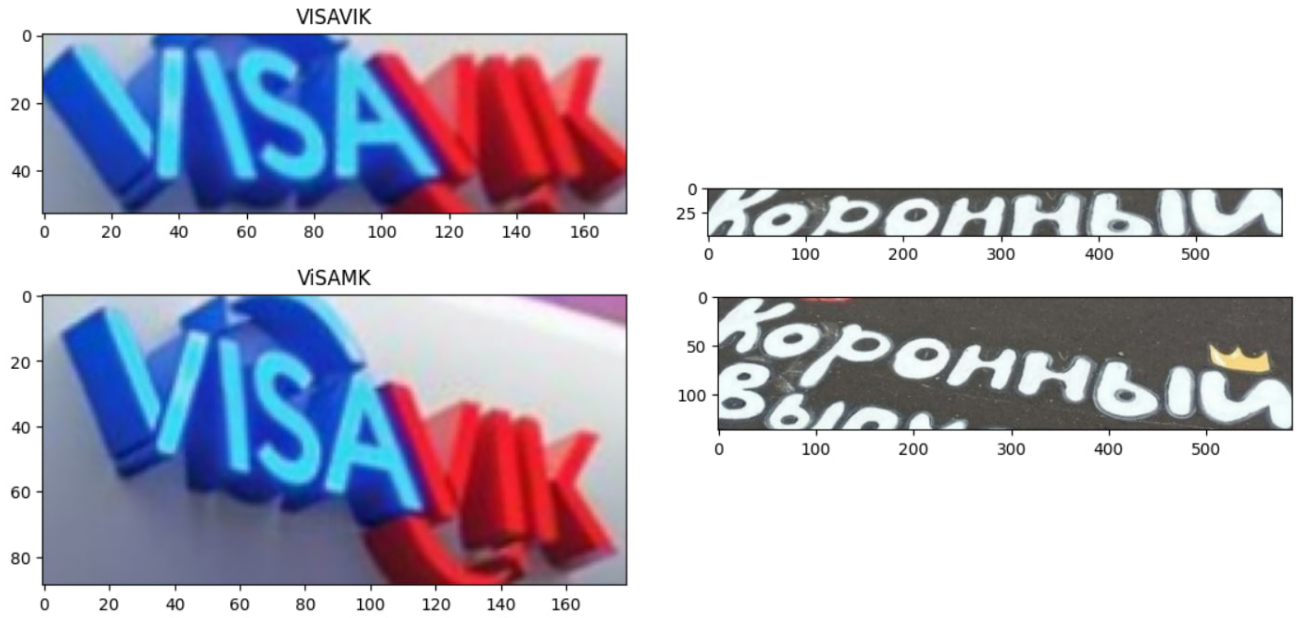

Рис. 4.57: Примеры эффективности и неэффективности поворотов

Также можно отметить, что уверенность модели в предсказаниях может служить индикатором качества распознавания, что позволяет использовать результаты как с повер-

нутых, так и с неповернутых изображений, выбирая наиболее вероятный вариант. График взаимосвязи разницы уверенностей и разницы метрик между повернутым и неповернутым входным изображением представлен на Рисунке 4.58.

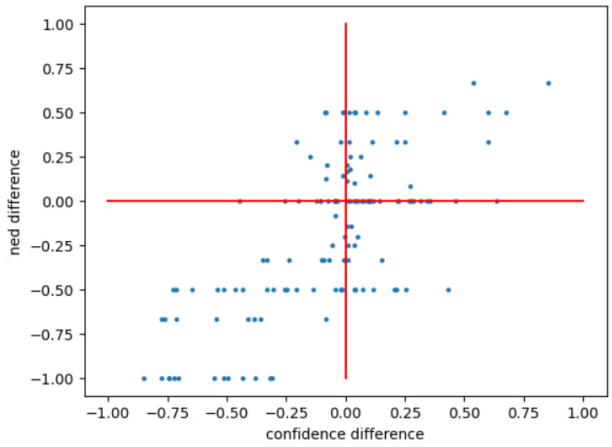

Рис. 4.58: График взаимосвязи разницы уверенностей и разницы метрик между повернутым и неповернутым входным изображением

Кроме того, дообучение модели на новой синтетике и реальных данных без применения поворотов привело к улучшению качества на неповернутых изображениях относительно всех предыдущих экспериментов с advance словарем(accuracy 0.85, ned 0.947), однако на повернутых изображениях результаты были хуже. Это подтверждает, что поворот изображения может усложнять распознавание текста в некоторых случаях.

По результатам анализа можно заключить, что использование поворотов на этапе инференса может дать прирост в качестве, если уверенность на повернутом изображении хотя бы на 1 процент выше, чем на неповернутом, что отражается в следующих значениях ned: 0.947 без использования поворотов, 0.949 с использованием поворотов, ориентируясь на уверенность. Таким образом, результаты указывают на возможную пользу от использования данных с поворотом во время инференса.

4.4.10 Эксперименты со смешиванием разных видов синтетики

В ходе анализа текстов из новой синтетической выборки, было замечено, что средняя длина получившихся текстов меньше, чем в датасете с реальными данными. Средняя длина текста в новой синтетике оказывается меньше, возможно, потому что изображения в SynthText являются реальными фотографиями и картинками, на которых сложно разместить длинный текст, так как однородные области достаточно маленькие. При этом анализ результатов обученных на новой синтетике моделей не выявил смещения качества в зависимости от размера текстов.

