МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ

«АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет математики и информационных технологий

Кафедра информатики

**Разработка веб–сервиса по автоматическому созданию текстовых описаний изображений с использованием нейронных сетей Hugging Face Transformers**

выпускная квалификационная работа

(магистерская диссертация)

Выполнил:

магистрант группы 4.306М–3,

Графова Анастасия Вениаминовна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

Научный руководитель:

к.ф.–м.н., доцент

Ласковец Екатерина Валерьевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

Допустить к защите:

Зав. кафедрой, к.ф.–м.н., доцент

Козлов Денис Юрьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Работа защищена:

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ГЭК:

д.т.н., профессор

Леонов Сергей Леонидович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2025

**РЕФЕРАТ**

Тема магистерской диссертации: «Разработка веб–сервиса по автоматическому созданию текстовых описаний изображений с использованием нейронных сетей Hugging Face Transformers».

Цель работы – разработка веб–сервиса, который будет автоматически генерировать текстовые описания изображений с использованием нейронных сетей библиотеки Hugging Face Transformers.

Предмет исследования – применение нейронных сетей библиотеки Hugging Face Transformers.

Объект исследования – создание текстовых описаний изображений с помощью нейронных сетей.

В ходе магистерской работы были решены следующие задачи: проанализированы современные подходы к генерации текстовых описаний изображений и существующие веб–сервисы в данной области, разработан и подготовлен собственный датасет с аннотациями, выполнено дообучение модели BLIP–large на этих данных, реализован веб–сервис с функцией автоматической генерации описаний и их оценки с помощью метрик BLEU, ROUGE–L и METEOR. Проведено тестирование и сравнительный анализ с аналогичными решениями.

Объем работы – 75 страниц, количество рисунков – 30, таблиц – 2, приложений – 3, 36 использованных источников литературы.

Ключевые слова: нейронные сети, текстовые описания изображений, Hugging Face Transformer.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc200828601)

[Глава 1. Анализ предметной области и существующих решений 8](#_Toc200828602)

[1.1. Эволюция технологий обработки изображений 8](#_Toc200828603)

[1.2. Современные мультимодальные технологии 21](#_Toc200828604)

[1.3. Обзор существующих веб–сервисов для генерации текстовых описаний изображений 29](#_Toc200828605)

[Глава 2. Разработка и обучение модели на основе Hugging face transformers 34](#_Toc200828606)

[2.1. Инструменты разработки 34](#_Toc200828607)

[2.2 Подготовка и предобработка данных для обучения модели 41](#_Toc200828608)

[2.3. Оптимизация модели для повышения точности и скорости работы 43](#_Toc200828609)

[Глава 3. Разработка веб–сервиса для автоматической генерации текстовых описаний 47](#_Toc200828610)

[3.1. Реализация клиентской части: HTML, CSS, JavaScript 47](#_Toc200828611)

[3.2. Реализация серверной части и интеграция модели BLIP 50](#_Toc200828612)

[Глава 4. Тестирование и оценка качества разработанного сервиса 56](#_Toc200828613)

[4.1. Тестирование сервиса на реальных данных 56](#_Toc200828614)

[4.2. Оценка точности сгенерированных текстовых описаний 59](#_Toc200828615)

[4.3. Сравнительный анализ с существующими решениями 61](#_Toc200828616)

[ЗАКЛЮЧЕНИЕ 68](#_Toc200828617)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 70](#_Toc200828618)

[ПРИЛОЖЕНИЯ 74](#_Toc200828619)

# **ВВЕДЕНИЕ**

Сегодня, когда искусственный интеллект и машинное обучение уже не просто технические тренды, а повседневные инструменты, ключевым направлением развития становятся нейросетевые технологии, способные не просто обрабатывать данные, а «понимать» их с разных сторон. Особенно это заметно в сфере мультимодальных моделей – архитектур, способных работать с несколькими типами информации одновременно: изображениями, текстами, звуками и даже видеофрагментами. Один из наиболее интересных и практически значимых векторов здесь – генерация текстовых описаний по изображениям.

Эта задача на первый взгляд может показаться простой: «увидел – рассказал». Но на практике она требует не только точного распознавания объектов, но и способности уловить контекст сцены, выразить его словами, сохранить логическую связность и ещё при этом быть понятной как человеку, так и машине. Это особенно актуально, например, в системах помощи людям с нарушением зрения, в интеллектуальных поисковых движках или при формировании метаданных для цифровых архивов.

При этом классические методы, такие как жёстко заданные шаблоны или ручная аннотация изображений, сегодня уже не соответствуют требованиям ни по точности, ни по масштабируемости. Их ограниченность становится особенно заметной в тех случаях, когда требуется не просто назвать объекты, но и передать тонкие нюансы сцены – настроение, действие, взаимоотношение между элементами. Современные решения опираются на достижения в области трансформеров и глубоких языковых моделей, но даже с их помощью задача остаётся сложной, многослойной и всё ещё открытой для совершенствования.

Объектом исследования является процесс автоматического создания текстовых описаний изображений. Для этого используются данные из различных источников, таких как базы стоковых изображений, датасеты с текстовыми аннотациями и социальные сети.

Предметом данной работы является исследование применения нейронных сетей библиотеки Hugging Face Transformers, в частности модели BLIP, для решения задачи автоматической генерации текстовых описаний изображений.

Целью данной работы является разработка веб–сервиса, который будет автоматически генерировать текстовые описания изображений с использованием нейронных сетей библиотеки Hugging Face Transformers, реализованного на языке программирования Python.

В соответствии с целью, поставлены следующие задачи:

1. Изучить существующие решения в области генерации аннотаций и собрать данные для собственного набора, на котором будет предобучаться модель.

2. Проанализировать структуру и содержание собранных данных, определить параметры, оказывающие наибольшее влияние на качество генерации.

3. Выполнить предобучение модели BLIP на собственноручно собранном датасете, оптимизировав ключевые параметры генерации.

4. Разработать веб–сервис, интегрирующий предобученную модель, обеспечив удобный пользовательский интерфейс.

5. Провести тестирование разработанного сервиса и оценить качество получаемых текстовых описаний.

6. Провести сравнительный анализ предложенной модели с существующими решениями, оценив точность, скорость работы и качество сгенерированных описаний.

Гипотеза: использование моделей BLIP из библиотеки Hugging Face позволит добиться высокого уровня семантической связности и выразительности описаний.

При выполнении работы использовались теоретические (изучение литературы, обобщение, анализ) и практические методы исследования (тестирование).

Теоретическая значимость исследования состоит в улучшении понимания о методах автоматического создания текстовых описаний изображений с использованием современных нейросетевых моделей Hugging Face Transformers.

Практическая значимость исследования состоит в том, что разработанный веб–сервис может быть использован в различных сферах, таких как цифровой маркетинг, электронная коммерция, автоматизация архивных данных и доступность информации для людей с нарушениями зрения.

Работа разделена на две основные части – теоретическую и практическую. В теоретической части проводится анализ эволюции методов обработки изображений и текста, начиная с зарождения свёрточных нейронных сетей и заканчивая современными мультимодальными подходами. Особое внимание уделяется изучению возможностей библиотеки Hugging Face Transformers и модели BLIP (Bootstrapped Language–Image Pre–training) как инструментов для автоматической генерации текстовых описаний. Кроме того, рассматриваются теоретические аспекты языка программирования Python, его библиотек, а также метрик, применяемых для оценки качества сгенерированных текстовых аннотаций.

В практической части была разработана модель автоматической генерации текстовых описаний изображений на основе нейронных сетей Hugging Face Transformers. На базе этой модели создан веб–сервис, который позволяет пользователям загружать изображения и получать их текстовые аннотации.

Использование разработанного веб–сервиса упрощает процесс создания текстовых описаний изображений, что может быть полезно в таких областях, как цифровой маркетинг, электронная коммерция, автоматизация архивных данных и обеспечение доступности информации для людей с нарушениями зрения. Разработанный сервис позволяет сократить время, необходимое для ручного создания описаний, а также минимизировать ошибки и несоответствия.

# **Глава 1. Анализ предметной области и существующих решений**

Глава посвящена анализу предметной области, связанной с автоматической генерацией текстовых описаний изображений, а также обзору существующих решений в данной области. В условиях стремительного развития технологий искусственного интеллекта и машинного обучения обработка мультимодальных данных, включающих изображения и текст, приобретает всё большее значение [13, 19]. Автоматизация задач, таких как классификация изображений, сегментация и генерация текстовых аннотаций, открывает новые возможности для применения в различных сферах – от цифрового маркетинга и электронной коммерции до обеспечения доступности контента для людей с ограниченными возможностями [28]. В рамках главы рассматривается эволюция технологий обработки изображений, современные мультимодальные подходы, а также существующие веб–сервисы, предназначенные для генерации текстовых описаний изображений [1, 2].

## **1.1. Эволюция технологий обработки изображений**

Эволюция технологий обработки изображений и текста охватывает значительный период – от зарождения свёрточных нейронных сетей (Convolutional Neural Networks, CNN) в конце XX века до современных мультимодальных моделей, способных интегрировать анализ визуальных и текстовых данных [11, 26]. Этот процесс отражает значительный прогресс в области искусственного интеллекта и машинного обучения, обеспечивший автоматизацию таких задач, как классификация изображений, сегментация, распознавание объектов и генерация текстовых описаний [13, 24].

Ключевым этапом в развитии технологий обработки изображений стало внедрение глубоких нейронных сетей, в частности свёрточных нейронных сетей (CNN). Начиная с пионерской архитектуры LeNet5 (1994) и заканчивая более сложными моделями, такими как AlexNet (2012), VGG (2014), ResNet (2015) и ENet (2016), CNN стали основным инструментом для извлечения признаков из визуальных данных, обеспечивая высокую точность в задачах компьютерного зрения [16], [22]. Каждая из этих архитектур внесла вклад в развитие технологий: LeNet5 заложила основы свёрточных операций, AlexNet продемонстрировала эффективность глубоких сетей на больших датасетах, VGG показала преимущества малых фильтров, ResNet ввела остаточные связи для обучения сверхглубоких сетей, а ENet предложила компактное решение для задач реального времени [19].

1.1.1. Свёрточная нейронная сеть архитектуры LeNet5

В 1994 году была представлена одна из первых свёрточных нейронных сетей, заложившая основы для развития концепции глубокого обучения. Данная разработка, выполненная Яном Лекуном (Yann LeCun) и являющаяся результатом многолетних исследований, начатых ещё в 1988 году, получила название LeNet5 [22]. Эта архитектура стала важным этапом в эволюции нейросетевых технологий, особенно в контексте обработки изображений, и определила ключевые принципы, которые впоследствии легли в основу современных подходов к глубокому обучению (рис. 1.1.) [18].

Изображение выглядит как текст, линия, диаграмма, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.1. Схема архитектуры CNN LeNet5

Архитектура LeNet5 отличается фундаментальным значением для анализа изображений благодаря своей способности эффективно учитывать пространственное распределение признаков по всему входному изображению. Использование свёрточных операций с обучаемыми параметрами позволило извлекать идентичные свойства из различных участков изображения с применением ограниченного числа параметров. В условиях того времени, когда отсутствовали графические процессоры (GPU), способные ускорить вычисления, а производительность центральных процессоров (CPU) оставалась ограниченной, такая оптимизация имела решающее значение. В отличие от традиционных многослойных нейронных сетей, которые рассматривали каждый пиксель изображения как отдельный входной параметр, LeNet5 предлагала более рациональный подход. Сохранение параметров свёртки и результатов промежуточных вычислений существенно снижало вычислительную нагрузку, что являлось важным преимуществом в условиях ограниченных ресурсов [22].

Особое внимание в LeNet5 уделено отказу от использования отдельных пикселей в качестве входных данных на первом слое. Это решение обусловлено высокой пространственной корреляцией пикселей в изображениях. Прямое применение пикселей как независимых признаков не позволяло бы эффективно использовать эту корреляцию, что снижало бы качество извлечения значимых характеристик. Вместо этого модель опиралась на свёрточные операции, которые обеспечивали обобщение пространственных зависимостей [16].

Архитектура LeNet5, предложенная Яном Лекуном в середине 90–х годов, считается одной из первых успешно применённых свёрточных нейросетей, положивших начало эпохе глубокого обучения в задачах визуального анализа. В основе этой модели – стройная и продуманная последовательность трёх ключевых типов слоёв: свёрточных, подвыборки (pooling) и активационных. Такая комбинация позволила эффективно извлекать и иерархически обрабатывать визуальные признаки, от простых геометрических контуров до более абстрактных характеристик.

На первом этапе модель применяет свёрточные фильтры – ядра, которые сканируют изображение и выделяют локальные признаки вроде границ, углов и текстур. Эти фильтры не фиксированы, как в традиционных алгоритмах обработки изображений, а обучаются во время тренировки, благодаря чему сеть адаптируется к спецификам конкретного визуального контента.

Далее следуют слои подвыборки. Они уменьшают пространственное разрешение карты признаков, оставляя только самую важную информацию. Такой подход не только сокращает количество вычислений, но и повышает устойчивость модели к незначительным смещениям и искажениям входных данных.

В роли активационных функций в LeNet5 использовались tanh и sigmoid – на тот момент это были основные средства для введения нелинейности, необходимой для приближения сложных зависимостей. Несмотря на известные ограничения этих функций (например, проблема исчезающего градиента), их использование позволило добиться заметных успехов в задачах распознавания рукописного текста.

На выходе LeNet5 располагается многослойный перцептрон, выполняющий окончательную классификацию. Особенность архитектуры – в частичности связей между слоями. Вместо полной плотной структуры, здесь используется разрежённая схема, благодаря которой модель становится легче и быстрее, что критично при работе в условиях ограниченных ресурсов.

Именно изящная простота, структурная экономия и чёткая логика слоёв сделали LeNet5 отправной точкой для более мощных архитектур – AlexNet, VGG и ResNet. Эти сети не просто наследовали идеи Лекуна, а трансформировали их под задачи нового масштаба: от анализа медицинских изображений до распознавания лиц в потоковом видео [22, 26]. LeNet5 по праву считается той основой, без которой бурное развитие компьютерного зрения в XXI веке было бы невозможно.

1.1.2. Свёрточная нейронная сеть архитектуры AlexNet

В 2012 году Алексей Крижевский совместно с коллегами представил архитектуру свёрточной нейронной сети AlexNet, которая стала значительным шагом вперёд по сравнению с более ранними моделями, такими как LeNet. AlexNet одержала победу в соревновании ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 года, продемонстрировав выдающуюся производительность в задаче классификации изображений. Данная архитектура представляет собой углублённую и расширенную версию концепций, заложенных в LeNet, адаптированную для работы с более сложными датасетами и задачами, включающими распознавание объектов с высокой степенью вариативности (рис. 1.2.) [21].

