Федеральное государственное бюджетное учреждение науки
Институт системного программирования им. В. П. Иванникова РАН

На правах рукописи

Ананьев Владислав Валерьевич

Метод и программные средства адаптивного синтеза изображений высокого разрешения на основе диффузионных моделей при работе с гигапиксельными изображениями

Специальность 2.3.5 —
"Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей"

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
академик РАН, д.физ.-мат. наук
Аветисян Арутюн Ишханович

Москва — 2026

Оглавление

Введение.....4
Глава 1. Анализ предметной области и методов генеративного моделирования 13
1.1. Специфика гигапиксельных изображений как объекта автоматизированного анализа.....13
1.1.1. Структура и принципы обработки гигапиксельных изображений.....14
1.1.2. Репрезентативность обучающих выборок: доменный сдвиг и дисбаланс классов.....20
1.2. Анализ генеративных архитектур и методов доменной адаптации.....26
1.2.1. Сравнительный анализ GAN, VAE и диффузионных моделей.....26
1.2.2. Фундаментальные модели как механизм извлечения признаков и визуального обусловливания.....34
1.3. Методы управления генерацией и оценка вычислительной сложности диффузионных моделей.....43
Глава 2. Метод адаптивного синтеза изображений высокого разрешения.....49
2.1 Математическая модель диффузионного процесса в латентном пространстве.....49
2.1.1 Формализация латентного перехода с использованием SD3-VAE и визуального обусловливания UNI2-h.....49
2.1.2 Оптимизация архитектуры диффузионных трансформеров (DiT) на основе метода KV-компрессии.....53
2.2 Метод доменной специализации латентных диффузионных моделей.....56
2.2.1 Прогрессивное обучение и формирование набора базовых моделей....57
2.2.2 Доменная адаптация и перенос стиля средствами низкоранговых адаптеров (LoRA).....59
2.3 Алгоритм локального контекстно-ориентированного встраивания объектов.. 62
Глава 3. Программные средства и архитектура системы генерации.....66
3.1 Проектирование модульной архитектуры и программных конвейеров.....66
3.2 Алгоритмическое обеспечение прикладных сценариев синтеза.....71
3.2.1 Алгоритмическая реализация метода доменной специализации латентных диффузионных моделей.....72
3.2.2 Алгоритм контекстно-ориентированного встраивания объектов.....77
3.3 Аппаратно-программная оптимизация и интеграция с конвейерами прикладного анализа.....84
Глава 4. Экспериментальная оценка метода и программных средств..... 92

4.1. Методология оценки.....92
4.1.1. Описание тестовых стендов и характеристик исходных датасетов..... 93
4.1.2. Система метрик для оценки вычислительной эффективности и качества синтеза.....96
4.2. Результаты оценки эффективности оптимизаций и производительности программного обеспечения.....100
4.3. Апробация результатов на задачах цифровой патологии (сквозной эксперимент)..... 103
Заключение..... 108
Список литературы..... 110
Список сокращений и условных обозначений..... 117
Приложение А. Программный модуль подготовки гистологических изображений для задач машинного обучения..... 120
Приложение Б. Исходный код модулей для обучения моделей.....121
Б1. Исходный код модуля дообучения набора базовых моделей в режиме прогрессивного обучения..... 121
Б2. Исходный код модуля для обучения низкоранговых адаптеров LoRA..... 125
Приложение В. Исходный код модуля контекстно-ориентированного встраивания объектов.....132
Приложение Г. Исходный код модуля для интеграции механизма KV-compression в слои самовнимания (self-attention).....137
Приложение Д. Исходный код модуля для предварительного вычисления и кэширования эмбеддингов..... 138

Введение

Актуальность темы исследования. Методы обработки изображений с применением глубоких нейронных сетей (далее ГНС), получившие активное развитие в начале 2010-х годов, привели к их успешному внедрению в широком спектре задач компьютерного зрения. Тем не менее, в процессе обучения и применения ГНС постоянно возникают три фундаментальных проблемы: высокая вычислительная сложность, потребность в наличии крупных наборов данных и крайне высокая стоимость их экспертной разметки. Для преодоления дефицита аннотированных данных традиционно применяют техники аугментации, повышающие разнообразие обучающей выборки и снижающие риск переобучения. Распространены и альтернативные стратегии обучения — обучение без учителя (unsupervised learning) и самообучение (self-supervised learning), к которым относятся методы контрастивного обучения (SimCLR, BYOL) и архитектуры на основе маскирования патчей (Masked Autoencoders).

В ряде прикладных задач потенциала таких методов достаточно. Однако существуют предметные области, где классы изображений обладают рядом особенностей: специфичная топология, экстремально высокое разрешение и строгие требования к качеству структур и объектов, присутствующих на изображении. При работе с подобными данными ограничения традиционных подходов проявляются более остро. Показательный пример — анализ биомедицинских гигапиксельных изображений.

Структура гигапиксельных изображений (например, Whole-Slide Images, размер которых может превышать $10^5 \times 10^5$ пикселей) организована в виде мультимасштабной иерархической пирамиды разрешений (рисунок 1). В рамках вычислительного анализа такие изображения декомпозируются на фрагменты (тайлы) меньшего размера на различных уровнях оптического увеличения. Обработка полученных данных осложнена проблемой выраженного ковариантного сдвига (covariate shift), который возникает из-за различий в аппаратных характеристиках, матрицах камер и алгоритмах цветопередачи

сканирующего оборудования. Эти факторы порождают существенные расхождения между распределениями признаков обучающей и тестовой сред. Ситуация часто усугубляется экстремальным классовым дисбалансом, при котором целевые структуры иногда занимают менее 5% площади изображения.

Общепринятым стандартом в практике обучения моделей остается применение классических методов аугментирования — колориметрической нормализации и базовых геометрических преобразований. На репрезентативных выборках такие подходы зарекомендовали себя хорошо, выполняя функцию регуляризации и повышая обобщающую способность моделей. Однако при сильном дисбалансе классов или малом объеме выборки их эффективность резко снижается: они лишь интерполируют и пространственно трансформируют значения существующих пикселей, не порождая принципиально новых валидных топологических конфигураций объектов. Отсюда вытекает объективная необходимость разработки специализированных методов адаптивной генерации искусственных изображений.

Для решения задачи синтеза изображений ранее активно применялись генеративно-состязательные сети (Generative Adversarial Network, GAN) и вариационные автокодировщики (Variational Autoencoder, VAE). Современный этап характеризуется переходом к вероятностным диффузионным моделям (Denoising Diffusion Probabilistic Models, DDPM), в частности к латентным диффузионным моделям (Latent Diffusion Model, LDM) на базе трансформеров (Diffusion Transformer, DiT). Такой переход открывает технологическую возможность синтеза морфологически достоверных данных в рамках заданного предметного домена.

Существенный вклад в развитие методов анализа и синтеза изображений, составляющих основу данного исследования, внесли отечественные и зарубежные научные школы компьютерного зрения, обработки изображений и генеративного моделирования. В российской научной традиции следует отметить работы школы Ю. В. Визильтера в области компьютерного зрения и распознавания визуальных

объектов, работы Д. С. Ватолина и его научной школы по обработке изображений, видеоанализу и оценке качества визуальных данных, а также исследования научных коллективов ФИЦ ИУ РАН, Сколтеха и Института искусственного интеллекта AIRI, развивающих современные методы машинного обучения, мультимодального анализа данных и генеративного искусственного интеллекта. Среди зарубежных направлений ключевое значение имеют работы Я. Гудфеллоу и соавторов из Университета Монреаля по генеративно-состязательным сетям, Д. Кингмы и М. Веллинга из Амстердамского университета по вариационным автокодировщикам, а также Я. Сонга из Стэнфордского университета по вероятностным диффузионным и score-based моделям, ставшим основой современных подходов к синтезу изображений высокого разрешения.

Ключевую роль в управлении генерацией играет механизм обусловливания (conditioning), задающий характеристики синтезируемого объекта. Традиционно он опирается на текстовые описания (text prompt) либо на визуальные примеры (изображения-референсы). Однако при работе со сложноструктурированными изображениями, где требуется строгий контроль пространственного расположения объектов, текстовое обусловливание сталкивается с фундаментальными ограничениями: сбор исчерпывающей текстовой разметки крайне трудоемок, а естественный язык не позволяет однозначно описывать сложные морфологические паттерны, что приводит к непредсказуемым результатам синтеза. В связи с этим целесообразно сместить фокус на визуальное обусловливание на базе фундаментальных моделей. Предварительно обученные на десятках миллионов изображений, они позволяют извлекать векторы признаков с богатым семантическим описанием и выступают детерминированной альтернативой, опирающейся на композицию структур в многомерном векторном пространстве, а не на их словесное описание. Разработка методов адаптивного управления визуальным обусловливанием и процессом обратной диффузии в LDM представляет собой важную научно-техническую задачу, решение которой обеспечит синтез высококачественных обучающих выборок, минимизацию

доменного сдвига и повышение надежности прикладного программного обеспечения.

Целью диссертационной работы является разработка метода, алгоритмов и программных средств адаптивного синтеза изображений высокого разрешения на основе латентных диффузионных трансформеров с применением техник, направленных на компенсацию ковариантного сдвига и классового дисбаланса в обучающих выборках.

Для достижения поставленной цели сформулированы следующие **задачи исследования**:

1. Выполнить системный анализ методов генерации, формализовать схему латентной диффузии и разработать метод адаптивного синтеза, интегрирующий механизмы визуального обусловливания и архитектурную оптимизацию диффузионных трансформеров (KV-компрессию) для создания модульного и вычислительно эффективного конвейера.
2. Разработать метод доменной специализации латентных диффузионных моделей, объединяющий стратегию прогрессивного обучения для адаптации морфологии и концепцию низкоранговых адаптеров (LoRA) для тонкой параметрической настройки стиля под особенности конкретных наборов данных.
3. Разработать алгоритм локального контекстно-ориентированного встраивания объектов (inpainting) для направленного обогащения обучающих выборок, обеспечивающий пространственно-детерминированную интеграцию целевого объекта в фоновое изображение на уровне латентных представлений с процедурой автоматического колориметрического согласования.
4. Создать программно-алгоритмический комплекс для организации оптимизированных вычислительных процессов генерации (включая асинхронное кэширование и декомпозицию гигапиксельных слайдов), обосновать гибридную систему метрик и провести экспериментальную

верификацию разработанных методов на данных медицинской визуализации.

Основные положения, выносимые на защиту:

1. Метод адаптивного синтеза изображений высокого разрешения, основанный на архитектуре латентных диффузионных трансформеров.
2. Новый метод доменной специализации латентных диффузионных моделей.
3. Алгоритм локального контекстно-ориентированного встраивания объектов.
4. Программно-алгоритмический комплекс адаптивного синтеза с модульной архитектурой и гибридная методология многофакторной оценки качества.

Научная новизна основных результатов исследования заключается в следующем:

1. Предложен метод адаптивного синтеза гистологических изображений высокого разрешения, новизна которого заключается в гибридном объединении архитектуры латентных диффузионных трансформеров (DiT), механизмов визуального обусловливания и алгоритмов сжатия матриц внимания (KV-компрессии). Формируемый таким сочетанием вычислительно эффективный архитектурный фундамент, в отличие от стандартных монолитных решений, является модульным: он обеспечивает строгий контроль над морфологической достоверностью генерируемых структур и позволяет независимо применять компоненты генеративного конвейера при пакетной обработке гигапиксельных изображений.
2. Предложен метод доменной специализации латентных диффузионных моделей, новизна которого заключается в двухэтапной декомпозиции

процесса адаптации: применении прогрессивного обучения с фиксацией оптического увеличения для усвоения морфологии и использовании библиотеки низкоранговых адаптеров (LoRA) для изолированной коррекции стиля. В противовес доминирующей парадигме text-to-image, полный отказ от текстового обусловливания позволяет надежно удерживать геометрию структур базовым визуальным механизмом, в то время как стиль каждого конкретного набора данных кодируется отдельным компактным адаптером. Такое решение делает систему доменной адаптации масштабируемой, позволяя параметрически переключать визуальные характеристики синтеза без полного переобучения базовой модели.

3. Разработан алгоритм локального контекстно-ориентированного встраивания объектов, новизна которого определяется совмещением целевого объекта с фоном непосредственно в латентном пространстве признаков, дополненным автоматизированным гистограммным подбором фонового фрагмента и колориметрическим согласованием (в пространстве CIELAB). Отличие от существующих методов инпейнтинга заключается в достижении бесшовной интеграции объекта без использования дополнительных обучаемых архитектурных модулей и пространственной разметки.
4. Предложена гибридная методология оценки качества синтеза, новизна которой состоит в одновременном учете структурно-семантических свойств и генеративного разнообразия образцов. На основе данной методологии и созданного модульного программно-алгоритмического комплекса экспериментально доказана применимость синтетических данных для повышения качества обучения прикладных нейросетевых моделей в условиях экстремального классового дисбаланса

Практическая и теоретическая значимость работы. Предложен программно-алгоритмический комплекс для адаптивного синтеза гистологических

изображений высокого разрешения, позволяющий преодолевать дефицит клинических данных и компенсировать доменный сдвиг, возникающий из-за различных факторов, возникающих в процессе подготовки биомедицинских изображений. Разработаны алгоритмы локального контекстно-ориентированного синтеза и доменной адаптации, с помощью которых экспериментально продемонстрировано повышение средних значений метрик качества и снижение их дисперсии при обучении и тестировании прикладных моделей сегментации и классификации в рамках многоэтапных диагностических конвейеров.

Предложенные решения обладают значительным потенциалом и могут служить основой для будущих исследований в области разработки методов условного синтеза медицинских изображений с применением моделей латентной диффузии.

Сформулированные в диссертации теоретические концепции и накопленный эмпирический опыт использованы при разработке программы для ЭВМ “Программный модуль подготовки гистологических изображений для задач машинного обучения” (свидетельство № 2024663138 от 04.06.2024 г. [5]). Разработанные методические и программные решения внедрены в научную и практическую деятельность 1-го патологоанатомического отделения ФГБУ “НМИЦ АГП им. В.И. Кулакова” Минздрава России при выполнении исследований, связанных с применением методов машинного обучения и компьютерного зрения для анализа гистологических изображений: справка о внедрении результатов диссертационной работы от 28.05.2026.

Апробация работы. Основные результаты диссертации были изложены на следующих конференциях:

1. III Международная научно-практическая конференция “Анализ данных в медицине” 09 июня 2023 года.
2. IV Международная научно-практическая конференция “Анализ данных в медицине” (Data Science in Medicine), 17 мая 2024 года.

3. VIII Сеченовский Международный биомедицинский саммит: научно-технологическая кооперация в медицинской отрасли (SIBS-2024), 6 ноября 2024 года.
4. Открытая конференция ИСП РАН, 11-12 декабря 2024 года.
5. VIII Петербургский медицинский инновационный форум с международным участием, 17 мая 2025 года.
6. В период выполнения диссертационного исследования автор являлся победителем конкурсного отбора 2024 года на назначение стипендии Президента Российской Федерации для аспирантов и адъюнктов, проводящих научные исследования в рамках реализации приоритетов научно-технологического развития Российской Федерации; приказ Минобрнауки России от 03.07.2024 г. № 428.

Личный вклад. Все представленные в рамках диссертационной работы результаты получены лично автором.

Публикации. Основные результаты диссертации изложены в 4 печатных изданиях. Работы [1-3] входят в международные системы цитирования Scopus и Web of Science. Работа [4] индексируется в РИНЦ. Также получено 1 свидетельство о государственной регистрации программы для ЭВМ [5].

В публикации [1] совместно с соавторами сформулирована задача и методика подсчета статистических показателей; автором выполнена разработка базы программного кода для подсчета статистических показателей и анализа результатов, а также часть работы по подготовке к публикации. В статьях [2, 3] автором выполнены постановка задачи исследования, техническая реализация моделей и их обучение, проверка и валидация результатов; автор участвовал в составлении протокола аннотирования данных и подготовке текста публикаций. В публикации [4] автором разработаны постановка задачи и планирование экспериментов, выполнены проверка и валидация результатов и подготовка текста публикации.

Работа выполнена при поддержке программы НЦМУ “Цифровой биодизайн и персонализированное здравоохранение” в части разработки программных средств анализа биомедицинских изображений и формирования воспроизводимых вычислительных конвейеров.

Объем и структура работы. Диссертация состоит из введения, 4 глав, заключения, списка литературы, списка сокращений и условных обозначений и 5 приложений. Полный объем диссертации 140 страниц текста с 9 рисунками и 9 таблицами. Список литературы содержит 74 наименования.

Глава 1. Анализ предметной области и методов генеративного моделирования

Настоящая глава посвящена системному анализу предметной области и обоснованию выбора инструментальной базы исследования. Рассматриваются структурные особенности гигапиксельных изображений как класса данных, ограничения существующих подходов машинного обучения при работе с такими данными, а также эволюция генеративных архитектур. Экспериментальная валидация предложенных в работе решений ориентирована на прикладную задачу анализа гигапиксельных изображений в области цифровой патологии (Whole-Slide Images, WSI), а именно на локализацию пространственно-минорных целевых структур (лимфоваскулярной инвазии, ЛВИ) с применением глубоких нейронных сетей [11, 12]. Такой сценарий служит репрезентативным примером области гигапиксельных изображений, в которой ограничения классических методов машинного обучения проявляются наиболее остро.

1.1. Специфика гигапиксельных изображений как объекта автоматизированного анализа

Гигапиксельные изображения образуют самостоятельный класс данных компьютерного зрения, отличающийся экстремальным пространственным размером. Объем и внутренняя организация таких изображений принципиально отличают их от типичных объектов компьютерного зрения и предъявляют особые требования к методам вычислительной обработки. Ниже рассматриваются структурные характеристики гигапиксельных изображений, вытекающие из них принципы автоматизированного анализа (подраздел 1.1.1), а также проблемы репрезентативности обучающих выборок, такие как доменный сдвиг и дисбаланс классов (подраздел 1.1.2).

1.1.1. Структура и принципы обработки гигапиксельных изображений

Гигапиксельные изображения выделяются в первую очередь за счет экстремального пространственного размера. Например, линейные размеры некоторых гигапиксельных изображений достигают 10^5 и более пикселей, а суммарный объем несжатых данных одного такого изображения исчисляется несколькими гигабайтами. Данные такого масштаба возникают в ряде прикладных областей: дистанционном зондировании Земли и аэрокосмической съемке, панорамной и широкоформатной фотографии, оптической и электронной микроскопии. Однако, независимо от происхождения изображений, их объединяет общая вычислительная проблема, которая заключается в невозможности полного размещения данных в оперативной памяти (RAM) или видеопамяти (VRAM).

Сходство ограничений делает уместным единый взгляд на подходы к обработке таких данных. В дистанционном зондировании Земли спутниковые и аэрофотоснимки хранятся в форматах GeoTIFF и облачно-оптимизированного GeoTIFF (COG). В астрономии обзорные снимки неба и мозаики телескопических наблюдений используют формат FITS. В микроскопии гигапиксельные и терапиксельные массивы порождают цифровая патология, коннектомика и объемная флуоресцентная микроскопия, для которых часто выбирают контейнер OME-TIFF. Несмотря на различие физической природы сигнала, все перечисленные области используют схожий принцип организации данных в виде многомасштабных тайловых пирамид с произвольным доступом к регионам, а при их вычислительном анализе возникает необходимость в потоковой фрагментарной обработке.

Безусловно, между доменами существуют различия, но они сосредоточены в деталях предметной области, а не в базовых принципах обработки. Так, спутниковые снимки часто многоспектральны; астрономические изображения требуют фотометрической калибровки и работы с широким динамическим диапазоном; данные цифровой микроскопии могут включать оптические срезы по глубине (z-стек). Эти особенности влияют на этапы радиометрической нормализации и отбора информативных фрагментов, но не меняют общую схему

действий при вычислительном анализе: декомпозиция, фрагментарная обработка и агрегация результатов.

В качестве репрезентативного примера гигапиксельных изображений в настоящей работе рассматриваются упомянутые выше WSI — растровые копии микроскопических препаратов [16]. Хотя эти изображения выступают основной экспериментальной базой исследования, разрабатываемые на их примере принципы обработки носят универсальный характер. При замене предметно-специфичных компонентов предобработки (нормализации, критериев отбора фрагментов, доменного кодировщика признаков) предлагаемый в работе метод допускает прямой перенос на иные классы гигапиксельных изображений, а специфика WSI привлекается исключительно для иллюстрации предложенных подходов.

Хранение гигапиксельного изображения и доступ к нему организуют в виде мультимасштабной иерархической пирамиды разрешений. Изображение представлено набором уровней $L_0, L_1, \ldots, L_{K-1}$, где базовый уровень L_0 соответствует исходному (максимальному) разрешению, а каждый следующий уровень формируется с помощью понижающей передискретизации предыдущего с коэффициентом s (как правило, $s = 2$):

$$dim(L_k) = \lfloor W/s^k \rfloor \times \lfloor H/s^k \rfloor, \quad k = 0, 1, \ldots, K-1 \tag{1.1}$$

где $W \times H$ — разрешение базового уровня в пикселях, K — число уровней пирамиды, $\lfloor \cdot \rfloor$ — округление вниз. Пирамидальное представление обеспечивает быстрый произвольный доступ к фрагменту изображения на требуемом масштабе без полной декомпрессии файла и реализуется как универсальными контейнерами (пирамидальный OME-TIFF), так и специализированными отраслевыми форматами. Для WSI это форматы SVS, NDPI, MRXS и стандарт DICOM [13], а потоковое чтение регионов обеспечивают библиотеки наподобие OpenSlide или cuCIM [14, 15], предоставляющие единый интерфейс поверх разнородных проприетарных контейнеров.

Каждый уровень пирамиды, в свою очередь, разбит на фрагменты, которые хранятся независимо и сжимаются блочными кодеками (как правило, JPEG или JPEG 2000). Именно блочное хранение и делает возможным произвольный доступ к выбранному региону без декомпрессии всего файла. Подсистема чтения декодирует лишь те тайлы, что пересекаются с запрошенной областью на заданном уровне. Платой за компактность выступает сжатие с потерями, при котором вносимые кодеком артефакты добавляют программно обусловленный источник вариативности данных.

Физический масштаб уровня задается шагом дискретизации — линейным размером участка сканируемой области изображения, проецируемого на один пиксель. В отличие от аппаратного увеличения, этот параметр является объективной мерой пространственного разрешения и позволяет корректно сопоставлять данные, полученные с различных устройств. В задачах дистанционного зондирования Земли этот показатель известен как разрешение на местности (Ground Sampling Distance, GSD), а в цифровой микроскопии — как число микрометров на пиксель (Microns Per Pixel, MPP). На практике значение MPP ≈ 0,25 μm/px обычно соответствует объективу ×40, а MPP ≈ 0,5 μm/px объективу ×20. Иерархическая структура данных WSI и типичная схема их декомпозиции приведены на рисунке 1.1.

Обучение и применение глубоких нейронных сетей (далее — ГНС) оперируют фрагментами фиксированного размера, поэтому ключевой операцией предобработки гигапиксельного изображения выступает его декомпозиция (тайлинг) — разбиение выбранного уровня пирамиды на множество непересекающихся или частично перекрывающихся фрагментов (тайлов/патчей). Технологическая необходимость тайлинга обусловлена ограничениями объема видеопамяти графических ускорителей: даже один уровень гигапиксельной пирамиды не помещается в память ускорителя целиком. Размер тайла выбирают из ряда степеней двойки (256×256, 512×512, 1024×1024 и т.д.), что служит компромиссом между объемом охватываемого пространственного контекста и вычислительными затратами на обработку.

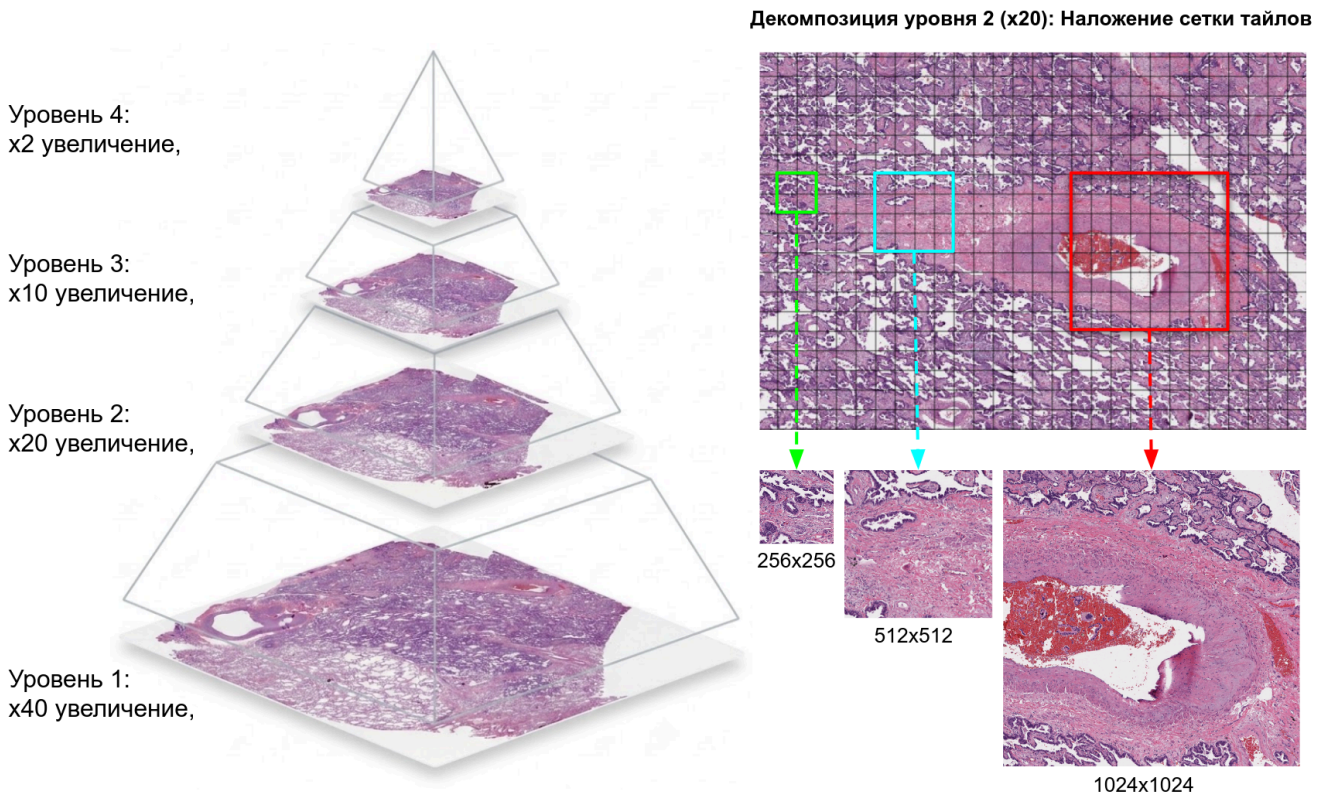


Рисунок 1.1 — Иерархическая структура гигапиксельного изображения WSI. Слева показана пирамида уровней оптического увеличения (×2-×40); справа приведен пример декомпозиции уровня с увеличением ×20 на фрагменты различного разрешения (256×256, 512×512, 1024×1024)

Выбор рабочего уровня пирамиды определяется природой целевой (downstream) задачи. Высокие разрешения сохраняют пространственно-текстурные признаки и геометрию мелких объектов, необходимые для надежной сегментации и классификации. Понижение разрешения ускоряет обработку, но ведет к утрате значимых деталей, а чрезмерно высокое разрешение сужает охватываемый контекст и кратно увеличивает число тайлов. Для морфологического анализа объектов на WSI базовыми являются увеличения ×20 и ×40.

Прикладные задачи анализа гигапиксельных изображений образуют иерархию по требуемой пространственной детальности. Классификация присваивает метку фрагменту или изображению целиком; детектирование локализует объекты ограничивающими рамками; семантическая сегментация формирует попиксельную карту структур. Чем выше детальность задачи, тем

дороже разметка и тем чувствительнее модель к качеству обучающих данных. В настоящей работе экспериментально исследуются обе крайние точки этого спектра — сегментация протяженных структур и классификация локальных целевых объектов на WSI.