Я экспериментировала с двумя подходами, которые потенциально могли улучшить качество модели: балансировкой доли реальных данных и новой синтетики в соотношении 15 к 85, это может скомпенсировать сложные случаи из которых состоит новая синтетика, и добавлением данных из старой синтетической выборки в обучение. Подход со старой синтетикой может оказаться полезным, поскольку в ней средняя длина текстов более приближена к данным на практике и включает больше простых слов, без спецсимволов. Обе синтетики в совокупности позволяют покрыть как можно больше разнообразных случаев, что способствует повышению устойчивости модели, так старая синтетика покрывает часто встречающиеся паттерны, в то время как новая синтетика покрывает как можно больше сложных случаев что делает модель робастной.

К сожалению, эксперимент по балансировке соотношения синтетических данных не привел к улучшению качества, результат оказался сильно хуже, чем в предыдущих экспериментах. В случае с добавлением старой синтетики были применены дополнительные настройки гиперпараметров обучения модели, такие как скорость обучения,количество эпох, по которому вычисляется скорость затухания lr, weight decay и количество warmup итераций. Для модели SVTR был использован Cosine scheduler для регулировки learning rate, который рассчитывается по формуле:

$$\mathrm{lr} = 0.05 \left(\cos \left(\mathrm{epoch} \cdot \frac{\pi}{\mathrm{epochs}} \right) + 1 \right)$$

с добавлением компоненты Linear Warmup для управления начальным этапом обучения. Также мы пробовали использовать различные оптимизаторы такие как Momentum, Adam и AdamW, однако AdamW, который использовался ранее, оказался значительно лучше остальных. При замене остальных гиперпараметров качество практически не менялось однако лучшая комбинация: 50 эпох, AdamW оптимизатор,количество warmup эпох 2, learning rate 5e-5. График сравнения всех рассмотренных комбинаций гиперпараметров для данной модели представлен на Рисунке 4.59.

Мы также попробовали другой способ совместить старую синтетику и новую: вначале модель обучается на смеси старой синтетики и реальных данных, после этого она дообучается на новой синтетике в сочетании с реальной выборкой.Дополнительно можно также немного дообучить после этого модель только на реальных данных. Такой подход позволит модели сначала выучить в большей степени буквенные символы, слова приближенные к реальным данным, найденные веса сформируют хороший базис, который впоследствии скорректируется дообучением на более сложном и разнообразном синтетическом датасете с целью выучить

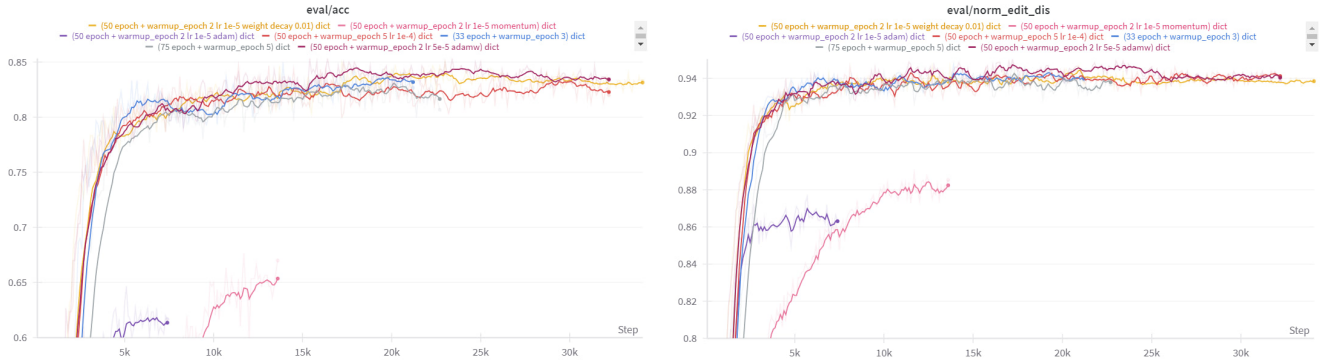

Рис. 4.59: График сравнения рассмотренных комбинаций гиперпараметров

более сложные комбинации и редкие символы. В завершении можно также дообучить модель на реальные данные чтобы немного подкорректировать модель под реальную среду.

Результаты поэтапного обучения не стали лучше обучения только на хорошей синтетике совместно с реальными данными.Была проанализирована модель полученная обучением на первых 2 из 3 выборок - качество в целом получается такое же, как и в случае обучения на всех 3 выборках, так как реальные данные присутствовали при обучении на других этапах.

Дополнительным улучшением стало включение поддержки пробельного символа в модель. В ходе анализа кода фреймворка было выявлено, что по умолчанию пробел не поддерживается, что могло ограничивать функциональность модели. Внесение этого изменения предоставило возможность более точной работы с текстовыми данными, где пробельный символ играет ключевую роль.