Изображение выглядит как зарисовка, диаграмма, План, рисунок

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.2. Схема архитектуры AlexNet

Основное отличие AlexNet от предшествующих моделей заключается в её способности эффективно обучаться на больших наборах данных, таких как ImageNet, содержащих миллионы изображений и тысячи классов. Это стало возможным благодаря увеличению глубины сети и внедрению ряда архитектурных и методологических нововведений, которые обеспечили значительное улучшение качества классификации. Рассмотрим ключевые особенности AlexNet:

* В отличие от традиционных функций активации, таких как гиперболический тангенс или сигмоид, использовавшихся в LeNet, AlexNet применяет линейную ректификацию (Rectified Linear Unit, ReLU) в качестве нелинейности. ReLU, определяемая как:

ускоряет сходимость градиентного спуска за счёт устранения проблемы исчезающего градиента, что особенно важно для глубоких сетей [19];

* Для предотвращения переобучения в полносвязных слоях AlexNet используется техника dropout, которая заключается в случайном отключении части нейронов во время обучения с заданной вероятностью (обычно 0.5). Это позволяет модели лучше обобщать, снижая зависимость от отдельных нейронов и способствуя более устойчивому обучению [18].
* В отличие от усредняющего пулинга (average pooling), применявшегося в LeNet, AlexNet использует максимальный пулинг (max pooling) с перекрытием (overlapping). Например, пулинг с окном 3×3 и шагом 2 (вместо шага, равного размеру окна) позволяет сохранить больше пространственной информации, избегая чрезмерного сглаживания, характерного для усредняющего пулинга [11].
* Для ускорения обучения AlexNet была реализована с использованием двух графических процессоров NVIDIA GTX 580. К 2012 году значительный рост числа вычислительных ядер в GPU позволил сократить время обучения примерно в 10 раз по сравнению с традиционными CPU [19].

AlexNet открыла дорогу целому поколению архитектур – VGG, Inception, ResNet – каждая из которых опиралась на её принципы и дополняла их собственными находками. Более того, с её победы термин «deep learning» стал означать не абстрактную идею, а конкретный, рабочий подход, способный распознавать образы, интерпретировать визуальные данные и находить закономерности в сложных структурах [16].

1.1.3. Свёрточная нейронная сеть архитектуры VGG

Модель VGG, созданная исследователями из Visual Geometry Group Оксфордского университета, по праву считается одним из важнейших этапов в развитии глубоких нейросетей. В отличие от своих предшественников, она предложила изящное и в то же время мощное решение: замену громоздких фильтров на серию небольших 3×3 свёрток. Такой подход оказался не только эффективным, но и концептуально новым – он позволил увеличить глубину сети, сохранив при этом стабильность и управляемое количество параметров [19].

Если в LeNet и AlexNet архитекторы отдавали предпочтение крупным ядрам (5×5, 7×7, вплоть до 11×11), то VGG сделала ставку на «узкие, но частые» операции. На первый взгляд идея могла показаться рискованной: свёртки 3×3 воспринимались как чересчур локальные, с ограниченной способностью к захвату пространственного контекста. Однако именно последовательное наложение таких фильтров оказалось способным имитировать более широкие рецептивные поля. Например, три подряд идущих слоя с 3×3 свёртками обеспечивали охват, эквивалентный одному 7×7 фильтру, но с меньшими вычислительными затратами и лучшей адаптивностью. Этот подход не только сохраняет способность сети извлекать сложные признаки, но и уменьшает количество параметров в каждом отдельном слое, что делает архитектуру более гибкой и масштабируемой. Данные идеи впоследствии были адаптированы и развиты в архитектурах Inception и ResNet [10].

Изображение выглядит как текст, Шрифт, чек, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.3. Конфигурация CNN архитектуры VGG

Для представления сложных признаков VGG использует глубокие последовательности свёрточных слоёв (рис. 1.3.). Например, в конфигурации VGG–E (19 слоёв) блоки 3, 4 и 5 содержат последовательности свёрточных слоёв с 256 и 512 фильтрами 3×3. Так, блок из трёх последовательных свёрточных слоёв с 512 фильтрами эквивалентен одному свёрточному слою с рецептивным полем 7×7, но с меньшим числом параметров и большей глубиной, что способствует лучшему обучению сложных зависимостей. Однако такая глубина и большое количество фильтров приводят к значительному росту числа параметров, что увеличивает вычислительную сложность. Например, конфигурация VGG–E содержит 144 миллиона параметров, что делает её одной из самых "тяжёлых" в семействе VGG [23].

Обучение таких глубоких сетей представляет собой сложную задачу. В отсутствие современных методов регуляризации, таких как dropout или batch normalization, которые стали стандартом в более поздних архитектурах, обучение VGG требовало значительных вычислительных ресурсов и тщательной настройки. Для упрощения процесса исследователи часто прибегали к поэтапному обучению: сначала обучали более мелкие конфигурации (например, VGG–A с 11 слоями), а затем постепенно добавляли новые слои, дообучая сеть. Это позволяло минимизировать проблемы, связанные с большим пространством параметров и отсутствием эффективных методов ограничения поиска оптимальных весов [26].

Высокая вычислительная сложность VGG обусловлена большим количеством признаков, извлекаемых на каждом слое. Для снижения нагрузки в последующих архитектурах, таких как Inception, были предложены bottleneck–слои, которые уменьшают количество фильтров перед применением свёртки, тем самым сокращая объём вычислений. Несмотря на эти ограничения, VGG остаётся важной вехой в развитии свёрточных нейронных сетей, демонстрируя эффективность глубоких архитектур с малыми фильтрами и последовательным объединением слоёв для задач классификации изображений [25].

1.1.4. Свёрточная нейронная сеть архитектуры ResNet

В декабре 2015 года, одновременно с представлением архитектуры Inception v3, была опубликована работа, посвящённая свёрточной нейронной сети ResNet (Residual Network), которая ознаменовала значительный прогресс в области глубокого обучения [11]. ResNet, разработанная исследователями из Microsoft Research, предложила инновационный подход к обучению глубоких нейронных сетей, основанный на использовании остаточных связей (residual connections). Этот подход позволил успешно обучать сети с беспрецедентной глубиной, достигающей сотен и даже тысяч слоёв, что ранее считалось практически невозможным из–за проблемы исчезающего градиента.

Ключевой идеей ResNet является введение остаточных блоков, в которых выход двух последовательных свёрточных слоёв складывается с входным сигналом , а результат передаётся на следующий слой после применения функции активации. Такой подход, известный как "identity shortcut" или остаточная связь, позволяет сети обучать остаточную функцию , где – желаемое отображение. Это упрощает процесс обучения, поскольку сеть может сосредоточиться на изучении разницы между входом и выходом, а в случае, если дополнительные слои не требуются, остаточная функция может быть близка к нулю, что эквивалентно пропуску этих слоёв (рис. 1.4.).

Изображение выглядит как текст, Шрифт, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.4. Общий вид остаточного блока основного строительного элемента архитектуры ResNet

Хотя идея остаточных связей обсуждалась ранее в литературе, ResNet впервые применила этот подход в больших масштабах, обходя сразу два свёрточных слоя [23]. Обход одного слоя, как показали эксперименты, не давал значительного улучшения, тогда как обход двух слоёв оказался ключевым новшеством. Такой остаточный блок можно интерпретировать как небольшой классификатор внутри сети, что делает ResNet своего рода "сетью в сети". Эта концепция позволила значительно увеличить глубину сети без потери производительности.

Для управления вычислительной сложностью в глубоких версиях ResNet (например, ResNet–50, ResNet–101, ResNet–152) был внедрён bottleneck–дизайн, аналогичный используемому в архитектуре Inception. Bottleneck–блок состоит из трёх свёрточных слоёв: первый слой (1×1) уменьшает размерность входных данных (например, с 256 до 64 фильтров), второй слой (3×3) выполняет основную обработку, а третий слой (1×1) восстанавливает исходную размерность (например, обратно до 256). Такой подход позволяет существенно сократить количество параметров и вычислений, сохраняя при этом способность сети комбинировать сложные признаки (рис. 1.5.). В сравнении с более сложными stem–модулями в Inception v3 и v4, bottleneck–блоки ResNet отличаются большей простотой и вычислительной эффективностью.

Изображение выглядит как текст, диаграмма, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.5. Пример bottleneck–блока, используемого в более глубоких версиях ResNet

На финальном этапе ResNet использует слой глобального усредняющего пулинга (global average pooling), за которым следует слой softmax для классификации. Этот подход заменил традиционные полносвязные слои, что дополнительно уменьшило количество параметров и снизило риск переобучения.

Архитектура ResNet также допускает различные интерпретации её структуры. Одной из них является представление сети как комбинации параллельных и последовательных модулей: входной сигнал в остаточных блоках передаётся параллельно через shortcut–связь, тогда как выходы блоков соединяются последовательно. Кроме того, ResNet можно рассматривать как ансамбль модулей, где сеть оперирует блоками относительно небольшой глубины (20–30 слоёв), работающими параллельно, а не как единая последовательная структура.

ResNet задала новую архитектурную логику, которая сегодня применяется в десятках производных моделей – от ResNeXt до EfficientNet. Именно благодаря ResNet стало возможным обучение сверхглубоких сетей без разрушения градиентов, что сделало её одной из наиболее значимых разработок в истории компьютерного зрения.

1.1.5. Свёрточная нейронная сеть архитектуры ENet

ENet (Efficient Neural Network), созданная Адамом Пазке, стала ответом на растущую потребность в лёгких, но эффективных архитектурах для задач реального времени. В отличие от громоздких моделей с миллионами параметров, ENet демонстрирует, что можно добиться высокой точности, не перегружая вычислительные ресурсы. Именно благодаря такому подходу эта архитектура особенно востребована в мобильных приложениях, робототехнике и встраиваемых системах, где скорость и энергоэффективность – на первом месте.

ENet основана на концепции энкодер–декодерной архитектуры. Энкодер, построенный по традиционной схеме свёрточной нейронной сети (CNN), выполняет задачу извлечения признаков и категоризации, постепенно уменьшая пространственное разрешение входного изображения. Декодер, в свою очередь, представляет собой сеть с повышением дискретизации (upsampling network), которая восстанавливает исходное разрешение изображения, распространяя категориальные признаки обратно на пиксельный уровень для выполнения сегментации. Важно отметить, что ENet использует исключительно нейросетевые методы для сегментации, не прибегая к дополнительным алгоритмам, что подчёркивает её универсальность и автономность.

Изображение выглядит как текст, снимок экрана, График, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.6. Сравнение различных архитектур CNN по удельной точности

Одной из ключевых особенностей ENet является её высокая удельная точность (top–1 accuracy density), которая, согласно сравнительному анализу, превосходит показатели других известных архитектур, таких как VGG, ResNet, Inception и GoogLeNet (рис. 1.6.) [31]. Например, при объёме параметров всего 0.7 миллиона (в формате fp16) ENet демонстрирует точность сегментации, сопоставимую или превосходящую более тяжёлые нейросетевые модели, такие как ResNet–50 или VGG–16, которые содержат десятки и сотни миллионов параметров. Такая эффективность делает ENet особенно подходящей для применения на устройствах с ограниченными ресурсами, например, в мобильных системах или встраиваемых устройствах.

Систематическая оценка модулей свёрточных нейронных сетей, проведённая в рамках исследований, выявила ряд рекомендаций, которые были учтены при разработке ENet и могут быть полезны для оптимизации других архитектур:

* Использование нелинейности ELU (Exponential Linear Unit) без пакетной нормализации (batch normalization) или ReLU с нормализацией показало лучшие результаты в сравнении с традиционными подходами.
* Применение обучаемой трансформации цветового пространства RGB позволяет улучшить качество извлечения признаков.
* Использование линейного уменьшения скорости обучения (linear learning rate decay) способствует более стабильной сходимости.
* Суммирование результатов среднего и максимального пулинга (average и max pooling) повышает качество извлечения пространственных признаков.
* Оптимальный размер мини–пакета составляет 128 или 256. При уменьшении размера пакета (например, из–за ограничений памяти видеокарты) рекомендуется пропорционально снижать скорость обучения.
* Использование полносвязных слоёв в качестве свёрточных с последующим усреднением прогнозов для получения финального результата улучшает обобщающую способность модели.
* При увеличении размера обучающего датасета необходимо убедиться, что обучение не достигло плато, а также уделять внимание чистоте данных, которая имеет большее значение, чем их объём.
* Если увеличение размера входного изображения невозможно, снижение шага (stride) в последующих слоях может дать аналогичный эффект.
* Для сетей с высокооптимизированной структурой, таких как GoogLeNet, любые изменения должны вноситься с осторожностью, чтобы не нарушить баланс между производительностью и эффективностью.

ENet демонстрирует выдающуюся эффективность в задачах сегментации изображений, обеспечивая высокую точность при минимальном объёме параметров. Её архитектурные решения и оптимизации делают её перспективным инструментом для применения в условиях ограниченных вычислительных ресурсов, а также служат основой для дальнейших исследований в области компактных и производительных нейронных сетей.

## **1.2. Современные мультимодальные технологии**

Мультимодальные модели (Multimodal Large Language Models, MLLM) и традиционные большие языковые модели (Large Language Models, LLM) представляют собой два различных подхода к обработке данных, хотя оба базируются на нейронных сетях. Основное различие между ними заключается в типах данных, с которыми они работают, и в задачах, которые они способны решать.

Традиционные LLM, такие как GPT–3 или BERT, разработаны исключительно для работы с текстовой модальностью. Их функционал включает задачи обработки естественного языка (NLP), такие как анализ текста, генерация текстов, перевод. LLM не может установить связь между текстом, например, "собака на пляже" и изображением, на котором изображена эта сцена, так как она не способна обработать визуальную информацию.

Мультимодальные модели, напротив, способны выявлять сложные взаимосвязи между различными модальностями. Например, в задаче визуального вопрос–ответного диалога (Visual Question Answering, VQA) мультимодальная модель может ответить на вопрос "Какого цвета мяч на изображении?", анализируя как текст вопроса, так и визуальные данные. Взаимодействие между модальностями делает такие модели особенно полезными в сценариях, где информация из разных источников дополняет друг друга, обеспечивая более полное понимание контекста.

Архитектура традиционных LLM базируется на трансформерных моделях, которые обрабатывают последовательности токенов в тексте. Такие модели обучаются предсказывать следующий токен в последовательности или выполнять задачи, связанные с пониманием текста, такие как классификация или извлечение информации. Трансформеры, используемые в LLM, эффективно справляются с задачами NLP, но не могут напрямую работать с визуальными или другими типами данных.