Обработке предшествует этап отбора информативных фрагментов. Полезное содержимое нередко занимает лишь часть площади гигапиксельного изображения (для WSI это области с тканями на фоне пустого предметного стекла), поэтому сплошной тайлинг породил бы значительную долю неинформативных фрагментов. Выделение области интереса выполняют пороговыми методами в подходящих цветовых пространствах и каналах. Например, по насыщенности в модели HSV с последующей очисткой результата операциями математической морфологии (замыканием мелких разрывов, удалением шумовых компонент). На дальнейшие этапы обработки передаются лишь те фрагменты, доля полезного содержимого в которых превышает заданный порог.

Цветовое (канальное) представление гигапиксельных изображений зависит от способа их получения и влияет на последующую обработку. В общем случае данные представляют собой многоканальный растр, а корректное сопоставление изображений из разных источников может требовать радиометрической или колориметрической нормализации к эталону. В отдельных областях применяются специализированные фотометрические модели, учитывающие физику формирования сигнала. Однако их подробное рассмотрение выходит за рамки настоящей работы, посвященной вычислительным и программным аспектам синтеза.

Дополнительной особенностью при вычислительном анализе гигапиксельных изображений выступает преобладание слабой (грубой) разметки. Экспертная аннотация чаще доступна на уровне всего гигапиксельного изображения, тогда как обработка ведется на уровне фрагментов. Данная особенность порождает парадигму обучения с множественными представлениями (multiple-instance learning, MIL), в рамках которой изображение трактуется как “пакет“ фрагментов с единой меткой, а вклад отдельных тайлов агрегируется. В то

же время подробная попиксельная разметка, необходимая для задач сегментации, формируется вручную и остается наиболее дорогостоящим ресурсом всего конвейера; для WSI такую работу выполняют профильные специалисты предметной области.

Дефицит размеченных данных отчасти компенсируется крупными публичными коллекциями, существующими для многих областей, где проводится анализ гигапиксельных изображений. Их состав, однако, смещен в сторону распространенных классов и типовых протоколов получения данных, тогда как редкие целевые объекты обычно малочисленны. Применимость публичных данных к конкретной задаче ограничивается и доменным сдвигом между изображениями, возникающим из-за различий в условиях их получения. Для прикладной задачи настоящей работы такими коллекциями служат архивы WSI (в частности, DHMC и наборы профильных соревнований), и проблема репрезентативности обучающей выборки сохраняется даже при их наличии.

Гигапиксельный масштаб предъявляет жесткие требования к вычислительной инфраструктуре. Многократное чтение тайлов с дисковой подсистемы в ходе обучения превращает подсистему ввода-вывода в узкое место: пропускная способность хранилища и стоимость декодирования сжатых тайлов ограничивают загрузку графического ускорителя, который значительную долю времени простаивает в ожидании данных. Дополнительную нагрузку создает извлечение признаков предобученными кодировщиками, которое выполняется для каждого фрагмента. Перечисленные ограничения определяют необходимость специализированных приемов аппаратно-программной оптимизации вычислительного конвейера, которым посвящена третья глава.

Совокупность таких факторов, как гигапиксельный масштаб, иерархическая организация, необходимость декомпозиции, дороговизна аннотирования и интенсивная нагрузка на систему ввода-вывода, формирует специфический вычислительный контекст, в котором традиционные приемы расширения данных оказываются недостаточно эффективными. Наиболее уязвимым звеном

становится репрезентативность обучающего множества, к анализу которой переходит следующий подраздел.

1.1.2. Репрезентативность обучающих выборок: доменный сдвиг и дисбаланс классов

Обобщающая способность нейросетевой модели напрямую зависит от репрезентативности обучающей выборки, а именно от того, насколько распределение обучающих данных согласовано с распределением данных, поступающих на вход модели в условиях ее эксплуатации. При работе с гигапиксельными изображениями это согласование систематически нарушается. Ключевыми причинами выступают доменный (ковариантный) сдвиг и дисбаланс классов.

В терминах статистического машинного обучения это различие между выборками формализуется следующим образом. Пусть $x \in \mathrm{X}$ — вектор признаков отдельного фрагмента (тайла), $y \in \mathrm{Y}$ — сопоставленная ему метка (класс объекта или попиксельная маска). Обучение выполняется на выборке из исходного (source) распределения $P_s(x, y)$ а применение модели происходит на данных целевого (target) распределения $P_t(x, y)$. Совместное распределение раскладывается как $P(x, y) = P(y \mid x)\, P(x)$, и в зависимости от того, какой из сомножителей изменяется при переходе от обучения к эксплуатации, различают несколько типов рассогласования данных (dataset shift) [17, 18]. Доминирующим для рассматриваемой области является ковариантный сдвиг — ситуация, при которой апостериорная зависимость метки от признаков сохраняется, а маргинальное распределение признаков меняется:

$$P_s(x) \neq P_s(y), \quad P_s(y \mid x) = P_t(y \mid x) \tag{1.2}$$

Условие (1.2) означает, что семантика объектов, а именно связь “изображение→класс”, остается неизменной, тогда как визуальное распределение самих изображений смещается. Модель, обученная минимизацией эмпирического

риска на P_s, в общем случае не минимизирует риск на P_t, что проявляется в падении качества при переносе модели на данные из новой среды [17].

Ковариантный сдвиг следует отличать от смежных типов рассогласования. При сдвиге априорных вероятностей (prior probability shift) изменяется маргинальное распределение меток $P(y)$ при сохранении правдоподобия $P(x \mid y)$ — этому случаю соответствует изменение баланса классов. При концептуальном сдвиге (concept shift) меняется сама зависимость $P(y \mid x)$, что для рассматриваемых задач нехарактерно, так как критерий принадлежности объекта к классу инвариантен относительно оборудования. Теоретическая коррекция ковариантного сдвига известна — несмещенную оценку целевого риска дает взвешивание обучающих примеров отношением плотностей $\omega(x) = P_t(x)/P_s(y)$ (importance weighting) [17]. Практическая применимость приема ограничена, поскольку надежная оценка отношения плотностей в высокоразмерном пространстве изображений сама является трудной задачей, а при малом объеме минорного класса становится статистически неустойчивой. Указанное обстоятельство смещает фокус с прямой коррекции весов на методы, изменяющие саму обучающую выборку.

Связь доменного сдвига с обобщающей способностью формализуется в теории адаптации доменов. Согласно результатам, полученным авторами работы [23], ошибка модели на целевом домене ограничена сверху суммой ошибки на исходном домене и меры расхождения между доменами в признаковом пространстве. Из этой оценки следует принципиальный вывод о том, что сократить целевую ошибку можно либо уменьшая расхождение распределений, либо повышая качество обучения на исходном домене. Методы доменной адаптации реализуют первое направление; направленный синтез дополнительных обучающих образцов работает со вторым, расширяя и уплотняя исходное распределение в окрестности недостаточно представленных классов. Эти пути дополняют друг друга, что обосновывает выбор генеративного подхода в качестве основного инструмента исследования.

Источники ковариантного сдвига при получении гигапиксельных изображений носят преимущественно аппаратный характер. К ним относятся различия в характеристиках сенсоров и оптических трактов, спектральных параметрах источников освещения, алгоритмах автоматической фокусировки и калибровки баланса белого; собственный вклад вносит сжатие с потерями на этапе хранения. В рамках решаемой в настоящей работе прикладной задачи к перечисленным факторам добавляется вариативность пробоподготовки и окрашивания препаратов (например, колебания концентрации и времени экспозиции красителей при окраске гематоксилином и эозином). Эта предметная специфика, оставаясь медицинской по происхождению, полностью укладывается в общую модель смещения распределения признаков (1.2): даже идентичные по содержанию изображения, полученные в разных условиях, демонстрируют различающиеся цветовые и текстурные статистики, и распределение $P_s(x)$ обучающей выборки отклоняется от распределения $P_t(x)$ целевой среды [22].

Описанная проблематика в той или иной форме проявляется во всех областях, связанных с анализом гигапиксельных изображений, что дополнительно подтверждает ее фундаментальный характер. В дистанционном зондировании ковариантный сдвиг порождается различиями сенсоров спутниковых платформ, сезонными и атмосферными условиями съемки, углом освещения и географической спецификой территорий, вследствие чего модель, обученная на снимках одного региона или сезона, систематически теряет качество на других. В астрономических обзорах источником сдвига выступают различия инструментов, экспозиций и условий наблюдения. Систематический характер подобных межколлекционных смещений и их влияние на обобщающую способность были отмечены еще в общих исследованиях смещения наборов данных в компьютерном зрении [67], где показано, что модель, обученная на одной коллекции, заметно теряет в качестве при переносе на другую даже в пределах одной задачи. Аналогично проявляется и проблема дисбаланса классов: редкость целевого объекта характерна не только для медицинских данных, но и, например, для задач

обнаружения малоразмерных или редких объектов на аэрокосмических снимках, где целевые структуры занимают ничтожную долю площади кадра. Общность обеих проблем для разных типов гигапиксельных изображений означает, что и подход к их преодолению средствами направленного генеративного синтеза обладает потенциалом переноса за пределы исходной предметной области, хотя в настоящей работе он разрабатывается и проверяется на полигоне WSI.

Стандартным ответом на ковариантный сдвиг служит нормализация изображений к эталону — радиометрическая в одних областях, колориметрическая в других. Для гистологических изображений роль такой процедуры выполняют методы нормализации окраски (Reinhard, Macenko и их модификации) [19, 20], выравнивающие цветовые статистики фрагментов. Подобные преобразования снижают межисточниковую вариативность, однако действуют исключительно в пространстве уже существующих пикселей и не привносят новой информации о форме и взаимном расположении объектов. Их регуляризирующий потенциал ограничен тем, что на репрезентативных выборках они эффективны, но в условиях дефицита данных не компенсируют структурную неполноту обучающего множества.

Более общий арсенал доменной адаптации включает методы выравнивания признаковых распределений — приведение статистик скрытых представлений исходного (source) и целевого (target) доменов (в частности, согласование вторых моментов признаков, CORAL) и состязательную адаптацию, при которой доменный дискриминатор вынуждает кодировщик формировать инвариантные к домену представления. Эти подходы сокращают разрыв между доменами, но нацелены на адаптацию решающей модели к уже имеющимся данным и не порождают новых образцов минорного класса. Когда редкий класс представлен единичными примерами, выравнивание признаков не устраняет первопричину — структурную неполноту выборки. Нормализацию входных изображений при этом удобно трактовать как частный, низкоуровневый случай доменной адаптации, действующий непосредственно в пространстве пикселей.

Второй фактор, усугубляющий ситуацию, — экстремальный дисбаланс классов. Целевые объекты на гигапиксельных изображениях нередко являются пространственно-минорными и занимают доли процента площади изображения. Показательным примером является профиль выборки, использованной в прикладной части настоящей работы, где из 8212 размеченных структур сосудистого русла на 207 WSI лишь 216 объектов (менее 3%) содержали признаки целевого минорного класса. При столь выраженной асимметрии стандартное обучение смещается в пользу мажоритарного класса, а чувствительность модели к редким, но критически важным паттернам деградирует [21]. Дисбаланс дополнительно затрудняет доменную адаптацию, когда при переходе к данным, полученным на новом оборудовании, минорный класс оказывается представлен еще скуднее.

Дисбаланс классов традиционно смягчают на уровне выборки и на уровне функции потерь. К первой группе относятся методы балансировки самой выборки, такие как прореживание доминирующего класса (undersampling), дублирование редких образцов или их синтез методом интерполяции (алгоритм SMOTE и его аналоги). Ко второй группе относятся изменения на уровне алгоритма: введение различных штрафов за ошибки классификации редких объектов и использование взвешенных функций потерь. Примером таких функций служит фокальная потеря (focal loss), которая искусственно снижает влияние простых, уверенно распознаваемых примеров. Перечисленные приемы балансируют влияние классов, однако обладают тем же фундаментальным ограничением, что и колориметрические аугментации, — они оперируют имеющимися образцами и не создают морфологически новых, пространственно достоверных конфигураций минорных объектов. Интерполяция в признаковом пространстве для изображений со сложной топологией к тому же способна порождать артефактные, неправдоподобные образцы.

Прикладным воплощением проблемы минорного класса в настоящей работе выступает целевая (downstream) задача выявления лимфоваскулярной инвазии (ЛВИ) на WSI. Задача относится к области цифровой патологии, однако с

инженерной точки зрения сводится к выявлению редкого, морфологически вариативного и пространственно локального целевого класса в условиях сильного дисбаланса. Она избрана в качестве экспериментальной базы именно потому, что концентрирует оба рассмотренных ограничения — ковариантный сдвиг между условиями получения данных и экстремальный дефицит целевого класса.

Принципиальная ограниченность классических подходов раскрывается в сочетании этих факторов. Колориметрические и геометрические (аффинные) преобразования, как и методы перебалансировки, математически отображают значения существующих пикселей или перераспределяют веса примеров, но не порождают новых валидных топологических конфигураций объектов. С точки зрения распределения данных классические аугментации перераспределяют вероятностную массу в пределах носителя исходного распределения $supp(P_s)$, тогда как генеративный синтез способен расширять сам носитель, порождая образцы в ранее не покрытых областях признакового пространства. В терминах формулы (1.2) интерполяционные методы способны лишь частично сбалансировать $P_s(x)$, не восполняя семантический дефицит выборки. Преодоление этого ограничения требует перехода от интерполяционных аугментаций к генеративному синтезу — направленному порождению искусственных, но достоверных изображений, расширяющих обучающую выборку в недостаточно представленных областях.

Идея восполнения дефицита данных средствами генеративных моделей не нова. Еще до распространения диффузионных архитектур аугментацию выборок выполняли генеративно-состязательными сетями, и ряд работ подтвердил положительный эффект синтетических изображений на качество прикладных классификаторов [4]. Практическое применение генеративно-состязательных сетей, однако, сдерживалось нестабильностью обучения, склонностью к коллапсу мод и ограниченным контролем над морфологией синтезируемых структур. Эти ограничения мотивировали переход к более устойчивому классу генеративных моделей и определили направление, развиваемое в настоящей работе.

Сформулируем требования, которым должен удовлетворять метод синтеза, пригодный для рассматриваемой предметной области:

- Морфологическая достоверность. Сохранение корректной геометрии и текстуры объектов, выступающих обучающим сигналом.
- Управляемость. Точное задание характеристик целевого объекта без использования трудоемкой текстовой разметки.
- Адаптивность к домену. Воспроизведение визуального стиля конкретной выборки для компенсации ковариантного сдвига.
- Бесшовная интеграция. Встраивание синтезированных объектов в окружающий контекст без граничных артефактов.

Перечисленные критерии используются далее как основа сравнительного анализа генеративных архитектур, выполненного в следующем разделе.

1.2. Анализ генеративных архитектур и методов доменной адаптации

Установленные в разделе 1.1 требования к методу синтеза: морфологическая достоверность, управляемость, адаптируемость к доменному стилю и бесшовность интеграции, служат критериальной основой для выбора генеративной архитектуры. В подразделе 1.2.1 проводится сравнительный анализ трех доминирующих семейств генеративных моделей и обосновывается выбор латентных диффузионных моделей. Подраздел 1.2.2 рассматривает фундаментальные модели как механизм извлечения признаков и реализации визуального обусловливания, на основе которого строится генеративный конвейер настоящей работы.

1.2.1. Сравнительный анализ GAN, VAE и диффузионных моделей

Задача генеративного моделирования состоит в восстановлении неизвестного распределения данных $p_{data}(x)$ по конечной выборке и последующем порождении новых образцов из него. Три семейства моделей: генеративно-состязательные сети, вариационные автокодировщики и

диффузионные модели, решают эту задачу принципиально разными способами, что и определяет их сравнительные достоинства и ограничения.

Генеративно-состязательные сети. Одним из наиболее известных семейств глубоких моделей, обеспечивших качественный прорыв в синтезе реалистичных изображений, стали генеративно-состязательные сети (Generative Adversarial Networks, GAN) [4]. Фундаментальная концепция данной архитектуры опирается на динамическое противоборство двух независимых нейросетевых компонентов. В рамках данной архитектуры генератор G преобразует вектор шума z из априорного распределения p_z в синтетический образец $G(z)$. В свою очередь, дискриминатор D оценивает вероятность того, что предъявленный образец принадлежит к истинному набору данных, а не является результатом генерации. Математически этот состязательный процесс формализуется в виде следующей оптимизационной задачи:

$$min_G max_D \mathbb{E}_{x\sim p_{data}}[logD(x)] + E_{z\sim p_z}[log(1 - D(G(z)))] . \quad (1.3)$$

Состязательная постановка обеспечивает высокую визуальную четкость генерируемых изображений и быстрый (одношаговый) инференс. Практическое применение GAN, однако, сопряжено с рядом известных трудностей: неустойчивостью обучения, чувствительностью к гиперпараметрам, отсутствием явной функции правдоподобия и, в особенности, феноменом коллапса мод, при котором генератор воспроизводит лишь узкое подмножество наблюдаемого разнообразия. Развитие GAN шло по двум направлениям. Архитектурное совершенствование привело от полносверточных генераторов (DCGAN) к моделям высокого разрешения с прогрессивным наращиванием и стилевым управлением (StyleGAN [30]); параллельно уточнялись целевые функции — условная генерация (conditional GAN) для управления синтезом и формулировки с расстоянием Вассерштейна (WGAN) для стабилизации обучения. Эти усовершенствования частично смягчают перечисленные проблемы, но не устраняют их полностью. Для задач, где принципиально важно полное покрытие

вариативности редкого класса, перечисленные ограничения оказываются критичными, что подтверждается и ограниченной эффективностью аугментации на основе GAN в прикладных областях, включая анализ гистологических изображений.

Вариационные автокодировщики. Важным этапом в развитии методов направленного синтеза стали вариационные автокодировщики (Variational Autoencoders, VAE) [5]. Они предлагают строгое вероятностное решение задачи генерации за счет выучивания информативного скрытого пространства. Модель VAE относится к классу латентных вероятностных моделей и состоит из кодировщика, аппроксимирующего апостериорное распределение $q_{\phi}(z \mid x)$ скрытых переменных, и декодировщика $p_{\theta}(x \mid z)$ восстанавливающего наблюдение по латентному коду. Обучение максимизирует нижнюю оценку логарифма правдоподобия (Evidence Lower Bound, ELBO):

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q_{\phi}}(z \mid x)\left[log\, p_{\theta}(x \mid z)\right] - D_{KL}(q_{\phi}(z \mid x) \,||\, p(z)), \quad (1.4)$$

где ϕ и θ — обучаемые параметры (весовые коэффициенты) нейронных сетей кодировщика и декодировщика соответственно. В формуле (1.4) первое слагаемое отвечает за точность реконструкции исходного наблюдения, а второе, дивергенция Кульбака-Лейблера, регуляризует латентное пространство, притягивая апостериорное распределение к априорному $p(z)$. Архитектура VAE отличается устойчивостью процесса обучения и формирует структурированное непрерывное латентное пространство, удобное для последующих манипуляций. Сквозное обучение такой модели опирается на прием репараметризации (reparameterization trick), обеспечивающий обратное распространение градиента через стохастический слой. Однако платой за вероятностную регуляризацию выступает характерная размытость реконструкций: гауссовы допущения приводят к усреднению значений пикселей и потере высокочастотных деталей.

Дискретные модификации (например, VQ-VAE) частично преодолевают это ограничение за счет квантования представлений. В свою очередь, латентные диффузионные модели (LDM) [7] используют VAE в качестве компонента перцептивного сжатия. Для этой цели применяется усиленная версия классического автокодировщика, обучаемая с добавлением перцептивной и состязательной (на уровне фрагментов) функций потерь. Такая модель сжимает изображение с высоким коэффициентом при сохранении воспринимаемых деталей, перенося основную нагрузку по моделированию распределения на диффузионную часть сети.

Диффузионные модели. Закономерным этапом эволюции генеративных архитектур, обеспечившим беспрецедентное качество и стабильность синтеза, стало появление диффузионных моделей. Их вычислительная парадигма кардинально отличается от состязательного или вариационного обучения. Вероятностные диффузионные модели (Denoising Diffusion Probabilistic Models, DDPM) [6] порождают данные, обращая постепенный процесс зашумления. Прямой процесс представляет собой марковскую цепь, на каждом шаге которой к образцу добавляется гауссов шум по заданному расписанию дисперсий $\{\beta_t\}_{t=1}^{T}$:

$$q(x_t \mid x_{t-1}) = N(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t \mathrm{I}). \tag{1.5}$$

Свойства гауссова распределения позволяют выразить состояние на произвольном шаге t в замкнутой форме непосредственно через исходный образец x_0:

$$q(x_t \mid x_0) = N(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t)\mathrm{I}), \text{ где } \bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i). \tag{1.6}$$

Генерация соответствует обратному процессу: нейросеть $\epsilon_\theta(x_t, t)$ обучается предсказывать добавленный шум, а оптимизируется упрощенная среднеквадратичная функция потерь:

$$\mathcal{L}_{simple} = \mathbb{E}_{x_0,\,\epsilon,\,t}\left[\left|\left|\epsilon - \epsilon_\theta(x_t,\,t)\right|\right|^2\right], \quad (1.7)$$

где $\epsilon \sim N(0,\,\mathrm{I})$ — целевой шум, а шаг t равномерно выбирается из множества $\{1, \ldots, T\}$. Декомпозиция сложного преобразования на множество элементарных шагов шумоподавления (denoising) обеспечивает диффузионным моделям устойчивое обучение, полное покрытие мод распределения и высокое разнообразие образцов.

Принципиальным недостатком классических диффузионных моделей выступает вычислительная стоимость этапа применения (инференса): порождение одного образца требует последовательного выполнения десятков и сотен шагов шумоподавления. Ускоренные схемы семплирования (в частности, детерминированный алгоритм DDIM) сокращают число шагов на порядок, однако не отменяют многошаговую природу процесса, из-за чего вычислительные затраты на генерацию остаются существенно выше, чем у одношаговых аналогов.

С теоретической точки зрения диффузионные модели допускают непрерывную трактовку: прямой и обратный процессы описываются стохастическими дифференциальными уравнениями, а обучаемая сеть аппроксимирует градиент логарифма плотности зашумленного распределения (score function). Единая формулировка связывает модели шумоподавления с методами сопоставления функций вклада (score matching) и объясняет существование детерминированных траекторий генерации, лежащих в основе ускоренных алгоритмов сэмплирования.

Латентные диффузионные модели. Вычислительная стоимость диффузии в пространстве пикселей послужила мотивацией к переносу диффузионного процесса в сжатое латентное пространство — концепцию латентной диффузионной модели (Latent Diffusion Model, LDM) [7]. LDM объединяет сильные стороны двух предыдущих семейств: предобученный автокодировщик (как правило, на основе VAE) выполняет перцептивное сжатие изображения в компактное латентное представление, а диффузионный процесс разворачивается

уже в этом пространстве пониженной размерности. Разделение на этап перцептивного сжатия и этап латентной диффузии радикально снижает вычислительные затраты при сохранении качества, что и сделало LDM доминирующей архитектурой современного генеративного моделирования изображений высокого разрешения. Подчеркнем, что LDM является обобщенной концепцией и не фиксирует конкретную реализацию сети шумоподавления.

В исходных реализациях диффузионных и латентных диффузионных моделей роль сети шумоподавления выполняла сверточная архитектура U-Net с присущим ей локальным индуктивным смещением. Последующее развитие выявило преимущество базовых трансформерных моделей: диффузионный трансформер (Diffusion Transformer, DiT) [8] заменяет сверточный остов механизмом самовнимания (self-attention), представляя латентное изображение как последовательность токенов-патчей и обрабатывая ее слоями трансформера. DiT демонстрирует лучшую масштабируемость по числу параметров и объему данных, а его пропускная способность ограничивается квадратичной по длине последовательности сложностью механизма внимания — обстоятельство, существенное для изображений высокого разрешения и подробно рассматриваемое в главах 2 и 3. В терминологии данной работы DiT трактуется как частный случай LDM, в котором этап шумоподавления реализуется трансформерной, а не сверточной сетью; это разграничение принципиально и выдерживается на протяжении всего изложения.

Принципиально, что в концепции LDM диффузионный трансформер оперирует не пикселями, а латентным представлением, формируемым автокодировщиком. Латентный тензор разбивается на патчи, линейно проецируемые в токены, над последовательностью которых и работают слои self-attention. Тем самым DiT и VAE образуют две сопряженные части единой модели: автокодировщик задает пространство, в котором действует трансформер, а его коэффициент сжатия напрямую определяет длину обрабатываемой последовательности токенов и, следовательно, вычислительную стоимость внимания. Например, при сжатии изображения 1024×1024 в латент 128×128 и

единичном размере патча трансформер обрабатывает последовательность из 16384 токенов; ее рост при повышении разрешения и обусловливает остроту проблемы квадратичной сложности слоев self-attention. Обусловливание по временному шагу внедряется в DiT через адаптивную нормализацию слоев (в варианте AdaLN-Zero), а масштабируемость архитектуры выражается монотонным улучшением качества генерации с увеличением вычислительного бюджета модели. Сводное сопоставление трех семейств по ключевым для настоящей работы критериям приведено в таблице 1.

Таблица 1 — Сравнительная характеристика генеративных архитектур

Критерий	**GAN**	**VAE**	**Диффузионные модели**
Устойчивость обучения	низкая	высокая	высокая
Качество (четкость) образцов	высокое	среднее (размытость)	высокое
Покрытие мод / разнообразие	ограниченное (коллапс мод)	полное	полное
Скорость инференса	высокая (один шаг)	высокая (один шаг)	низкая (многошаговая)
Явная оценка правдоподобия	отсутствует	да (ELBO)	да
Управляемость генерации	ограниченная	умеренная	высокая

Сопоставление с требованиями подраздела 1.1.2 однозначно указывает на диффузионные модели. Морфологическая достоверность обеспечивается высоким качеством образцов и полным покрытием мод распределения; критическое для задачи разнообразие минорного класса недостижимо в рамках GAN из-за коллапса мод и ограничено для VAE из-за размытости реконструкций. Единственный значимый недостаток диффузии, стоимость инференса, носит сугубо вычислительный характер и устраняется как переходом в латентное пространство (LDM), так и методами архитектурной оптимизации, разрабатываемыми в

настоящей работе. Высокая управляемость диффузионных моделей, наконец, реализуется через механизмы обусловливания, отвечающие требованию точного задания характеристик целевого объекта.

Эволюция методов синтеза в предметной области. Рассмотренная смена архитектурных парадигм полностью отражает фактическую историю развития методов генерации изображений в области цифровой патологии. Первое поколение моделей для аугментации данных в этой сфере опиралось на генеративно-состязательные сети (GAN). Хотя архитектуры семейства StyleGAN [30] существенно повысили качество и контролируемость синтеза текстур, они унаследовали фундаментальные недостатки состязательного обучения: вычислительную нестабильность и неполноту покрытия вероятностного распределения. Для цифровой патологии это оказалось особенно критичным, так как редкие морфологические конфигурации при генерации часто терялись. Именно эти ограничения обусловили смещение фокуса у исследователей к латентным диффузионным моделям (LDM), как более устойчивому и выразительному классу алгоритмов. Таким образом, выбор диффузионной парадигмы в настоящем исследовании продиктован не только ее теоретическим превосходством, но и объективной траекторией развития методов вычислительного анализа гигапиксельных изображений WSI.

Механизмы внедрения управляющего сигнала (обусловливания). Контроль над процессом генерации в диффузионных моделях достигается за счет интеграции дополнительного сигнала c на этапе восстановления данных из шума. При этом архитектура сети шумоподавления принимает условный вид $\epsilon_\theta(x_t, t, c)$. Исторически закрепились два основных механизма управления траекторией генерации:

- *Управление с использованием классификатора* (classifier guidance): корректирует процесс шумоподавления с помощью градиентов от независимо обученного классификатора.
- *Бесклассификаторное управление* (classifier-free guidance): основано на совместном обучении единой генеративной сети как в условном, так и в

безусловном режимах. Этот подход исключает необходимость использования внешней модели, что упрощает вычислительный конвейер и повышает степень контроля над синтезом.

В трансформерных архитектурах сигнал с внедряется преимущественно посредством механизма перекрестного внимания (cross-attention), где токены матрицы-условия выступают в роли ключей (K) и значений (V). Природа этого сигнала, текстовая или визуальная, определяет границы применимости алгоритма к конкретной прикладной задаче, что подробно рассматривается в следующем подразделе.