Финальной моделью была выбрана модель обученная в 2 этапа: вначале старая синтетика причем количество было увеличено до 30 тысяч совместно с реальными данными,а после новая синтетика совестно с реальными данными, мы использовали словарь advance где нет заглавных букв, также добавили поддержку пробельного символа.Повороты для данной модели не использовались,так как конкретно в случае этой модели часто предсказания на повернутых изображениях имели завышенную уверенность, но при этом были хуже. Качество данной модели на реальной валидационной выборке стало лучшим из всех поставленных экспериментов: точность 0.86, ned: 0.96, accuracy filtered: 0.9, ned filtered: 0.9665. Результаты экспериментов представлены в Таблице 4.11

Таблица 4.11: Результаты экспериментов

Model Name	**Accuracy**	**NED**	**Acc Filter**	**NED Filter**
pretrained crnn	0.18	0.498		
svtr 12K synth	0.6	0.819		
abinet 12K synth	0.268	0.42		
spin 12K synth	0.463	0.713		
robust scanner 12K synth	0.745	0.8		
svtr synth+real	0.786	0.932	0.83	0.94
svtr balanced eng-rus	0.846	0.939	0.87	0.944
svtr advance little dict	0.843	0.944	0.876	0.952
svtr light dict			0.86	0.942
svtr light little dict			0.876	0.953
svtr extrapolation	0.826	0.936	0.853	0.941
svtr rotation	0.84	0.937	0.866	0.944
svtr good_synth rotation	0.83	0.943	0.856	0.949
svtr good synth+real	0.85	0.947	0.87	0.951
svtr good synth+real balanced	0.81	0.942	0.842	0.945
svtr good+old synth+real best parameters	0.843	0.95	0.866	0.948
svtr combined	0.843	0.944	0.856	0.948
best model (combined, spaces, advance little)	0.866	0.96	0.9	0.966

5 Реализация сервиса

В процессе написания сервиса были использованы следующие библиотеки:

- FastAPI: Современный, быстрый веб-фреймворк для создания API
- Uvicorn: Легковесный, быстрый и асинхронный сервер, разработанный для эффективной работы с FastAPI.
- Aiogram: Фреймворк для создания ботов в Telegram на Python, использующий асинхронный подход.
- gTTS (Google Text-to-Speech): Библиотека для взаимодействия с Google Text-to-Speech API, позволяющая преобразовывать текст в речь.
- httpx: Современная и быстрая библиотека HTTP клиента, поддерживающая асинхронную и синхронную передачу данных.

Модели, которые используются в данном проекте, очень разнообразные, они имеют большое количество зависимостей, которые могут конфликтуют друг с другом. Чтобы избежать подобных проблем, было принято решение использовать Docker, это позволяет каждой отдельной модели запускаться в своём окружении, не мешая остальным.

Для обеспечения передачи информации между моделями была необходима конфигурация связи между Docker-контейнерами. Она была реализована с помощью настройки сети между контейнерами с использованием инструмента Docker compose.

Каждый контейнер включает в себя сервис, написанный с использованием FastAPI, что позволяет поддерживать REST API интерфейс для взаимодействия между моделями. Для каждой модели разработаны специальные API-ручки, которые позволяют удобно отправлять данные в модель и получать нужные результаты. Это обеспечивает гибкость решения.

Так как модели требуют значительных вычислительных ресурсов, для увеличения производительности, было принято решение развернуть систему на сервере с тремя графическими картами Tesla T4. Это позволяет значительно ускорить обработку данных и выполнение моделей в условиях высоких требований к вычислительной мощности.

Чтобы обеспечить более удобный и комфортный пользовательский опыт, было принято решение реализрвать взаимодействие с нашим сервисом через асинхронного телеграмбота. Это позволяет пользователям легко и удобно получать необходимую информацию и доступ к сервису прямо из популярного мессенджера. При этом асинхронная обработка данных способствует эффективному использованию вычислительных мощностей и ускоряет время получения ответа пользователем

Итоговая система работает следующим образом:

1. Изначально пользователь отправляет картинку телеграм боту

2. Телеграм бот отправляет POST запрос на API-ручку detect_GUI вместе с картинкой. Данный микросервис, обработав изображение, формирует zip архив с отдельными фрагментами интерфейса (блоки текста, картинки, кнопки, иконки и тд) и возвращает его обратно (вместе с JSON файлом, где содержится вся необходимая информация для дальнейшей обработки).

3. После получения zip архива, телеграм бот отправляет POST запросы в 4 сервиса: caption (для описание изображения), age (для определения возраста и пола человека), cartoons (для классификации изображений на реалистичное и не реалистичное) и detect (для распознавания текста). Для оптимизации времени исполнения все 4 микросервиса обрабатывают запросы асинхронно. После обработки zip архива, они создают JSON файлы со всей необходимой информацией и возвращают их обратно телеграм боту

4. Получив JSON файлы от различных микросервисов, телеграм бот агрегирует их и собирает один общий JSON файл, в котором содержится вся необходимая информация об изображении

5. Чтобы отдавать пользователю результат в удобном для восприятия формате, вся нужная информация из JSON собирается в компактном текстовом формате, с использованием заранее разработанного шаблона.

6. Данное итоговое описание подаётся на вход TTS модели для дальнейшей озвучки

7. В конечном итоге, бот отправляет пользователю 3 сообщения:

   – Описание в удобном виде

   – Голосовое сообщение, которое озвучивает данное описание

   – Итоговый JSON файл с полной информацией об изображении

Схема работы сервиса изображена на Рисунке 5.1.