С развитием трансформерных архитектур, начиная с 2017 года, технологии обработки изображений и текста претерпели значительные изменения, что открыло новые возможности для интеграции мультимодальных данных [35]. В 2017 году была представлена модель Transformer. Изначально разработанная для задач обработки естественного языка (Natural Language Processing, NLP), Transformer заменила рекуррентные нейронные сети (RNN) благодаря механизму внимания (attention mechanism). Этот механизм позволяет эффективно моделировать зависимости между словами в последовательности, независимо от их расстояния друг от друга, что значительно ускорило обучение и улучшило качество обработки текста в задачах, таких как машинный перевод, генерация текста и анализ тональности.

Применение трансформерных архитектур к обработке изображений началось позже, с появлением Vision Transformer (ViT) в 2020 году. ViT полностью отказался от свёрточных слоёв, традиционно используемых в CNN, заменив их трансформерными блоками (Рис. 1.7.). В ViT изображение разбивается на фиксированные патчи (например, 16×16 пикселей), которые преобразуются в последовательность эмбеддингов с добавлением позиционных эмбеддингов для учёта пространственного расположения. Эти эмбеддинги затем обрабатываются трансформером, аналогично тому, как трансформеры обрабатывают текст. ViT продемонстрировал высокую эффективность на задачах классификации ImageNet: например, ViT–Large достиг топ–1 точности около 88.5%, что превзошло многие свёрточные архитектуры, такие как ResNet–152, при меньших вычислительных затратах на этапе инференса.

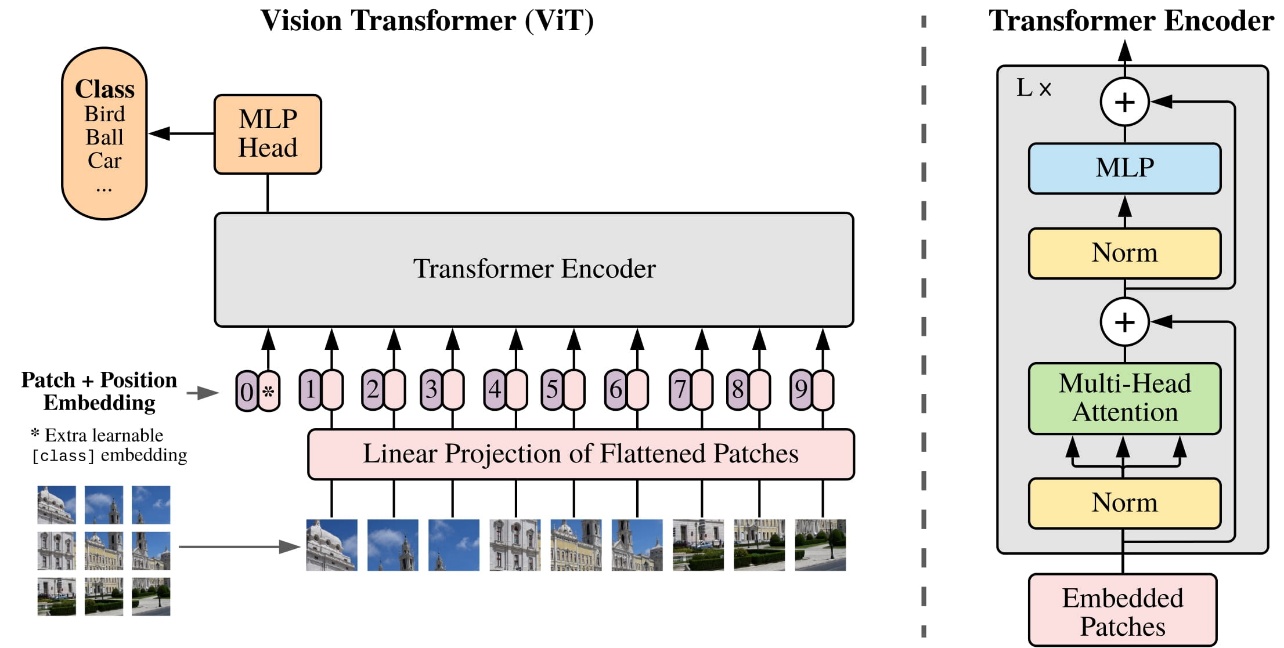


Рис. 1.7. Архитектура ViT

Успех трансформеров в обработке изображений и текста стал основой для развития мультимодальных моделей. Одной из первых таких моделей стала CLIP, представленная OpenAI в 2021 году. CLIP использует ViT для обработки изображений и трансформерный энкодер для текста, обучаясь на парах "изображение–текст" с помощью контрастивного обучения (Рис. 1.8.). Это позволяет модели устанавливать семантические связи между визуальной и текстовой информацией, что делает её эффективной для задач zero–shot классификации и поиска изображений по текстовым запросам.

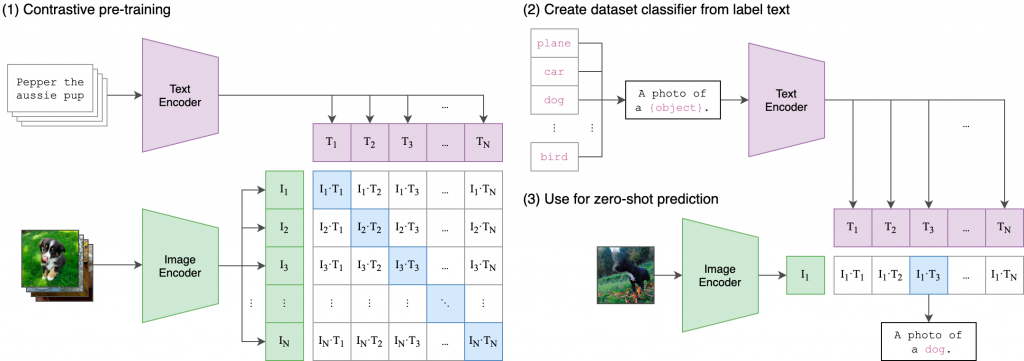


Рис. 1.8. Схема работы модели CLIP

Дальнейшее развитие мультимодальных технологий привело к появлению модели BLIP, представленной в 2022 году. Ее архитектура основана на сочетании визуальных и языковых трансформеров, что позволяет эффективно анализировать визуальные объекты и формировать их текстовые интерпретации. Кроме того, BLIP интегрирует принципы обучения с подкреплением, что способствует созданию высококачественных текстовых описаний. BLIP была создана как универсальная мультимодальная модель, объединяющая обработку изображений и текста, и предназначена для задач:

* Автоматической генерации текстовых описаний изображений (Image Captioning).
* Вопросно–ответных систем по изображениям (Visual Question Answering, VQA).
* Оценки соответствия изображения и текста (Image–Text Matching).

Поскольку в рамках данной дипломной работы предполагается использование модели BLIP, то подробнее рассмотрим каждую из этих задач и разберем, как именно BLIP справляется с их решением.

Задача автоматической генерации текстовых описаний изображений (Image Captioning) заключается в создании текстового описания, отражающего содержимое изображения. BLIP решает эту задачу, эффективно комбинируя визуальную и текстовую информацию, что позволяет генерировать точные и контекстуально релевантные описания (Рис. 1.9.) [3].

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.9. – Cхема работы Image Captioning

Сначала нейросеть "смотрит" на изображение с помощью специального модуля (Image Encoder), который помогает понять, что на нем изображено. Потом другая часть нейросети (Text Decoder) берет эту информацию и превращает ее в текст. Чтобы описание было точным, модель использует механизм внимания (Cross–Attention), который помогает связать объекты на изображении со словами [12].

Задача визуального вопрос–ответного диалога (Visual Question Answering, VQA) заключается в предоставлении ответа на вопрос, связанный с содержимым изображения. BLIP демонстрирует высокую эффективность в этой задаче благодаря способности интегрировать визуальную и текстовую информацию (Рис. 1.10.) [8].

Изображение выглядит как текст, снимок экрана, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.10. – Cхема работы Visual Question Answering

Сначала модель "смотрит" на изображение и запоминает, что на нем есть (Image Encoder). Потом она читает вопрос и пытается понять, что от нее хотят (Image–grounded Question Encoder). Далее модель соединяет то, что она "видит", с вопросом, чтобы дать правильный ответ (Cross–Attention). Наконец, специальная часть модели (Answer Decoder) "говорит" ответ [12].

Задача оценки соответствия изображения и текста (Image–Text Matching) заключается в определении, насколько текст соответствует содержимому изображения. Данная задача может помочь произвести поиск изображений по текстовым запросам (рис. 1.11.).

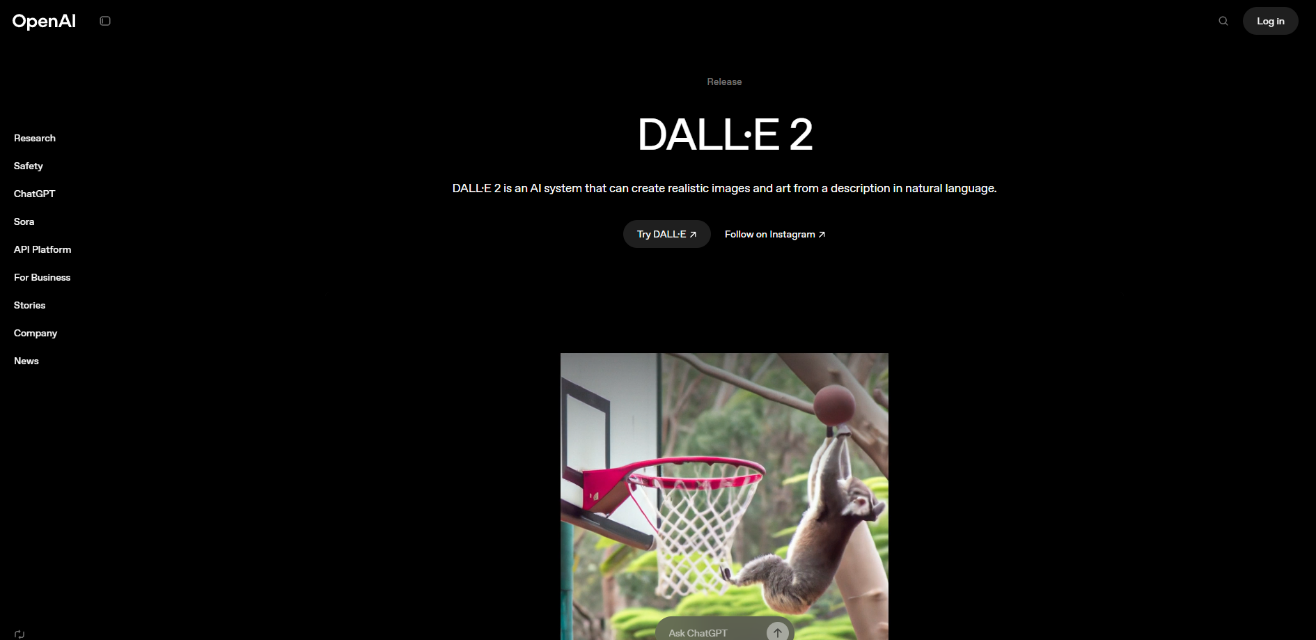
Изображение выглядит как текст, снимок экрана, млекопитающее

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.11. – Cхема работы Image–Text Matching

Модель "смотрит" на изображение и запоминает, что на нем есть (Image Encoder). Потом она читает текст и пытается понять его смысл (Text Decoder). Затем модель сравнивает картинку и текст, чтобы решить, подходят ли они друг к другу (Bi–Self–Attention и Cross–Attention). В конце она выдает результат: "Matched " (совпадает) или " Not matched" (не совпадает) [12].

Развитие трансформерных архитектур также способствовало появлению моделей для генерации изображений из текста, таких как DALL–E и Stable Diffusion. DALL–E, разработанная OpenAI, использует трансформерные архитектуры для создания изображений на основе текстовых описаний, демонстрируя высокое качество генерации (например, изображение "кот в шляпе, сидящий на луне") (Рис. 1.12.) [4, 19].

 Рис. 1.12. – Нейросеть DALL–E

Stable Diffusion, представленная в 2022 году, основана на диффузионных моделях, но также использует трансформерные компоненты для обработки текста, что делает её более компактной и доступной для использования на потребительских устройствах (Рис. 1.13.) [8].

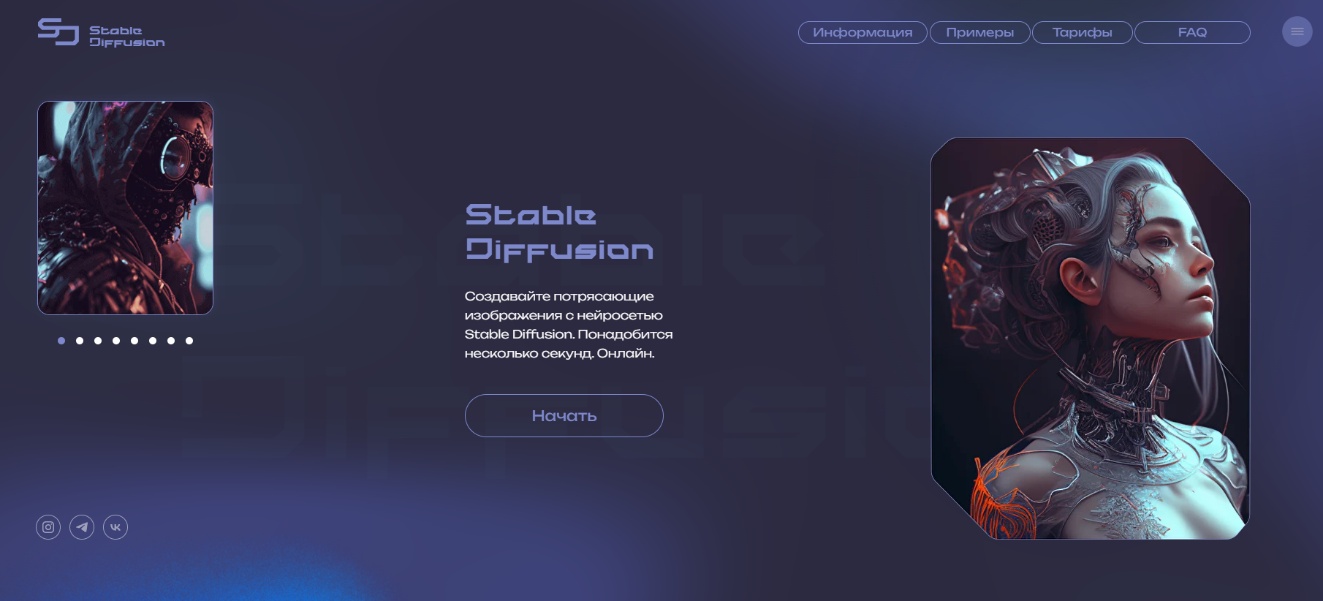


Рис. 1.13. – Нейросеть Stable Diffusion

Мультимодальные технологии находят применение в широком спектре задач, где требуется интеграция данных из различных источников. К числу таких задач относятся:

* Мультимодальные модели анализируют аудио, видео и текст субтитров, учитывая мимику и жесты для повышения точности перевода.
* Системы позволяют пользователям находить информацию, комбинируя текстовые запросы с изображениями, что особенно полезно в электронной коммерции.
* Модели анализируют рентгеновские снимки и текстовые записи из медицинской карты пациента, что улучшает точность диагностики.