1.2.2. Фундаментальные модели как механизм извлечения признаков и визуального обусловливания

Выбор диффузионной архитектуры, обоснованный в подразделе 1.2.1, оставляет открытым вопрос о природе управляющего сигнала. Именно сигнал c, направляющий процесс восстановления данных из шума, определяет, насколько точно и каким способом задаются характеристики синтезируемого объекта. В данном подразделе рассматриваются фундаментальные модели как источник такого сигнала и обосновывается переход от текстовых условий к визуальным.

Фундаментальные модели и самообучение. Под фундаментальной моделью (foundation model) понимают масштабную нейросетевую архитектуру, предварительно обученную на огромных массивах неразмеченных данных. Она позволяет формировать универсальные переносимые представления, пригодные для широкого спектра прикладных задач. Обучение таких моделей опирается на парадигму самообучения (self-supervised learning, SSL): полезный сигнал извлекается из самой структуры данных без привлечения разметки. Исторически сложились три основных семейства SSL-методов:

- контрастивное обучение сближает представления различных аугментаций одного изображения и отдаляет их от представлений других образцов [1];
- самодистилляция без меток (DINO, DINOv2 [25]) согласует выходы моделей “ученика” и “учителя” без использования негативных примеров;

- маскированное моделирование (MAE [3]) обучается восстанавливать преднамеренно скрытые фрагменты входных данных.

Все три подхода позволяют получить кодировщики, преобразующие входное изображение в компактный вектор признаков (эмбеддинг) с богатым семантическим содержанием. Многочисленные исследования подтверждают, что такие эмбеддинги кодируют информацию, достаточную для решения дискриминативных задач классификации и поиска без дополнительного обучения модели либо с его минимальным объемом.

Доминирующая в компьютерном зрении общего назначения парадигма “текст-изображение” (text-to-image) опирается на пары “изображение — текстовое описание”, в изобилии доступные в открытых веб-источниках. Однако для гигапиксельных изображений специализированных предметных областей этот ресурс либо полностью отсутствует, либо ограничен грубой разметкой. Формирование детальной текстовой разметки на уровне фрагментов требует колоссальных временных затрат. Так, по опубликованным оценкам [10], аннотирование лишь одной коллекции WSI (TCGA-BRCA) потребовало бы около 40000 часов работы профильного эксперта. Более того, естественный язык принципиально ограничен в способности однозначно фиксировать сложную морфологию и взаимное пространственное расположение микроструктур. Словесное описание оставляет значительную свободу интерпретации, что в задачах с жесткими требованиями к достоверности приводит к неконтролируемой вариативности генерации. Таким образом, использование текстовых описаний оказывается малопригодным или трудно применимым для рассматриваемого класса данных.

Существует альтернативный подход, который позволяет использовать эмбеддинги фундаментальных моделей в качестве управляющего сигнала для диффузионного синтеза. Ключевая гипотеза состоит в том, что SSL-представления достаточно выразительны, чтобы служить в качестве замены детальной ручной разметки. В этом случае обученный кодировщик выступает в роли автоматического “аннотатора”, поставляющего плотный попиксельный сигнал

условия на высоком разрешении [10]. Диффузионная модель обучается обращать это отображение — генерировать изображение, согласованное с заданным эмбеддингом (проецировать вектор признаков обратно в пространство пикселей). Данный метод обладает двумя принципиальными достоинствами:

- процесс генерации полностью детерминирован заданным условием и не требует ручного труда, поскольку эмбеддинги извлекаются из референсных образцов автоматически;
- подход инвариантен к источнику признаков. Целевой вектор может быть извлечен как из реального изображения-референса, так и сформирован вспомогательной моделью из данных другой модальности.

Эффективность механизма подтверждена не только в цифровой патологии, но и в задачах дистанционного зондирования Земли (спутниковые снимки) [10], что свидетельствует о его универсальности при работе с гигапиксельными изображениями вне зависимости от конкретной предметной области.

Синтез изображений произвольного размера. Визуальное обусловливание открывает путь к генерации изображений, превосходящих по размеру обучающие фрагменты. Крупное изображение представляется как пространственная решетка SSL-эмбеддингов, каждый из которых кодирует окрестность соответствующего участка. Целевой результат синтезируется как набор согласованных фрагментов, в котором локальные свойства задаются поэлементным обусловливанием, а глобальная структура — взаимным расположением эмбеддингов в решетке. Изменение пространственной компоновки этих векторов равносильно редактированию семантической организации всего полотна. Подход позволяет порождать изображения практически произвольного масштаба без существенного роста вычислительных затрат (относительно базовой модели) и сохраняет при этом пространственные зависимости на больших расстояниях, что является принципиальным условием при работе с гигапиксельными изображениями.

Для гигапиксельных изображений цифровой патологии сформировалось семейство специализированных фундаментальных моделей, различающихся парадигмой предварительного обучения и уровнем агрегации признаков.

Архитектура UNI [9] представляет собой трансформер (ViT), обученный методом самодистилляции DINOv2 на крупном корпусе фрагментов изображений цифровой патологии; ее расширенная версия UNI2-h позволяет формировать векторы с богатой морфологической семантикой на уровне отдельного тайла. Близкую нишу занимает Virchow-2 [32] — кодировщик большой емкости, обученный на масштабной коллекции слайдов и обеспечивающий высокую дискриминативность тайловых признаков. Иную парадигму реализует CONCH [33]: мультимодальная архитектура “зрение-язык”, обученная контрастивным методом на парах данных (“изображение-текст”), что открывает дополнительные возможности для текстовых запросов. Наконец, модель GigaPath [34] работает на уровне всего слайда, объединяя тайловый кодировщик с агрегатором, моделирующим контекст в пределах всего WSI.

В рамках архитектуры настоящей работы перечисленные решения играют две различные роли. Тайловая модель UNI2-h используется как источник эмбеддингов, управляющих генерацией; кодировщики того же или иного семейства (в частности, Virchow-2) применяются в качестве экстракторов признаков, формирующих пространство для объективной оценки качества синтезированных изображений. Важно отметить, что кодировщик выступает в роли заменяемого программного компонента. Его доменная специализация не нарушает общую вычислительную схему: при переносе алгоритма на другой класс гигапиксельных данных он легко заменяется соответствующим универсальным экстрактором, таким как DINOv2 [25].

Линия архитектур PixArt. Непосредственным предшественником принятой в данном исследовании базовой генеративной модели выступает семейство диффузионных трансформеров PixArt. Модель PixArt-α [28] продемонстрировала, что DiT-архитектуры класса “текст-изображение” могут обучаться при кардинально сниженных вычислительных затратах благодаря декомпозиции процесса оптимизации и эффективному отбору данных. Ее развитие, модель PixArt-Σ [29], реализует стратегию прогрессивного обучения “от слабого к сильному” (weak-to-strong training). Она позволяет последовательно повышать

качество генерации за счет использования лучших данных и двух ключевых усовершенствований: уточненных текстовых описаний и эффективного механизма сжатия токенов. Именно в PixArt-Σ был предложен механизм сжатия ключей и значений (KV-компрессия) — специализированный модуль внимания, снижающий вычислительную сложность слоев самовнимания с $O(N^2)$ до $O(N^2/R^2)$, где R — коэффициент сжатия. Это новшество сделало осуществимой генерацию изображений разрешения (вплоть до 4K) при компактном размере модели (порядка 0,6 млрд параметров). Данный механизм заложен в основу архитектурной оптимизации, разрабатываемой в настоящей работе, и подробно формализуется в подразделе 2.1.2.

Генеративная фундаментальная модель PixCell. Перечисленные принципы: диффузионный трансформер семейства PixArt, прогрессивное обучение и визуальное управление с помощью SSL-эмбеддингов, объединены в модели PixCell [24]. Модель PixCell адаптирует архитектуру и подходы к обучению семейства PixArt [28, 29] для работы с WSI. Фактически она позиционируется как первая генеративная фундаментальная модель в этой предметной области и развивает концепцию диффузии, управляемой представлениями (representation-guided diffusion) [10]. Перцептивное сжатие в PixCell выполняет предобученный автокодировщик SD-3 VAE [26], отображающий фрагмент 1024×1024 пикселей в скрытое (латентное) представление размерностью $128\times128\times16$. Управляющее условие задается эмбеддингами UNI2-h (один вектор размерностью 1×1536 на изображение 256×256). Обучение проводилось на наборе PanCan-30M (около 30,8 млн фрагментов 1024×1024 из 69184 WSI) с применением трехэтапной прогрессивной схемы обучения с последовательным наращиванием разрешения от 256×256 до 1024×1024. Практическое применение синтезированных PixCell изображений для аугментации повышает качество прикладных диагностических моделей в среднем на 3% на различных наборах данных [24], что подтверждает состоятельность подхода и обоснованность его выбора в качестве отправной точки. Быстрое признание PixCell исследовательским сообществом укрепляет ее статус как базового эталона:

последующие генеративные архитектуры для гистологии (в частности, представленная в 2026 году модель CytoSyn [31]) разрабатываются и оцениваются в прямом сопоставлении с ней. Эволюция семейства PixArt и процесс его адаптации в PixCell обобщены в таблице 2.

Таблица 2 — Эволюция архитектур семейства PixArt и их адаптация в PixCell

Модель	**Магистраль**	**Разрешение**	**Ключевое нововведение**	**Домен / обусловливание**
PixArt-α [28]	латентный DiT	до 1024×1024	эффективное обучение DiT, декомпозиция процесса	естественные изображения / текст
PixArt-Σ [29]	латентный DiT	до 4096×4096 (4K)	KV-компрессия, обучение “от слабого к сильному”	естественные изображения / текст
PixCell [24]	латентный DiT (линия PixArt)	до 1024×1024 (выше с дообучением)	адаптация к гистологии, SD-3 VAE, прогрессивное обучение	гигапиксельные изображения / визуальное (UNI2-h)

PixCell наследует генеративное ядро линии PixArt. При этом механизм текстового управления заменяется на использование визуальных эмбеддингов UNI2-h, что позволяет адаптировать архитектуру к домену гигапиксельных данных. Важно отметить, что алгоритм сжатия ключей и значений (KV-компрессия), предложенный в PixArt-Σ, сохраняется в качестве эффективного средства архитектурной оптимизации, которое активно задействуется и в настоящей работе.

Помимо PixCell, в последние годы сформировалась целая линия генеративных моделей гистологических изображений. Их развитие происходит преимущественно вокруг концепции LDM, что подтверждает общую зрелость данного направления. Заметный вклад в эту область внесла лаборатория

компьютерного зрения Университета Стоуни-Брук (CVLab @ Stony Brook). В частности, в их исследовании по сжатию изображений цифровой патологии с помощью предварительно обученных автокодировщиков [68] показано, что формируемые скрытые (латентные) представления отлично подходят для компактного кодирования гистологических данных. Они образуют оптимальное пространство для последующего генеративного моделирования, что напрямую согласуется со схемой латентного синтеза, принятой в данной работе. Например, модель PathLDM [70] реализовала латентную диффузию с текстовым управлением для цифровой патологии. Она продемонстрировала принципиальную возможность управляемой генерации, но в то же время на практике подтвердила описанные ранее ограничения текстовых условий для данного домена. Модель ZoomLDM [69] решает специфичную для гигапиксельных данных задачу многомасштабной генерации: единая диффузионная модель порождает согласованные изображения на разных уровнях увеличения, что строго соответствует пирамидальной структуре WSI. Синтез крупных изображений совместно с их попиксельными масками проработан в архитектуре DiffInfinite [71], где итоговое полотно формируется за счет параллельного и согласованного диффузионного процесса на множестве фрагментов. Перечисленные работы очерчивают актуальный научный контекст исследования, в рамках которого LDM надежно закрепились в качестве доминирующей архитектуры генерации гистологических снимков.

На фоне описанного исследовательского ландшафта новизна и актуальность настоящей работы определяются сочетанием трех фокусных направлений, которые лишь частично затрагивались в предшествующих публикациях:

- использование визуального сигнала в качестве основного механизма управления генерацией;
- целенаправленное восполнение данных именно миноритарного класса посредством локального контекстно-ориентированного синтеза;
- систематическая аппаратно-программная оптимизация вычислительного конвейера.

Доменная адаптация генеративных моделей. Базовая генеративная модель, обученная на обширном многодоменном корпусе, воспроизводит усредненный стиль и не учитывает специфику конкретной целевой выборки, то есть не компенсирует ковариантный сдвиг, рассмотренный в подразделе 1.1.2. Адаптация модели к целевому домену достигается путем ее дообучения. Полное дообучение всех параметров вычислительно затратно и сопряжено с риском катастрофического забывания, поэтому в современной практике преобладают методы параметрически-эффективной адаптации (parameter-efficient fine-tuning). Среди них выделяется алгоритм низкоранговой адаптации (Low-Rank Adaptation, LoRA [27]), при котором обновление весовой матрицы аппроксимируется произведением двух матриц малого ранга, что сокращает число обучаемых параметров на несколько порядков при сохранении качества. Применение LoRA к генеративным моделям позволяет кодировать стиль отдельного домена компактным адаптером, не затрагивая веса базовой модели. Тем самым доменная специализация становится масштабируемой и обратимой. Данный принцип положен в основу метода адаптации, разрабатываемого в главе 2.

Оценка качества генерации. Объективная оценка генеративных моделей опирается на сопоставление распределений реальных и синтезированных изображений в пространстве признаков. Стандартной мерой служит расстояние Фреше (Fréchet Distance) между параметрами гауссовых аппроксимаций двух наборов эмбеддингов; его классический вариант (FID) использует признаки сети Inception, тогда как для специализированных доменов корректнее применять доменно-ориентированные экстракторы. Применительно к гистологическим изображениям расстояние Фреше вычисляют с использованием как универсальных, так и профильных кодировщиков (например, Virchow-2), что позволяет оценивать достоверность синтеза одновременно в общем и в доменно-специфичном контексте. Система метрик для оценки качества генерации и полезности синтезированных данных в прикладных задачах определяется в главе 4.

Рассмотренные компоненты сводятся в единую функциональную схему латентной диффузионной модели, приведенную на рисунке 1.2.

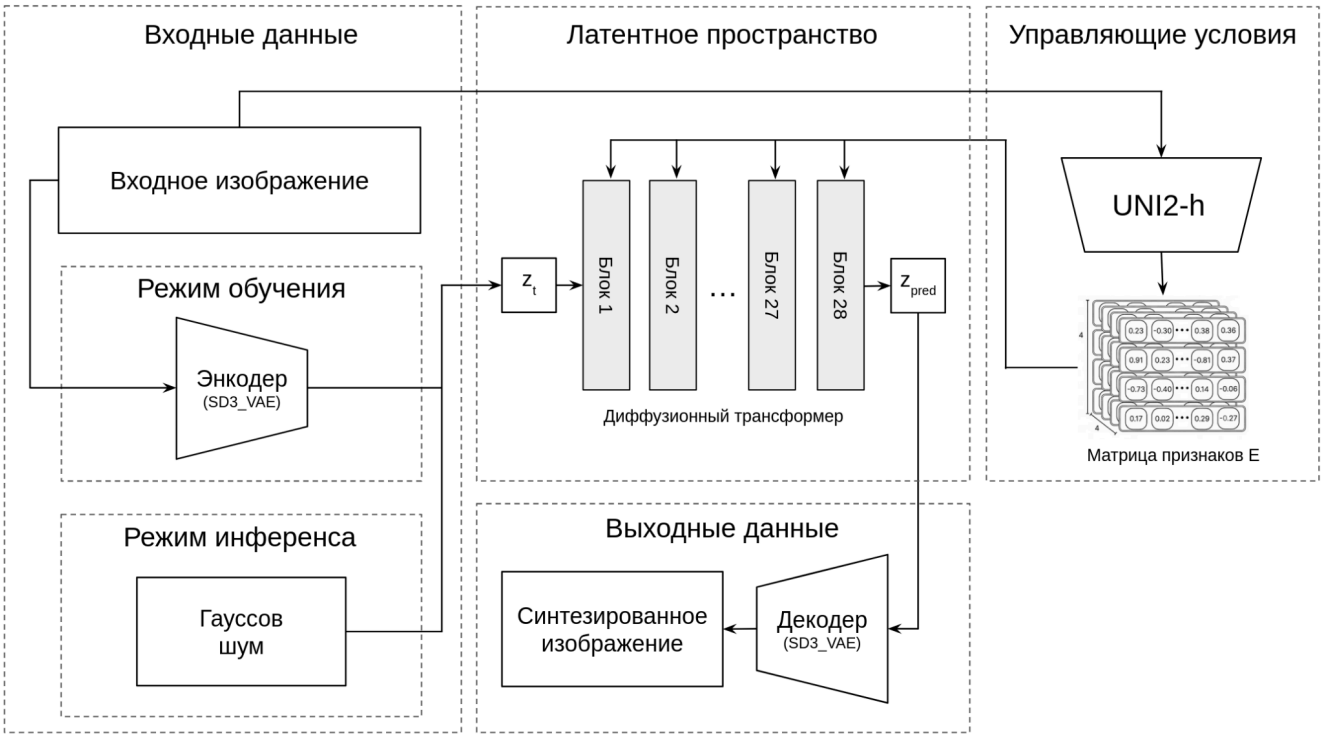


Рисунок 1.2 — Функциональная схема латентной диффузионной модели, демонстрирующая взаимодействие базовых компонентов (автоэнкодера и диффузионного трансформера) с механизмом визуального обусловливания на основе матрицы признаков UNI2-h в режимах обучения и инференса

Вычислительный конвейер образован тремя функциональными блоками. Перцептивное сжатие реализует автокодировщик SD3-VAE [26], отображающий изображение из пространства пикселей в латентное представление пониженной размерности и обратно. Генеративное ядро, диффузионный трансформер, выполняет обратный диффузионный процесс непосредственно в латентном пространстве, что и обеспечивает вычислительную эффективность архитектуры LDM. Контроль над генерацией осуществляет модуль визуального управления: эмбеддинг, извлеченный кодировщиком UNI2-h из референсного изображения, направляет процесс шумоподавления к семантически согласованному результату. Внедрение управляющих сигналов в DiT происходит двумя путями: временной шаг t подается через механизм адаптивной нормализации слоев (AdaLN), а условный эмбеддинг – через слои cross-attention, где токены условия служат в

качестве ключей и значений. Подробная математическая формализация латентного перехода, обусловливания и обратного процесса вынесена в главу 2.

Выбранная конфигурация — латентная диффузионная модель с трансформерной сетью шумоподавления, перцептивным автокодировщиком SD3-VAE и механизмом визуального управления по эмбеддингам UNI2-h, реализованная на базе архитектуры PixCell, удовлетворяет всем требованиям, сформулированным в подразделе 1.1.2, и принимается за основу разрабатываемого метода.

1.3. Методы управления генерацией и оценка вычислительной сложности диффузионных моделей

Завершая обзор методов генеративного моделирования, рассмотрим механизмы управления процессом генерации и вопросы вычислительной сложности диффузионных моделей. Эти два аспекта определяют выбор конкретных алгоритмических и оптимизационных решений, реализуемых в последующих главах.

Механизмы управления генерацией. Помимо базового обусловливания через перекрестное внимание (подраздел 1.2.1), для тонкого управления диффузионными моделями разработаны специализированные архитектурные надстройки. Метод ControlNet [35] добавляет к предобученной модели обучаемую копию ее блоков, подключаемую через “нулевые свертки” (zero convolutions). Эта ветвь принимает пространственную карту-условие (контуры, карту глубины или маску сегментации), что обеспечивает контроль над компоновкой генерируемого изображения. Поскольку исходный ControlNet рассчитан на сверточные U-Net-модели, для трансформерных сетей была предложена его адаптация — ControlNet-Transformer [36]. Иной принцип реализует IP-Adapter [37] (image prompt adapter), который внедряет признаки референсного изображения через отдельную ветвь перекрестного внимания, позволяя задавать содержание и стиль с помощью изображения-примера. Обе группы методов представляют собой

мощный инструмент управляемой генерации изображения целиком по заданному условию.

Локальное встраивание объектов. Задача обогащения выборки объектами миноритарного класса, решаемая в настоящей работе, обладает иной структурой, нежели генерация изображения целиком. Требуется интегрировать заранее заданный целевой объект в указанную область уже существующего фонового фрагмента, сохранив остальную часть фона неизменной и обеспечив согласованность границ. Подобная локальная замена области по своей сути является задачей дорисовки (inpainting) или контекстного встраивания: синтезируется только содержимое заданной маски, тогда как окружающий контекст выступает фиксированным условием. Применение к этой задаче механизмов управляемой генерации целого изображения сопряжено с тремя принципиальными ограничениями. Во-первых, алгоритмы ControlNet и IP-Adapter порождают изображение заново и не гарантируют попиксельного сохранения исходного фона, который для целей аугментации должен оставаться валидным реальным образцом. Во-вторых, оба механизма требуют обучения дополнительного модуля на парных данных (пространственных картах-условиях для ControlNet или согласованных парах изображений для IP-Adapter). Для редкого класса такие данные отсутствуют, что возвращает к исходной проблеме дефицита разметки. В-третьих, базовая модель уже контролируется визуальными эмбеддингами; добавление IP-Adapter дублировало бы имеющийся механизм управления, а ControlNet потребовал бы создания детальных пространственных аннотаций, то есть того самого ручного экспертного труда, которого данная работа стремится избежать. Задача inpainting, напротив, реализуется непосредственно поверх существующей модели через маскированное шумоподавление в латентном пространстве и дополняется автоматическим колориметрическим согласованием объекта с фоном. Такое решение не вводит дополнительных обучаемых модулей и не требует труднодоступных пар данных для обучения. Сказанное не умаляет достоинств ControlNet и IP-Adapter как универсальных средств направленной генерации. Напротив, речь идет о соответствии метода строгим требованиям

конкретной прикладной задачи: попиксельному сохранению фона, отсутствию дополнительной разметки и бесшовности интеграции. Формальное описание алгоритма локального встраивания приведено в подразделе 2.3.

Ускорение сэмплирования. Многошаговая природа обратного диффузионного процесса определяет высокую вычислительную стоимость инференса (подраздел 1.2.1), для снижения которой разработан ряд ускоренных численных схем. Детерминированный алгоритм сэмплирования DDIM [38] переходит от стохастической марковской цепи к немарковскому процессу, эквивалентному решению обыкновенного дифференциального уравнения, что сокращает число шагов с порядка тысячи до нескольких десятков без переобучения модели. Алгоритмы семейства DPM-Solver [39] используют аппроксимацию высокого порядка и дополнительно уменьшают число шагов при сохранении качества. Радикальное ускорение обеспечивает дистилляция согласованности: латентные консистентные модели (Latent Consistency Models, LCM [40]) обучаются отображать любую точку траектории обратного процесса непосредственно в ее начало, что позволяет генерировать изображение за 2-4 шага. Именно этот прием применен, например, в модели PixArt-δ [36] для ускорения вывода. В конечном итоге выбор конкретного алгоритма сэмплирования всегда представляет собой компромисс между скоростью генерации и качеством получаемых образцов.

Оценка вычислительной сложности. Анализ временной сложности позволяет формализовать вычислительные затраты диффузионных моделей и обоснованно выбирать оптимизации. Сложность принято выражать в нотации “О-большое” как асимптотику числа элементарных операций в зависимости от размера входа. Доминирующей операцией в диффузионном трансформере выступает механизм self-attention, вычислительная сложность которого квадратична по длине последовательности токенов:

$$C_{attn} = O(N^2 d), \tag{1.8}$$

где N — число токенов (определяемое разрешением латентного представления и размером патча), d — размерность вектора признаков. Полная вычислительная стоимость генерации одного изображения складывается из стоимости работы всех слоев на всех шагах шумоподавления:

$$C_{gen} = O(T\,L\,N^2 d), \tag{1.9}$$

где T — число шагов сэмплирования, L — число трансформерных слоев. Выражение (1.9) указывает на два независимых направления снижения затрат: уменьшение числа шагов T за счет ускоренных алгоритмов сэмплирования и снижение квадратичной зависимости вычислительной сложности от длины последовательности N. Механизм KV-компрессии (раздел 1.2 и подраздел 2.1.2) воздействует на второе направление, понижая сложность внимания до $O(N^2 d/R^2)$, где R — коэффициент сжатия. Это особенно значимо при повышении разрешения, когда число токенов N растет квадратично по отношению к линейному размеру изображения. Реализуемые в работе алгоритмы вариативного синтеза и локального встраивания действуют именно в рамках этой модели сложности: первый снижает затраты за счет KV-компрессии и прогрессивного обучения, второй — за счет ограничения области генерации рамками маски.

При анализе вычислительных затрат важно разделять арифметическую сложность и нагрузку на подсистему памяти. В слое внимания с размером мини-партии B, количеством параллельных потоков вычислений (“голов”) H, длиной последовательности N и размерностью векторного представления признаков d_k(размерность матриц ключей K и значений V в рамках одной “головы” механизма внимания) матрица оценок внимания принимает размерность $B{\times}H{\times}N{\times}N$. Следовательно, даже при умеренном значении d_k основной вклад в пиковое потребление памяти вносят не статические веса модели, а матрицы промежуточных вычислений и их градиенты. Этим обусловлен квадратичный рост требований к памяти графического ускорителя при увеличении разрешения, когда

удвоение пространственного разрешения данных приводит к увеличению числа токенов N в четыре раза, а объем матрицы внимания в шестнадцать раз.

Механизм KV-компрессии разработан с целью нивелировать этот доминирующий фактор вычислительной сложности. При коэффициенте пространственного сжатия R эффективная длина последовательности токенов для тензоров ключей и значений сокращается до N/R^2. Вследствие этого матрица оценок внимания принимает асимметричную размерность $B\times H\times N\times(N/R^2)$. В результате асимптотическая сложность механизма внимания снижается с $O(N^2 d_k)$ до $O((N^2/R^2)\, d_k)$. Поскольку тензоры запросов Q вычисляются в исходном пространственном разрешении, способность модели генерировать высокодетализированные признаки не деградирует. Данный анализ подтверждает, что рассматриваемая оптимизация носит фундаментальный архитектурный характер.

Выводы по первой главе:

- Гигапиксельные изображения образуют специфический класс данных, обработка которого осложнена их иерархической организацией, необходимостью декомпозиции, интенсивной нагрузкой на подсистему ввода-вывода при чтении фрагментов и дороговизной экспертной разметки. Репрезентативность обучающих выборок в этой области систематически нарушается ковариантным сдвигом и экстремальным дисбалансом классов. Классические интерполяционные аугментации, оперирующие значениями существующих пикселей, не способны восполнить этот семантический дефицит.
- Сравнительный анализ генеративных архитектур показал, что строгим требованиям морфологической достоверности, управляемости и полноты покрытия распределения наилучшим образом отвечают латентные диффузионные модели (LDM) с трансформерной архитектурой. При этом LDM трактуется как обобщенная концепция, а диффузионный трансформер

DiT как ее частный случай, совмещенный с автокодировщиком в рамках единого генеративного конвейера.

- Визуальное обусловливание на основе эмбеддингов фундаментальных кодировщиков выступает эффективной детерминированной альтернативой текстовому управлению, применимой к гигапиксельным изображениям из различных предметных областей. В качестве базовой генеративной модели обоснованно выбрана архитектура PixCell (семейство PixArt с механизмом KV-компрессии), а для задачи доменной адаптации — низкоранговые адаптеры (LoRA).
- Анализ методов управления генерацией показал, что для локального обогащения выборки объектами миноритарного класса наиболее адекватным алгоритмическим решением является контекстное достраивание (inpainting), а не использование универсальных механизмов генерации целого изображения. Оценка вычислительной сложности выявила два ключевых направления архитектурной оптимизации: сокращение числа шагов сэмплирования и снижение квадратичной сложности механизма self-attention. Данные направления определяют логику разработки методов, предлагаемых в последующих главах.

Глава 2. Метод адаптивного синтеза изображений высокого разрешения

Настоящая глава раскрывает концептуальное ядро исследования и содержит математическое и алгоритмическое обеспечение разработанного метода адаптивного синтеза сложноструктурированных изображений высокого разрешения. В разделе 2.1 формализуется диффузионный процесс в латентном пространстве. Он опирается на автокодировщик SD3-VAE и механизм визуального управления, реализуемый кодировщиком UNI2-h; здесь же излагается метод оптимизации диффузионного трансформера за счет сжатия матриц ключей и значений. Раздел 2.2 определяет двухэтапный метод доменной специализации латентных диффузионных моделей, а раздел 2.3 описывает алгоритм локального контекстно-ориентированного встраивания объектов. Изложение опирается на принятую в первой главе базовую модель PixCell и развивает ее до алгоритмического конвейера, отвечающего сформулированным требованиям.