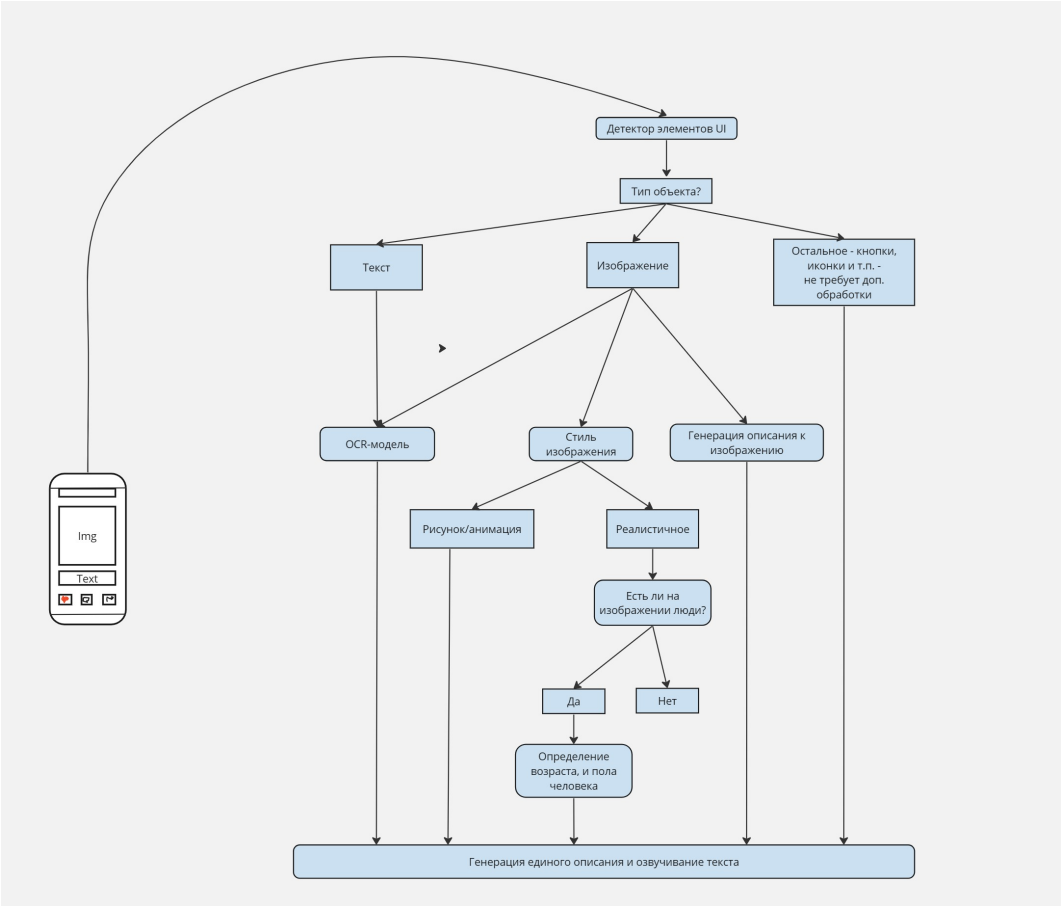

Рис. 5.1: Высокоуровневая схема работы сервиса.

Хочется отметить, что, хоть наш сервис и предоставляет итоговые описания на английском языке, он в состоянии детектировать русский текст на изображениях. Кроме того,

голосовая модель настроена таким образом, что может озвучивать как русский, так и английский текст. Последовательности текста на разных языках проходят озвучку и склеиваются в одну звуковую дорожку, которая предоставляется пользователю.

Итоговый вывод нашего сервиса формируется следующим обазом: изображение делится на 9 частей, собирая только те элементы интерфейса, котрые находятся в выбранной области, и описывается по шаблону. Пример вывода представлен на Рисунках 5.2, 5.3