Таким образом, с развитием трансформерных архитектур технологии обработки изображений и текста перешли на новый уровень, обеспечивая более эффективную интеграцию мультимодальных данных. Такие технологии уже сегодня применяются в системах навигации для слабовидящих, в цифровом маркетинге, в контентной модерации, в умных ассистентах и в расширенном поиске [7].

## **1.3. Обзор существующих веб–сервисов для генерации текстовых описаний изображений**

На данный момент существует множество веб–сервисов и приложений, которые используют технологии обработки изображений и текста, включая мультимодальные модели, подобные BLIP [14]. Однако конкретно сервисов, полностью основанных на BLIP, может быть не так много, поскольку многие компании адаптируют существующие решения под свои нужды или комбинируют несколько моделей для достижения лучших результатов.

Примером использования BLIP является сайт «products.aspose.ai», который основан на моделях машинного обучения, таких как: BLIP, CLIP, Vision Transformers (ViT) и OCR–технологии. Этот веб–сервис предоставляет бесплатное описание к изображениям. Пользователь может выбрать язык аннотации, также загрузить изображение из различных веб–сайтов [1].

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.14. – Сайт products.aspose.ai

На сайте «products.aspose.ai» пользователь сталкивается с простой механикой работы: пользователь может загрузить изображение из локальной файловой системы или указать ссылку на файл из интернета. После загрузки изображения необходимо нажать кнопку «Описать изображение», чтобы сгенерировать аннотацию. При этом важно отметить, что при повторной загрузке одного и того же изображения аннотация может отличаться, что позволяет пользователю выбрать наиболее подходящее описание для своих целей. Пример работы сайта предоставлен на рисунке 1.14.

Также, примером применения технологии BLIP является сайт «Astica.ai» [2]. предоставляет функциональность автоматической генерации текстовых описаний для изображений с использованием технологий машинного обучения и компьютерного зрения. Этот сервис может быть полезен для различных задач, таких как создание метаданных для изображений, улучшение доступности контента (например, для людей с ограниченными возможностями) и анализ визуальной информации.

Ключевыми особенностями платформы являются:

* Использует предобученные модели, такие как BLIP, для обеспечения высокой точности и выразительности текстовых аннотаций.
* Сервис позволяет генерировать описания на различных языках, что делает его универсальным решением для международных проектов.
* Разработчики могут использовать платформу через RESTful API для автоматизации процессов в своих приложениях.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1.15. – Сайт Astica.ai

Пример работы «Astica.ai» показывает, что сервис способен эффективно обрабатывать изображения различной сложности, включая фотографии с большим количеством деталей, а также изображения с текстом (рис. 1.15).

Рассмотренные веб–сервисы, такие как «products.aspose.ai» и «Astica.ai», демонстрируют высокую эффективность в задачах автоматической генерации текстовых описаний изображений, однако разрабатываемый в рамках данной магистерской диссертации проект обладает рядом отличительных особенностей, которые выделяют его на фоне существующих решений.

В отличие от «products.aspose.ai» и «Astica.ai», которые не предоставляют функцию редактирования сгенерированных аннотаций, разработанный сервис позволяет пользователям корректировать описания непосредственно в веб–интерфейсе, адаптируя их под конкретные цели, такие как использование в научных публикациях или маркетинговых материалах, а также экспортировать аннотации в форматы .txt, .docx и .pdf, что отсутствует в приведенных выше веб–сервисах. Следует отметить, что доступ к «products.aspose.ai» в некоторых регионах возможен только при использовании VPN, что создаёт дополнительное препятствие для пользователей. По сравнению с «Astica.ai», ориентированным на высокую точность и API–интеграцию, разработанный сервис, реализованный с использованием FastAPI, также поддерживает интеграцию, но дополнительно предлагает адаптивный веб–интерфейс с поддержкой мобильных устройств, таких как смартфоны и планшеты [18]. Кроме того, «Astica.ai» требует регистрации, создавая барьеры для пользователей, тогда как разработанный сервис предоставляет бесплатный и открытый доступ без необходимости регистрации, упрощая процесс взаимодействия.

Ещё одним преимуществом является повышенное внимание к безопасности данных: в отличие от рассмотренных решений, где политика хранения данных недостаточно прозрачна, разработанный сервис не сохраняет пользовательские данные после завершения сессии, а передача осуществляется через защищённые протоколы (HTTPS), что важно для работы с чувствительной информацией, например, в медицинской или юридической сферах. Дополнительно сервис поддерживает выбор языка аннотации, например английский, русский, немецкий.

Таким образом, разработанный веб–сервис выделяется среди «products.aspose.ai» и «Astica.ai» благодаря интерактивному интерфейсу с возможностью редактирования аннотаций, экспорту в различные форматы, адаптивному дизайну, повышенной безопасности данных и бесплатному доступу без регистрации, что делает его более универсальным и адаптированным к разнообразным сценариям, подчёркивая его практическую значимость и перспективность.

Выводы по первой главе

В рамках данной главы был проведён анализ предметной области, связанной с автоматической генерацией текстовых описаний изображений, с акцентом на эволюцию технологий обработки изображений, современные мультимодальные подходы и существующие веб–сервисы. Рассмотрение эволюции технологий обработки изображений показало, что развитие свёрточных нейронных сетей (LeNet5, AlexNet, VGG, ResNet, ENet) и последующий переход к трансформерным архитектурам, таким как Vision Transformer (ViT), создали прочную основу для современных мультимодальных моделей, таких как CLIP и BLIP [14]. Эти модели используются в веб–сервисах, таких как «products.aspose.ai» и «Astica.ai». Они продемонстрировали высокое качество текстовых описаний изображений, однако выявили и ограничения, такие как отсутствие функции редактирования аннотаций, необходимость регистрации и дополнительный доступ через VPN.

# **Глава 2. Разработка и обучение модели на основе Hugging face transformers**

Глава посвящена разработке и обучению модели для генерации текстовых описаний изображений с применением библиотеки Hugging Face Transformers. В рамках главы описаны основные инструменты, использованные при разработке, включая язык Python, его библиотеки и метрики. Отдельное внимание уделено процессу подготовки и предобработки данных для обучения модели BLIP, а также методам оптимизации, направленным на повышение качества и производительности. Представленные материалы заложили техническую основу для создания конечного веб–сервиса.

## **2.1. Инструменты разработки**

Основу разработки модели генерации текстовых описаний изображений составляет использование проверенных и широко применяемых инструментов в сфере машинного обучения и построения веб–приложений. В качестве основного языка программирования выбран Python, который по праву считается стандартом в задачах, связанных с анализом данных, компьютерным зрением и созданием нейросетевых архитектур [7]. Популярность языка обусловлена читаемым синтаксисом, активным сообществом и богатой экосистемой специализированных библиотек, что особенно важно при создании мультимодальных решений.

2.1.1. Язык программирования Python

Python – один из самых востребованных языков программирования в сфере машинного обучения и разработки веб–приложений. Его синтаксис интуитивно понятен, что упрощает написание и сопровождение кода даже при реализации сложных архитектур.

В процессе разработки сервиса генерации текстовых описаний изображений на основе моделей Hugging Face именно Python обеспечил необходимую гибкость и совместимость с ключевыми библиотеками. Модули transformers, torch, PIL, OpenCV, FastAPI и множество вспомогательных инструментов (включая asyncio, tqdm, re, googletrans и др.) представлены в виде легко подключаемых пакетов и активно поддерживаются сообществом [5, 7, 9, 30].

Отдельное внимание заслуживает стандартная библиотека Python, позволяющая решать задачи по работе с файлами, регулярными выражениями, многопоточностью и сетевыми запросами без необходимости установки стороннего ПО. Также язык поддерживает парадигмы объектно–ориентированного, функционального и процедурного программирования, что удобно при построении масштабируемой архитектуры.

Python – кроссплатформенный язык: написанный код одинаково работает в Windows, macOS и Linux. Это упрощает тестирование и развёртывание как локально, так и в облаке. Дополнительным преимуществом является его доступность: Python является полностью бесплатным, что особенно важно при выполнении курсовых и научных проектов с ограниченными ресурсами. Распространённость языка обеспечивает наличие обширной документации, форумов и обучающих материалов, что значительно упрощает решение возникающих технических задач.

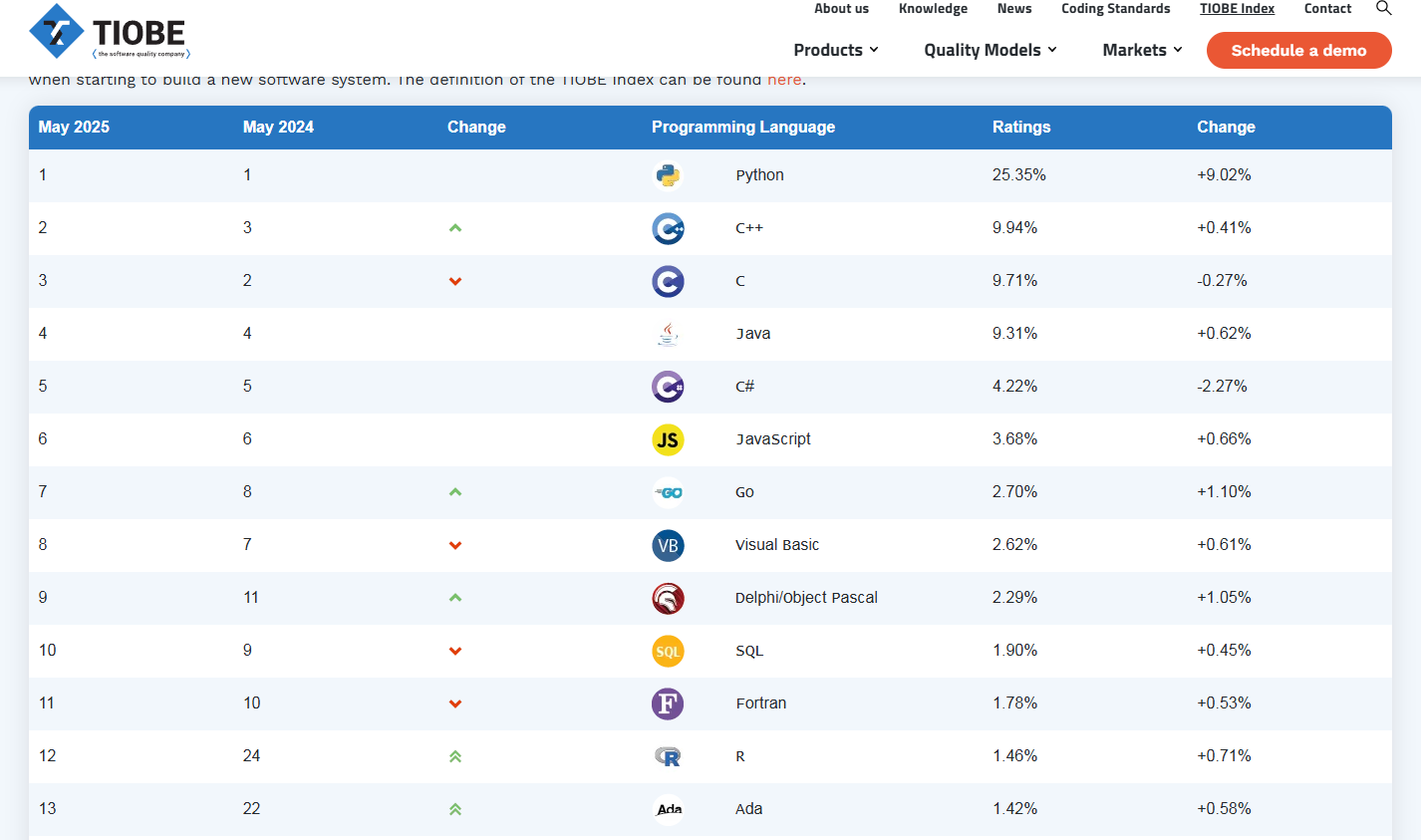


Рис. 2.1. – Рейтинг TIOBE

По данным Google Trends за май 2025 года, PYPL (PopularitY of Programming Language) составил рейтинг наиболее популярных языков программирования. Согласно этому рейтингу, в первую десястку наиболее популярных языков входят: Python, C++, C, Java, C#, JavaScript, GO, Visual Basic, Delphi/Object Pascal, SQL (см. рисунок 2.1.).

2.2.1. Библиотеки Python

Разработка веб–сервиса для автоматического создания текстовых описаний изображений требует использования широкого спектра специализированных библиотек на языке Python. Они обеспечивают как работу с изображениями, так и использование нейросетевых моделей, реализацию интерфейса, обработку запросов, генерацию текстов и взаимодействие между компонентами системы.

Одной из ключевых библиотек, задействованных в проекте, является Transformers от Hugging Face. Она предоставляет доступ к множеству предобученных моделей, включая BLIP и BLIP–2, специально предназначенных для мультимодальной генерации [7]. С помощью BlipProcessor и BlipForConditionalGeneration можно обрабатывать изображения и получать текстовые описания на основе визуального контекста. Эта библиотека активно развивается и поддерживает актуальные архитектуры, что делает её незаменимой при построении решений на базе трансформеров.

Для загрузки изображений, предварительной обработки и преобразования данных применяется Pillow (PIL) – удобная и легковесная библиотека, позволяющая выполнять базовые операции: изменение размера, поворот, фильтрация, коррекция яркости и контрастности. Наряду с ней используется OpenCV, обеспечивающая более низкоуровневую работу с изображениями: извлечение контуров, преобразование цветовых пространств, применение пороговых фильтров и др [30]. Эти библиотеки позволяют адаптировать входные данные под требования модели и повысить качество итоговых описаний [19].

Важную роль играет библиотека torch (PyTorch) – основной фреймворк для работы с нейросетевыми моделями. Она используется как для запуска предобученных моделей, так и для организации пайплайна обработки изображений. Модули torch.utils.data.Dataset и DataLoader позволяют формировать батчи, управлять потоками данных и реализовать эффективную загрузку при необходимости масштабирования.