2.1 Математическая модель диффузионного процесса в латентном пространстве

Данный раздел формализует генеративный конвейер, концептуальная схема которого была приведена на рисунке 1.2. Сначала определяется переход в латентное пространство и механизм визуального управления (подраздел 2.1.1), а затем метод снижения вычислительной сложности трансформерной архитектуры (подраздел 2.1.2).

2.1.1 Формализация латентного перехода с использованием SD3-VAE и визуального обусловливания UNI2-h

Синтез изображений выполняется в сжатом латентном пространстве, что определяет вычислительную эффективность модели. Перевод исходных изображений в латентное пространство осуществляет модуль перцептивного сжатия SD3-VAE. Изображение $x \in \mathbb{R}^{3\times H\times W}$ отображается в латентное представление:

$$z_0 = E_{VAE}(x), \quad z_0 \in \mathbb{R}^{c \times H/8 \times W/8}, \tag{2.1}$$

где $E_{VAE}(x)$ — кодировщик SD3-VAE, $c = 16$ — число каналов в латентном представлении, пространственный коэффициент сжатия равен 8. Так, например, для изображения 1024×1024 пикселя латентное представление принимает размер $16 \times 128 \times 128$. Обратное отображение задает декодировщик D_{VAE}, так что $x \approx D_{VAE}(z_0)$. Выбор архитектуры SD3-VAE обусловлен ее способностью минимизировать пространственные искажения и потери резкости на границах микроструктурных элементов и контуров, что обеспечивает необходимые условия для последующего бесшовного объединения фрагментов результирующего изображения.

Перенос диффузионного процесса в латентное пространство соответствует двухэтапной декомпозиции генеративной задачи. Автокодировщик отвечает за моделирование высокочастотных перцептивных деталей и сжимает изображение по площади в $8 \times 8 = 64$ раза, тогда как диффузионная модель оперирует семантически насыщенным, но пространственно компактным представлением. Такое разделение позволяет значительно сократить размерность пространства, в котором разворачивается ресурсоемкий марковский процесс, и служит главным источником вычислительной эффективности латентных диффузионных моделей.

Прямой диффузионный процесс определяется в латентном пространстве как марковская цепь зашумления (ср. формулы (1.5) — (1.6)). Состояние на произвольном шаге t выражается в замкнутой форме через исходное представление z_0:

$$z_t = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon, \quad \varepsilon \sim N(0, \mathrm{I}), \tag{2.2}$$

где $t \in \{1, \ldots, T\}$ — номер шага диффузии, $\{\beta_t\}_{t=1}^{T}$ — заданное расписание дисперсий шума, $\bar{\alpha}_t = \prod_{i=1}^{t}(1 - \beta_i)$ — кумулятивный параметр расписания, ε — выборка гауссова шума.

Управление обратным процессом осуществляется вектором визуального условия, извлекаемым фундаментальной моделью UNI2-h. Для изображения x формируется матрица признаков:

$$e = E_{UNI}(x), \quad e \in \mathbb{R}^{16\times1536}, \tag{2.3}$$

где $E_{UNI}(x)$ — кодировщик UNI2-h. Исходное изображение размером 1024×1024 пикселя разбивается на регулярную сетку 4×4 из 16 локальных фрагментов (размером 256×256 каждый). При этом каждая из 16 строк матрицы e представляет собой вектор размерностью 1536, кодирующий семантику соответствующего фрагмента. Благодаря такой структуре условие e задает точное пространственно-распределенное морфологическое описание объекта, исключая семантическую неопределенность текстовых аннотаций.

Пространственная организация матрицы e принципиальна для управления процессом генерации. Строки-условия находятся в строгом соответствии с участками на исходном изображении, благодаря чему модель получает не глобальный, а локализованный в пространстве семантический ориентир. При обработке изображений иного разрешения число строк условия изменяется согласованно: например, при синтезе фрагмента размером 256×256 пикселей размерность матрицы признаков сводится к одной строке 1×1536. Это свойство непосредственно используется в стратегии прогрессивного обучения (подраздел 2.2.1) и в алгоритме контекстного синтеза (подраздел 2.3).

Процесс восстановления данных моделируется параметризованной сетью шумоподавления $\varepsilon_\theta(z_t, t, e)$ — диффузионным трансформером, который предсказывает добавленный шум при заданном условии e. Вектор параметров θ

настраивается путем минимизации взвешенной среднеквадратичной ошибки предсказания шума:

$$\mathcal{L}(\theta) = \mathbb{E}_{z_0,\, e,\, \varepsilon,\, t}\left[\omega(t)\left|\left|\varepsilon - \varepsilon_\theta(z_t,\, t,\, e)\right|\right|^2\right], \tag{2.4}$$

где $\omega(t)$ — весовая функция шага, балансирующая вклад различных уровней зашумления (в работе применяется метод Min-SNR-γ [42], повышающий устойчивость сходимости). Условие e внедряется в трансформер через слои cross-attention (e служит источником ключей K и значений V), а номер шага t — через адаптивную нормализацию слоев (AdaLN). Выбор шума ε в качестве цели предсказания, вместо самого латентного представления, обеспечивает численно устойчивую и однородную по шагам задачу обучения. Обученная сеть параметризует обратный переход $p_\theta(z_{t-1} \mid z_t,\, e)$; многократное применение этого перехода к тензору случайного шума приводит к формированию целевого латентного представления. На этапе вывода для усиления следования условию применяется техника бесклассификаторного управления (classifier-free guidance) [43]: итоговое предсказание шума комбинируется из условного и безусловного режимов с коэффициентом наведения s:

$$\hat{\varepsilon}_\theta(z_t,\, t,\, e) = \varepsilon_\theta(z_t,\, t,\, \emptyset) + s(\varepsilon_\theta(z_t,\, t,\, e) - \varepsilon_\theta(z_t,\, t,\, \emptyset)), \tag{2.5}$$

где $\emptyset$ обозначает пустое условие, а $\hat{\varepsilon}_\theta$ — итоговое (направленное) предсказание шума. Итеративное применение обратного шага к начальному шуму $z_T \sim N(0,\, \mathrm{I})$ с последующим декодированием D_{VAE} порождает результирующее искусственное изображение, согласованное с заданным визуальным условием.

Описанная модель функционирует в двух режимах, представленных на рисунке 1.2. В режиме обучения реальное изображение кодируется в латент, зашумляется согласно (2.2), и сеть шумоподавления настраивается на предсказание шума по (2.4) при условии e, извлеченном из того же изображения. В режиме генерации процесс инициализируется чистым шумом, и обратный

переход итеративно применяется под управлением условия e, порождая новое изображение. Единообразный способ задания условия в обоих режимах обеспечивает согласованность обучения и генерации и служит основой для последующей доменной специализации.

2.1.2 Оптимизация архитектуры диффузионных трансформеров (DiT) на основе метода KV-компрессии

Главным вычислительным ограничением диффузионного трансформера при обработке изображений высокого разрешения выступает механизм self-attention.

При длине входной последовательности в N токенов классический механизм self-attention [41] описывается соотношением:

$$Attn(Q,\ K,\ V)\ =\ softmax(\frac{QK^T}{\sqrt{d_k}})\ V\ , \tag{2.6}$$

Алгоритмическая реализация этого механизма сопряжена с вычислением матрицы попарных совместимостей QK^T размера $N{\times}N$, что обусловливает квадратичную асимптотическую сложность $O(N^2d)$. В данном выражении $Q,\ K,\ V\ \in \mathbb{R}^{N\times d}$ выступают матрицами запросов, ключей и значений, d обозначает размерность признаков токена, а d_k — размерность вектора ключа. Число токенов N растет квадратично линейным размером изображения, поэтому прямое масштабирование до разрешения $1024{\times}1024$ и выше становится вычислительно неэффективным.

Для устранения этого барьера в работе реализован механизм пространственного сжатия матриц ключей и значений (KV-компрессия), предложенный в модели PixArt-Σ [29]. Перед вычислением матрицы внимания к тензорам K и V применяется оператор пространственного понижения f_R, сокращающий число токенов в R^2 раз посредством агрегации в локальных окнах $R{\times}R$:

$$Attn_{KV}(Q,\ K,\ V) = softmax(\frac{Qf_R(K)^T}{\sqrt{d_k}})\, f_R(V)\ ,\ \ f_R(K),\ f_R(V)\ \in \mathbb{R}^{N/R^2 \times d}\ , \quad (2.7)$$

где R — коэффициент пространственного сжатия, равный размеру стороны локального окна. В данном случае матрица совместимостей принимает размер $N\times(N/R^2)$, что снижает вычислительную сложность слоя внимания до $O(N^2d/R^2)$

На уровне операций сжатие выполняется в три шага: последовательность токенов преобразуется в двумерную пространственную карту, к ней применяется свертка с шагом R, после чего результат вновь разворачивается в последовательность сокращенной длины. Схема механизма приведена на рисунке 2.1.

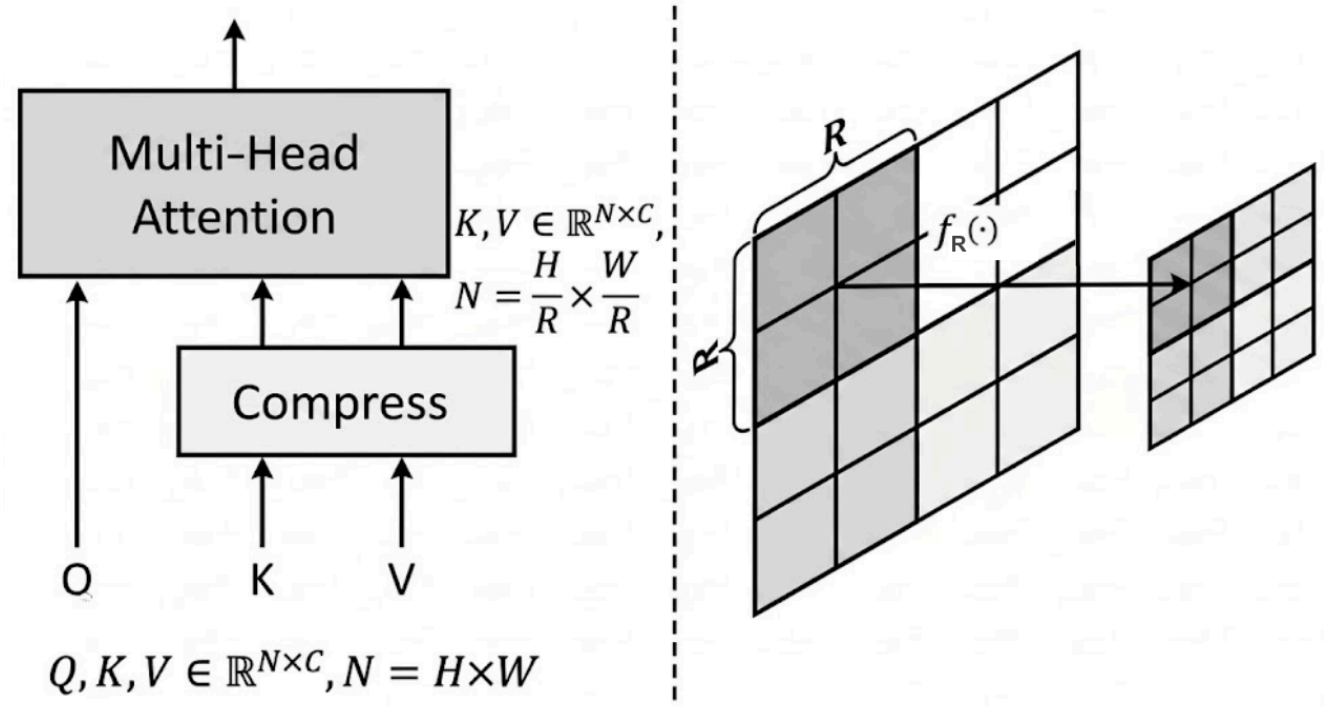


Рисунок 2.1 — Схема пространственного сжатия ключей и значений (KV-компрессия) в слое самовнимания диффузионного трансформера

Оператор f_R реализован как глубинная (depthwise) свертка с размером ядра и шагом, равными R; ее весовые коэффициенты инициализируются значением $1/R^2$, что эквивалентно операции усредняющего пулинга. Подобная инициализация принципиальна: на старте дообучения оператор не вносит искажений в

предварительно обученные представления, и оптимизация начинается из неискажающего состояния, постепенно адаптируя сжатие под данные.

Коэффициент сжатия R задает компромисс между вычислительной экономией и детальностью моделирования пространственных зависимостей на больших расстояниях (большие значения R сильнее снижают вычислительную сложность, но огрубляют разрешение ключей и значений). Существенно также, что сжатие применяется исключительно к слоям self-attention, тогда как слои cross-attention, через которые поступает визуальное условие e, остаются неизменными. Тем самым полнота управляющего сигнала UNI2-h сохраняется, и снижение вычислительной сложности не сказывается на точности обусловливания.

На уровне тензоров процедура для одного слоя выглядит следующим образом. Входная последовательность токенов $X \in \mathbb{R}^{B\times N\times C}$ преобразуется в двумерную пространственную карту признаков $X_{map} \in \mathbb{R}^{B\times C\times H\times W}$, где $N = H \cdot W$. К тензорам ключей и значений применяется один и тот же оператор пространственного сжатия $f_R(\cdot)$, реализованный с помощью операции свертки с шагом (stride), равным R: $K_c = f_R(K)$, $V_c = f_R(V)$. После обратного преобразования в одномерный формат получаются последовательности K_c, V_c длины N/R^2. Они передаются в стандартную операцию self-attention. Сохранение исходного внешнего интерфейса блока трансформера означает, что оптимизация не выходит за границы слоя и не требует адаптации остальных компонентов модели.

Выбор коэффициента R является компромиссом между скоростью и точностью. При малом R выигрыш по памяти незначителен, а при чрезмерно большом R пространственные представления ключей и значений излишне сглаживаются и могут утратить информацию о локальной структуре изображения. Поэтому R следует рассматривать как гиперпараметр вычислительной аппроксимации, который определяет степень агрегации контекста при сохранении

полного набора детализированных запросов. В экспериментальной части данный параметр оценивается не изолированно, а в составе комплексного конвейера оптимизаций, поскольку итоговая производительность зависит также от размера мини-партии, разрядности вычислений и низкоуровневой реализации ядер самовнимания.

2.2 Метод доменной специализации латентных диффузионных моделей

Базовой генеративной моделью служит предварительно обученная фундаментальная модель PixCell (раздел 1.2). Ее диффузионное ядро используется как готовая отправная точка, содержащая обширные морфологические представления о гистологических изображениях. Метод доменной специализации осуществляет тонкую настройку данной модели под целевой макро-домен не прибегая к ресурсоемкому обучению ядра с нуля и не разрушая накопленных в нем представлений. В основу метода положена декомпозиция настройки на две составляющие, действующие на разном уровне и опирающиеся на разные данные: специализацию морфологических представлений под целевой домен и тонкую коррекцию визуального стиля под отдельные наборы данных. Соответственно, метод реализуется в виде двух последовательных этапов, представленных на рисунке 2.2, — прогрессивного дообучения базовой модели под целевой домен (подраздел 2.2.1) и доменной адаптации средствами низкоранговых адаптеров (подраздел 2.2.2). Формальное описание процедур обучения и соответствующие соотношения вынесены в подраздел 3.2.1.

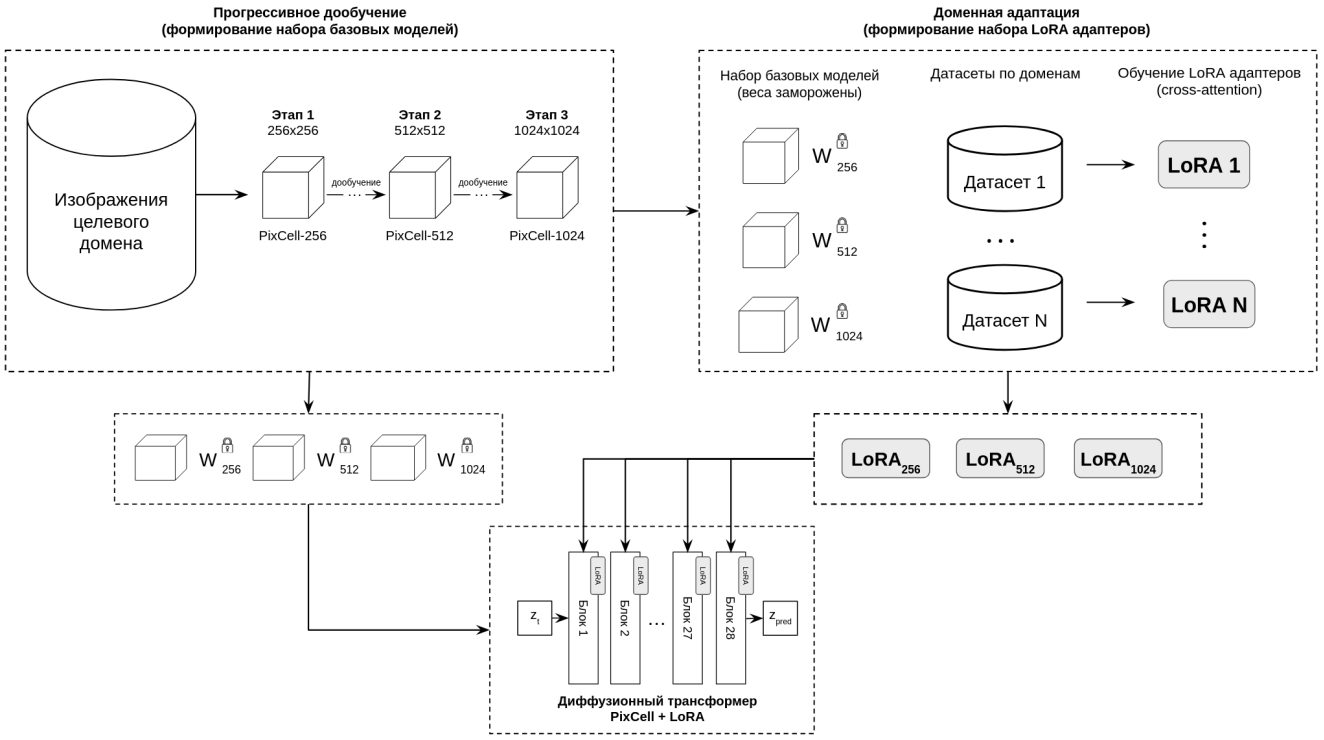


Рисунок 2.2 — Двухэтапная концептуальная схема метода доменной специализации латентных диффузионных моделей

2.2.1 Прогрессивное обучение и формирование набора базовых моделей

Специализация диффузионного ядра под целевой домен осуществляется методом прогрессивной тонкой настройки (progressive fine-tuning) предобученной модели PixCell на изображениях целевой выборки. Чтобы процесс дообучения оставался вычислительно эффективным и устойчивым при работе с высоким разрешением (где алгоритмическая сложность механизма self-attention растет квадратично относительно числа токенов, а число токенов квадратично относительно линейного размера изображения), он организуется по стратегии поэтапного наращивания разрешения.

Концептуально процедура заключается в последовательной оптимизации весов единой модели на возрастающих разрешениях (256×256, 512×512 и 1024×1024 пикселей): отправной точкой служит предобученная модель PixCell-256, а модель каждого следующего этапа инициализируется весами предыдущего и доуточняется на данных более высокого разрешения.

Такая стратегия согласуется с принципом “от слабого к сильному” (weak-to-strong), реализованным в моделях PixArt-Σ и PixCell.

Эффект прогрессивного обучения имеет двойную природу — вычислительную и оптимизационную. С вычислительной стороны ранние этапы оперируют короткими последовательностями токенов и потому требуют существенно меньших затрат на шаг; основная доля итераций приходится на дешевые низкоразрешающие этапы, а в вычислительно затратном режиме с данными высокого разрешения выполняется лишь финальная доводка. С оптимизационной стороны поэтапное наращивание разрешения соответствует принципу обучения от грубого к детальному (coarse-to-fine): модель сначала осваивает устойчивые макромасштабные закономерности, после чего переходит к высокочастотным элементам, не разрушая уже сформированных базовых представлений. Это сглаживает ландшафт функции потерь и снижает риск расхождения, характерный для обучения высокоразрешающих моделей “с нуля”.

Принципиальной особенностью метода является строгая фиксация уровня оптического увеличения на всех этапах. При наращивании разрешения изменяется охватываемая площадь изображения (поле зрения), а не масштаб дискретизации, благодаря чему модель изучает морфологические паттерны, согласованные по масштабу. С ростом разрешения согласованно увеличивается и число строк матрицы визуального обусловливания e от одной строки для фрагмента 256×256 до шестнадцати строк для фрагмента 1024×1024 (подраздел 2.1.1). Дообучение на этом этапе ведется в режиме без учителя, на изображениях целевого домена без привлечения разметки. Такой выбор рационален: в прикладных задачах доля размеченных данных мала, тогда как основной объем составляют неразмеченные WSI изображения, и прогрессивное дообучение задействует именно этот обширный ресурс неразмеченных данных для специализации морфологических представлений под домен (количественные характеристики выборок приведены в главе 4).

Результатом этапа является набор доменно-специализированных базовых моделей $W_0 = \{W_0^{256*}, W_0^{512}, W_0^{1024}\}$ — адаптированных под особенности целевого домена версий PixCell для соответствующих разрешений; в частности, W_0^{1024} соответствует модели, обозначаемой далее как PixCell-1024. Веса набора W_0 фиксируют морфологию целевого домена и в дальнейшем остаются замороженными, а каждая модель применяется для генерации изображений соответствующего разрешения.

2.2.2 Доменная адаптация и перенос стиля средствами низкоранговых адаптеров (LoRA)

Базовые модели набора W_0 фиксируют морфологию целевого домена, однако воспроизводят стиль, усредненный по разнородным источникам обучающей выборки, и потому не вполне соответствуют визуальным характеристикам каждого отдельного набора данных, различающихся условиями оцифровки. Тонкая коррекция стиля под конкретный набор данных выполняется методом низкоранговой адаптации (Low-Rank Adaptation, LoRA), не затрагивающим базовые веса. Идея метода состоит в том, что необходимое приращение весовой матрицы аппроксимируется произведением двух матриц существенно меньшего ранга. В процессе дообучения оптимизируются только эти компактные матрицы, что сокращает число обновляемых параметров на несколько порядков. Адаптер инициализируется таким образом, чтобы в начальный момент времени он не вносил возмущений в работу сети.

Адаптеры внедряются в слои cross-attention. Выбор именно этих слоев методологически обоснован: слои self-attention кодируют пространственно-геометрическую структуру изображения, которую следует сохранить, тогда как cross-attention определяет отклик модели на управляющее условие и потому является естественным местом для внедрения механизма коррекции стиля. Замороженность базовых весов исключает риск катастрофического забывания доменно-инвариантных признаков.

Каждый адаптер обучается на размеченном подмножестве соответствующего набора данных, тогда как доменно-инвариантная морфология уже зафиксирована на этапе прогрессивного дообучения. Такое разделение источников — обширные неразмеченные изображения для морфологии и компактные размеченные подмножества для стиля — позволяет эффективно использовать все имеющиеся данные. Совокупность обученных адаптеров формирует библиотеку, в которой каждому специфическому набору данных соответствует свой профиль. Активация нужного адаптера позволяет переключать стиль генерации без перезагрузки и переобучения базовой модели, что делает доменную специализацию масштабируемой и обратимой. Тем самым один набор базовых моделей W_0 обслуживает произвольное число доменов, а компенсация ковариантного сдвига (раздел 1.1) сводится к подключению соответствующего адаптера.

Адаптер перенастраивает реакцию слоев cross-attention на один и тот же визуальный сигнал e. При неизменном морфологическом условии активный адаптер сдвигает колориметрические и текстурные характеристики результата к статистикам конкретного набора данных. Геометрия объекта, заданная условием e, и стиль его отрисовки, заданный адаптером, разделяются и управляются независимо — в чем и состоит суть параметрического переноса стиля без вмешательства в геометрические представления.

Практическая ценность такого решения многогранна. Объем одного адаптера составляет малую долю от размера базовой модели, поэтому библиотека из десятков доменных адаптеров хранится компактно. Переключение между доменами сводится к подмене активного адаптера без выгрузки базовой модели из памяти ускорителя, что упрощает процесс развертывания и пакетную обработку нескольких доменов. В отличие от ряда других методов параметрически-эффективной адаптации, низкоранговое разложение не вводит дополнительных слоев в граф вычислений на этапе вывода: приращение при необходимости сливается с базовой матрицей и не увеличивает задержку инференса.

Параметрическую эффективность LoRA можно выразить аналитически. Для исходной весовой матрицы $W \in \mathbb{R}^{d\times k}$ количество параметров составляет $d\times k$. При использовании низкоранговой адаптации оригинальные веса модели замораживаются, операции линейных проекций дополняются обучаемым приращением $\Delta W = BA$, где матрицы $A \in \mathbb{R}^{r\times k}$ и $B \in \mathbb{R}^{d\times r}$ содержат в сумме $r(d + k)$ параметров. Поскольку ранг $r \ll min(d, k)$, отношение $\frac{r(d+k)}{dk}$ оказывается пренебрежимо малым. В контексте текущего диссертационного исследования это свойство играет важную роль, поскольку доменная специализация обеспечивается за счет интеграции компактных адаптеров, что нивелирует необходимость хранения полных копий весов базовой модели.

С программно-инженерной точки зрения библиотека адаптеров реализуется как индексируемое хранилище приращений весов ΔW, а не как набор самостоятельных моделей. Для каждого адаптера фиксируются метаданные: домен, рабочее разрешение, версия базовой модели, ранг r, масштабирующий коэффициент α, дата обучения, контрольная сумма обучающего набора данных и метрики валидации. Такой подход позволяет автоматически верифицировать совместимость адаптера с базовой моделью перед этапом инференса.

2.3 Алгоритм локального контекстно-ориентированного встраивания объектов

Метод доменной специализации (раздел 2.2) обеспечивает синтез изображений в стиле целевого набора данных, однако не решает задачу прицельного обогащения выборки редким классом, когда целевой минорный объект должен быть размещен в корректном контексте. Для этого разработан алгоритм локального контекстно-ориентированного встраивания объектов (контекстно-адаптивного inpainting), направленный на компенсацию дисбаланса минорных классов и повышение их разнообразия. Концептуальная схема алгоритма приведена на рисунке 2.3; его детальное описание с формулами и псевдокодом приведено в подразделе 3.2.2.

Рисунок 2.3 — Концептуальная схема алгоритма контекстно-адаптивного встраивания объектов

Пусть заданы целевой объект D (донорский фрагмент изображения с минорным объектом), фоновое изображение T и бинарная маска объекта M; в роли генеративной модели G_θ выступает доменно-специализированная модель из раздела 2.2 — базовая модель W_0^{1024} с активным LoRA-адаптером целевого набора данных (или без него). Требуется встроить объект D в фон T с сохранением

исходной морфологии объекта и обеспечением бесшовной согласованности границ совмещения. Алгоритм поддерживает два режима подготовки фона: использование реального фрагмента либо генерацию синтетического фона диффузионной моделью. Независимо от режима работа концептуально декомпозируется на три последовательных этапа.

Этап 1. Цель данного этапа заключается в подготовке донорского объекта и фона к совмещению с минимизацией визуального рассогласования между ними. В режиме реального фона фоновый фрагмент подбирается автоматически из базы подготовленных изображений, что снижает исходное рассогласование еще до коррекции. Донорский объект поворачивается так, чтобы согласовать ориентацию его клеточных структур с ориентацией структур фона, и переносится в тканево-плотную область целевого изображения. Затем выполняется хроматическая гармонизация: цветовые статистики объекта приводятся к статистикам фона в перцептивном цветовом пространстве CIELAB [20], выбор которого обусловлен приближенной перцептивной однородностью этого пространства. Этап завершается композицией объекта и фона по бинарной маске. Принципиально, что полученный композит уже содержит реальный объект в целевом контексте — это отличает подход от классического инпейнтинга, синтезирующего содержимое маски “с нуля”.