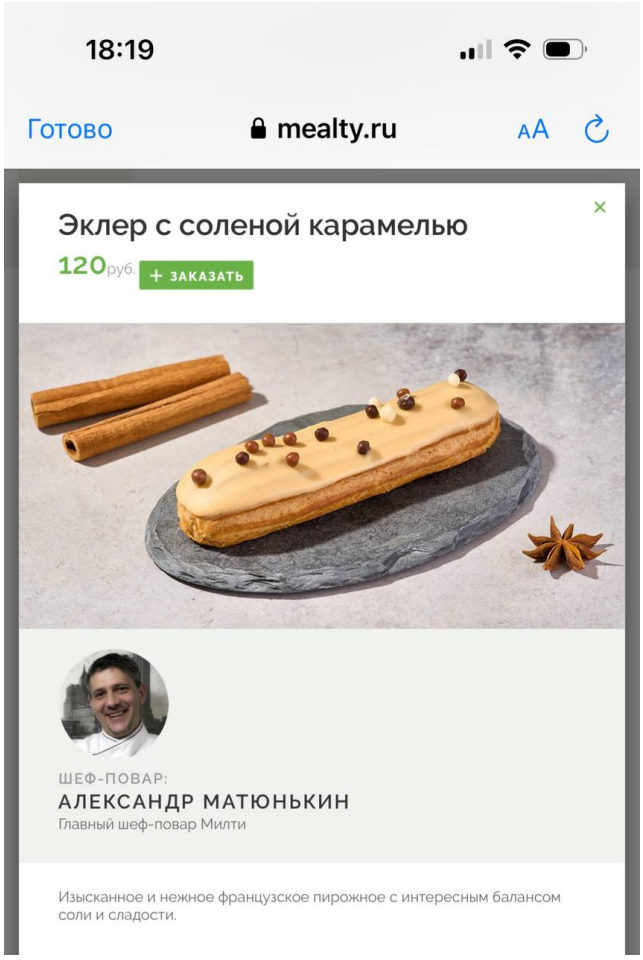

Рис. 5.2: Изображение

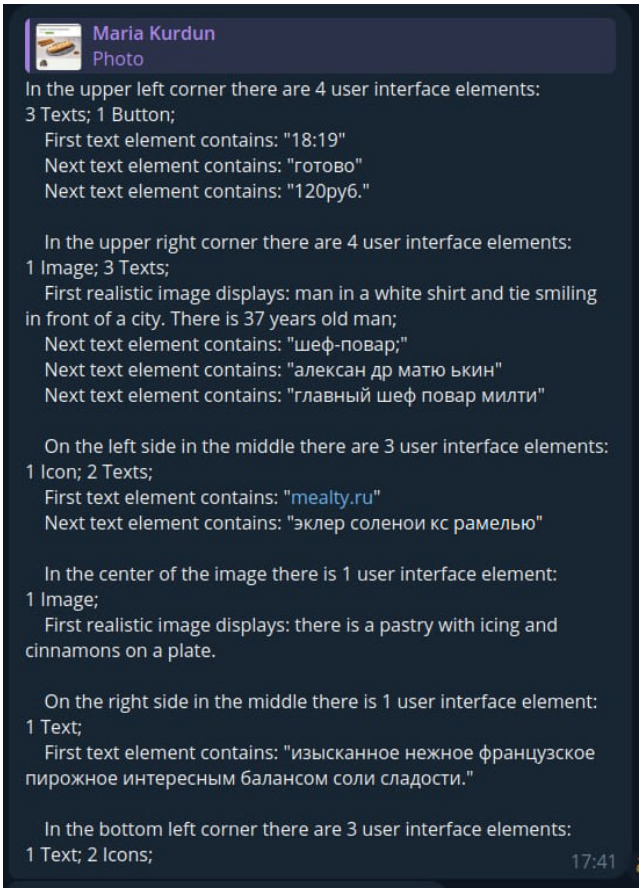

Рис. 5.3: Ответ сервиса

5.1 Сервис детекции элементов интерфейса

Первый этап после отправки изображения телеграм-боту - передача изображения в сервис детекции элементов интерфейса. Данный сервис представляет собой один Docker-контейнер, в котором работает модель-детектор YOLOv9. Основные модули, используемые сервисом, - FastAPI, использующийся для реализации API, и ultralytics - фреймворк, необходимый для работы YOLOv9.

Основная ручка сервиса (handler) - "detect_GUI". Эта функция - "обертка"над функцией, генерирующей инференс YOLOv9, и несколькими функциями-эвристиками. Разберем

её работу более подробно:

- Изображение из потока сохраняется локально в докер-контейнере.
- По изображению генерируются предсказания модели YOLOv9 (с порогом уверенности, равным 0.5, так как эмпирически было обнаружено, что результаты, выдаваемые моделью с более низкой уверенностью, в основном неудачны).
- Выход модели представляет собой структуру данных, по которой собирается JSON-файл с информацией про UI-элементы на изображении, а также - zip-архив с вырезанными прямоугольными участками фотографий, соответствующими bounding box'ам, выданным YOLOv9.
- В zip-архиве все "вырезанные"изображения разбиты по субдиректориям, каждая из которых соответствует некоторому классу.
- JSON-файл, содержит следующую информацию про объекты-элементы интерфейса: его bounding box, класс объекта, путь в zip-архиве до соответствующего объекту "вырезанного"изображения. Также важно упомянуть, что все объекты JSON-файла разбиты на 9 одинаковых по площади непересекающихся пространственных блоков для упрощения последующего формирования ответа сервиса. Принадлежность к некоторому блоку выбирается по тому, какому из блоков принадлежит центр изображения. Эмпирически было определено, что общая площадь блоков, выбираемых таким образом, и bounding box'ов объектов практически всегда максимальна по сравнению с площадями пересечений этих же bounding box'ов и других блоков, а следовательно, в восприятии человека скорее всего такой выбор наиболее логичен.
- zip-архив и JSON-файл передаются обратно основному сервису, вся информация, переданная данному сервису, удаляется.

5.2 Сервис детекции людей и предсказание возраста и пола по изображению

Для подзадачи детекции людей и предсказания возраста и пола человека по изображению был сформирован один Docker контейнер, так как модель предсказания работает непосредственно через детекцию. В этом сервисе используется фреймворк FastAPI.