Дополнительно применяется torchvision.transforms, позволяющая проводить цепочки преобразований изображений, включая нормализацию, случайные обрезки и преобразования, что особенно важно при тонкой настройке или тестировании модели на различных входных данных.

Для работы с оптическим распознаванием текста подключены библиотеки PaddleOCR и EasyOCR. Их использование позволяет обогащать описание изображений текстовой информацией, присутствующей на самих изображениях (например, плакатах, документах, интерфейсах), а также расширять функциональность сервиса за пределы генеративных моделей.

Для перевода текстов, полученных в результате генерации, используется библиотека googletrans, обеспечивающая автоматический перевод описаний на нужный язык. Это особенно актуально в случае, если модель возвращает описания на английском языке, а интерфейс сервиса ориентирован на русскоязычную аудиторию.

Реализация серверной части построена с помощью FastAPI – современного асинхронного фреймворка, позволяющего быстро и удобно создавать RESTful API [5]. Он обеспечивает обработку изображений, получаемых через HTTP–запросы, и возвращает пользователю текстовое описание. В сочетании с uvicorn и nest\_asyncio обеспечивается высокопроизводительный запуск сервиса как в локальной среде, так и через туннелирование (с использованием pyngrok), что упрощает тестирование и демонстрацию результата без развертывания на внешнем сервере.

Отдельное место занимают утилиты для работы с текстом, файлами и пользовательскими документами. Библиотека docx используется для генерации и сохранения текстов в формате Word, а reportlab – для создания PDF–файлов с описаниями. Это расширяет функциональность веб–сервиса и позволяет сохранять результаты работы в различных форматах.

Кроме этого, в проекте применяются стандартные библиотеки Python: os, re, logging, random, asyncio, io и другие. Они необходимы для организации структуры проекта, управления файловой системой, логирования процессов, генерации уникальных имен, работы с асинхронными задачами и буферизацией данных.

Таким образом, эффективное взаимодействие всех вышеупомянутых библиотек позволяет построить рабочий прототип веб–сервиса, способного автоматически анализировать изображения и выдавать к ним текстовые описания. Большинство используемых инструментов являются открытыми и активно поддерживаются, что делает их удобными для интеграции, расширения и масштабирования. Выбор именно этих библиотек обусловлен как их функциональной полнотой, так и совместимостью с фреймворком Hugging Face и моделью BLIP.

2.3.1 Метрики оценивания моделей

В задачах автоматической генерации текстовых описаний изображений оценка качества выходных данных является ключевым этапом, позволяющим определить, насколько точно модель воспроизводит смысл и структуру, сопоставимые с реальными описаниями [16]. Для этой цели используются специальные метрики, адаптированные к анализу сходства между сгенерированным и эталонным текстом. В рамках данной работы применяются три наиболее распространённые метрики: BLEU, ROUGE–L и METEOR. Каждая из них анализирует текст с разных позиций: от совпадений по словам до учёта морфологии и порядка следования фраз.

1. BLEU (Bilingual Evaluation Understudy) – одна из первых метрик, применяемых в области машинного перевода, а позже – и в задачах генерации текста. Она измеряет степень совпадения между предсказанным и эталонным описанием на основе н–грамм (последовательностей из n слов). BLEU оценивает точность (precision), то есть, сколько к–грамм в предсказанном тексте также встречаются в эталонном. При этом используется модифицированная точность с учётом максимально допустимого количества повторов [28].

Формула вычисления BLEU для к–грамма:

где,

* – модифицированная точность для n–грамм;
* ​ – вес (обычно равный );
* – штраф за длину (brevity penalty), уменьшающий значение метрики, если предсказание слишком короткое по сравнению с эталоном.

Значение BLEU находится в диапазоне от 0 до 1, где 1 означает полное совпадение.

1. ROUGE–L (Recall–Oriented Understudy for Gisting Evaluation) – метрика, ориентированная на полноту (recall) и структуру текста. В отличие от BLEU, здесь акцент делается на нахождении самой длинной общей подпоследовательности (LCS, Longest Common Subsequence) между предсказанным и эталонным описанием. Это позволяет учитывать не только точные совпадения слов, но и их порядок [28].

Формула для ROUGE–L основана на сочетании точности и полноты:

где,

* – коэффициент, регулирующий вес точности и полноты (обычно равен 1.2).

ROUGE–L особенно полезна при оценке более длинных фраз, где важен не только состав слов, но и их порядок.

1. METEOR (Metric for Evaluation of Translation with Explicit ORdering) – метрика, сочетающая точность, полноту и семантическое сходство. Она позволяет выявлять совпадения не только на уровне точных слов, но и с учётом синонимов, различных форм слова и перестановок. Для достижения этого METEOR использует морфологический анализ и лемматизацию [28].

Формула METEOR:



где,

* – количество совпадающих слов;
* – количество фрагментов, в которых порядок слов сохраняется;
* и – параметры штрафа (обычно ).

METEOR демонстрирует хорошую корреляцию с человеческой оценкой качества текста, особенно в задачах с несколькими эталонами на одно изображение [7].

Каждая из этих метрик имеет свои преимущества и ограничения. Выбор конкретной метрики зависит от задачи, приоритетов и контекста применения модели. Важно учитывать все аспекты моделирования, а не полагаться только на одну метрику, чтобы получить полное представление о производительности модели.

Совместное использование трёх различных метрик обеспечивает более объективную и надёжную оценку работы модели, особенно в условиях наличия нескольких референсных описаний для одного изображения.

## **2.2 Подготовка и предобработка данных для обучения модели**

Качество обучающей выборки напрямую влияет на точность и выразительность сгенерированных текстов. Особенно это актуально в задачах генерации описаний изображений, где важно не просто сопоставить изображению набор ключевых слов, а передать его смысл, структуру и визуальные детали. Поэтому на этапе подготовки особое внимание уделяется содержательности аннотаций и корректному соответствию между изображением и описанием [16].

Для обучения модели использовался собственноручно сформированный набор данных, в котором каждое изображение сопровождалось текстовым описанием на английском языке. Основу датасета составили разнообразные визуальные сцены, охватывающие повседневные ситуации: человек на сцене, девочка на траве, женщина у стены, девушка с книгой и т. д. Это позволяет обучить модель на разнотипных контекстах и повысить универсальность выходных аннотаций.

Данные изначально сохранялись в формате JSON, так как он удобен для хранения пар "изображение – описание"[15]. Каждая запись представляла собой объект с двумя полями:

* image – имя файла изображения (например, "littlegirl.jpg");
* caption – текстовое описание на естественном языке.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 2.2. – Пример структуры датасета

Такой подход исключает необходимость последующего перевода или конвертации в другие форматы, упрощая этап подготовки. Кроме того, фиксированная структура JSON позволяет легко считывать данные в процессе обучения и использовать их напрямую при вызове моделей [15].

На каждое изображение создавалось одно описание, составленное вручную. Эталоны формулировались полно и содержательно, с акцентом на детали, атмосферу, цветовую палитру и действия на сцене. Это повышает вероятность того, что модель будет ориентироваться не только на ключевые визуальные объекты, но и на их связи, настроение и контекст. Пример описания: «На изображении представлена маленькая девочка, сидящая на зелёной траве. Её лицо и белое платье украшены яркими разноцветными пятнами краски, что подчёркивает её увлечённость творческим процессом...».

Каждое описание составлялось так, чтобы оно могло выступать полноценным абзацем текста. Это делалось с прицелом на генерацию более богатых и связанных по смыслу выходов, а не отдельных разрозненных фраз. Такой подход также даёт преимущества при оценке метриками, чувствительными к полноте и структурной связности.

На этапе подготовки изображения сохранялись в папке проекта без изменения разрешения или сжатия. Ни аугментация, ни фильтрация изображений не применялись, так как основная цель заключалась не в классификации объектов, а в формировании осмысленного текста по картинке. При этом изображения подбирались с учётом разнообразия: в датасет включались как портреты, так и сцены с несколькими персонажами, чтобы модель могла научиться выделять разные уровни композиции

Подготовленные данные в дальнейшем использовались в процессе адаптации модели BLIP к целевой задаче генерации описаний. Благодаря ручной разметке и содержательности аннотаций, модель получила возможность опираться на качественные и семантически насыщенные примеры при формировании выходного текста. Это позволило не только улучшить выразительность сгенерированных описаний, но и адаптировать поведение модели к более детализированному и стилистически выверенному формату подачи информации.

Таким образом, этап подготовки данных оказался минималистичным, но целенаправленным: основной упор был сделан на осмысленность описаний и стабильную структуру датасета.

## **2.3. Оптимизация модели для повышения точности и скорости работы**

В задачах генерации текстовых описаний важно учитывать не только качество создаваемых аннотаций, но и скорость их генерации. Особенно это критично при разработке веб–сервиса, где пользователь ожидает быстрый результат. Поэтому на этапе внедрения и настройки модели основной задачей стало достижение баланса между точностью, выразительностью текста и временем отклика [7].

После выбора архитектуры BLIP было принято решение предобучить модель на собственном датасете, сформированном вручную. Это позволило адаптировать генерацию описаний под конкретные визуальные сцены, повысить соответствие между изображением и текстом, а также улучшить метрики качества при тестировании. Итоговая версия модели получила обозначение BLIP–finetuned, так как была модифицирована на основе предобученной BLIP–large.

Процесс предобучения проводился с использованием библиотеки Hugging Face Transformers. Базовая модель Salesforce/blip–image–captioning–large была загружена и дополнительно обучена на кастомном наборе данных, состоящем из 500 пар «изображение–описание».

Для оптимизации производительности модель была переведена в режим инференса через model.eval() и загружена на доступное устройство с помощью model.to(device). Это позволило задействовать графический процессор и тем самым сократить время генерации. Также использовалась асинхронная обработка в рамках FastAPI, благодаря чему удалось минимизировать время ответа сервера при параллельной обработке изображений.

Ключевую роль в оптимизации сыграла детальная настройка параметров генерации. Значения подбирались таким образом, чтобы обеспечивать разнообразие, логичность и стабильность выходных текстов.

Применялись следующие параметры:

* max\_new\_tokens: максимальное число токенов – рассчитывается автоматически на основе заданного пользователем количества слов;
* min\_new\_tokens: половина от word\_count – обеспечивает минимальную длину;
* num\_beams: 10 – даёт связные и последовательные описания;
* temperature: 0.9 – поддерживает естественность формулировок;
* top\_k: 100 – ограничивает выбор наиболее вероятных токенов;
* top\_p: 0.95 – обеспечивает разнообразие;
* repetition\_penalty: 1.3 – снижает повторяемость;
* length\_penalty: 2.0 – поощряет более длинные формулировки;
* no\_repeat\_ngram\_size: 2 – исключает повторы биграмм;
* do\_sample: включено – генерация в семплирующем режиме;
* early\_stopping: отключено – позволяет генерировать до достижения лимита.

Эти параметры обеспечивают необходимый баланс между точностью и свободой генерации [32]. Например, num\_beams усиливает связность, top\_p и temperature вносят вариативность, а no\_repeat\_ngram\_size предотвращает повторения. Кроме того, пользователь может указать желаемое количество слов (word\_count), и параметры автоматически адаптируются под него. Это делает сервис гибким и удобным в различных сценариях – от кратких описаний до более подробных аннотаций.

Таким образом, модель BLIP–finetuned стала центральным компонентом веб–сервиса. Благодаря грамотной оптимизации параметров, аппаратной настройки и асинхронной обработки удалось добиться стабильной и достаточно быстрой работы сервиса без потери точности и выразительности описаний.

Выводы по второй главе

В рамках данной главы была реализована основная логика формирования и настройки модели генерации текстовых описаний изображений на основе архитектуры BLIP. Проведён выбор инструментов и библиотек, необходимых для обработки изображений, работы с текстами и взаимодействия с моделью. В качестве инструмента разработки выбран язык программирования Python с его развитой экосистемой, включая библиотеки Transformers, Torch, а также фреймворк FastAPI для построения серверной части веб–сервиса.

Для предобучения модели использован собственноручно сформированный датасет, где каждому изображению соответствовало вручную составленное содержательное описание. Данные хранились в формате JSON, что обеспечило удобную структуру и прямую совместимость с библиотекой Transformers [15].

Базовая модель BLIP–large была предобучена на сформированном датасете. Это позволило адаптировать генерацию под особенности лексики, стилистики и визуальных сцен, представленных в обучающих данных. Полученная модель, обозначенная как BLIP–finetuned, демонстрирует улучшенные результаты по сравнению с оригинальной, в том числе более точное соответствие визуальному содержимому и выразительные формулировки в текстах.

Дополнительно были настроены параметры генерации: количество beam–лучей, длина, температура, вероятность выбора токенов, штрафы за повторы и другие. Это позволило сбалансировать связность, выразительность и скорость отклика. Модель была переведена в режим eval() и оптимизирована под доступное вычислительное устройство.

Таким образом, были реализованы все ключевые этапы подготовки модели: от сбора и структурирования данных до настройки параметров генерации и интеграции в прикладной сервис. Полученные результаты легли в основу последующей оценки качества и тестирования веб–приложения, что будет подробно рассмотрено в следующей главе.

# **Глава 3. Разработка веб–сервиса для автоматической генерации текстовых описаний**

Глава посвящена практической реализации веб–сервиса, предназначенного для автоматической генерации текстовых описаний к изображениям. Создание удобного пользовательского интерфейса и эффективной серверной логики позволяет обеспечить полноценную работу модели в прикладном контексте. Реализация сервиса охватывает клиентскую часть с интерфейсом для загрузки изображений и выбора параметров генерации, а также серверную часть, обеспечивающую обработку запросов, интеграцию модели и возвращение результатов. Кроме того, в рамках сервиса реализован механизм оценки качества сгенерированных описаний с помощью автоматических метрик и построения визуальной аналитики в реальном времени.

## **3.1. Реализация клиентской части: HTML, CSS, JavaScript**

Клиентская часть веб–сервиса выполняет роль интерфейса взаимодействия пользователя с системой генерации описаний. Она позволяет загружать изображение, задавать параметры генерации, выбирать язык вывода и получать результат в удобной текстовой форме. Простота, отзывчивость и визуальная читаемость интерфейса напрямую влияют на восприятие конечного продукта, особенно при использовании в прикладных задачах.

Для создания интерфейса были использованы технологии HTML, CSS и JavaScript. HTML отвечает за структуру страницы, CSS – за визуальное оформление, а JavaScript – за динамическое взаимодействие с сервером и реализацию функциональных элементов без перезагрузки страницы. Вёрстка выполнена адаптивной, благодаря чему сервис корректно отображается как на компьютере, так и на мобильных устройствах [18].