Этап 2. На данном этапе композит служит отправной точкой управляемого диффузионного преобразования в латентном пространстве (рисунок 2.3). Из композита извлекается визуальное условие энкодером UNI2-h; вычисленное по самому композиту, оно удерживает обратный процесс в семантике окружающей ткани, благодаря чему регенерируемая зона шва остается согласованной с контекстом. Ключевая идея этапа — частичное, а не полное зашумление, когда композит зашумляется до некоторого промежуточного шага диффузии, после чего запускается обратный процесс диффузии. Ограниченная глубина зашумления сохраняет крупномасштабную структуру композита, а значит, и морфологию встроенного объекта, регенерируя лишь детали в зоне сращивания. Меньшая глубина оставляет композит почти нетронутым, но слабее сглаживает шов;

большая усиливает гармонизацию ценой риска исказить объект, поэтому глубина выбирается как компромисс между сохранностью структуры и качеством шва.

Этап 3. Диффузионное преобразование может затронуть фоновые структуры вне зоны встраивания; чтобы извлечь из его результата только полезную работу по сращиванию швов, применяется аппарат математической морфологии [45]. На основе расстояния до границы объекта строится гладкая приграничная карта влияния, определяющая, где результат диффузии замещает исходный композит. Влияние максимально у самого шва, где артефакты совмещения наиболее вероятны, и затухает вглубь как объекта, так и фона; благодаря этому внутренняя структура донора и фон вдали от границы сохраняются практически без изменений. Итоговое изображение собирается как взвешенная композиция исходного композита и результата диффузии под управлением этой карты. Строгая локализация изменений делает переход незаметным, а вносимые моделью правки — ограниченными зоной шва.

В целом алгоритм декомпозирует задачу на три последовательных перехода: выбор или генерацию фона, композицию объекта в заданной области и локальную диффузионную гармонизацию. Это разделение и обеспечивает заявленные свойства: сохранение морфологии донорского объекта (он является результатом композиции, а не синтезируется моделью заново), неизменность фона вне маски и бесшовность границ.

Алгоритмически поиск координат встраивания сводится к выбору оптимальной позиции на дискретной сетке фона. Каждая точка-кандидат проверяется по ряду ограничений: отсутствие пересечений с запрещенными зонами, расстояние до краев изображения и цветовое соответствие. Среди допустимых вариантов выбирается позиция, минимизирующая суммарный штраф. Такой детерминированный подход автоматизирует ручной поиск и эффективен для пакетной генерации.

Смешивание композита и результата генерации выполняется с помощью сглаженной карты воздействия (весовая карта). На основе бинарной маски объекта M вычисляется карта расстояний до ее границ $D(M)$, которая задает функцию

веса $w \in [0, 1]$. Вес функции максимален в зоне стыка (“шва”) и плавно убывает при удалении от него. Такая процедура строго локализует изменения, вносимые моделью, и делает переход незаметным.

В отличие от диффузионного инпейнтинга, восстанавливающего замаскированную область из шума под управлением контекста [44], предложенный алгоритм не порождает объект заново, а переносит реальный донорский объект в новый контекст, используя диффузионную модель лишь для гармонизации границ. Это, во-первых, гарантирует морфологическую достоверность редкой структуры, а во-вторых, обеспечивает рост разнообразия минорного класса: один и тот же донорский объект может быть многократно встроен в различные фоновые контексты с разными геометрическими преобразованиями, порождая семейство достоверных, но не повторяющихся образцов.

При недостатке реальных фоновых фрагментов фон порождается по схеме контекстного масштабирования. На базе малого исходного фрагмента размерностью 256×256 пикселей формируется набор его версий со случайным поворотом (всего 16 изображений). Векторы признаков этих изображений, извлеченные моделью UNI2-h, объединяются в единое плотное условие, эквивалентное управляющему сигналу полного фрагмента 1024×1024 ($e \in \mathbb{R}^{16\times1536}$). Под управлением этого сигнала модель выполняет полный процесс генерации, порождая синтетическое изображение репрезентативного фона. Детальная математическая формализация всех трех этапов, включая уравнения и псевдокод, приведена в подразделе 3.2.2.

Глава 3. Программные средства и архитектура системы генерации

В настоящей главе приводится описание архитектуры, алгоритмического обеспечения и программной реализации разработанного генеративного комплекса, построенного на базе экосистемы библиотеки Hugging Face [57], а также методов аппаратно-программной оптимизации вычислений при обработке гигапиксельных изображений. Раздел 3.1 определяет модульную программную архитектуру комплекса и принципы организации вычислительных конвейеров. Раздел 3.2 содержит алгоритмическое обеспечение двух прикладных сценариев синтеза — доменной специализации (подраздел 3.2.1) и контекстно-ориентированного встраивания объектов (подраздел 3.2.2). Раздел 3.3 описывает методы аппаратно-программной оптимизации и интеграцию комплекса с конвейерами прикладного анализа. Изложение опирается на математический аппарат и концепции, введенные в главах 1 и 2, и доводит их до уровня воспроизводимой программной реализации.

3.1 Проектирование модульной архитектуры и программных конвейеров

Генеративный комплекс спроектирован как модульная архитектура, в рамках которой масштабируемое взаимодействие обеспечивают четыре независимых функциональных контура (рисунок 3.1), обменивающихся данными через фиксированные интерфейсы. В основу проектирования положен принцип разделения ответственности: каждый контур решает один класс подзадач, а фиксированный формат интерфейсов обмена обеспечивает слабую связанность компонентов и допускает асинхронное исполнение этапов вычислительного конвейера. Такое разделение повышает сопровождаемость и расширяемость системы и позволяет независимо оптимизировать отдельные стадии обработки.

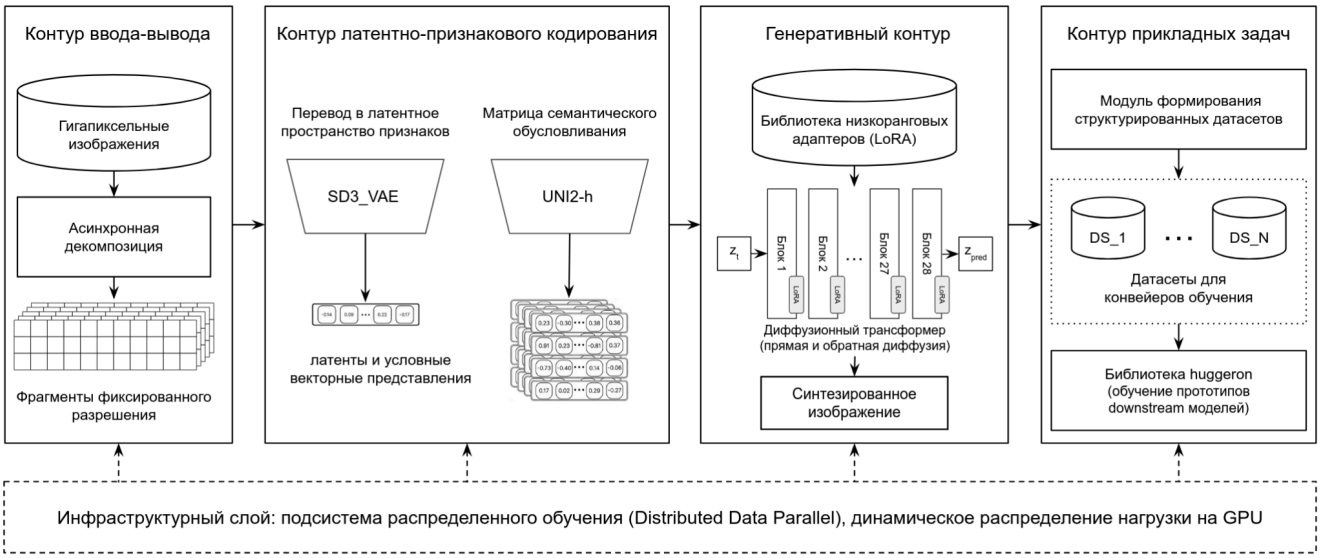


Рисунок 3.1 — Схема взаимодействия независимых функциональных контуров в модульной архитектуре генеративной системы

Контур ввода-вывода реализует логику асинхронной декомпозиции гигапиксельных изображений на отдельные тайлы фиксированного разрешения. Поскольку WSI хранится в виде многоуровневой пирамиды и не помещается в оперативную память целиком (подраздел 1.1.1), контур опирается на оптимизированные инструменты потокового чтения произвольных регионов нужного уровня увеличения (OpenSlide или cuCIM). Возможность организации асинхронного чтения и декодирования тайлов важна потому, что именно подсистема ввода-вывода является характерным узким местом конвейеров обработки гигапиксельных данных. Вынесение этих операций в асинхронный поток предотвращает простаивание вычислительного ядра в ожидании данных. Контур оперирует координатами регионов на выбранном уровне пирамиды и формирует поток тайлов фиксированного разрешения, при необходимости с управляемым перекрытием смежных фрагментов для последующей бесшовной сборки. Буферизация готовых тайлов сглаживает неравномерность обращений к дисковой подсистеме и поддерживает постоянную загрузку вычислительного ядра.

Контур латентно-признакового кодирования объединяет вариационный автокодировщик SD3-VAE и фундаментальную модель-кодировщик UNI2-h для параллельного отображения исходных изображений в латентное признаковое пространство. Контур реализует переход, формализованный в подразделе 2.1.1:

сеть SD3-VAE отображает фрагмент в латентное представление, а UNI2-h формирует матрицу визуальных признаков. Обе модели заморожены, что не только соответствует логике визуального управления генерацией, но и открывает возможность однократного предварительного вычисления их выходов с последующим кэшированием (данный прием подробно рассматривается в разделе 3.3). Параллельность отображения двумя независимыми моделями повышает пропускную способность на стадии кодирования. Обработка выполняется в пакетном режиме: данные группируются, после чего SD3-VAE возвращает латентные представления исходных изображений, а UNI2-h – матрицу векторных представлений (эмбеддингов) с пространственной организацией по локальным областям (подраздел 2.1.1).

Генеративный контур образует вычислительное ядро системы. Он реализует этапы прямой и обратной диффузии на базе модифицированной архитектуры диффузионного трансформера (подраздел 2.1.2) и динамической библиотеки низкоранговых адаптеров LoRA. Принципиальной особенностью контура является наличие точки внедрения алгоритмических модификаций в граф вычислений. Через нее реализуются перехват слоев перекрестного внимания при доменной специализации (подраздел 3.2.1) и маскированное смешивание представлений при контекстном встраивании (подраздел 3.2.2). Программно ядро выполнено на основе модификации абстракции `DiffusionPipeline` библиотеки diffusers [53], что обеспечивает совместимость с обширной экосистемой готовых компонентов: планировщиками шума, утилитами работы с условиями и инструментами обработки весов. Модификация затрагивает прежде всего цикл шумоподавления и механизм передачи условий: стандартная процедура обратной диффузии дополняется перехватом промежуточных представлений и управляющими воздействиями, специфичными для каждого прикладного сценария. Такое решение позволяет локализовать оригинальные алгоритмические модули в пределах генеративного контура, не нарушая контрактов с соседними контурами.

Контур прикладных задач реализует программные интерфейсы сопряжения генеративной подсистемы с прикладными конвейерами анализа. Через

специализированную библиотеку huggeron (надстройку над фреймворком Hugging Face) контур автоматизирует преобразование результатов синтеза в структурированные наборы данных, совместимые с конвейерами обучения моделей классификации и сегментации. Тем самым выход системы напрямую включается в прикладной цикл обучения распознающих алгоритмов (подраздел 1.1.1), а синтезированные изображения экспортируются потоковым методом без ручного преобразования форматов. Экспортируемые наборы формируются как согласованные совокупности изображений, масок и аннотаций в формате, принятом в экосистеме Hugging Face, что обеспечивает их прямую загрузку конвейерами обучения сегментационных и классификационных моделей. Потоковая запись позволяет обрабатывать объемы, превышающие оперативную память, не накапливая весь синтетический набор целиком.

Фиксированный формат интерфейсов обмена данными обеспечивает слабую связанность компонентов и позволяет выполнять этапы вычислительного конвейера асинхронно: пока контур ввода-вывода готовит очередной пакет изображений, контур кодирования и генеративное ядро обрабатывают предыдущий. Для динамического распределения нагрузки на графические ускорители применяется модуль `DistributedDataParallel`, реплицирующий вычислительное ядро по доступным устройствам и агрегирующий результаты. Модульность архитектуры обеспечивает и расширяемость: точка внедрения в контуре управления допускает подключение альтернативных механизмов обусловливания (например, трансформерного варианта ControlNet) без перестройки остальных контуров, хотя в составе защищаемых сценариев синтеза такие механизмы не используются по причинам, изложенным в подразделе 1.3.

На стадиях обучения и пакетной генерации контуры ввода-вывода и кодирования могут замещаться чтением предварительно вычисленных данных из кэша (раздел 3.3).

Описанная декомпозиция отражена и в структуре программной реализации комплекса, организованной в виде пакета модулей. Конфигурационные параметры и пути сосредоточены в модуле конфигурации; работа с WSI изображениями и

подготовка данных в инструментальных модулях (потоковое чтение и нарезка тайлов, формирование наборов); загрузка и кэширование латентно-признаковых представлений в модулях наборов данных, включая вариант с предвычисленными признаками; вычислительное ядро (обертки базовой модели, адаптеры, планировщик шума, реализация KV-компрессии) в модулях моделей и утилит; сквозные службы профилирования, журналирования и трекинга экспериментов вынесены в отдельные утилиты. Соответствие функциональных контуров и групп программных модулей приведено в таблице 3.

Таблица 3 — Соответствие функциональных контуров архитектуры и групп программных модулей реализации

Функциональный контур	**Назначение**	**Группы программных модулей**
Ввода-вывода	Декомпозиция WSI на тайлы, потоковое чтение.	Инструменты работы с WSI, подготовки данных.
Латентно-признакового кодирования	Отображение в латент и эмбеддинг (SD3-VAE, UNI2-h).	Модули наборов данных обертки кодировщиков.
Генеративный (ядро)	Прямая/обратная диффузия, DiT + KV-компрессия, библиотека LoRA.	Модули моделей, планировщики шума, утилита KV-компрессии.
Прикладных задач	Экспорт в наборы данных, интерфейс huggeron.	Инструменты подготовки наборов, интерфейс huggeron.
Сквозные службы	Профилирование, журналирование, трекинг, конфигурация.	Утилиты бенчмаркинга и логирования, модуль конфигурации.

Принятая архитектура решает двойную задачу: с одной стороны, она инкапсулирует научную новизну работы в генеративном контуре, оставляя его интерфейсы стабильными; с другой стороны, обеспечивает воспроизводимость и

переносимость за счет использования стандартизованных компонентов экосистемы Hugging Face и единого формата обмена данными.

Архитектура целенаправленно отвечает требованиям, сформулированным в главе 1. Асинхронный контур ввода-вывода и потоковое чтение снимают остроту проблемы дискового доступа при работе с пирамидальными изображениями; распределенное исполнение и пакетная обработка обеспечивают масштабируемость синтеза изображений высокого разрешения; контур прикладных задач замыкает синтез на прикладной цикл обучения распознающих моделей, ради которого и выполняется генерация. Тем самым модульная декомпозиция выступает не самоцелью, а средством согласования вычислительной эффективности, расширяемости и прикладной применимости комплекса. Воспроизводимость экспериментов поддерживается централизованным управлением конфигурациями и фиксацией путей к данным и моделям в выделенном модуле, а также сквозным журналированием и трекингом параметров запусков. Дальнейшие разделы раскрывают алгоритмическое наполнение генеративного контура (раздел 3.2) и методы оптимизации, повышающие эффективность всего конвейера (раздел 3.3).

3.2 Алгоритмическое обеспечение прикладных сценариев синтеза

В рамках генеративного контура системы реализованы два прикладных сценария целенаправленного расширения обучающих выборок: гибкий вариативный синтез, выступающий алгоритмической реализацией метода доменной специализации (подраздел 3.2.1) и контекстно-ориентированное встраивание объектов (подраздел 3.2.2). Оба сценария транслируют теоретические концепции, изложенные в главе 2, и приводят их к виду детерминированных вычислительных конвейеров.

Все рассматриваемые сценарии реализованы в виде независимых программных модулей в составе единого вычислительного комплекса. В их состав входят: специализированные программные модули для дообучения базовой модели, тренировки доменных адаптеров, условного вариативного синтеза и

локального контекстного встраивания. Использование общей библиотеки конфигураций и унифицированного формата обмена данными позволяет запускать эти этапы как в строгой последовательности, так и интегрировать их во внешние системы без модификации базовых алгоритмов.

3.2.1 Алгоритмическая реализация метода доменной специализации латентных диффузионных моделей

Вычислительный алгоритм гибкого вариативного синтеза опирается на двухэтапный метод (раздел 2.2), и процедурно разделяется на две независимые фазы: фазу предварительного обучения (подготовку базовых моделей и библиотеки адаптеров) и фазу вывода (управляемую генерацию). Обе фазы представлены на рисунке рисунке 3.2.

Фаза обучения. Формирование набора базовых моделей выполняется в процессе прогрессивной тонкой настройки архитектуры PixCell (подраздел 2.2.1). Обучение организуется как последовательность этапов $s = 1, \ldots, S$ с монотонно возрастающими разрешениями r_s. Модель каждого этапа инициализируется весами предыдущего и настраивается путем минимизации целевой функции $\mathcal{L}$ на данных соответствующего разрешения:

$$W_s = arg\,min_W \mathcal{L}(W; \mathcal{D}_{r_s}),\ \ W_s\big|_{init} = W_{s-1},\ \ W_1\big|_{init} = W_{PixCell}, \qquad (3.1)$$

где $W_{PixCell}$ — веса базовой модели PixCell-256, $\mathcal{D}_{r_s}$ — выборка изображений целевого домена разрешения r_s, без разметки. Поскольку начальная модель W_1 инициализируется весами PixCell, выражение (3.1) описывает каскадную оптимизацию единой модели, а не обучение независимых сетей с нуля.

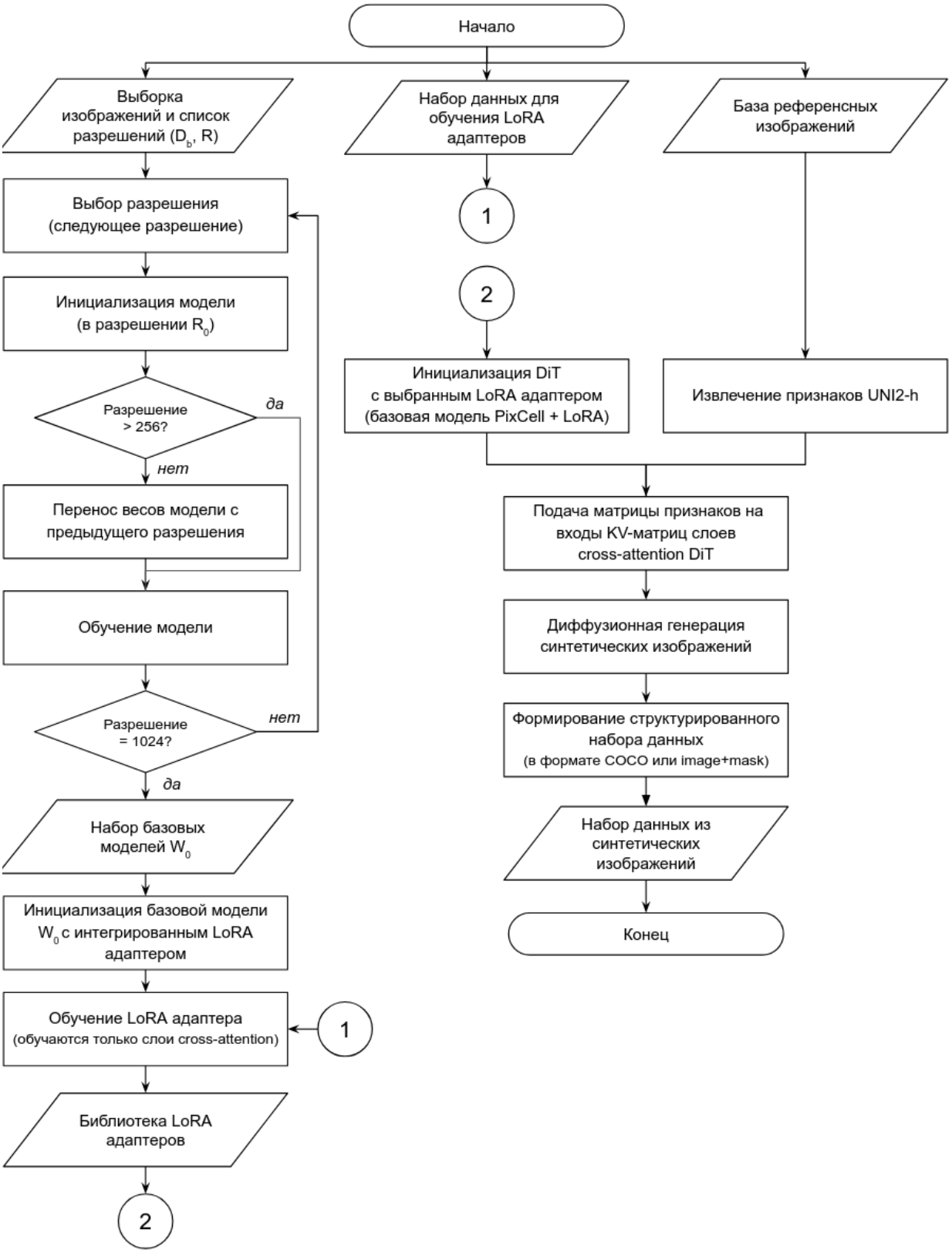


Рисунок 3.2 — Схема алгоритма гибкого вариативного синтеза. Левая ветвь иллюстрирует процесс дообучения и формирование библиотеки адаптеров; правая ветвь — конвейер диффузионной генерации

Результатом фазы является набор доменно-специализированных базовых моделей $W_0 = \{W_0^{256}, W_0^{512}, W_0^{1024}\}$, веса которых далее замораживаются.

Тонкая коррекция стиля под отдельный набор данных выполняется обучением низкорангового адаптера (подраздел 2.2.2). Для замороженной весовой матрицы $W_0 \in \mathbb{R}^{d\times k}$, слоя cross-attention эффективная матрица на этапе адаптации принимает вид:

$$W' = W_0 + \Delta W = W_0 + \frac{\alpha}{r}BA, B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}, r \ll min(d,k) \quad (3.2)$$

где r — ранг адаптера, α — масштабирующий коэффициент. Обучаются исключительно матрицы A и B (W_0 заморожена). Матрица A инициализируется случайным гауссовым распределением, а B нулями, благодаря чему в начальный момент $\Delta W = 0$, и адаптированная модель полностью тождественна базовой. Обучение ведется на размеченном подмножестве соответствующего набора данных, с использованием стратегии взвешивания Min-SNR-γ, повышающей устойчивость сходимости. Каждый обученный адаптер сохраняется в библиотеку, в которой одному набору данных отвечает один компактный набор матриц $\{A, B\}$. На каждом шаге обучения адаптера прямой проход выполняется через базовую модель с активным приращением ΔW, а градиент распространяется только на параметры A и B; малое число обучаемых параметров снижает требования к объему размеченных данных и ускоряет сходимость. Различные значения ранга r и коэффициента α задают компромисс между выразительностью адаптера и риском переобучения на ограниченном размеченном подмножестве.

Фаза вывода. Конвейер семантически обусловленной генерации комбинирует визуальные условия с контролируемой активацией адаптеров. На предварительном шаге исследователь вручную формирует набор управляющих визуальных референсов X_{ref}, представляющих дефицитные подмножества данных, которые требуется обогатить. Такой полуавтоматический отбор переносит экспертное знание о дефицитных категориях непосредственно в управляющий

сигнал: референсы фиксируют желаемую морфологию редких структур, которую генеративный контур затем тиражирует в нужном стиле и объеме. В процессе генерации алгоритм перехватывает управление вычислительным графом в слоях cross-attention каждого из 28 трансформерных блоков модели W_0^{1024} (PixCell-1024) и разворачивает на графе два параллельных управляющих потока данных. Перехват реализуется через точку внедрения в генераторном контуре.

Первый поток — пространственное управление. Для каждого образца из набора референсов извлекается матрица признаков:

$$e = E_{UNI}(X_{ref}) \in \mathbb{R}^{16\times1536}, \quad (3.3)$$

задающая ограничение на макроструктуру целевых объектов. Эта матрица подается на входы формирования ключей и значений слоев перекрестного внимания, удерживая морфологию и внутреннюю структуру синтезируемого объекта. Второй поток (коррекция стиля) активирует весовые приращения ΔW выбранного из библиотеки низкорангового адаптера, что переносит на результат визуальный стиль целевого набора данных.

Формально оба потока сходятся в операции cross-attention каждого блока. Запросы формируются из текущего латентного представления z синтезируемого изображения, а ключи и значения — из условия e, причем проекционные матрицы заменяются адаптированными согласно (3.2):

$$Q = zW_Q^{'}, K = eW_K^{'}, V = e\,W_V^{'}, CrossAttn(z,\, e) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V\,W_0^{'}, \quad (3.4)$$

где $W_Q^{'}$, $W_K^{'}$, $W_V^{'}$, $W_0^{'}$ — адаптированные матрицы проекций запросов, ключей, значений и выхода соответствующего домена. Выражение (3.4) делает явной природу двух потоков: морфология поступает через содержимое матриц K и V, формируемых из e, а доменный стиль — через приращения весов ΔW в проекциях. Тем самым морфология и стиль управляются независимыми механизмами в пределах одной операции cross-attention.

Управляемый описанным образом обратный диффузионный процесс (раздел 2.1) порождает изображение высокого разрешения, согласованное с заданными референсами по структуре и с целевым набором данных по стилю. Алгоритм допускает пакетную генерацию: при фиксированных референсах и активном адаптере формируется серия изображений, различающихся реализацией стохастического процесса, что обеспечивает вариативность образцов при сохранении заданной морфологии и стиля. Замена активного адаптера без перезагрузки базовой модели позволяет в рамках единого запуска порождать данные для нескольких целевых наборов.

Вычислительная процедура фазы генерации приведена в виде псевдокода (листинг 3.1); вызов модели $\varepsilon_\theta(z_i, i, e)$ в нем подразумевает перекрестное внимание (3.4), в котором ключи и значения формируются из условия e адаптированными проекциями. Листинг соответствующего исходного программного модуля вынесен в приложение Б.

Листинг 3.1 — Псевдокод фазы вывода: семантически обусловленная генерация с доменной специализацией

```
Входные данные:
        X_REF     — управляющие визуальные референсы;
        W_0^1024  — базовая модель (28 блоков DiT);
        {A, B}    — адаптер целевого домена;
        E_UNI     — энкодер;
        D_VAE     — декодер;
        T         — число шагов;
        s         — коэффициент наведения CFG
Параметры:
        r = 4     — ранг адаптера;
        α = 4     — масштабирующий коэффициент.
 Выходные данные:
        X         — синтетическое изображение высокого разрешения
e ← E_UNI(X_REF)        # поток 1 (морфология): условие e ∈ R^{16×1536}
для каждого из 28 блоков DiT:      # поток 2 (стиль): активация адаптера
    W'* ← W_0* + (α / r) · B · A    # в проекциях cross-attention (Q, K, V)

z_T ← ε,   ε ~ N(0, I)              # инициализация шумом
для i = T, T−1, …, 1:               # обратная диффузия
    ε_COND   ←ε_θ(z_i, i, e)        # K, V формируются из e (см. (3.4))
    ε_UNCOND ←ε_θ(z_i, i, ∅)
    ε^ ← ε_UNCOND + s · (ε_COND − ε_UNCOND) # CFG
    z_{i-1} ← Денойзинг(z, ε^, i)
X ← D_VAE(z_0)
вернуть X
```

3.2.2 Алгоритм контекстно-ориентированного встраивания объектов

Алгоритм контекстно-ориентированного встраивания объектов (inpainting) реализует метод, концептуально изложенный в разделе 2.3. В данном разделе приводится детальная формализация работы алгоритма (в соответствии с этапами программной реализации). Пошаговая логика работы алгоритма представлена на рисунке 3.3. Листинг с исходным кодом соответствующего программного модуля приведен в приложении В.