Алгоритм работы сервиса начинается с обработки изображения, которое поступает через телеграм-бота. Инициализируется процесс детекции элементов GUI с использованием

специализированной модели. После успешной обработки сервис телеграм-бота делает запрос к API-ручке "age_estimate передавая zip-файл с классифицированными элементами. Из этого файла извлекаются объекты класса Image. Далее, изображения подаются на вход модели детекции, которая определяет наличие человека или группы людей на фотографии. В случае обнаружения человека, активируется модель MiVOLO, которая анализирует возраст и пол каждого лица на изображении, иначе возращается пустое содержание. Результаты работы модели образуются в формате JSON, содержащем мета-информацию по каждому изображению, включая координаты рамок обнаружения (bounding box), предсказанный возраст, пол и степень уверенности в предсказаниях пола.

В ходе экспериментов выявлено, что модель детекции может сталкиваться с трудностями в непривычных условиях, таких как обработка маленьких изображений или ошибки в классификации изображений. В таких случаях модель может ошибочно идентифицировать объекты, демонстрируя уверенность выше стандартного порога в 0.4. Для коррекции этих ошибок используется выход другой модели классификатора мультик / не мультик, который проводит дополнительную проверку реалистичности изображения. Таким образом, предсказания модели корректируются на основе анализа, нужно ли вообще проводить детекцию для данного изображения, основываясь на его реалистичности.

Для обеспечения интуитивно понятного вывода результатов детекции была разработана следующая эвристика: объекты, идентифицированные на изображении, перечисляются в порядке убывания площади их рамок детекции. Этот подход позволяет первоначально представить объекты, расположенные на переднем плане, последовательно переходя к объектам, расположенным дальше. При этом для сортировки используется максимальная площадь рамки (bounding box), независимо от того, относится ли она к туловищу или лицу. Эта методика улучшает визуальное восприятие результатов детекции, подчеркивая более крупные и значимые элементы на изображении. Кроме того, было решено не включать в число успешно детектированных объектов те случаи, когда на изображении присутствует только туловище или части тела без лица. Это решение было принято в связи с тем, что такие детекции снижают качество вывода модели и не представляют необходимости для выполнения задачи оценки возраста и пола.

В процессе формирования вывода результатов модели, возраст каждого идентифицированного человека арифметически округляется до ближайшего целого числа. На основе полученных данных далее формируется соответствующий ответ: если возраст детектированного человека меньше 18 лет, то в ответе указывается "boy"или "girl в зависимости от определённого пола. Для лиц старше 18 лет вывод форматируется как "man"или

"woman"соответственно. Сам конечный ответ выглядит, как "[age] years old man / woman / boy / girl / person где [age] — это округлённый возраст.

В процессе анализа результатов модели определения пола было установлено, что средняя уверенность модели при неверных предсказаниях составляет 0.924 на наборе данных UTKFace, в то время как для верных предсказаний этот показатель достигает 0.978. Поэтому было принято решение установить порог уверенности на уровне 0.94, что должно способствовать повышению точности оценки пола. В случаях, когда уверенность модели ниже этого порога, идентифицированный объект классифицируется как "person что позволяет избегать ошибочного определения пола при низкой уверенности предсказаний.

5.3 Сервис детекции и распознавания текста

Для того чтобы удобно развертывать и пользоваться моделями детекции и распознавания текста, был написан сервис с использованием фреймворка FastAPI и поднято окружение в докере отдельно для модели детекции и модели распознавания, что позволяет с легкостью изолировать процесс работы с моделями в различных условиях.

После обработки изображения,поданного на вход телеграм-боту, моделью детекции элементов GUI, сервис телеграм-бота обращается к ручке recognize сервиса с моделью распознавания. Мы распознаем текст на текстовых блоках и непосредственно на изображениях. Каждый блок проходит этап детекции текста (вызываем ручку detect сервиса с моделью детекции), а после каждое задетектированное поле вырезается из блока и подается на вход модели распознавания. Для ускорения совместной работы сервисов мы используем асинхронность, пользуясь библиотекой httpx для отправления асинхронных запросов в сервис детекции отдельно для текстовых блоков и для блоков с изображением. Дожидаясь первого результата детекции мы начинаем обрабатывать его: вырезаем картинки с текстом из блоков и подаём в модель распознавания. После того как закончится детекция другого типа блоков они аналогично обрабатываются.

Так как модель детекции выделяет небольшие части текста, отдельные слова, то после их распознавания внутри каждого из блоков отдельные слова надо собрать обратно в цельный текст. Для этого мы специальным образом сортируем части текста внутри каждого блока, основываясь на расположении этих частей текста. Цель - посортировать части таким образом чтобы порядок был схож с тем как человек прочитал бы этот текст. Так как мы читаем сверху вниз и слева направо, то первой идеей было посортировать задетектированные рамки по левому верхнему углу сначала по оси Oy от меньшего к большему а потом по