Интерфейс содержит следующие ключевые элементы:

* форму для загрузки изображения;
* выпадающий список для выбора языка аннотации (английский/русский/немецкий);
* числовое поле, где можно указать желаемое количество слов в сгенерированном описании;
* функциональный блок о возможностях веб–сервиса;
* поле вывода текста аннотации, появляющееся после генерации;
* кнопка «Редактировать», позволяющая вручную изменить сгенерированный текст;
* кнопка «Скопировать», копирующая аннотацию в буфер обмена;
* кнопка «Экспорт», позволяющая сохранить описание в различных форматах (TXT, DOCX, PDF);
* кнопки оценки качества сгенерированного текста;
* кнопка «Загрузить новое изображение», очищающая форму и позволяющая начать процесс заново.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.1. – Клиентской часть сервиса перед генерацией аннотации

Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.2. – Клиентская часть сервиса после генерации аннотации

Процесс генерации построен по следующей схеме: пользователь загружает изображение, выбирает параметры и отправляет запрос. Изображение передаётся на сервер, где происходит генерация аннотации с помощью модели BLIP. Результат возвращается в виде текста и отображается на странице. Если пользователь хочет отредактировать аннотацию, он может внести изменения в текстовом поле и сохранить их. Кнопки «Экспорт» и «Скопировать» позволяют использовать полученный текст в других приложениях – в документах, социальных сетях, презентациях и т. д. Также реализованы элементы взаимодействия – пользователь может оценить результат генерации с помощью кнопок «нравится» и «не нравится», что делает интерфейс более интерактивным.

В нижней части интерфейса размещён функциональный блок, представленный в виде карточек с иконками. Он отображает основные преимущества сервиса: быструю загрузку, работу с ИИ, редактируемость, экспорт, мобильную совместимость и безопасность данных. Этот блок не только визуально дополняет интерфейс, но и помогает пользователю быстрее понять возможности системы.

Таким образом, клиентская часть веб–сервиса представляет собой гибкий, функциональный и адаптивный интерфейс, ориентированный на удобство пользователя. Простота взаимодействия, поддержка нескольких языков, редактируемость текста и чистый визуальный стиль делают работу с системой интуитивной и приятной даже для пользователя без технической подготовки.

## **3.2. Реализация серверной части и интеграция модели BLIP**

Серверная часть веб–сервиса отвечает за обработку входящих запросов, генерацию текстовых описаний на основе изображений, оценку качества аннотаций и передачу результатов обратно клиенту. Вся логика построена с использованием языка программирования Python и фреймворка FastAPI, что позволило достичь высокой скорости отклика, читаемости кода и масштабируемости архитектуры [5].

FastAPI был выбран в качестве основного инструмента разработки серверной части благодаря своей асинхронной природе, простоте настройки и встроенной поддержке OpenAPI–документации [5]. Этот фреймворк позволяет легко создавать RESTful–интерфейсы, эффективно обрабатывать файлы, формы и возвращать ответы в формате JSON [15]. Серверный код запускается на локальном хосте и принимает HTTP–запросы от фронтенда при загрузке изображения и отправке формы генерации.

Загрузка изображения реализуется через специальный POST–метод, где объект типа UploadFile принимает файл от пользователя. Далее изображение обрабатывается при помощи библиотеки PIL, приводится к формату RGB и подаётся на вход модели.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.4. – Обработка запроса изображения на сервере с валидацией параметров

Дополнительно в серверной части реализован механизм оценки качества сгенерированного текста [34]. Используются метрики BLEU, ROUGE–L и METEOR, которые загружаются с помощью библиотеки Evaluate. Для каждой новой аннотации рассчитываются значения этих метрик на основе заранее заданных эталонных описаний. Это позволяет фиксировать результат и впоследствии проводить сравнение между различными генерациями или настройками параметров [7, 31].

Результаты генерации и метрик логируются в CSV–файл с указанием даты, имени файла, текста описания и значений BLEU, ROUGE–L и METEOR. Такой подход позволяет постепенно накапливать данные о работе модели и использовать их для анализа. Файл создаётся автоматически при первом запуске, а последующие записи добавляются построчно.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.5. – Логирование результатов генерации и метрик оценки качества

Для перевода описаний предусмотрена интеграция с внешним API. Если выбран язык, отличный от английского, текст аннотации переводится с помощью библиотеки googletrans. Это позволяет поддерживать многоязычность без потери основной смысловой структуры.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.6. – Перевод генераций аннотаций

Для генерации аннотаций используется модель BLIP (Bootstrapped Language–Image Pretraining), загруженная из библиотеки Hugging Face Transformers. Для повышения качества результата была предобучена модель BLIP–finetuned, созданная на основе BLIP–large и адаптированная под вручную составленный датасет. Предобучение проводилось на специально подготовленных парах «изображение–описание», что позволило улучшить стилистическое и содержательное соответствие генерируемых текстов [7].

Для оптимизации производительности модель была переведена в режим инференса через model.eval() и загружена на доступное устройство с помощью model.to(device).

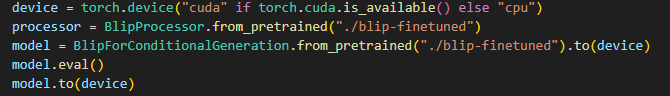


Рис. 3.7. – Загрузка модели BLIP– finetuned

Интеграция модели началась с загрузки необходимых компонентов: BlipProcessor и BlipForConditionalGeneration. При запуске сервера модель и процессор инициализируются один раз и переводятся на доступное устройство – CPU или GPU. Это позволяет избежать повторной загрузки модели при каждом запросе, что значительно снижает время отклика. Генерация текста происходит в асинхронной функции, вызываемой на стороне сервера при получении изображения.

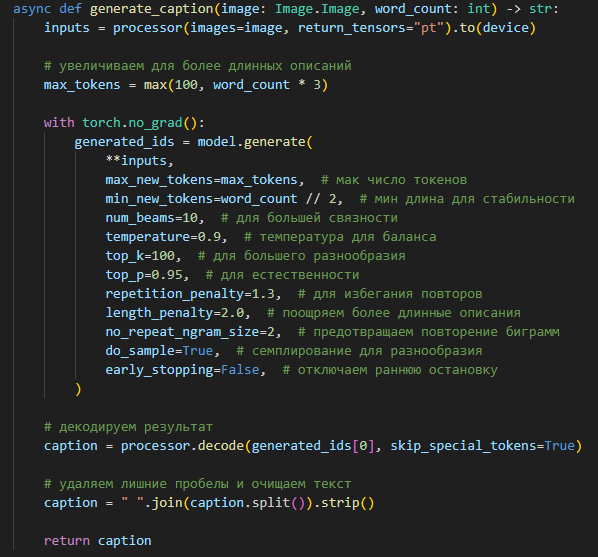


Рис. 3.8. – Настройка параметров генерации описания в модели BLIP

Вызов генерации происходит в отдельной асинхронной функции. Используются расширенные параметры модели, такие как max\_new\_tokens, min\_new\_tokens, num\_beams, temperature, top\_k, top\_p, length\_penalty и другие. Эти параметры подобраны вручную, чтобы обеспечить баланс между разнообразием и связностью текста, его длиной и логической завершённостью. Результат возвращается в виде строки и далее очищается от лишних символов.

Таким образом, серверная часть веб–сервиса представляет собой устойчивую и расширяемую архитектуру, построенную на Python и FastAPI, с полной интеграцией модели BLIP и логикой обработки текста. Она обеспечивает не только генерацию аннотаций, но и сбор качественных метрик, логирование, поддержку перевода и базовую защиту от ошибок, что делает систему полноценной с технической точки зрения.

Выводы по третьей главе

В рамках третьей главы был разработан полноценный веб–сервис, позволяющий автоматически генерировать текстовые описания изображений с использованием нейронной модели BLIP, предобученной на собственном датасете. Реализация клиентской части выполнена на HTML, CSS и JavaScript, с акцентом на простоту, доступность и функциональность интерфейса. Пользователь получает возможность загрузить изображение, выбрать параметры генерации, отредактировать результат и сохранить его в нужном формате.

Серверная часть построена на Python с использованием FastAPI. В неё интегрирована предобученная версия модели BLIP–large, адаптированная под визуальные образы и стилистику кастомного датасета. Настройка параметров генерации позволила добиться баланса между связностью, выразительностью и разнообразием аннотаций. Реализована система логирования и сохранения всех сгенерированных описаний с расчётом метрик BLEU, ROUGE–L и METEOR. Также добавлена возможность перевода описания на другой язык, что делает сервис универсальным и многоязычным.

Таким образом, реализованная система представляет собой функциональный, надёжный и расширяемый веб–сервис, в котором объединены удобный интерфейс и глубокая серверная логика. Интеграция предобученной модели BLIP–finetuned стала ключевым фактором повышения качества генерации, а реализованные функции делают систему готовой к практическому применению и дальнейшему развитию.

# **Глава 4. Тестирование и оценка качества разработанного сервиса**

После завершения разработки веб–сервиса наступает этап практического тестирования. Он позволяет оценить, насколько эффективно работает модель генерации описаний, насколько удобно взаимодействовать с интерфейсом и насколько качественными получаются текстовые аннотации [1], [2]. Проверка производится на реальных изображениях, не входивших в обучающую выборку, чтобы исключить переобучение и получить объективную картину.

Основная цель тестирования – выявить сильные и слабые стороны сервиса, определить точность генерации, убедиться в стабильности работы всех компонентов и собрать статистику по ключевым метрикам качества. В процессе тестирования также проводится запись результатов в журнал логов, что позволяет впоследствии анализировать поведение модели, отслеживать метрики и выявлять закономерности [34].

В данной главе описывается, как проходило тестирование сервиса, какие изображения использовались, как рассчитывались метрики BLEU, ROUGE–L и METEOR, а также какие результаты были получены в ходе генерации описаний. На основе собранных данных делаются выводы о работе сервиса и качестве сгенерированного контента.

## **4.1. Тестирование сервиса на реальных данных**

Для оценки эффективности разработанного веб–сервиса было проведено тестирование на реальных изображениях, не входивших в обучающую выборку. Такой подход позволил проверить, насколько успешно модель справляется с генерацией описаний в условиях, близких к практическому использованию. В процессе тестирования учитывались разнообразие сцен, композиции, освещение, выражения лиц и тип контента [7].

Изображения загружались вручную через клиентский интерфейс веб–сервиса. После загрузки автоматически происходила генерация текстового описания с использованием предобученной модели BLIP–finetuned. Далее отображался результат, рассчитывались метрики качества, и данные сохранялись в лог–файл. Отдельно проверялась стабильность работы интерфейса, корректность обработки изображений различных форматов (JPEG, PNG), а также поведение сервера при некорректных вводах (например, при слишком маленьком изображении).

Для тестирования были отобраны изображения, отражающие разные контексты: портреты, уличные сцены, концертные выступления, предметная съёмка и повседневные ситуации. Это позволило оценить универсальность модели и то, как она интерпретирует визуальные признаки, ранее ей не встречавшиеся. Особое внимание уделялось смысловой точности, грамматической связности и стилевой выразительности сгенерированных текстов.

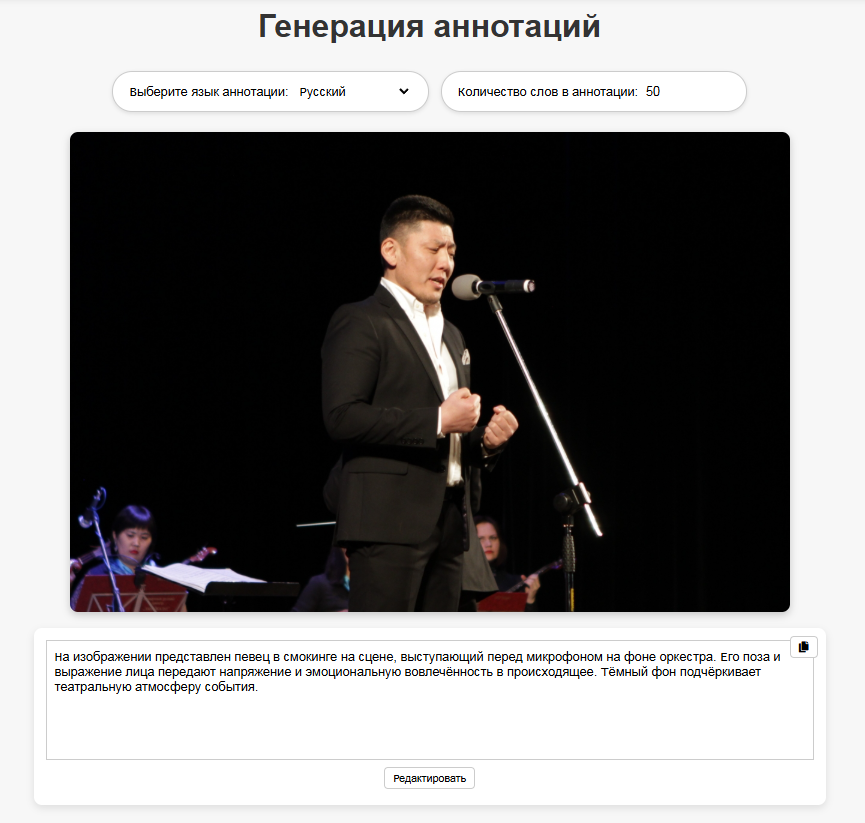


Рис. 4.1. – Пример тестового изображения с концертной сценой

На этапе тестирования фиксировались следующие параметры:

* исходное изображение;
* сгенерированное описание;
* эталонные описания (для последующей оценки метриками);
* значения BLEU, ROUGE–L и METEOR;
* дата и имя загруженного файла.

Интерфейс также позволял копировать или экспортировать аннотацию, что проверялось отдельно. В случае выбора другого языка происходил автоматический перевод, и его корректность сравнивалась вручную. Важно, что модель оставалась устойчивой к различиям в освещении, ракурсе, детализации и даже в случае отсутствия очевидных объектов фокусировки. Генерация происходила без сбоев, а текст в большинстве случаев отражал основную суть изображения.