Ко входным данным алгоритма относятся: донорский объект D, его бинарная маска M, фоновое изображение T, базовая модель G_θ (доменно-специализированная модель W_0^{1024} с активным LoRA адаптером или без него), энкодер E_{UNI} и автокодировщик (E_{VAE}, D_{VAE}).

Алгоритм поддерживает два режима подготовки фонового изображения T: использование реального фрагмента ($T_{real} \sim p_{real}(x)$) либо генерацию синтетического фона ($T_{real} \sim p_\theta(x \mid E_{UNI})$) диффузионной моделью. Независимо от выбранного режима работу алгоритма можно разделить на три последовательных этапа:

- пространственное и хроматическое выравнивание;
- семантически обусловленная диффузионная генерация;
- морфологическая коррекция границ.

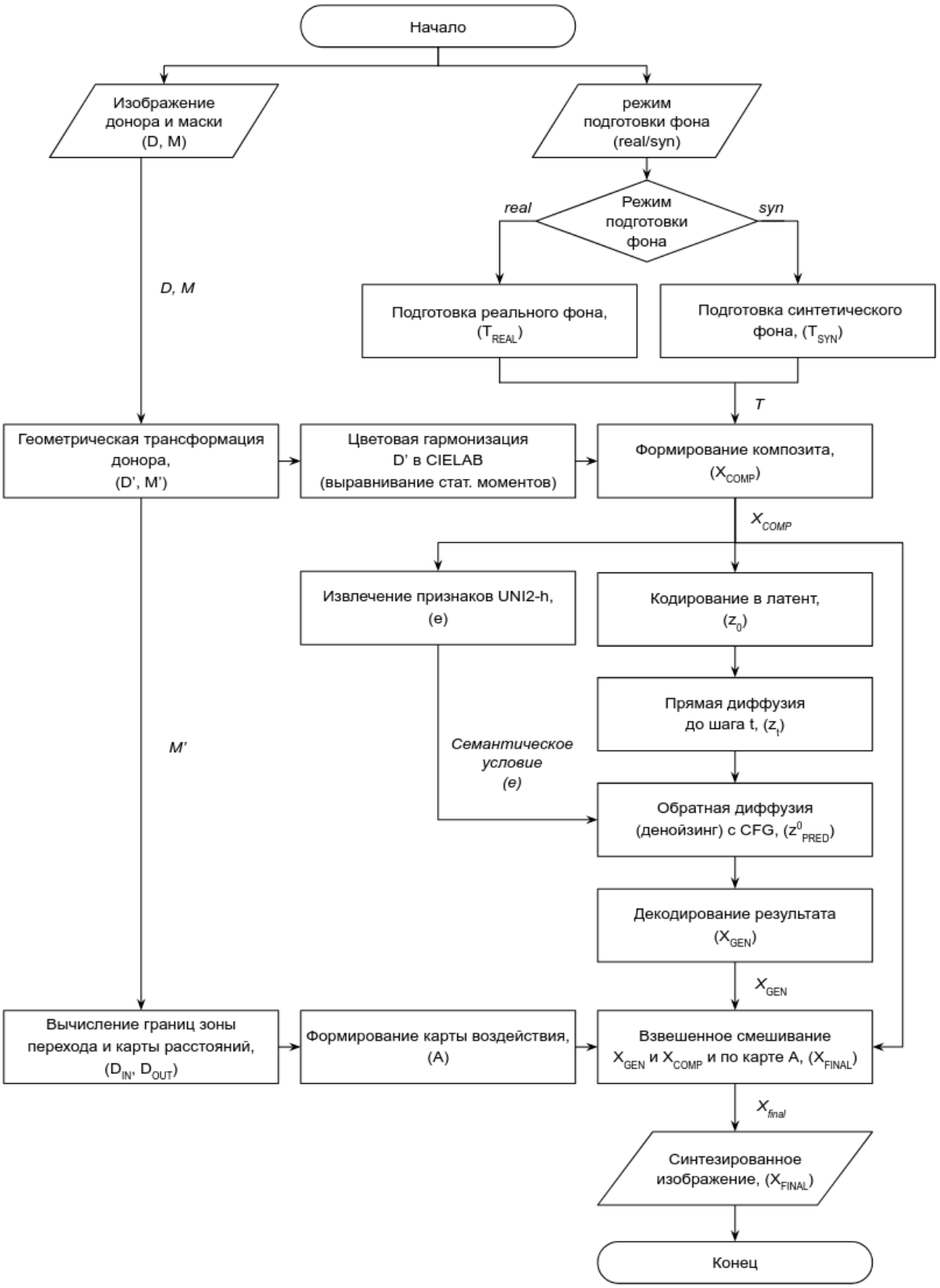


Рисунок 3.3 — Схема алгоритма контекстно-ориентированного inpainting на основе семантически обусловленной диффузионной модели и метода морфологического пограничного смешивания

Пространственное и хроматическое выравнивание. На данном этапе донорский объект приводится к виду, пространственно и колориметрически согласованному с фоновым изображением. Для пространственного согласования структур предварительно оцениваются доминирующие ориентации клеточных ядер фона φ_T и донора φ_D на основе их медианного угла наклона. Затем донор поворачивается на расчетный угол $\varphi = (\varphi_D - \varphi_T)\, mod\, 360^{\circ}$. После ротации объект переносится на фоновое изображение в точку с координатами (x, y). Алгоритмически поиск оптимальной позиции (x, y) осуществляется на дискретной сетке фона: точки-кандидаты оцениваются с учетом удаленности от краев, отсутствия пересечений с запрещенными зонами и цветового соответствия. Итоговыми координатами становится позиция, минимизирующая суммарный штраф. Полученные пространственно согласованные версии донора и его маски обозначаются D' и M'. Для устранения колориметрического рассогласования выполняется выравнивание статистических моментов донорского объекта относительно соответствующих характеристик фона. Данное преобразование осуществляется в перцептивно однородном цветовом пространстве CIELAB каждого канала $k \in \{L,\ a,\ b\}$:

$$\hat{x}_D^k = \left(\frac{x_D^k - \mu_D^k}{\sigma_D^k}\right)\sigma_T^k + \mu_T^k, \tag{3.5}$$

где μ_D^k, σ_D^k μ_T^k, σ_T^k— средние значения и стандартные отклонения пиксельных интенсивностей в канале k для донора D' и фона T Для предотвращения избыточных цветовых искажений результат преобразования сглаживается путем линейного смешивания с коэффициентом $\lambda = 0,85$. С применением бинарной маски M' формируется предварительный композит из донора D' и фона T:

$$X_{comp} = (1 - M') \odot T + M' \odot D', \tag{3.6}$$

где $\odot$ — поэлементное умножение.

Семантически обусловленная диффузионная генерация. Из композита извлекается матрица признаков $e = E_{UNI}(X_{comp}) \in \mathbb{R}^{16x1536}$, после чего композит переводится в латентное представление $z_0 = E_{vae}(X_{comp})$ и подвергается зашумлению в рамках прямого процесса диффузии до промежуточного временного шага $t = \lfloor \tau \cdot t_{max} \rfloor$:

$$z_t = \sqrt{\overline{\alpha}_t} \cdot z_0 + \sqrt{1 - \overline{\alpha}_t} \cdot \varepsilon, \quad \varepsilon \sim N(0, I), \tag{3.7}$$

где гиперпараметр силы $\tau = 0,55$ задает ограничение на глубину зашумления, а $\overline{\alpha}_t$ — кумулятивный параметр расписания дисперсий шума для шага t. Инициализация обратной диффузии из состояния z_t ограничивает глубину преобразования и сохраняет общую макрогеометрию композита, направляя генеративное воздействие преимущественно на зону стыка фрагментов. Процесс шумоподавления выполняется по методу бесклассификаторного управления (формула (2.5)) при коэффициенте управления $s = 2,5$, числе шагов 50, и завершается декодированием латентного представления обратно в пространство изображений:

$$X_{gen} = D_{VAE}(z_0^{pred}), \tag{3.8}$$

Визуальное условие e, вычисленное по самому композиту, удерживает обратный процесс в семантике окружающей ткани, благодаря чему регенерируемая зона сопряжения согласуется с контекстом. Меньшие значения параметра τ сохраняют композит почти без изменений, но слабее сглаживают границу стыка, тогда как большие усиливают гармонизацию ценой риска исказить морфологию объекта.

Морфологическая коррекция границ. Чтобы извлечь из результата диффузии только полезную работу по бесшовному слиянию фрагментов и не допустить

искажения внутренних областей донора и фона, строится гладкая приграничная карта воздействия на основе преобразования расстояния. Вычисляются поля расстояний до границы маски изнутри и снаружи:

$$d_{in} = DT(M'), \quad d_{out} = DT(1 - M'), \tag{3.9}$$

где $DT(\cdot)$ — преобразование расстояния (для каждого пикселя вычисляется евклидово расстояние до ближайшего нулевого элемента аргумента). Карта воздействия задается произведением двух функций Гаусса от этих расстояний с последующей нормировкой к диапазону [0, 1]:

$$A = exp\left(-\frac{d_{out}^2}{2\sigma^2}\right) \cdot exp\left(-\frac{d_{in}^2}{2\sigma^2}\right), \quad A \leftarrow \frac{A}{max\ A}, \tag{3.10}$$

с характерным масштабом $\sigma = 15$ пикселей. Значения карты A максимальны непосредственно на самой границе, где оба расстояния близки к нулю, и экспоненциально затухают вглубь как донора, так и фона. Итоговое изображение вычисляется как взвешенная комбинация композита и результата диффузии:

$$X_{final} = (1 - A) \odot X_{comp} + A \odot X_{gen} \tag{3.11}$$

Поскольку вдали от границы карта A обращается в нуль, результат диффузии замещает композит лишь в узкой полосе шириной порядка σ вокруг шва, тогда как внутренняя часть донора и фон вдали от границы воспроизводятся композитом без изменений, что и обеспечивает заявленные в разделе 2.3 сохранение морфологии и неизменность фона. Весь конвейер компактно записывается единым оператором граничного смешивания $\mathcal{B}$:

$$X_{final} = \mathcal{B}(G_\theta(E_{vae}(X_{comp}),\ E_{UNI}(X_{comp})),\ X_{comp},\ M') \tag{3.12}$$

Псевдокод алгоритма представлен в листинге 3.2. В псевдокоде опущена часть функционала, связанная с подготовкой фона, и предполагается, что фоновое изображение подготовлено отдельно.

Листинг 3.2 — Псевдокод алгоритма контекстно-ориентированного встраивания объектов (режим подготовки фона опущен)

```
Входные данные:
        T           — фон;
        D           — объект-донор;
        M           — бинарная маска объекта.
Параметры:
        τ = 0,55
        λ = 0,85
        s = 2,5
        σ = 15
Выходные данные:
        X_FINAL     — изображение со встроенным объектом;
        M'          — преобразованная маска объекта.
```

```
# Этап 1: Пространственное и хроматическое выравнивание
φ_T ← ОценкаОриентации(T);
φ_D ← ОценкаОриентации(D, M)
(D', M') ← ГеометрическоеПреобразование(D, M, φ_T, φ_D)
D'       ← ХроматическоеВыравнивание(D', T, M', λ)  # CIELAB
X_COMP   ← (1 − M') ⊙ T + M' ⊙ D'                    # Формирование композита

# Этап 2: Частичная диффузионная генерация
e    ← E_UNI(X_COMP)
z_0  ← E_VAE(X_COMP)
t    ← ⌊τ · t_MAX⌋
z_t  ← √ā_t · z_0 + √(1 − ā_t) · ε                   # Зашумление композита
z_0^pred ← ОбратнаяДиффузия(z_t, e, s, t)            # Обратная диффузия
X_GEN ← D_VAE(z_0^pred)

# Этап 3: Морфологическое смешивание
d_IN  ← ПреобразованиеРасстояний(M')
d_OUT ← ПреобразованиеРасстояний(1 − M')
A     ← ВычислениеВесовойКарты(d_IN , d_OUT, σ)
X_FINAL ← (1 − A) ⊙ X_COMP + A ⊙ X_GEN               # Локальное слияние
вернуть X_FINAL, M'
```

В режиме реального фона фрагмент выбирается из имеющихся реальных изображений. При недостатке подходящих реальных фрагментов задействуется режим синтетического фона, реализующий контекстное масштабирование реального тайла малого размера $x_{tile} \in \mathbb{R}^{3\times256\times256}$ до полного целевого 1024×1024. Из тайла формируется набор его вариаций с поворотом $\phi = \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$, для каждого извлекается эмбеддинг $e^{(\phi)} = E_{UNI}(\mathcal{R}_\phi(x_{tile}))$, а их конкатенация образует плотное условие:

$$e_{clean} = Concat(e^{(0)}, e^{(90)}, e^{(180)}, e^{(270)}) \in \mathbb{R}^{1\times16\times1536} \quad (3.13)$$

под управлением которого модель G_θ выполняет полный обратный диффузионный процесс из чистого гауссова шума, порождая синтетический фон T_{syn}. По размерности (16 строк) такое условие эквивалентно матрице условия для реального фрагмента 1024×1024, что позволяет получить репрезентативный фоновый контекст из одного реального тайла малого размера.

Программная реализация этапов опирается на компоненты генеративного контура (раздел 3.1): частичное зашумление и шумоподавление выполняются через планировщик шума библиотеки diffusers [53] в режиме “изображение-в-изображение”; преобразование в пространство CIELAB и обратно, а также оценка ориентации ядер и преобразование расстояния осуществляются средствами библиотеки обработки изображений OpenCV. Алгоритм возвращает не только синтезированное изображение, но и бинарную маску вставки M', в результате чего формируются синтетические пары (изображение, маска), что делает наборы синтезированных изображений пригодными для обучения моделей сегментации. Подбор фонов, ориентаций и положений донора, а также реализаций стохастического процесса порождает разнообразные, но морфологически достоверные образцы редкого класса, чем и достигается заявленная в разделе 2.3 цель повышения разнообразия минорного класса.

Резюмируем ключевые свойства, обеспечиваемые алгоритмом, и их формальные математические основания. Морфологическая достоверность редкой структуры гарантируется тем, что целевой объект формируется путем композиции из реального биологического фрагмента (3.6) и подвергается лишь частичному зашумлению (3.7), а не генерируется моделью случайным образом с нуля. Кроме того, весовая карта воздействия A экспоненциально затухает вглубь донора, сохраняя его внутреннюю макроструктуру нетронутой. Неизменность фонового контекста вне зоны встраивания математически следует из того факта, что на достаточном удалении от контура вставки функция A также обращается в нуль.

Согласно уравнению (3.11), итоговое изображение в этих областях строго совпадает с начальным композитом, а, следовательно, и с исходным фоном. Плавность и бесшовность границ сопряжения достигается комплексным применением геометрического выравнивания ориентации ядер, хроматического согласования (3.5) и диффузионной гармонизации зоны стыка, строго локализованной при помощи карты границ (3.9)–(3.10). Таким образом, каждое из заявленных в разделе 2.3 свойств опирается на конкретный элемент математической формализации, что делает поведение алгоритма полностью предсказуемым и теоретически обоснованным.

3.3 Аппаратно-программная оптимизация и интеграция с конвейерами прикладного анализа

Подсистема оптимизации проектировалась как слой, не меняющий математическую постановку метода. Каждая оптимизация подключается через отдельный программный флаг и имеет обратимый характер. Например, отключение кэширования возвращает конвейер к классическому варианту чтения исходных изображений с диска, отключение KV-компрессии возвращает механизм самовнимания к исходному виду, а отключение JIT-компиляции (Just-In-Time) к стандартному режиму работы PyTorch. Такая декомпозиция необходима для корректной абляционной оценки, поскольку позволяет измерять вклад каждого инженерного решения независимо от остальных.

Обучение и инференс диффузионного трансформера на изображениях разрешения 1024×1024 сопряжены с двумя взаимосвязанными ограничениями. Первое ограничение связано с квадратичным рост вычислительной сложности механизма self-attention относительно числа токенов: при пространственном размере латентного представления 128×128 последовательность содержит $N = 16384$ элемента и построение матрицы внимания размера $N\times N$ доминирует как во времени вычислений, так и в потреблении видеопамяти. Второе ограничение носит конвейерный характер: при “наивной” организации обучения каждый шаг включает чтение и декодирование исходных изображений с диска,

прямой проход вариационного автокодировщика и извлечение эмбеддингов фундаментальной моделью-кодировщиком. Поскольку параметры используемых на этих этапах моделей заморожены, обработка одного и того же изображения всегда приводит к идентичному результату (с точностью до естественной стохастичности автокодировщика). Регулярное повторение этих неизменных вычислений создает “узкое место” и графический ускоритель простаивает в ожидании данных, а допустимый размер пакета (batch size) жестко ограничивается. Для их преодоления в работе применен комплекс методов аппаратно-программной оптимизации, описываемых ниже; их количественная оценка приведена в главе 4.

Парадигма латентных диффузионных моделей разделяет генеративный процесс на стадию перцептивного сжатия и стадию диффузии (раздел 2.1). Поскольку кодировщики SD3-VAE и UNI2-h заморожены, результаты их вычислений строго детерминированы и не меняются от эпохи к эпохе, что позволяет выполнить их предварительный расчет лишь один раз. Для каждого изображения заблаговременно вычисляются и сохраняются в иерархическую базу формата HDF5 [55] параметры апостериорного распределения латентного кода и эмбеддинг UNI2-h. Принципиально, что сохраняются не сэмплы латентных векторов, а пары (μ, σ) — математическое ожидание и стандартное отклонение диагонального гауссова распределения, формируемого энкодером (32 канала: 16 для μ и 16 для σ). На этапе обучения латентный код восстанавливается путем репараметризации $z = \mu + \sigma \cdot \varepsilon, \quad \varepsilon \sim N(0, \mathrm{I})$, что полностью сохраняет стохастичность исходной схемы кодирования и исключает переобучение на фиксированных сэмплах. При первом чтении с диска признаки кэшируются в RAM, благодаря чему цикл обучения освобождается от множественных обращений к дисковой подсистеме; веса VAE и UNI2-h при этом вовсе не загружаются на ускоритель, высвобождая видеопамять. Длительность разовой фазы загрузки кэша (порядка 70 секунд) пренебрежимо мала на фоне многочасового процесса обучения модели. Аналогичный офлайн-расчет признаков применяют и авторы PixArt-Σ [29].

Метод пространственного сжатия матриц ключей и значений (KV-компрессии), формализованный в подразделе 2.1.2 (формула 2.7), заимствован из работы PixArt-Σ [29] и алгоритмически восходит к идеям пирамидальных трансформеров PVT v2 [46]. Мотивация данного подхода опирается на естественную избыточность визуальных признаков: семантика токенов внутри локального окна размером $R \times R$ в значительной степени дублируется. Вследствие этого матрицы ключей K и значений V допускают пространственное сжатие без существенной потери информации, тогда как полный набор запросов Q сохраняется неизменным. Асимптотическая вычислительная сложность слоя self-attention при этом снижается с $O(N^2)$ до $O(N^2/R^2)$. Оператором сжатия выступает поканальная свертка с ядром и шагом $R = 2$, инициализированная по специальной схеме “Conv-Avg-Init“ с весовыми коэффициентами $1/R^2$. На начальных этапах обучения это алгоритмически эквивалентно пулингу с усреднением (average pooling), благодаря чему оптимизация начинается из осмысленного и неразрушающего начального состояния. При этом доля дополнительных обучаемых параметров составляет лишь 0,018% от общего объема параметров модели. Ключевой фрагмент программной реализации инициализации приведен в листинге 3.3, а полная версия исходного кода вынесена в приложение Г.

Листинг 3.3 — Инициализация оператора KV-компрессии (Conv-Avg-Init)

```
# поканальная (depthwise) свертка для пространственного сжатия матриц K и V,
# инициализированная как усредняющий пулинг

self.sr = nn.Conv2d(
    hidden_size, hidden_size,
    kernel_size=sr_ratio, stride=sr_ratio,
    groups=hidden_size,                          # depthwise: канал-в-канал
)
self.sr.weight.data.fill_(1.0 / sr_ratio ** 2) # Conv-Avg-Init: w = 1 / R²

self.sr.bias.data.zero_()
```

Механизм torch.compile, появившийся в PyTorch 2.x, позволяет выполнять динамическую компиляцию модели без изменения ее исходного кода [47]. Подсистема TorchDynamo (JIT-компилятор уровня Python) извлекает

последовательности операций в граф вычислений посредством механизма перехвата исполнения фреймов (Frame Evaluation Hooks) интерпретатора CPython. Компонент AOTAutograd заблаговременно трассирует обратный проход, а компиляторный бэкенд TorchInductor транслирует полученный граф в высокооптимизированные вычислительные ядра (kernels) на языке Triton, адаптированные для графического ускорителя [48]. Ключевым источником ускорения выступает слияние цепочек поэлементных операций в единые ядра, что сокращает число обращений к глобальной памяти ускорителя и снижает накладные расходы на их запуск. Согласно данным разработчиков, на выборке из более чем 180 реальных архитектур применение бэкенда TorchInductor обеспечивает среднее геометрическое ускорение в 1,41× в режиме обучения на графических процессорах NVIDIA A100 [48]. Вычислительной платой за данную оптимизацию служит разовая компиляция графа на первых итерациях, требующая от десятков секунд до нескольких минут.

Как отмечалось ранее, стандартная реализация self-attention требует вычисления промежуточной матрицы совместимостей QK^T размера $N \times N$ в глобальной памяти ускорителя. Для последовательности из N=16384 токенов это приводит к выделению сотен мегабайт видеопамяти на каждую голову self-attention в каждом слое (для хранения промежуточных активаций, необходимых при обратном проходе). Алгоритм FlashAttention устраняет необходимость явного хранения данной матрицы: внимание вычисляется математически точно, но с учетом аппаратной иерархии памяти GPU. Метод поблочного вычисления минимизирует число операций чтения и записи между высокопропускной глобальной памятью (HBM) и быстрой внутрикристальной памятью (SRAM). Функция softmax вычисляется поблочно с потоковой нормализацией, а требуемые для обратного прохода градиенты пересчитываются динамически [49]. Несмотря на введение дополнительных арифметических операций, метод демонстрирует более высокую общую скорость за счет кратного снижения нагрузки на шину HBM, обеспечивая ускорение до 3× и линейную зависимость потребления памяти от длины последовательности. В программном

комплексе данный класс ядер подключается посредством библиотеки xformers [50]. Примечательно, что внутренняя реализация поканальной свертки (KV-компрессии) использует совместимые примитивы эффективного управления памятью, благодаря чему оба метода оптимизации комбинируются без конфликтов.

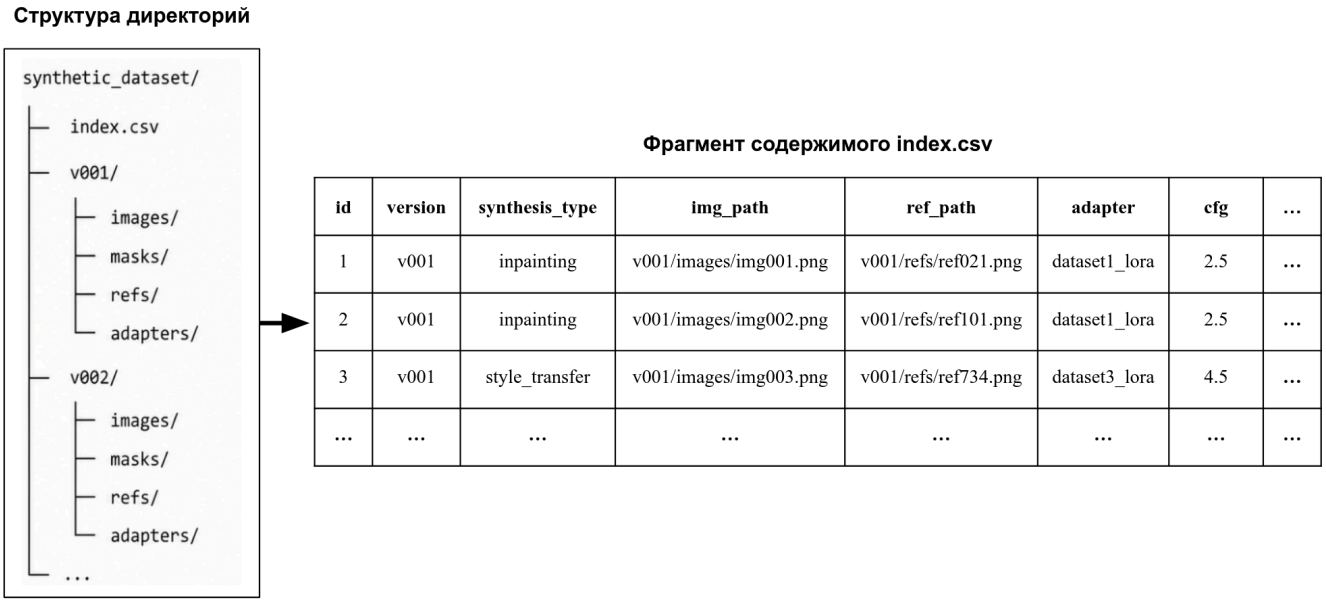
Выполнение прямого и обратного проходов над параметрами, активациями и градиентами осуществляется в формате половинной точности FP16, тогда как обновление весовых коэффициентов производится в формате FP32 [51]. На тензорных вычислительных ядрах микроархитектуры Volta и новее операции матричного умножения в формате FP16 обладают кратно большей пиковой производительностью по сравнению с FP32. Это особенно эффективно для архитектур на базе трансформеров, где доминируют матричные умножения, при этом экономия видеопамяти составляет порядка 50%. Узкий динамический диапазон формата FP16 требует применения механизма масштабирования функции потерь (loss scaling), в рамках которого значения функции потерь умножаются на коэффициент S перед этапом обратного распространения ошибки, что выводит малые градиенты из зоны потери значимости (денормализованных чисел). Перед обновлением весов модели градиенты делятся на этот же коэффициент, причем значение S подстраивается динамически при обнаружении численных переполнений. Управление вычислениями со смешанной точностью делегировано программной библиотеке Accelerate [54], инкапсулирующей логику масштабирования градиентов и безопасного приведения типов.

Перечисленные методы компонуются без взаимных конфликтов: предварительное вычисление признаков, KV-компрессия, JIT-компиляция, память-эффективные ядра и смешанная точность образуют кумулятивную конфигурацию, последовательное включение элементов которой и оценивается в главе 4. Принятая методика измерений наряду с количественными результатами, подробно описана в разделе 4.2.

Оптимизированный генеративный контур интегрируется в цикл прикладных задач через функциональный контур прикладных задач (раздел 3.1) и

специализированную библиотеку huggeron. Результаты обоих сценариев синтеза автоматически преобразуются в структурированные наборы данных, совместимые с конвейерами обучения моделей. Тем самым формируется единый замкнутый технологический цикл: предварительное вычисление признаков, управляемый синтез, формирование аннотированной обучающей выборки и обучение прикладной модели. Синтетические наборы используются для целенаправленного обогащения обучающих данных в задачах сегментации и классификации. Детальные параметры объемов выборок и архитектуры прикладных моделей (сверточная нейронная сеть семейства U-Net для сегментации сосудистых структур и специализированный классификатор) подробно рассматриваются в главе 4. Подобная интеграция преобразует генеративную систему из автономного инструмента синтеза в органичный компонент прикладного конвейера анализа гигапиксельных изображений. Это полностью отвечает инженерной постановке задачи целенаправленного расширения данных и логически завершает описание программного комплекса.

На уровне файловой структуры результаты синтеза сохраняются не как набор несвязанных изображений, а как датасет с индексным файлом и версионированием (рисунок 3.3).



id	version	synthesis_type	img_path	ref_path	adapter	cfg	…
1	v001	inpainting	v001/images/img001.png	v001/refs/ref021.png	dataset1_lora	2.5	…
2	v001	inpainting	v001/images/img002.png	v001/refs/ref101.png	dataset1_lora	2.5	…
3	v001	style_transfer	v001/images/img003.png	v001/refs/ref734.png	dataset3_lora	4.5	…
…	…	…	…	…	…	…	…

Рисунок 3.3 — Формат хранения синтезированного набора данных

В индекс включаются тип синтеза, путь к изображению, путь к маске, исходный референс, использованный адаптер, коэффициент CFG, сила зашумления, координаты вставки и контрольные суммы входных объектов. Это обеспечивает трассируемость синтетических примеров, когда любой образец, попавший в обучающую выборку прикладной модели, может быть связан с конкретным запуском генеративного конвейера и параметрами его получения.