оси Ox также от меньшего к большему. Такой подход является достаточно простым, однако он не учитывает, что наши рамки текста могут слегка отличаться по оси Oy, но при этом будут находиться визуально в одной строке и соответственно человек их прочитал бы друг за другом. Поэтому для того чтобы компенсировать небольшой сдвиг по оси Oy было решено ввести параметр h, с помощью которого блок разбивался на строки равной ширины, и теперь вместо того чтобы сортировать по координате y, мы будем сортировать по номеру строки. Мы пробовала задавать h константным или брать равным половине средней высоты задетектированных рамок в блоке, чтобы алгоритм хорошо работал и в случае мелкого текста, и в случае крупного. Однако такой подход не учитывает, что в размерах задетектированных частей может быть достаточно сильный разброс. В таком случае нам бы хотелось чтобы ширина строчек была динамической. Поэтому предлагается следующий алгоритм: мы выбираем из еще не рассмотренных рамок ту, у которой левый верхний угол расположен выше, чем у всех остальных.Назовем найденную часть текста - ориентиром. Зададим ширину текущей строки равной половине высоты ориентира. И посортируем все части текста которые левым верхним углом попали в текущую строку по координате Ox (совместно с ориентиром), и запишем ответ в таком порядке. Далее повторяем алгоритм, пока не рассмотрим все рамки. Такая эвристика конечно покрывает не все случаи расположения частей текста, в некоторых случаях даже неочевидно в каком порядке человек прочитал бы эти части. Однако такой алгоритм показывает достаточно неплохие результаты.

В случае с текстовыми блоками, они могут содержать абсолютно разный объем текста: от одного слова до целых абзацев. В случае когда это одно слово, пропуск этапа с детекцией текста моей моделью – разумное решение для ускорения распознавания. Однако все текстовые блоки нельзя отправлять сразу в модель распознавания, так как это могут быть огромные абзацы, и модель не предполагает распознавание текста на картинке с многострочным текстом. Поэтому текстовые блоки отправляются прямо в модель распознавания, и также отправляются в модель детекции, откуда вырезанные задетектированные части текста обрабатываются моделью распознавания. Далее мы выбираем какой из распознанных текстов выбрать, основываясь на уверенности модели распознавания (средняя уверенность в случае если задетектированных частей текста несколько). Таким образом мы покрываем два возможных сценария и тем самым улучшаем качество распознавания.

6 Заключение

В рамках данной курсовой работы был успешно разработан сервис для описания изображений, предназначенный для помощи слабовидящим людям. Этот сервис способен описывать различный визуальный контент, встречающийся в повседневной жизни, включая фотографии, рисунки, а также графические интерфейсы сайтов и мобильных приложений. Данные сервис позволяет пользователям легко и удобно получать необходимую информацию и доступ к сервису прямо из популярного мессенджера. Итоговая информация об изображении может быть получена в нескольких форматах: текстовым сообщением, аудиозаписью или JSON-файлом. Планируется продолжать развивать данный сервис, чтобы сделать медиа контент более доступным для слабовидящих пользователей.

Список литературы

[1] Cong Yao Baoguang Shi Xiang Bai. «An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition». В: *arXiv* (2015). arXiv: arXiv:1507.05717v1 [cs.CV].

[2] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu и Magy Seif El-Nasr. «VINS: Visual Search for Mobile User Interface Design». В: *arXiv* (2021). arXiv: arXiv:2102.05216v1 [cs.HC].

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov и Sergey Zagoruyko. «End-to-End Object Detection with Transformers». В: *arXiv* (2020). arXiv: arXiv:2005.12872v3 [cs.CV].

[4] *Cartoons*. https://www.kaggle.com/datasets/volkandl/cartoon-classification. (дата обр. 13.02.2024).

[5] Zhanzhan Cheng Chengwei Zhang Yunlu Xu. «SPIN: Structure-Preserving Inner Offset Network for Scene Text Recognition». В: *arXiv* (2021). arXiv: arXiv:2005.13117v4 [cs.CV].

[6] *Clothes*. https://www.kaggle.com/datasets/rajkumarl/people-clothing-segmentation. (дата обр. 11.02.2024).

[7] Hugging Face Datasets. *SBU Captions Dataset*. https://huggingface.co/datasets/sbu_captions. (дата обр. 13.02.2024).

[8] Othón González-Chávez, Guillermo Ruiz, Daniela Moctezuma и Tania A. Ramirez-delReal. «Are metrics measuring what they should? An evaluation of image captioning task metrics». В: *arXiv* (2022). arXiv: arXiv:2207.01733 [cs.CV].

[9] *IMDB-WIKI: a dataset for age and gender estimation on movie stars*. https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/. (дата обр. 12.02.2024).

[10] Aditya Jain. *Flickr30k*. https://www.kaggle.com/datasets/adityajn105/flickr30k/data. (дата обр. 13.02.2024).

[11] Maksim Kuprashevich и Irina Tolstykh. «MiVOLO: Multi-input Transformer for Age and Gender Estimation». В: *arXiv* (2023). arXiv: arXiv:2307.04616v2 [cs.CV].

[12] Chae Young Lee, Youngmin Baek и Hwalsuk Lee. «TedEval: A Fair Evaluation Metric for Scene Text Detectors». В: *arXiv* (2019). arXiv: arXiv:1907.01227v1 [cs.CV].

[13] Junnan Li, Dongxu Li, Caiming Xiong и Steven Hoi. «BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation». В: *arXiv* (2022). arXiv: arXiv:2201.12086v2 [cs.CV].

[14] Xiang Li, Wenhai Wang, Wenbo Hou, Ruo-Ze Liu, Tong Lu и Jian Yang. «Shape Robust Text Detection with Progressive Scale Expansion Network». В: *arXiv* (2018). arXiv: arXiv:1806.02559v1 [cs.CV].

[15] Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen и Xiang Bai. «Real-time Scene Text Detection with Differentiable Binarization». В: *arXiv* (2019). arXiv: arXiv:1911.08947v2 [cs.CV].