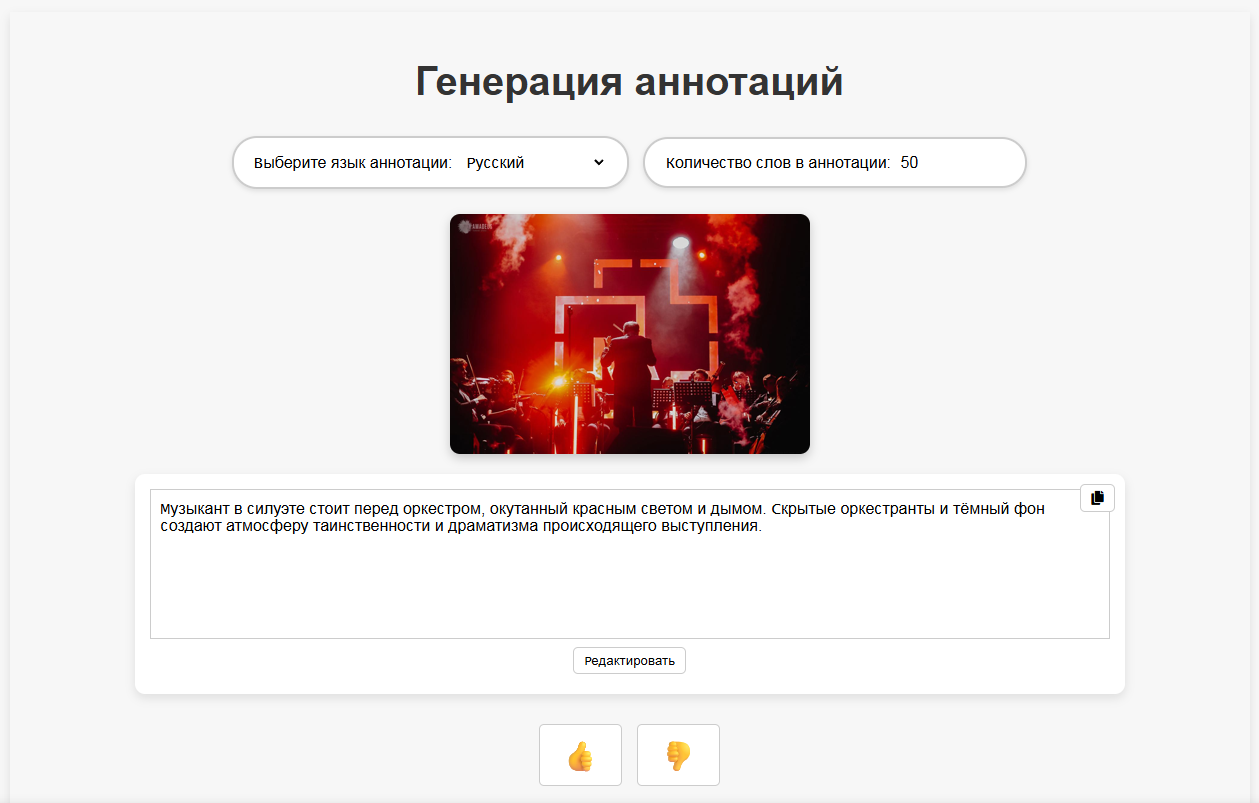


Рис. 4.2. – Тестовое изображение со сложной визуальной композицией

Таким образом, тестирование на реальных данных подтвердило работоспособность всех ключевых компонентов веб–сервиса: загрузка изображений, генерация, метрики, логирование и взаимодействие с интерфейсом. Полученные данные легли в основу дальнейшего анализа качества сгенерированных описаний, который представлен в следующем пункте.

## **4.2. Оценка точности сгенерированных текстовых описаний**

После этапа тестирования была проведена оценка качества полученных аннотаций с помощью автоматических метрик BLEU, ROUGE–L и METEOR. Эти метрики позволяют количественно определить степень совпадения между сгенерированным описанием и эталонными текстами, что даёт возможность объективно сравнивать результаты и отслеживать изменения при разных настройках генерации [7].

Для проверки метрик оценивания были отобраны от трёх до пяти эталонов, точно передающих суть изображения. Такой подход обеспечил более надёжную и реалистичную оценку – метрики показывают лучшие результаты при наличии нескольких вариантов, так как учитывают разнообразие формулировок [34].

Расчёт метрик выполнялся автоматически после каждой генерации и выводился как в терминал разработчика, так и в лог–файл metrics\_log.csv. В таблицу попадали следующие данные: имя файла, сгенерированный текст, значения BLEU, ROUGE–L и METEOR, а также временная метка. Такой формат позволил в дальнейшем визуализировать результаты и выявить тенденции в работе модели. Пример изображения и метрик, соответствующих записи в журнале логов:

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4.3. –Значения метрик BLEU, ROUGE–L и METEOR на тестовых изображениях

Значения метрик интерпретировались следующим образом:

* BLEU отражает степень точного совпадения по n–граммам, чувствителен к порядку слов.
* ROUGE–L фиксирует наличие длиннейших общих подстрок и оценивает пересечения по фразам.
* METEOR учитывает синонимы, форму слов и порядок, давая более гибкую оценку смысла.

На практике наиболее стабильной оказалась METEOR, так как она лучше распознаёт смысловое соответствие даже при расхождениях в формулировке. BLEU, наоборот, часто занижал результат при наличии дополнительных деталей или изменённом порядке слов. Также в логах наблюдались случаи, когда BLEU стремился к нулю, хотя описание визуально полностью соответствовало изображению. Это подтверждает важность использования нескольких метрик одновременно [7]. Такие наблюдения легли в основу принятого решения использовать не только BLEU, но и более гибкие метрики в дальнейшей работе. ROUGE–L показывал промежуточные значения, особенно хорошо срабатывая на описаниях, где ключевые фразы были сохранены.

Результаты оценки в процентах (с округлением до сотых долей) варьировались в зависимости от сложности изображения, уровня детализации и близости эталонов. В среднем METEOR достигал значений от 55 до 70%, ROUGE–L – от 30 до 60%, BLEU – от 0 до 45%. На рисунке 4.4. представлена сводная таблица с результатами генерации описаний и рассчитанными метриками BLEU, ROUGE–L и METEOR для различных изображений. Данные собраны из логов веб–сервиса в процессе тестирования.

Таблица 4.1. – Сводная таблица значений метрик для сгенерированных описаний

|  |  |  |  |
| --- | --- | --- | --- |
| Изображение | BLEU | ROUGE–L | METEOR |
| Девочка в краске | 35.2% | 48.6% | 53.5% |
| Жен. в хиджабе | 27.4% | 41.9% | 46.2% |
| Девушка с книгой | 42.1% | 56.3% | 60.1% |
| Парень на сцене | 38.7% | 51.2% | 58.4% |
| Мужчина в кафе | 30.8% | 45.0% | 50.3% |

Таким образом, автоматические метрики помогли зафиксировать качество генерации и подтвердить, что модель, предобученная на пользовательском датасете, способна создавать описания, близкие по смыслу и структуре к ручным вариантам. Эти данные послужили основой для последующего анализа и визуализации результатов.

## **4.3. Сравнительный анализ с существующими решениями**

Генерация текстовых описаний изображений – активно развивающееся направление, в котором на данный момент существует несколько общедоступных решений. Основное различие заключается в доступности, гибкости настройки, языке генерации и точности получаемых текстов.

Большинство коммерческих решений представляют собой «чёрные ящики», к которым пользователь может лишь передать изображение, получив в ответ краткое описание. Генерация чаще всего происходит быстро, но результат имеет общий характер и редко содержит стилистические детали или эмоциональную окраску. Визуальное содержание часто сводится к сухому перечню объектов, без их интерпретации и связности текста.

Среди доступных мультимодальных моделей можно выделить BLIP–2, GIT и MiniGPT–4. Каждая из них имеет свои особенности. Например, BLIP–2 способен комбинироваться с языковыми моделями, но требует тонкой настройки и большого объёма памяти. GIT показывает хорошие результаты в мультимодальных задачах, но фокусируется на генерации более общего характера. MiniGPT–4 в большинстве случаев применяется в демонстрационных целях и не обладает стабильной производительностью при реальном использовании.

На раннем этапе разработки в проекте предпринималась попытка интеграции модели BLIP–2, однако она оказалась провальной. В результате было принято решение предобучить модель BLIP–large на собственном датасете и использовать полученную версию – BLIP–finetuned [7].

Разработанный веб–сервис отличается от большинства доступных решений. Для объективной оценки качества генерации аннотаций было проведено сравнение с существующим онлайн–сервисом, использующий схожую технологию. В качестве внешнего решения был выбран веб–сайт products.aspose.ai и astica.ai. Эти сервисы предоставляет автоматическую генерацию описания изображения.

В рамках сравнения одно и то же изображение – мужчина на сцене в концертной обстановке – было обработано в разработанном веб–сервисе и на других платформах. Ниже представлены изображения полученных текстовых аннотаций.

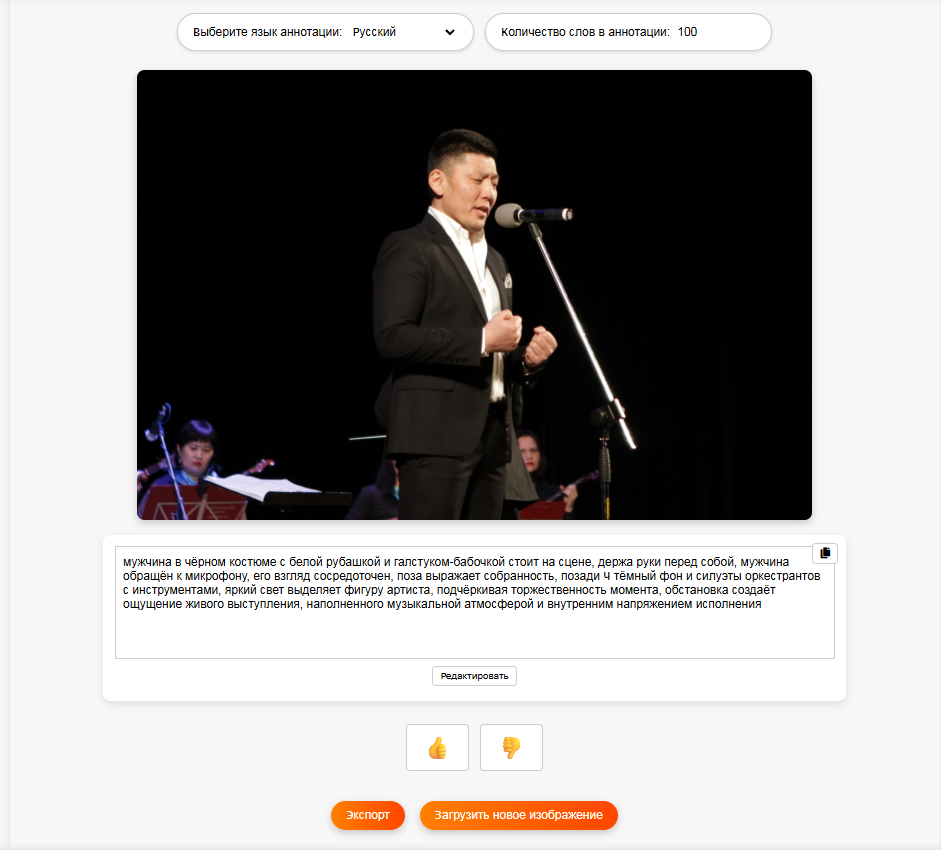


Рисунок 4.4. – Сгенерированное описание в разработанном веб–сервисе

Аннотация, сгенерированная в собственном сервисе, отличается большей образностью, стилистической выразительностью и сценическим контекстом. В ней уделено внимание свету, эмоциональному тону сцены и общей атмосфере, что придаёт тексту глубину. Использование формулировок вроде «внутреннее напряжение исполнения» или «торжественность момента» расширяет рамки простого описания объектов, переходя к художественной интерпретации происходящего.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.5. – Сгенерированное описание через сервис Aspose.ai

Описание от Aspose.ai, в свою очередь, более сухое и обобщённое. Оно ограничивается описанием объектов и не включает в себя стилистических деталей или сценического анализа. Несмотря на наличие базовой логики и связности, в тексте отсутствует эмоциональная окраска и многослойность восприятия [1].

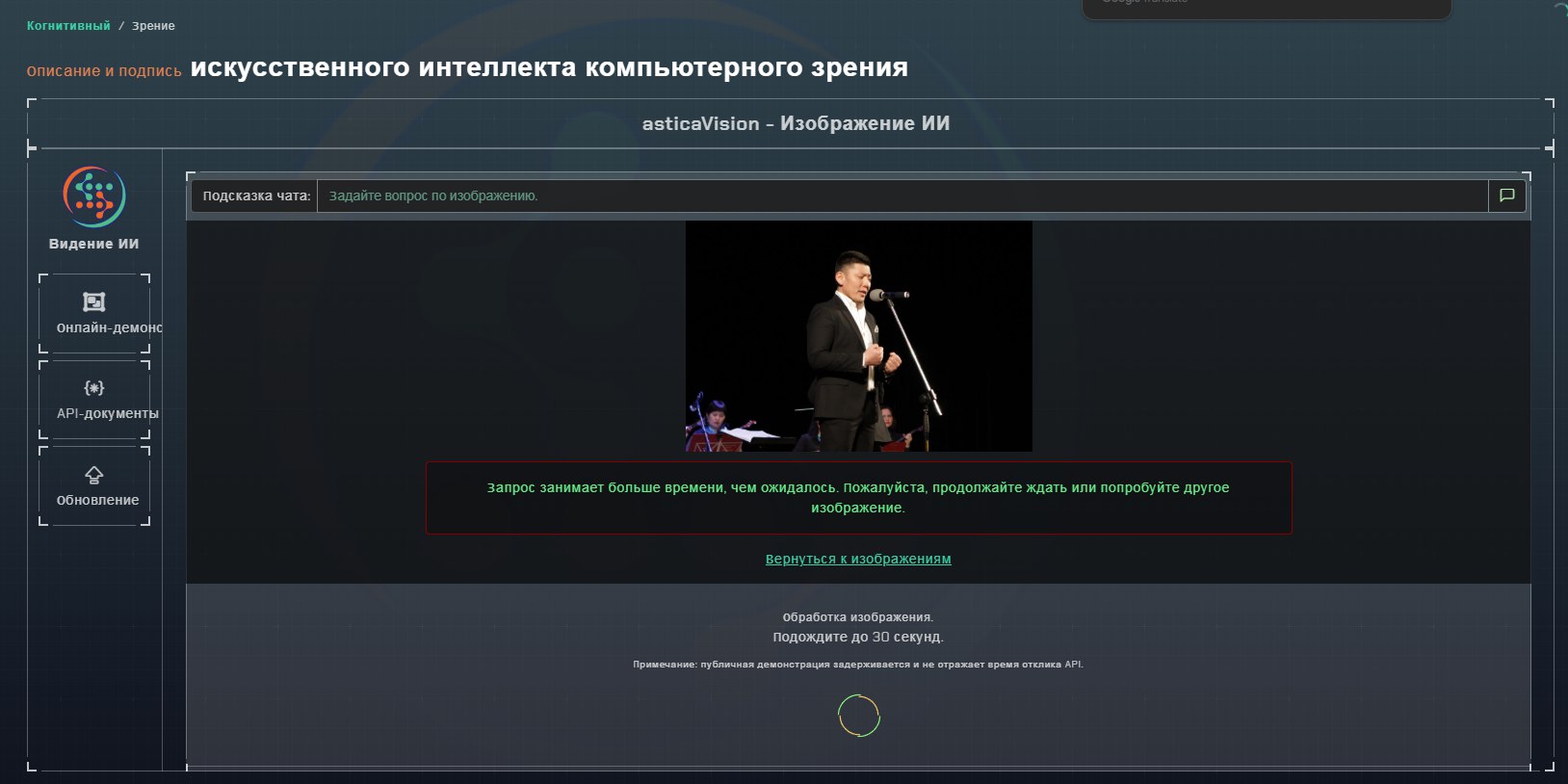


Рисунок 4.6. – Сгенерированное описание через сервис Astica.ai

Сервис Astica.ai не выдал результат даже спустя 10 минут ожидания, что делает его использование в реальных условиях затруднительным. Это ещё раз подчёркивает преимущество локального решения, управляемого и оптимизированного под конкретную задачу [2].