Выводы по третьей главе

В третьей главе методы, формализованные в главе 2, доведены до уровня воспроизводимой программной реализации: разработаны архитектура, алгоритмическое обеспечение и средства оптимизации генеративного комплекса.

Спроектирована модульная программная архитектура, состоящая из четырех независимых функциональных контуров ввода-вывода, латентно-признакового кодирования, генеративного ядра и прикладных задач, взаимодействующих через фиксированные интерфейсы. Такая декомпозиция обеспечивает слабую связанность компонентов, асинхронное исполнение вычислительного конвейера. Оригинальные алгоритмические решения инкапсулированы в пределах генеративного контура посредством модификации абстракции `DiffusionPipeline`, что гарантирует совместимость со стандартизированными компонентами экосистемы Hugging Face.

Дано детальное алгоритмическое описание двух прикладных сценариев синтеза. Метод доменной специализации формализован как двухфазная процедура, где на фазе предварительной подготовки выполняется прогрессивная настройка базовой модели и обучение библиотеки низкоранговых адаптеров, а на фазе применения осуществляется перехват слоев перекрестного внимания с разделением потоков пространственного управления и коррекции стиля. Алгоритм контекстно-ориентированного встраивания объектов представлен в виде трехэтапного конвейера, включающего пространственно-хроматическое выравнивание, частичную диффузионную генерацию и морфологическую коррекцию границ. Логика обоих алгоритмов представлена в виде схем (рисунки 3.2 и 3.3) и псевдокода (листинги 3.1 и 3.2).

Описан комплекс методов аппаратно-программной оптимизации: предварительное вычисление и кэширование латентных представлений и эмбеддингов, пространственная KV-компрессия, JIT-компиляция вычислительного графа, память-эффективные ядра внимания и вычисления со смешанной точностью. Показано, что методы компонуются без взаимных конфликтов и в совокупности снижают вычислительные ограничения синтеза изображений высокого разрешения.

Глава 4. Экспериментальная оценка метода и программных средств

В настоящей главе приводится экспериментальная верификация разработанных метода, алгоритмов и программных средств по трем ключевым направлениям: вычислительная эффективность методов оптимизации, морфологическая достоверность синтезированных изображений и оценка влияния синтетических данных на качество обучения прикладных нейросетевых моделей компьютерного зрения. Данные направления последовательно обосновывают заявленные результаты исследования: оценка вычислительной эффективности подтверждает состоятельность инженерных решений (глава 3), морфологическая достоверность подтверждает корректность математического метода и алгоритмов (главы 2 и 3), а повышение метрик качества у прикладных моделей — практическую полезность всего комплекса (глава 1). Таким образом, экспериментальное исследование обеспечивает комплексную верификацию предложенного решения.

Раздел 4.1 описывает общую методологию проводимых экспериментов, а раздел 4.2 содержит результаты оценки эффективности методов аппаратно-программной оптимизации, описанных в разделе 3.3. Раздел 4.3 посвящен оценке качества синтеза и полезности порождаемых данных в двух задачах цифровой патологии: семантической сегментации кровеносных сосудов и выявлении сосудистой инвазии (задача классификации).

4.1. Методология оценки

В разделе описана методология экспериментальной оценки и фиксируются условия, общие для всех приводимых далее результатов. Описываются вычислительный стенд и тестовые выборки гистологических изображений (подраздел 4.1.1), а также система метрик, по которым оцениваются вычислительная эффективность, морфологическая достоверность синтеза и его прикладная полезность. Единство аппаратно-программной среды и метрик обеспечивает сопоставимость результатов разделов 4.2 и 4.3.

4.1.1. Описание тестовых стендов и характеристик исходных датасетов

Вычислительный конвейер развернут на аппаратном комплексе, в состав которого входит один графический ускоритель Nvidia A100 с поддержкой архитектуры параллельных вычислений. Компоненты конвейера реализованы с использованием актуальных версий библиотек CUDA и PyTorch. Базовая генеративная модель PixCell предварительно обучена на многодоменном корпусе из 69184 WSI, охватывающем 28 текстурно-графических доменов. Единая аппаратно-программная среда сохраняется во всех экспериментах, что обеспечивает сопоставимость результатов.

Обучение и оценка проводились на трех независимых выборках WSI с выраженным дефицитом аннотаций (таблица 4). Данные всех коллекций гигапиксельных изображений получены из различных, непересекающихся источников и различаются условиями подготовки и оцифровки; такая разнородность порождает междоменный сдвиг, типичный для цифровой патологии (подраздел 1.1.2). Совокупно из 449 WSI размечено лишь 207 — этот дисбаланс и определяет задачу, на которую направлены предложенные алгоритмы.

Таблица 4 — Состав исходных клинических выборок

Выборка	**Аннотированных WSI**	**Всего WSI**
DHMC[56]	102	143
LCNOV	54	161
NLST[58]	51	145
Всего	207	449

Все WSI обрабатывались по единому протоколу: из каждого слайда на фиксированном уровне увеличения извлекались тайлы размером 1024×1024 пикселя (служат элементарной единицей как при обучении генеративной модели, так и при формировании прикладных выборок) с аннотированными сосудами. Для задачи классификации фрагментов сосудов (классификация ЛВИ) все сосуды дополнительно дробились на фрагменты размером 256×256 пикселей. Единство

протокола на всех этапах устраняет ещё один источник рассогласования и обеспечивает корректность сквозного сравнения конфигураций.

Нарезка обучающих наборов для диффузионной модели выполнялась на 242 неразмеченных WSI, не пересекающихся с данными прикладных задач: из данных размеченной части (207 WSI) сформированы обучающие выборки для прикладных задач, а из неразмеченной части данных (242 WSI) формировались выборки для обучения генеративной модели. Тем самым PixCell не обучается на слайдах, из которых берется контрольная выборка, что исключает утечку между синтезом и оценкой прикладных моделей. Прогрессивное дообучение PixCell (подраздел 2.2.1) проводилось последовательно на трех разрешениях; объем данных закономерно убывает с ростом размера тайла: 112921 тайлов (256×256 пикселей) для PixCell-256, 28228 тайлов (512×512 пикселей) для PixCell-512 и 7058 тайлов (1024×1024 пикселей) для PixCell-1024. Каждый LoRA-адаптер (подраздел 2.2.2) обучался на 10000 тайлах размера 1024×1024, извлеченных из неразмеченных WSI соответствующего набора данных (см. таблицу 5). Выборки данных этапа дообучения и обучения LoRA-адаптеров также не пересекаются друг с другом. Оптимизация параметров диффузионной модели базировалась на минимизации среднеквадратичной ошибки (MSE) между добавленным и предсказанным шумом. Обновление весов осуществлялось алгоритмом AdamW с постоянным шагом (constant scheduler). Скорость обучения (learning rate) на этапе прогрессивного дообучения составляла 1e-6, а при настройке LoRA-адаптеров — 1e-4. Расписание дисперсии для прямого и обратного диффузионных процессов задавалось алгоритмом DDPMScheduler.

Разделение на обучающую, валидационную и контрольную части выполнялось на уровне слайдов (WSI), а не тайлов. Такой подход принципиален поскольку тайлы одного слайда коррелированы, и их попадание в разные части выборки может вызывать утечку признаков, при которой модель “узнаёт” контрольные фрагменты по смежным обучающим и, как следствие, завышает метрики. Контрольная (тестовая) часть в задачах классификации и сегментации фиксировалась один раз и оставалась неизменной между разбиениями. Обучение

моделей проводилось в режиме 10-кратной кросс-валидации с пропорцией $train\,/\,val\,/\,test\;=\;0.70\,/\,0.15\,/\,0.15$. Синтетические данные добавлялись только в обучающую часть каждого разбиения, тогда как валидация и контроль состояли исключительно из реальных изображений. Оценка по 10 разбиениям дает средние значения метрик и их разброс, позволяя отделить устойчивый эффект расширения выборки от случайных колебаний в различных подвыборках.

Исходная разметка включала 8212 кровеносных сосудов на 207 WSI. Объемы реальных и синтетических данных по обеим задачам приведены в таблице 5. По маскам сосудов сформирована выборка из 5764 реальных изображений, дополненная 5172 синтетическими изображениями в качестве регуляризирующего дополнения.

Из 8212 сосудов признаки инвазии имели лишь 216 (против 7 996 “чистых”) — экстремальный классовый дисбаланс. Чтобы нарастить объем минорного класса, анализ переведен с целых сосудов на их локальные фрагменты размером 256×256 пикселей. Таким образом, на базе 216 инвазивных сосудов было получено 573 фрагмента с признаками инвазии, которые вместе с 1302 фрагментами здоровых сосудов образовали реальную выборку из 1875 изображений. Для балансировки классов дополнительно сгенерирован набор из 4520 синтетических фрагментов с разбиением по классам: 2256 инвазивных и 2264 “чистых”.

Таблица 5 — Объем обучающих выборок по задачам

Этап	**Задача**	**Реальные изображения**	**Синтетические изображения**
Обучение генеративной модели	Дообучение: 256	112921	-
	Дообучение: 512	28228	-
	Дообучение: 1024	7058	-
	Обучение LoRA	10000	-
Обучение прикладных моделей	Сегментация сосудов	5764	5172
	Классификация инвазии	1875	4520

4.1.2. Система метрик для оценки вычислительной эффективности и качества синтеза

Оценка генеративной модели — многомерная задача, для объективного решения которой недостаточно опоры на единственную метрику. Для всесторонней оценки сформирован набор показателей, покрывающий четыре взаимодополняющих аспекта качества: согласованность распределений данных, сохранение доменно-специфичной семантики, локальное разнообразие с реализмом отдельных образцов, а также уникальность генерируемых изображений. Совместный анализ этих аспектов позволяет выявлять специфические дефекты генерации — от размытия мелких деталей до «коллапса мод» и запоминания обучающей выборки.

Принципиальной особенностью методологии является вычисление метрик в признаковых пространствах нескольких различных моделей-кодировщиков. Признаковое пространство кодировщика определяет, какие свойства изображения метрика “замечает”, а какие оставляет без внимания, поэтому оценка в единственном пространстве неизбежно односторонняя. В работе применяются кодировщики двух типов: универсальный кодировщик общего назначения Inception-v3 [64], отражающий реализм в общевизуальном смысле, и фундаментальные модели-кодировщики гистологических изображений Virchow-2 [32] и UNI2-h [9], чувствительные к доменно-специфичным морфологическим признакам. Такое сочетание позволяет оценить достоверность синтеза одновременно с общей и с предметной точек зрения. Это особенно важно для гистологии, где визуально правдоподобное изображение может оказаться некорректным с точки зрения морфологии.

Согласованность распределений реальных и синтетических данных измеряется расстоянием Фреше (Fréchet Inception Distance, FID) и ядерным расстоянием (Kernel Inception Distance, KID) в пространстве признаков Inception-v3 [59, 60]. Расстояние Фреше аппроксимирует каждое из двух множеств признаков многомерным нормальным распределением и вычисляет расстояние между этими распределениями по их средним значениям и ковариациям; чем

ближе распределение синтетических изображений к распределению реальных, тем меньше значение метрики. Кернельное расстояние оценивает ту же близость через ядровую меру расхождения распределений и не опирается на предположение о нормальности. Совместное использование метрик оправдано тем, что ядерная оценка несмещена на выборках ограниченного объема, что делает ее более устойчивой в условиях дефицита данных, характерного для рассматриваемых выборок. Дополнительно вычисляется расстояние Фреше в признаковом пространстве CLIP (метрика CLIP-FID в признаковом пространстве модели CLIP ViT-B/32) [61, 62]. В отличие от Inception-v3, кодировщик CLIP чувствителен прежде всего к согласованности макроструктур, а не только к локальной текстуре, что делает CLIP-FID индикатором структурной согласованности взаимного расположения макроструктур ткани. Для всех трех мер меньшее значение соответствует лучшему качеству.

Локальное разнообразие и реалистичность отдельных образцов оцениваются метриками точности (Precision) и полноты (Recall), рассчитываемыми на основе оценки многообразий данных методом k ближайших соседей [63]. Идея метода состоит в аппроксимации многообразия (области, занимаемой данными в признаковом пространстве) для реальной и синтетической выборок и в измерении их взаимного перекрытия. Метрика Precision характеризует долю синтетических образцов, попадающих в многообразие реальных данных, и потому отражает реализм: высокое значение означает, что синтетические изображения неотличимы от правдоподобных реальных. Метрика Recall характеризует долю реальных образцов, покрываемых многообразием синтетических данных, и потому отражает разнообразие: высокое значение означает, что модель воспроизводит весь спектр вариаций реальных данных, а не их узкое подмножество. Заметное снижение Recall служит индикатором “коллапса мод” — патологического сужения разнообразия, при котором модель порождает ограниченный набор похожих образцов. Для обеих мер большее значение соответствует лучшему качеству.

Сохранение доменно-специфичной семантики, определяемое как соответствие синтетического образца заданному референсному изображению (подраздел 3.2.1), оценивается мерой косинусного сходства (Cosine Similarity). Эта метрика измеряет угол между векторами признаков (эмбеддингами) и позволяет количественно охарактеризовать точность следования морфологическому условию. Вычисление метрики выполняется в пространстве независимого кодировщика Virchow-2 [32], не участвующего в обусловливании генеративной модели, что исключает методологическую предвзятость при расчете.

Степень уникальности синтезированных изображений оценивается метрикой расстояния до ближайшего соседа (Nearest Neighbour Distance, NND). Для каждого сгенерированного образца вычисляется расстояние до наиболее похожего на него реального изображения — так формируется показатель “synth→real”. Полученная средняя величина удаленности интерпретируется относительно эталона — типичного внутреннего расстояния между группами, состоящими из реальных изображений внутри обучающей выборки (“real→real”). Близость значения «synth→real» к значению «real→real» свидетельствует об отсутствии прямого копирования обучающих паттернов, поскольку синтетические образцы отстоят от реальных не ближе, чем реальные образцы естественно отстоят друг от друга. Напротив, существенно меньшее расстояние указывало бы на то, что алгоритм просто запомнил и воспроизводит обучающую выборку. Такой дефект обесценивает генерацию, поскольку синтезированные данные перестают быть источником новизны.

Помимо прямой оценки качества генерации, практическая ценность синтетических данных оценивается опосредованно — по результатам обучения прикладных нейросетевых моделей, ради улучшения которых синтез и выполняется. Качество модели сегментации кровеносных сосудов (UNet++ [65]) характеризуется тремя метриками. Индекс пересечения по объединению (IoU) измеряет долю совпадения предсказанной и эталонной областей сосуда и служит интегральным показателем точности сегментации; чувствительность (Sensitivity) отражает полноту обнаружения истинных сосудистых структур, а специфичность

(Specificity) — корректность отнесения фоновых пикселей к фону. Качество модели классификации участков инвазии (EfficientNet-b1 [66]) характеризуется F1-мерой — гармоническим средним точности и полноты, устойчивым к дисбалансу классов, — а также чувствительностью и специфичностью, раздельно описывающими способность модели выявлять инвазию и отвергать ее отсутствие. Совместное рассмотрение чувствительности и специфичности значимо в прикладном контексте, где цена пропуска события и цена ложного срабатывания различны.

Вычислительная эффективность алгоритмов контролируется тремя показателями. Пропускная способность (throughput), измеряемая числом обработанных изображений в секунду, отражает скорость обучения и инференса. Длительность эпохи обучения характеризует совокупные временные затраты на полный проход по обучающей выборке. Пиковый объем используемой видеопамяти определяет требования к оборудованию и косвенно — допустимый размер мини-партии. Совместный учет времени выполнения и памяти критически важен: оптимизация, ускоряющая вычисления, но резко повышающая пиковую память, может оказаться непригодной для практического конвейера. В совокупности эти показатели характеризуют степень полезного использования вычислительных ресурсов и применяются в разделе 4.2 для оценки эффекта от использования методов оптимизации.

В совокупности описанная система метрик реализует принцип взаимной перекрестной проверки, где ни один аспект качества не оценивается изолированно, а характерные дефекты генерации (недостаток реализма, сужение разнообразия, отклонение от заданной семантики и запоминание обучающих данных) обнаруживаются независимыми группами показателей. Такая избыточность оценки повышает доверие к итоговому заключению о качестве синтеза и снижает риск переоценки модели по одному выгодному показателю. Конкретные значения всех перечисленных метрик и их интерпретация применительно к разработанным алгоритмам приведены в разделе 4.3.

4.2. Результаты оценки эффективности оптимизаций и производительности программного обеспечения

В данном разделе оценивается вклад методов оптимизации (раздел 3.3) в вычислительную эффективность конвейера обучения. Измерения выполнялись по принципу кумулятивного наслаивания: методы внедрялись последовательно, образуя цепочку конфигураций. Для обеспечения корректности при сравнении результатов каждая конфигурация тестировалась в изолированном процессе при неизменных внешних условиях (разрешение 1024×1024 пикселей, единый аппаратный стенд, отсутствие других вычислительных задач в системе). Метрики фиксировались в установившемся режиме после фазы прогрева, что исключило искажение результатов за счет разовых издержек на инициализацию и компиляцию вычислительного графа. Чтобы нивелировать случайные колебания времени на отдельных итерациях, пропускная способность θ усреднялась по фиксированному окну наблюдений:

$$\theta = B \cdot \frac{n}{T}, \qquad (4.1)$$

где B — размер мини-партии, n — количество последовательных шагов оптимизации, образующих окно, а T — суммарное время, затраченное на выполнение этих n шагов. Результаты замеров представлены в таблице 6.

Таблица 6 — Оценка вычислительной эффективности методов оптимизации

Конфигурация	**Длительность эпохи, мин**	**Пиковое VRAM, ГБ**	**Пропускная способность, изобр/с**	**Ускорение**
baseline	1459,4	48,20	0,28	×1,0
+ precompute	1174,4	45,19	0,35	×1,24
+ kv-compression	1009,2	46,18	0,41	×1,45
+ JIT + xformers + fp16	161,2	32,65	2,56	×9,06

Базовый режим (baseline) соответствует прямолинейной реализации конвейера, при которой признаки кодировщиков SD3-VAE и UNI2-h извлекаются из изображений заново (“на лету”) на каждом шаге обучения. Переход от этого режима к стратегии предварительного вычисления признаков (+ precompute), в рамках которой предварительно вычисленные вектора признаков единожды загружаются с диска в RAM, позволил сократить время обучения на одну эпоху на 19,5% и увеличить пропускную способность в 1,24 раза. Такое ускорение достигается за счет устранения узкого места в виде множественных обращений к дисковой подсистеме ввода-вывода. Отметим, что кэшируются не сами латентные векторы, а параметры их распределения, из которых тензоры восстанавливаются путем репараметризации на каждом шаге (подраздел 3.3). Это позволяет ускорить вычисления, сохраняя стохастичность кодирования и не допуская обеднения обучающего сигнала.

Последующее внедрение метода KV-компрессии дополнительно сократило время обучения на 14%, доведя совокупное ускорение до 1,45 раза. Наблюдаемый при этом незначительный рост пикового потребления видеопамяти (с 45,19 до 46,18 ГБ) связан с добавлением параметров оператора сжатия (подраздел 2.1.2). Оптимизации precompute и kv-compression повышают эффективность исключительно за счет реорганизации потоков данных и математического сжатия, не меняя низкоуровневый режим работы ускорителя.

Радикальный прирост производительности достигнут за счет комплексного применения JIT-компиляции графа (torch.compile), оптимизированных ядер внимания (xformers) и вычислений со смешанной точностью (FP16). Данная конфигурация в совокупности с precompute и kv-compression обеспечила суммарное ускорение расчетов в 9,06 раза при снижении потребления видеопамяти на ~32% — с 48,20 до 32,65 ГБ относительно базовой конфигурации. Расчетная длительность эпохи сократилась с 24,3 до 2,7 часа.

Дополнительным эффектом перехода на смешанную точность стало двукратное сокращение объема кэша, используемого для хранения

предварительно вычисленных векторов признаков в оперативной памяти — с 99 до 50 ГБ.

Применение JIT-компиляции демонстрирует высокую эффективность именно в контексте циклически повторяющегося вычислительного графа диффузионных моделей. Поскольку топология операций на каждом шаге шумоподавления остается неизменной, а меняются исключительно значения тензоров и индекс шага, транслятор получает возможность выполнить глубокую низкоуровневую оптимизацию.

Кумулятивный характер достигнутого ускорения объясняется тем, что каждый метод оптимизирует свое, независимое узкое место конвейера. Предварительное вычисление признаков снимает ограничения дисковой подсистемы, KV-компрессия снижает квадратичную вычислительную сложность механизма self-attention, JIT-компиляция минимизирует накладные расходы на запуск ядер, xformers оптимизирует трафик видеопамяти, а использование смешанной точности позволяет задействовать тензорные ядра ускорителя. Поскольку эти подходы не конкурируют за одни и те же ресурсы, их эффекты успешно суммируются.

Внедрение описанного комплекса методов позволяет существенно смягчить аппаратные ограничения, характерные для синтеза изображений высокого разрешения. На системном уровне обучение переводится из режима, лимитированного скоростью дискового ввода-вывода, в состояние высокоэффективной утилизации ресурсов графического ускорителя. Достигнутое при этом высвобождение резервов видеопамяти открывает возможность увеличить размер мини-партии для повышения стабильности обучения. С практической точки зрения сокращение длительности эпохи более чем в девять раз качественно меняет рабочий процесс: эксперименты, ранее требовавшие около суток вычислений, теперь укладываются в несколько часов. Это позволяет провести существенно больше итераций подбора гиперпараметров и конфигураций в пределах фиксированного вычислительного бюджета. Таким образом, эффект оптимизаций не сводится к тривиальной экономии времени, а

принципиально расширяет пространство практически реализуемых экспериментов на доступном оборудовании, что становится критически важным фактором при работе на единственном графическом ускорителе.

4.3. Апробация результатов на задачах цифровой патологии (сквозной эксперимент)

Раздел объединяет две группы экспериментов: прямую оценку качества генеративной модели (подраздел 4.1.2) и опосредованную оценку практической ценности синтетических данных. Двухуровневая верификация взаимно дополняет друг друга: прямые метрики характеризуют реализм, разнообразие и оригинальность самих изображений, тогда как прикладной эксперимент отвечает на главный для инженерной задачи вопрос о том, приносит ли синтез изображений пользу конечным распознающим моделям. Согласованность выводов двух уровней повышает доверие к результату в целом.

Результаты оценки качества синтетических данных (таблица 7) демонстрируют высокую согласованность распределений. Значения FID (15,6866) и KID (4,9407×10−3) в пространстве Inception-v3 находятся на уровне современных генеративных моделей, а низкое значение CLIP-FID (0,7029) подтверждает достоверность пространственной организации и взаимного расположения тканевых структур. Высокие показатели точности (Precision = 0,8653) и полноты (Recall = 0,8819) свидетельствуют о реализме отдельных образцов и сохранении их разнообразия, исключая риск “коллапса мод”. Сохранение доменно-специфичной семантики подтверждается высоким косинусным сходством в независимых пространствах сразу трех фундаментальных моделей: Virchow-2 (0,8660), H-Optimus-1 (0,7960) и UNI-2h (0,8008). Согласованность результатов, полученных в признаковых пространствах как универсальных, так и специализированных моделей, подтверждает объективность и достоверность итоговой оценки.

Уникальность синтеза оценивалась метрикой NND в пространстве Inception-v3.

Таблица 7 — Оценка качества сгенерированных изображений (↓ — меньшее значение лучше, ↑ — большее лучше; KID приведен в масштабе $\times 10^3$)

Аспект качества	**Метрика**	**Пространство признаков**	**Значение метрики**
Согласованность распределений (общевизуальная)	FID ↓	Inception-v3	15.6866
	KID x10^3 ↓	Inception-v3	4.9407
Макроструктурная согласованность	CLIP-FID ↓	CLIP ViT-B/32	0.7029
Новизна (отсутствие копирования)	NND (Synth→Real)	Inception-v3	0.1324
	NND (Real→Real)	Inception-v3	0.1317
Качество выборки (реализм и разнообразие)	Precision ↑	Inception-v3	0.8653
	Recall ↑	Inception-v3	0.8819
Сохранение доменно-специфичной семантики	Cosine Similarity ↑	Virchow-2	0.8660
		H-Optimus-1	0.7960
		UNI-2h	0.8008

Среднее расстояние от синтетического изображения до ближайшего реального (0,1324) практически совпало с типичным расстоянием внутри подгрупп изображений обучающей выборки (0,1317). Это подтверждает, что алгоритм порождает новые морфологические вариации, а не дублирует выученные паттерны. Доказательство отсутствия прямого копирования принципиально для последующих этапов: оно гарантирует, что наблюдаемый прирост качества прикладных моделей обусловлен генерацией подлинно новых, достоверных данных.

Практическая ценность синтезированных данных оценивалась по эффекту от расширения обучающих выборок. Модели обучались в трех конфигурациях: базовые изображения (source-data), базовые изображения с классическими

аугментациями (source-data + aug) и выборка, дополнительно расширенная синтетическими данными (source-data + aug + synth). Классические геометрические и колориметрические аугментации не порождают новой морфологии, в отличие от алгоритмов генерации. Подобная трехступенчатая схема строго изолирует вклад предложенных методов, не позволяя ошибочно приписать итоговый прирост качества эффекту простого пространственного размножения примеров. Результаты представлены в таблицах 8 и 9 в формате “среднее значение ± стандартное отклонение”).

Таблица 8 — Качество модели сегментации сосудов (UNet++) при разных конфигурациях обучающей выборки

Конфигурация	**IoU**	**Sensitivity**	**Specificity**
source-data	0,7218 ± 0,0067	0,8778 ± 0,0062	0,8975 ± 0,0056
source-data + aug	0,7292 ± 0,0018	0,8887 ± 0,0043	0,8963 ± 0,0042
source-data + aug + synth	0,7376 ± 0,0014	0,8781 ± 0,0036	0,9094 ± 0,0034

Таблица 9 — Качество модели классификации участков инвазии (EfficientNet-b1) при разных конфигурациях обучающей выборки

Конфигурация	**F1-score**	**Sensitivity**	**Specificity**
source-data	0,9279 ± 0,0149	0,9477 ± 0,0135	0,9580 ± 0,0146
source-data + aug	0,9323 ± 0,0137	0,9616 ± 0,0216	0,9554 ± 0,0064
source-data + aug + synth	0,9577 ± 0,0090	0,9721 ± 0,0116	0,9744 ± 0,0079

Результаты эксперимента показывают, что интеграция синтетических изображений в обучающие выборки оказывает выраженный регуляризирующий эффект: растут средние значения метрик и кратно снижается их дисперсия. В задаче сегментации сосудов среднее значение IoU достигло 0,7376 при

существенном сужении доверительного интервала (стандартное отклонение снизилось с 0,0067 на базовой выборке до 0,0014). В задаче классификации инвазии зафиксирован комплексный прирост: F1-мера возросла до 0,9577 (±0,0090), а чувствительность и специфичность — до 0,9721 и 0,9744 соответственно. Поскольку класс инвазии является дефицитным (подраздел 4.1.1), этот результат подтверждает эффективность алгоритма контекстного встраивания для компенсации дисбаланса. Рост средних значений достигнут без просадки сопутствующих показателей, что говорит об отсутствии нежелательных смещений (bias). Снижение дисперсии метрик при добавлении синтетических данных имеет самостоятельное практическое значение: меньший разброс означает большую предсказуемость качества обученной модели при смене разбиения данных, что существенно для воспроизводимости результатов в реальных прикладных условиях.

Сопоставление результатов между различными конфигурациями позволяет объективно оценить относительный вклад каждого из двух источников расширения выборки. В задаче классификации переход от базовых данных (source-data) к использованию классических аугментаций (source-data + aug) дает лишь умеренное улучшение F1-меры, тогда как интеграция синтетических изображений на следующей ступени обеспечивает заметно более существенный прирост. Это наглядно указывает на то, что эффект генеративного обогащения превосходит вклад от классических аугментаций и привносит в модель новые полезные признаки, а не дублирует существующие. Аналогичная динамика наблюдается и в задаче сегментации, где основной рост специфичности и существенное сокращение разброса показателя IoU приходятся именно на конфигурацию с синтетическими данными. При этом улучшение целевых метрик носит строго монотонный характер от эксперимента к эксперименту. Подобная картина подтверждает согласованное, а не конкурирующее действие двух различных методов расширения обучающей выборки.