[16] Minghui Liao, Zhisheng Zou, Zhaoyi Wan, Cong Yao и Xiang Bai. «Real-Time Scene Text Detection with Differentiable Binarization and Adaptive Scale Fusion». В: *arXiv* (2022). arXiv: arXiv:2202.10304v1 [cs.CV].

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick и Piotr Dollár. «Microsoft COCO: Common Objects in Context». В: *arXiv* (2015). arXiv: arXiv:1405.0312v3 [cs.CV].

[18] Yiming Lin, Jie Shen, Yujiang Wang и Maja Pantic. «FP-Age: Leveraging Face Parsing Attention for Facial Age Estimation in the Wild». В: *arXiv* (2021). eprint: 2106.11145 (cs.CV).

[19] Igor Markov, Sergey Nesteruk, Andrey Kuznetsov и Denis Dimitrov. «RusTitW: Russian Language Text Dataset for Visual Text in-the-Wild Recognition». В: *arXiv* (2023). arXiv: arXiv:2303.16531v1 [cs.CV].

[20] *mobile-ui-design: a dataset for object detection tasks with a focus on detecting elements in mobile UI designs*. https://huggingface.co/datasets/mrtoy/mobile-ui-design/. (дата обр. 01.02.2024).

[21] *Portraits*. https://www.kaggle.com/datasets/frenchbot/portrait-dataset-for-training-gans. (дата обр. 11.02.2024).

[22] Joseph Redmon, Santosh Divvala, Ross Girshick и Ali Farhadi. «You Only Look Once: Unified, Real-Time Object Detection». В: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, с. 779—788.

[23] Rasmus Rothe, Radu Timofte и Luc Van Gool. «Deep expectation of real and apparent age from a single image without facial landmarks». В: *International Journal of Computer Vision* 126.2-4 (2018), с. 144—157. DOI: 10.1007/s11263-017-1059-5.

[24] Yuxin Wang Shancheng Fang Hongtao Xie. «Read Like Humans: Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition». В: *arXiv* (2021). arXiv: arXiv:2103.06495v1 [cs.CV].

[25] *UTKFace: Large Scale Face Dataset*. https://susanqq.github.io/UTKFace/. (дата обр. 11.02.2024).

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser и Illia Polosukhin. «Attention Is All You Need». В: *arXiv* (2023). arXiv: arXiv:1706.03762v7 [cs.CL].

[27] Paul Viola и Michael Jones. «Rapid Object Detection using a Boosted Cascade of Simple Features». В: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*. Т. 1. IEEE. 2001, с. I—I.

[28] *VisionBot: Сервис распознавания картинок и текста на них*. https://visionbot.ru/. (дата обр. 14.02.2024).

[29] Chien-Yao Wang, I-Hau Yeh и Hong-Yuan Mark Liao. «YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information». В: *arXiv* (2024). arXiv: arXiv:2402.13616 [cs.CV].

[30] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu и Lijuan Wang. «GIT: A Generative Image-to-text Transformer for Vision and Language». В: *arXiv* (2022). arXiv: arXiv:2205.14100v5 [cs.CV].

[31] Pengfei Wang, Chengquan Zhang, Fei Qi, Zuming Huang, Mengyi En, Junyu Han, Jingtuo Liu, Errui Ding и Guangming Shi. «A Single-Shot Arbitrarily-Shaped Text Detector based on Context Attended Multi-Task Learning». В: *arXiv* (2019). arXiv: arXiv:1908.05498v1 [cs.CV].

[32] *WebUI: A Dataset for Enhancing Visual UI Understanding with Web Semantics*. https://github.com/js0nwu/webui/tree/main?tab=readme-ov-file/. (дата обр. 12.02.2024).

[33] Chenhao Lin Xiaoyu Yue Zhanghui Kuang. «RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition». В: *arXiv* (2020). arXiv: arXiv:2007.07542v2 [cs.CV].

[34] Caiyan Jia Yongkun Du Zhineng Chen. «SVTR: Scene Text Recognition with a Single Visual Model». В: *arXiv* (2022). arXiv: arXiv:2205.00159v2 [cs.CV].

[35] Li Yuan, Qibin Hou, Zihang Jiang, Jiashi Feng и Shuicheng Yan. «VOLO: Vision Outlooker for Visual Recognition». В: *arXiv* (2021). eprint: 2106.13112v2 (cs.CV).

[36] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li и Yu Qiao. «Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks». B: *IEEE Signal Processing Letters* 23.10 (2016), с. 1499—1503.

[37] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He и Jiajun Liang. «EAST: An Efficient and Accurate Scene Text Detector». B: *arXiv* (2017). arXiv: arXiv:1704.03155v2 [cs.CV].

[38] Yiqin Zhu, Jianyong Chen, Lingyu Liang, Zhanghui Kuang, Lianwen Jin и Wayne Zhang. «Fourier Contour Embedding for Arbitrary-Shaped Text Detection». B: *arXiv* (2021). arXiv: arXiv:2104.10442v2 [cs.CV].

[39] Zhuofan Zong, Guanglu Song и Yu Liu. «DETRs with Collaborative Hybrid Assignments Training». B: *arXiv* (2023). arXiv: arXiv:2211.12860v6 [cs.CV].