Кроме анализа содержательности текстов, также проведена оценка времени отклика каждого сервиса. Замер производился с момента отправки изображения до получения текстовой аннотации.

Таблица 4.2. – Таблица результатов замеров по генерации аннотации

|  |  |  |
| --- | --- | --- |
| Сервис | Время генерации | Аннотация |
| Разработанный сервис (BLIP–finetuned) | 01:21:58 | Мужчина в чёрном костюме с белой рубашкой и галстуком бабочкой стоит на сцене, держа руки перед собой, он обращён к микрофону, его взгляд сосредоточен, поза выражает собранность, позади тёмный фон и силуэты оркестрантов с инструментами. Яркий свет выделяет фигуру артиста, подчёркивая торжественность момента, обстановка создаёт ощущение живого выступления, наполненного музыкальной атмосферой и внутренним напряжением исполнения |
| Aspose.ai | 00:42:78 | На изображении мужчина в темном костюме и белой рубашке выступает на сцене с микрофоном. Вероятно, он поет, так как его руки сжаты в кулаки, и он выглядит эмоционально вовлеченным. На заднем плане видны музыканты с инструментами, что предполагает наличие оркестра или музыкального сопровождения. Сцена затемненная, внимание сосредоточено на выступающем. |
| Astica.ai | >10 минут | без результата |

Несмотря на то, что сервис Aspose.ai продемонстрировал более быстрое время генерации, его результат был менее выразительным и ориентирован скорее на поверхностное описание. В свою очередь, собственный веб–сервис, хоть и сработал чуть медленнее, сформировал содержательную и сценически точную аннотацию. При этом стабильность генерации и отсутствие технических сбоев также сыграли решающую роль.

Таким образом, несмотря на ограниченность по ресурсам и масштабу, разработанный веб–сервис решает конкретную задачу более эффективно, чем универсальные готовые решения. Разработанный сервис справляется с задачей более точно, демонстрируя способность не только распознавать объекты, но и передавать настроение, контекст и художественную составляющую изображения.

Выводы по четвертой главе

В рамках четвёртой главы была проведена комплексная проверка работоспособности веб–сервиса по генерации текстовых описаний изображений. Тестирование осуществлялось на реальных данных, что позволило убедиться в стабильности и корректности работы всех компонентов системы. Особое внимание уделялось качеству создаваемых аннотаций, которое оценивалось с помощью автоматических метрик BLEU, ROUGE–L и METEOR. Для повышения объективности использовались сразу несколько эталонных описаний на каждое изображение.

Результаты показали, что дообученная модель BLIP–finetuned способна формировать содержательные, стилистически выразительные и точные описания даже при сложной визуальной композиции. Наиболее информативной и устойчивой метрикой оказалась METEOR, в то время как BLEU был чувствителен к изменению порядка слов. Это подчёркивает необходимость использования нескольких метрик при оценке генеративных моделей.

Анализ логов позволил систематизировать собранные данные и выявить закономерности в работе модели. Отслеживались имена изображений, генерации, значения метрик, а также временные метки. На основе этого была построена диаграмма, визуализирующая изменение качества генерации. Дополнительно была реализована автоматическая система логирования, позволяющая собирать статистику без участия пользователя.

Для обоснованного сравнения разработанный веб–сервис был сопоставлен с внешним решением – платформой Aspose.ai, также использующей мультимодальные модели. В ходе сравнения использовалось одно и то же изображение. Полученные аннотации продемонстрировали различие по стилю, полноте и выразительности: собственный сервис формировал более подробные и художественно точные описания. Также был зафиксирован показатель времени генерации: BLIP–finetuned справился с задачей за 1 минуту 21 секунду, в то время как сервис Aspose.ai выдал результат за 42 секунды, а платформа Astica.ai не завершила обработку даже спустя 10 минут.

Таким образом, результаты главы подтвердили практическую применимость разработанного решения. Сервис успешно справляется с генерацией описаний, обеспечивает измеримость качества с помощью метрик и логирования, а также демонстрирует конкурентоспособность по сравнению с существующими онлайн–сервисами. Это позволяет сделать вывод о готовности системы к практическому использованию и дальнейшему развитию.

# **ЗАКЛЮЧЕНИЕ**

Основной целью магистерской работы было создание веб–сервиса для автоматической генерации текстовых описаний изображений на основе нейросетевой модели BLIP и библиотеки Hugging Face Transformers.

Для достижения поставленной цели были решены следующие задачи:

* проведён анализ существующих решений и мультимодальных технологий, применяемых для описания изображений;
* выбран стек технологий и программных инструментов, в том числе язык программирования Python, библиотеки Transformers, Torch и FastAPI;
* сформирован собственный датасет: изображения с вручную составленными аннотациями, сохранёнными в формате JSON;
* предобучена модель BLIP–large на пользовательском датасете для повышения точности и адаптивности к задаче;
* разработан и реализован веб–сервис с возможностью загрузки изображений, генерации описаний, перевода текста и экспорта результата;
* внедрены автоматические метрики оценки качества описаний (BLEU, ROUGE–L, METEOR) с логированием в файл для последующего анализа;
* проведено тестирование модели и интерфейса веб–сервиса на реальных данных;
* выполнен сравнительный анализ с внешним сервисом Aspose.ai, подтверждающий конкурентоспособность разработанного решения по качеству, стилю и устойчивости генерации.

В результате была создана полнофункциональная система, способная формировать содержательные, выразительные и грамматически корректные текстовые описания к изображениям. Реализация предобученной модели позволила учесть специфику собственных данных и добиться высокого качества генерации. Система демонстрирует стабильную работу, расширяемость и возможность дальнейшего развития.

Перспективами проекта являются обучение модели на автоматически расширяемом датасете, добавление новых языков аннотаций и адаптация сервиса под мобильные устройства.

# **БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Aspose AI. Image Description Service. URL: [https://products.aspose.ai/pdf/ru/image–description](https://products.aspose.ai/pdf/ru/image-description) (дата обращения: 07.03.2025).
2. Astica AI. AI–powered Image to Text API. URL: <https://astica.ai/> (дата обращения: 21.03.2025).
3. BLIP – Bootstrapped Language–Image Pretraining. Hugging Face. URL: https://huggingface.co/Salesforce/blip–image–captioning–base (дата обращения: 07.01.2025).
4. DALL–E. URL: https://openai.com/index/dall–e–2/ (дата обращения: 22.03.2025).
5. FastAPI. Documentation. URL: https://fastapi.tiangolo.com/ (дата обращения: 03.02.2025).
6. GitHub – Salesforce/BLIP. URL: <https://github.com/salesforce/BLIP> (дата обращения: 07.01.2025).
7. Hugging Face. Transformers Documentation. URL: https://huggingface.co/docs/transformers (дата обращения: 29.03.2025).
8. Stable Diffusion. URL: https://stablediffusion.com.ru/ (дата обращения: 22.03.2025).
9. Transformer Architectures Overview – Papers with Code. URL: https://paperswithcode.com/method/transformer (дата обращения: 15.03.2025).
10. Антонио, Д. Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow / Д. Антонио, П. Суджит ; перевод с английского А. А. Слинкин. – Москва: ДМК Пресс, 2018. – 294 с.
11. Баланов, А. Н. Машинное обучение и искусственный интеллект: учебное пособие для СПО / А. Н. Баланов. – Санкт–Петербург: Лань, 2024. – 80 с.
12. Баюк, Д. А. Практическое применение методов кластеризации, классификации и аппроксимации на основе нейронных сетей: монография / Д. А. Баюк, О. А. Баюк, Д. В. Берзин. – Москва: Прометей, 2020. – 448 с.
13. Бессмертный, И. А. Искусственный интеллект: учебное пособие / И. А. Бессмертный. – Санкт–Петербург: НИУ ИТМО, 2010. – 132 с.
14. Головко, В. А. Нейросетевые технологии обработки данных: учебное пособие / В. А. Головко, В. В. Краснопрошин. – Минск: БГУ, 2017. – 263 с.
15. Груздев, А. В. Предварительная подготовка данных в Python / А. В. Груздев. – Москва: ДМК Пресс, 2023 – Том 1: Инструменты и валидация – 2023. – 816 с.
16. Гудфеллоу, Я. Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвилль; перевод с английского А. А. Слинкина. – 2–е изд. – Москва: ДМК Пресс, 2018. – 652 с.
17. Евгеньев, А. Ценность ваших решений. Как современные технологии и искусственный интеллект меняют наше будущее / А. Евгеньев. – Москва: Альпина Паблишер, 2025. – 456 с.
18. Елисеев, А. И. Разработка программных интерфейсов веб–приложений с использованием фреймворка FastAPI: учебное пособие / А. И. Елисеев, Ю. В. Минин. – Тамбов: ТГТУ, 2024. – 81 с.
19. Йылдырым, C. Осваиваем архитектуру Transformer / C. Йылдырым, М. Асгари–Ченаглу; перевод с английского В. С. Яценкова. – Москва: ДМК Пресс, 2022. – 320 с.
20. Кацов, И. Искусственный интеллект на предприятии: руководство / И. Кацов; перевод с английского В. С. Яценкова. – Москва: ДМК Пресс, 2024. – 710 с.
21. Кэлер, А. Изучаем OpenCV 3. Разработка программ компьютерного зрения на C++ с применением библиотеки OpenCV / А. Кэлер, Г. Брэдски; перевод с английскБрэдски; Слинкина. – Москва: ДМК Пресс, 2017. – 826 с.
22. Лекун, Я. Как учится машина: Революция в области нейронных сетей и глубокого обучения / Я. Лекун. – Москва: Альпина Паблишер, 2021. – 351 с.
23. Лю, Ю. Обучение с подкреплением на PyTorch. Сборник рецептов: руководство / Ю. Лю; перевод с английского А. А. Слинкина. – Москва: ДМК Пресс, 2020. – 282 с.
24. Ма, К. Трехмерное глубокое обучение на Python: руководство / К. Ма, В. Хегде, Л. Йольан ; перевод с английского А. В. Логунова. – Москва: ДМК Пресс, 2023. – 226 с.
25. Матвеев, А. И. Цифровая обработка изображений в OpenCv. Практикум / А. И. Матвеев. – 2–е изд., стер. – Санкт–Петербург: Лань, 2023. – 104 с.
26. Митяков, Е. С. Искусственный интеллект и машинное обучение: учебное пособие для вузов / Е. С. Митяков, А. Г. Шмелева, А. И. Ладынин. – Санкт–Петербург: Лань, 2025. – 252 с.
27. Никитина, Т. П. Программирование. Основы Python для инженеров: учебное пособие для вузов / Т. П. Никитина, Л. В. Королев. – 2–е изд., стер. – Санкт–Петербург: Лань, 2025. – 156 с.
28. Рашка, С. Python и машинное обучение: крайне необходимое пособие по новейшей предсказательной аналитике, обязательное для более глубокого понимания методологии машинного обучения: руководство / С. РРашка;перевод с английского А. В. Логунова. – Москва: ДМК Пресс, 2017. – 418 с.
29. Ричардс, Т. Streamlit для Data Science: руководство / Т. Ричардс, А. Груздев; перевод с английского А. В. Груздева. – Москва: ДМК Пресс, 2024. – 354 с.
30. Селянкин, В. В. Компьютерное зрение. Анализ и обработка изображений / В. В. Селянкин. – 3–е изд., стер. – Санкт–Петербург: Лань, 2023. – 152 с.
31. Степанов, Ю. А. Системы искусственного интеллекта: учебное пособие / Ю. А. Степанов, А. В. Вылегжанина, Л. Н. Бурмин. – Кемерово: КемГУ, 2024. – 102 с.
32. Ферлитш, Э. Шаблоны и практика глубокого обучения / Э. Ферлитш; перевод с английского А. В. Логунова. – Москва: ДМК Пресс, 2022. – 538 с.
33. Филиппов, Ф. В. Нейросетевые технологии: учебное пособие / Ф. В. Филиппов. – Санкт–Петербург: СПбГУТ им. М.А. Бонч–Бруевича, 2020. – 129 с.
34. Храмов, А. Г. Методы и алгоритмы интеллектуального анализа данных: учебное пособие / А. Г. Храмов. – Самара: Самарский университет, 2019. – 176 с.
35. Эль, А. А. GPT–3 программирование на Python в примерах: руководство / А. А. Эль; перевод с английского В. С. Яценкова. – Москва: ДМК Пресс, 2023. – 218 с.
36. Ян, Э. С. Программирование компьютерного зрения на языке Python / Э. С. Ян; перевод с английского А. А. Слинкин. – Москва: ДМК Пресс, 2016. – 312 с.

# ***Приложение 1***

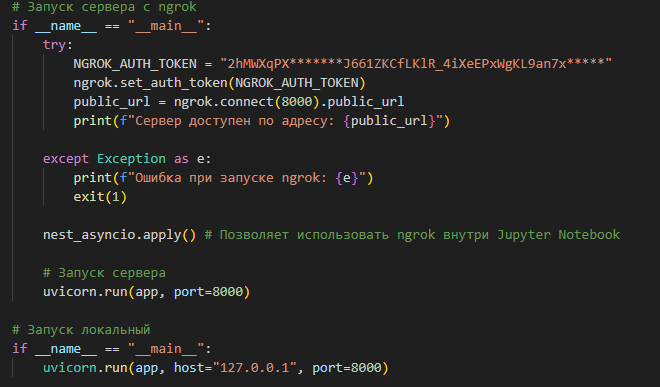
**Фрагмент предобучения модели**

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

***Приложение 2***

**Инструкция по запуску веб–сервиса**



***Приложение 3***

**Интерфейс веб–сервиса на телефоне**