Выводы по четвертой главе

Экспериментальная верификация разработанных методов и алгоритмов проведена по трем ключевым направлениям, всесторонне подтвердившим итоговую эффективность комплекса.

Анализ вычислительной производительности показал, что внедренные оптимизации ускоряют обучение в 9,06 раза и снижают пиковое потребление видеопамяти на 32%. Сокращение длительности эпохи с 24,3 до 2,7 часа доказывает практическую применимость конвейера для работы с гигапиксельными изображениями даже в условиях ограниченных аппаратных ресурсов.

Оценка качества синтетических данных подтвердила высокую согласованность распределений сгенерированных и реальных данных, сохранение доменно-специфичной морфологии и локального разнообразия без проявлений “коллапса мод”, а анализ метрики NND подтвердил отсутствие прямого копирования обучающих данных моделью.

Оценка влияния синтетических данных на качество прикладных моделей сегментации сосудов и классификации инвазии продемонстрировала стабильный рост точности распознавания и снижение дисперсии целевых метрик. Поскольку наибольший эффект достигается именно для дефицитных минорных классов, совокупность результатов доказывает, что разработанный комплекс успешно решает инженерную задачу направленного обогащения выборок в условиях нехватки и дисбаланса медицинских данных.

Заключение

Основные результаты работы заключаются в следующем:

1. Разработан метод адаптивного синтеза гистологических изображений высокого разрешения, основанный на гибридном объединении архитектуры латентных диффузионных трансформеров (DiT), механизмов визуального обусловливания и алгоритмов сжатия матриц внимания (KV-компрессии). Метод формирует вычислительно эффективный и модульный архитектурный фундамент, обеспечивающий строгий контроль над морфологической достоверностью генерируемых структур и позволяющий независимо применять компоненты генеративного конвейера при обработке гигапиксельных изображений.
2. Разработан метод доменной специализации латентных диффузионных моделей, отличающийся двухэтапной декомпозицией процесса: применением прогрессивного обучения с фиксацией оптического увеличения для адаптации морфологических представлений и библиотеки низкоранговых адаптеров (LoRA) для параметрической коррекции доменного стиля. Метод позволяет масштабируемо настраивать генеративные модели под целевые наборы данных без полного переобучения базовой архитектуры и при сохранении визуально обусловленной геометрии структур, эффективно компенсируя ковариантный сдвиг.
3. Разработан алгоритм локального контекстно-ориентированного встраивания объектов, отличающийся пространственно-детерминированной бесшовной интеграцией целевого объекта в контекст фонового изображения на уровне латентных представлений с процедурой автоматического колориметрического согласования. Алгоритм обеспечивает естественное объединение структур без привлечения дополнительных обучаемых архитектурных модулей и сложной пространственной разметки.

4. Разработаны программно-алгоритмический комплекс адаптивного синтеза с модульной архитектурой и гибридная методология многофакторной оценки качества, всесторонне характеризующая статистическую согласованность, семантическую достоверность и генеративное разнообразие выборок. Экспериментально доказано, что применение разработанных решений для обогащения обучающих выборок обеспечивает повышение метрик качества прикладных нейросетевых моделей сегментации и классификации в условиях классового дисбаланса.

Список литературы

1. Timakova A., **Ananev V.**, Fayzullin A., Makarov V., Ivanova E., Shekhter A., Timashev P. Artificial intelligence assists in the detection of blood vessels in whole slide images: practical benefits for oncological pathology // Biomolecules. – 2023. – Vol. 13, No. 9. – P. 1327.
2. Timakova A., **Ananev V.**, Fayzullin A., Zemnuhov E., Rumyantsev E., Zharov A., Zharkov N., Zotova V., Shchelokova E., Demura T., Timashev P., Makarov V. LVI-PathNet: Segmentation-classification pipeline for detection of lymphovascular invasion in whole slide images of lung adenocarcinoma // Journal of Pathology Informatics. – 2024. – Vol. 15. – P. 100395.
3. Timakova A., Fayzullin A., **Ananev V.**, Zemnuhov E., Alfimov V., Baranov A., Smirnova Y., Shatalov V., Konukhova N., Karpulevich E., Timashev P., Makarov V. Computer Vision-Assisted Spatial Analysis of Mitoses and Vasculature in Lung Cancer // Journal of Clinical Medicine. – 2025. – Vol. 14, No. 21. – P. 7526.
4. **Ананьев В. В**., Ефимов Э. В., Земнухов Е. С., Тимакова А. А., Карпулевич Е. А. Метод адаптивного синтеза изображений высокого разрешения на основе диффузионных трансформеров // Труды Института системного программирования РАН. – 2026. – Т. 38, вып. 4.
5. Земнухов Е. С., Ананьев В. В., Макаров В. А. “Программный модуль подготовки гистологических изображений для задач машинного обучения” / Свидетельство о государственной регистрации программы для ЭВМ, рег. №2024663138 от 04.06.2024 – Российская Федерация, 2024.
6. Serra J. Image analysis and mathematical morphology. – Academic Press, Inc., 1983.
7. Group T. H. D. F. Hierarchical Data Format, Version 5. [Электронный ресурс]. – 1997. – URL: https://www.hdfgroup.org/HDF5/.
8. Shimodaira H. Improving predictive inference under covariate shift by weighting the log-likelihood function //Journal of statistical planning and inference. – 2000. – Т. 90. – № 2. – С. 227-244.

9. Reinhard E. et al. Color transfer between images //IEEE Computer graphics and applications. – 2002. – T. 21. – № 5. – C. 34-41.
10. He H., Garcia E. A. Learning from imbalanced data //IEEE Transactions on knowledge and data engineering. – 2009. – T. 21. – № 9. – C. 1263-1284.
11. Macenko M. et al. A method for normalizing histology slides for quantitative analysis //2009 IEEE international symposium on biomedical imaging: from nano to macro. – IEEE, 2009. – C. 1107-1110.
12. Ben-David S. et al. A theory of learning from different domains //Machine learning. – 2010. – T. 79. – № 1. – C. 151-175.
13. Pantanowitz L. et al. Review of the current state of whole slide imaging in pathology //Journal of pathology informatics. – 2011. – T. 2. – № 1. – C. 36.
14. Torralba A., Efros A. A. Unbiased look at dataset bias //CVPR 2011. – IEEE, 2011. – C. 1521-1528.
15. Moreno-Torres J. G. et al. A unifying view on dataset shift in classification //Pattern recognition. – 2012. – T. 45. – № 1. – C. 521-530.
16. Goode A. et al. OpenSlide: A vendor-neutral software foundation for digital pathology //Journal of pathology informatics. – 2013. – T. 4. – № 1. – C. 27.
17. Kingma D. P., Welling M. Auto-encoding variational bayes //arXiv preprint arXiv:1312.6114. – 2013.
18. Goodfellow I. J. et al. Generative adversarial nets //Advances in neural information processing systems. – 2014. – T. 27.
19. Chen T. et al. Training deep nets with sublinear memory cost //arXiv preprint arXiv:1604.06174. – 2016.
20. Szegedy C. et al. Rethinking the inception architecture for computer vision //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – C. 2818-2826.
21. Heusel M. et al. Gans trained by a two time-scale update rule converge to a local nash equilibrium //Advances in neural information processing systems. – 2017. – T. 30.

22. Micikevicius P. et al. Mixed precision training //arXiv preprint arXiv:1710.03740. – 2017.
23. Vaswani A. et al. Attention is all you need //Advances in neural information processing systems. – 2017. – T. 30.
24. Bińkowski M. et al. Demystifying mmd gans //arXiv preprint arXiv:1801.01401. – 2018.
25. Zhou Z. et al. Unet++: A nested u-net architecture for medical image segmentation //International workshop on deep learning in medical image analysis. – Cham : Springer International Publishing, 2018. – C. 3-11.
26. Karras T., Laine S., Aila T. A style-based generator architecture for generative adversarial networks //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2019. – C. 4401-4410.
27. Kynkäänniemi T. et al. Improved precision and recall metric for assessing generative models //Advances in neural information processing systems. – 2019. – T. 32.
28. Paszke A. et al. PyTorch: An imperative style, high-performance deep learning library //Advances in neural information processing systems. – 2019. – T. 32.
29. Tan M., Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks //International conference on machine learning. – PMLR, 2019. – C. 6105-6114.
30. Wei J. W. et al. Pathologist-level classification of histologic patterns on resected lung adenocarcinoma slides with deep neural networks //Scientific reports. – 2019. – T. 9. – № 1. – C. 3358.
31. Chen T. et al. A simple framework for contrastive learning of visual representations //International conference on machine learning. – PmLR, 2020. – C. 1597-1607.
32. Grill J. B. et al. Bootstrap your own latent-a new approach to self-supervised learning //Advances in neural information processing systems. – 2020. – T. 33. – C. 21271-21284.

33. Ho J., Jain A., Abbeel P. Denoising diffusion probabilistic models //Advances in neural information processing systems. – 2020. – T. 33. – C. 6840-6851.

34. Song J., Meng C., Ermon S. Denoising diffusion implicit models //arXiv preprint arXiv:2010.02502. – 2020.

35. Stacke K. et al. Measuring domain shift for deep learning in histopathology //IEEE journal of biomedical and health informatics. – 2020. – T. 25. – № 2. – C. 325-336.

36. Wolf T. et al. Transformers: State-of-the-art natural language processing //Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations. – 2020. – C. 38-45.

37. Radford A. et al. Learning transferable visual models from natural language supervision //International conference on machine learning. – PmLR, 2021. – C. 8748-8763.

38. Dao T. et al. FlashAttention: Fast and memory-efficient exact attention with io-awareness //Advances in neural information processing systems. – 2022. – T. 35. – C. 16344-16359.

39. Gugger S. et al. Accelerate: Training and inference at scale made simple, efficient and adaptable //Computer software]. Hugging Face. https://github. com/huggingface/accelerate. – 2022.

40. He K. et al. Masked autoencoders are scalable vision learners //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2022. – C. 16000-16009.

41. Ho J., Salimans T. Classifier-free diffusion guidance //arXiv preprint arXiv:2207.12598. – 2022.

42. Hu E. J. et al. LoRA: Low-rank adaptation of large language models //Iclr. – 2022. – T. 1. – № 2. – C. 3.

43. Kynkäänniemi T. et al. The role of imagenet classes in frechet inception distance //arXiv preprint arXiv:2203.06026. – 2022.

44. Lefaudeux B. et al. xFormers: A Modular and Hackable Transformer Modelling Library [Электронный ресурс]. — 2022. — URL: https://github.com/facebookresearch/xformers.

45. Lu C. et al. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps //Advances in neural information processing systems. – 2022. – Т. 35. – С. 5775-5787.

46. Lugmayr A. et al. Repaint: Inpainting using denoising diffusion probabilistic models //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2022. – С. 11461-11471.

47. Mori I. Current status of whole slide image (WSI) standardization in Japan //Acta Histochemica et Cytochemica. – 2022. – Т. 55. – № 3. – С. 85-91.

48. Rombach R. et al. High-resolution image synthesis with latent diffusion models //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2022. – С. 10684-10695.

49. Von Platen P. et al. Diffusers: State-of-the-art diffusion models [Электронный ресурс]. — 2022. — URL: https://github.com/huggingface/diffusers.

50. Wang W. et al. Pvt v2: Improved baselines with pyramid vision transformer //Computational visual media. – 2022. – Т. 8. – № 3. – С. 415-424.

51. Aversa M. et al. Diffinfinite: Large mask-image synthesis via parallel random patch diffusion in histopathology //Advances in Neural Information Processing Systems. – 2023. – Т. 36. – С. 78126-78141.

52. Hang T. et al. Efficient diffusion training via min-snr weighting strategy //Proceedings of the IEEE/CVF international conference on computer vision. – 2023. – С. 7441-7451.

53. Luo S. et al. Latent consistency models: Synthesizing high-resolution images with few-step inference //arXiv preprint arXiv:2310.04378. – 2023.

54. Oquab M. et al. DINOv2: Learning robust visual features without supervision //arXiv preprint arXiv:2304.07193. – 2023.

55. Peebles W., Xie S. Scalable diffusion models with transformers //Proceedings of the IEEE/CVF international conference on computer vision. – 2023. – С. 4195-4205.

56. Ye H. et al. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models //arXiv preprint arXiv:2308.06721. – 2023.

57. Zhang L., Rao A., Agrawala M. Adding conditional control to text-to-image diffusion models //Proceedings of the IEEE/CVF international conference on computer vision. – 2023. – С. 3836-3847.

58. Ansel J. et al. PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation //Proceedings of the 29th ACM international conference on architectural support for programming languages and operating systems, volume 2. – 2024. – С. 929-947.

59. Chen J. et al. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis //International conference on learning representations. – 2024. – Т. 2024. – С. 57611-57640.

60. Chen J. et al. Pixart-δ: Fast and controllable image generation with latent consistency models //arXiv preprint arXiv:2401.05252. – 2024.

61. Chen J. et al. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation //European Conference on Computer Vision. – Cham : Springer Nature Switzerland, 2024. – С. 74-91.

62. Chen R. J. et al. Towards a general-purpose foundation model for computational pathology //Nature medicine. – 2024. – Т. 30. – № 3. – С. 850-862.

63. Esser P. et al. Scaling rectified flow transformers for high-resolution image synthesis //Forty-first international conference on machine learning. – 2024.

64. Graikos A. et al. Learned representation-guided diffusion models for large-image generation //Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 2024. – С. 8532-8542.

65. Lu M. Y. et al. A visual-language foundation model for computational pathology //Nature medicine. – 2024. – Т. 30. – № 3. – С. 863-874.

66. Xu H. et al. A whole-slide foundation model for digital pathology from real-world data //Nature. – 2024. – Т. 630. – № 8015. – С. 181-188.

67. Yellapragada S. et al. Pathldm: Text conditioned latent diffusion model for histopathology //Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. – 2024. – С. 5182-5191.

68. Zimmermann E. et al. Virchow2: Scaling self-supervised mixed magnification models in pathology //arXiv preprint arXiv:2408.00738. – 2024.

69. RAPIDS A. I. cuCIM-RAPIDS GPU-accelerated image processing library (2025) [Электронный ресурс].— URL: https://github.com/rapidsai/cucim.

70. Yellapragada S. et al. Pathology Image Compression with Pre-trained Autoencoders //International Conference on Medical Image Computing and Computer-Assisted Intervention. – Cham : Springer Nature Switzerland, 2025. – С. 442-452.

71. Yellapragada S. et al. PixCell: A generative foundation model for digital histopathology images //ArXiv. – 2025. – С. arXiv: 2506.05127 v1.

72. Yellapragada S. et al. Zoomldm: Latent diffusion model for multi-scale image generation //Proceedings of the Computer Vision and Pattern Recognition Conference. – 2025. – С. 23453-23463.

73. Duboudin T. et al. CytoSyn: a Foundation Diffusion Model for Histopathology--Tech Report //arXiv preprint arXiv:2603.18089. – 2026.

74. National Lung Screening Trial Research Team et al. Data from the National Lung Screening Trial (NLST)[Data set]. The Cancer Imaging Archive [Электронный ресурс]. – URL: https://www.cancerimagingarchive.net/collection/nlst/).

Список сокращений и условных обозначений

Аугментирование	Augmentation, метод искусственного расширения и повышения разнообразия обучающей выборки путем применения к исходным данным различных преобразований (геометрических, колориметрических и т.д.).
Доменный сдвиг	Ковариантный сдвиг, Covariate shift, статистическая проблема рассогласования данных, при которой распределение признаков в обучающей выборке смещается относительно распределения в реальной или целевой среде
Инференс	Inference, этап применения обученной модели машинного обучения для получения предсказаний на новых данных
Коллапс мод	Mode collapse, проблема при обучении генеративных моделей, выражающаяся в потере моделью способности воспроизводить разнообразие исходных данных и генерации ею лишь узкого подмножества
Латентное пространство	Latent space, сжатое математическое пространство признаков пониженной размерности, в которое исходное изображение отображается с помощью кодировщика
Обусловливание	Conditioning, управление, механизм, позволяющий осуществлять детерминированное управление процессом генерации путем внедрения дополнительного направляющего сигнала
Репараметризация	Reparameterization trick, математический прием, применяемый при обучении вариационных автокодировщиков, обеспечивающий возможность обратного распространения градиента через стохастический слой
Самодистилляция	Self-distillation, метод самообучения нейронных сетей без использования меток, при котором происходит согласование выходов двух моделей («ученика» и «учителя») на основе одной и той же структуры данных
Тайл	tile, patch, патч, небольшой прямоугольный фрагмент фиксированного размера
Тайлинг	Декомпозиция, процесс разбиения изображения на фрагменты
Эмбеддинг	Embedding, компактный вектор признаков, извлекается из исходных данных с помощью кодировщика и отражает смысловые характеристики исходного объекта

Inpainting	Контекстное достраивание/встраивание, алгоритм локального редактирования или генерации изображения, при котором синтезируется только содержимое заданной целевой области с обязательным сохранением остального контекста.
ГНС	DNN, Deep Neural Network, глубокая нейронная сеть
ЛВИ	лимфоваскулярная инвазия
CFG	Classifier-Free Guidance, управление процессом диффузии без использования отдельной модели классификатора
CORAL	Correlation Alignment, метод выравнивания признаковых распределений
DDPM	Denoising Diffusion Probabilistic Models, вероятностные диффузионные модели
DiT	Diffusion Transformer, диффузионный трансформер
ELBO	Evidence Lower Bound, нижняя оценка логарифма правдоподобия
FID	Fréchet Inception Distance, расстояние Фреше (метрика оценки качества генерации)
GAN	Generative Adversarial Network, генеративно-состязательная сеть
GSD	Ground Sampling Distance, разрешение на местности
HBM	High Bandwidth Memory, высокопропускная глобальная память видеокарты
IoU	Intersection over Union, индекс пересечения по объединению (метрика качества сегментации)
JIT	Just-In-Time, динамическая компиляция вычислительного графа
KID	Kernel Inception Distance, ядерное расстояние (метрика оценки согласованности распределений)
LDM	Latent Diffusion Model, латентная диффузионная модель
LoRA	Low-Rank Adaptation, алгоритм низкоранговой адаптации
MAE	Masked Autoencoders, генеративные архитектуры на основе преднамеренного маскирования патчей
MIL	Multiple-Instance Learning, обучение с множественными представлениями
MPP	Microns Per Pixel, число микрометров на пиксель
MSE	Mean Squared Error, среднеквадратичная ошибка

NND	Nearest Neighbour Distance, расстояние до ближайшего соседа (метрика уникальности синтеза)
SSL	Self-Supervised Learning, самообучение (обучение без явной разметки)
VAE	Variational Autoencoder, вариационный автокодировщик
WSI	Whole-Slide Images, полнокадровые растровые копии микроскопических препаратов (гигапиксельные гистологические изображения)

Приложение А

Программный модуль подготовки гистологических изображений для задач машинного обучения

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU2024663138

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ

Номер регистрации (свидетельства): **2024663138** Дата регистрации: **04.06.2024** Номер и дата поступления заявки: **2024661064 20.05.2024** Дата публикации и номер бюллетеня: **04.06.2024 Бюл. № 6** Контактные реквизиты: **vladimir.makarov@novsu.ru**	Автор(ы): **Земнухов Егор Сергеевич (RU),** **Ананьев Владислав Валерьевич (RU),** **Макаров Владимир Алексеевич (RU)** Правообладатель(и): **Федеральное государственное бюджетное образовательное учреждение высшего образования «Новгородский государственный университет имени Ярослава Мудрого» (RU)**

Название программы для ЭВМ:
Программный модуль подготовки гистологических изображений для задач машинного обучения

Реферат:
Программный модуль предназначен для преобразования гистологических изображений к унифицированным форматам данных, используемым при обучении моделей машинного обучения. Может использоваться в области цифровой патологии. Функциональные возможности: генерация конфигурируемых схем для фрагментации гистологического изображения, в результате фрагментации получается набор файлов заданного размера в форматах .png и .jpeg; предоставление API-интерфейса для взаимодействия с файлами, загрузка с диска, предварительная обработка, визуализация; возможность фильтрации аннотаций по площади размеченного объекта; поддержка возможности взаимодействия с аннотациями в формате GeoJSON: функционал для постобработки результатов работы моделей машинного обучения. Работа выполнена при финансовой поддержке Министерства науки и высшего образования Российской Федерации по программе создания и развития центра мирового уровня "Цифровой биодизайн и персонализированное здравоохранение" в рамках проекта № 075-15-2022-306. ОС: Windows, Unix.

Язык программирования: Python

Объем программы для ЭВМ: 27,1 КБ

Приложение Б

Исходный код модулей для обучения моделей

Б1. Исходный код модуля дообучения набора базовых моделей в режиме прогрессивного обучения

```
import os
import logging
import argparse
import glob
import math
import torch
from torch.utils.data import DataLoader
from torch.utils.data import default_collate
from safetensors.torch import save_file
from diffusers.optimization import get_scheduler
from accelerate import Accelerator, DistributedDataParallelKwargs
from tqdm.auto import tqdm

from torchvision.utils import make_grid
from diffusers import DiffusionPipeline

import histo_diffusion.config.paths as paths
from histo_diffusion.config.configs import get_model_config
from histo_diffusion.utils.logging_utils import setup_global_logging
from histo_diffusion.models.pixcell_lora_transformer import
PixCellTransformer2DModelLoRA
from histo_diffusion.utils.model_utils import (load_uni_model,
                                               load_vae,
                                               get_vae_params,
                                               load_train_scheduler,
                                               extract_uni_embeddings,
                                               compute_loss,
                                               compile_models,
                                               get_model_optimizer,
)
from histo_diffusion.utils.inp_args_processor import parse_args_lora
from histo_diffusion.dataset.datasets import VesselsDataset
from histo_diffusion.dataset.efficient_datasets import CachedFeaturesDataset
from histo_diffusion.utils.tracking import (add_clearml_args,
                                            init_clearml_tracker,
                                            log_metrics,
                                            get_accelerator_log_with,
)

logger = logging.getLogger(__name__)

def _diffusion_forward(vae_latents, uni_emb, transformer, noise_scheduler,
uncond_prob, snr_gamma=None):
    device = vae_latents.device
    bs = vae_latents.shape[0]
    n_channels = vae_latents.shape[1]

    noise = torch.randn_like(vae_latents)
    timesteps = torch.randint(0, noise_scheduler.config.num_train_timesteps, (bs,),
device=device, dtype=torch.long)
    noisy_latents = noise_scheduler.add_noise(vae_latents, noise, timesteps)

    transformer_unwrapped = transformer.module if hasattr(transformer, "module") else
transformer
```

```
    if uncond_prob > 0:
        uncond =
transformer_unwrapped.caption_projection.uncond_embedding.clone().tile(bs, 1, 1)
        drop = (torch.rand((bs, 1, 1), device=device) < uncond_prob).float()
        uni_emb_input = (1 - drop) * uni_emb + drop * uncond
    else:
        uni_emb_input = uni_emb
    epsilon_pred = transformer(noisy_latents, encoder_hidden_states=uni_emb_input,
timestep=timesteps, return_dict=False)[0]
    model_pred = epsilon_pred[:, :n_channels]

    loss = compute_loss(noise_scheduler, model_pred, noise, timesteps, vae_latents,
snr_gamma=snr_gamma)
    return loss

def training_step(batch, transformer, vae, vae_transform, uni_model, uni_transform,
noise_scheduler, vae_scale, vae_shift, image_size, uncond_prob, snr_gamma=None):
    img = batch["image"]
    device = img.device
    uni_emb = extract_uni_embeddings(img, uni_model, uni_transform, device,
image_size)
    img_norm = vae_transform(img)
    vae_latents = vae.encode(img_norm).latent_dist.sample()
    vae_latents = (vae_latents - vae_shift) * vae_scale

    return _diffusion_forward(vae_latents, uni_emb, transformer, noise_scheduler,
uncond_prob, snr_gamma)

def training_step_precomputed(batch, transformer, noise_scheduler, vae_scale,
vae_shift, uncond_prob, snr_gamma=None):
    uni_emb = batch["uni"]
    raw = batch["sd3_vae"]
    mean, std = raw[:, :16], raw[:, 16:]
    vae latest = mean + std * torch.randn_like(mean)
    vae_latents = (vae_latents - vae_shift) * vae_scale

    return _diffusion_forward(vae_latents, uni_emb, transformer, noise_scheduler,
uncond_prob, snr_gamma)

def main():
    args = parse_args_lora()

    from datetime import datetime
    timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    args.clearml_project = "progressive_full_training"

    if args.clearml_task is None:
        args.clearml_task =
f"train_progressive_full_{args.image_size}_{timestamp}_lr-{args.learning_rate}"

    os.makedirs(args.save_dir, exist_ok=True)
    os.makedirs(args.output_dir, exist_ok=True)

    setup_global_logging(log_file=f"{args.output_dir}/training_logs.log",
level=logging.INFO)


    logger.info("Mode: PRE-COMPUTED embeddings")
    uni_model, uni_transform = None, None
    vae, vae_transform = load_vae(args.path_to_sd3_vae)
    vae_scale, vae_shift = get_vae_params(vae)

    noise_scheduler = load_train_scheduler(args.path_to_pixcell)

    config = get_model_config(args.image_size)
    transformer = PixCellTransformer2DModelLoRA(**config)

    if args.resume_from_checkpoint:
        logger.info(f"Load previous weights and interpolate:
{args.resume_from_checkpoint}")
```

```
        if os.path.isdir(args.resume_from_checkpoint):
            safetensors_files = glob.glob(os.path.join(args.resume_from_checkpoint,
"*.safetensors"))
            chkpt_path = safetensors_files[0] if safetensors_files else None
        else:
            chkpt_path = args.resume_from_checkpoint

        if chkpt_path and os.path.exists(chkpt_path):
            transformer = PixCellTransformer2DModelLoRA.from_lower_resolution(
                config=config,
                source_checkpoint=chkpt_path,
                tile_uncond_embedding=True
            )
        else:
            logger.error(f"File not found: {args.resume_from_checkpoint}")
            raise FileNotFoundError(args.resume_from_checkpoint)
    else:
        chkpt_path = os.path.join(args.path_to_pixcell, "transformer",
"diffusion_pytorch_model.safetensors")
        logger.info(f"Base init from: {chkpt_path}")
        transformer = PixCellTransformer2DModelLoRA.from_lower_resolution(
            config=config,
            source_checkpoint=chkpt_path,
            tile_uncond_embedding=True
        )

    if args.gradient_checkpointing:
        transformer.enable_gradient_checkpointing()

    if args.enable_xformers:
        transformer.enable_xformers_memory_efficient_attention()
    if args.compile_model:
        transformer, = compile_models(models=[transformer], mode="default")

    if args.use_precomputed:
        ds_names = args.dataset_names or paths.get_ds_names(args.image_size)
        total_dataset = CachedFeaturesDataset(root=paths.DATASETS_ROOT,
dataset_names=ds_names, features_subdir="features", use_fp16=(args.mixed_precision ==
"fp16"))
    else:
        total_dataset = VesselsDataset.from_json(
            json_path=[str(paths.ds_by_name(d).coco) for d in args.coco_datasets],
            limit=-1,
            mask_representation=None
        )

    train_dataloader = DataLoader(
        total_dataset, batch_size=args.train_batch_size, shuffle=True,
        num_workers=args.num_workers, pin_memory=True, prefetch_factor=None,
persistent_workers=False,
        drop_last=True)


    transformer.requires_grad_(True)
    trainable_parameters = list(transformer.parameters())

    optimiser_kwargs = {"lr": args.learning_rate}
    optimiser_kwargs.update({"weight_decay": 1e-2, "betas": (0.9, 0.999), "eps":
1e-8})

    num_update_steps_per_epoch = math.ceil(len(train_dataloader) /
args.gradient_accumulation_steps)
    max_train_steps = args.num_epochs * num_update_steps_per_epoch

    optimizer = get_model_optimizer(o_type=args.optimizer,
model_params=trainable_parameters, **optimiser_kwargs)
    lr_sched = get_scheduler(args.lr_scheduler, optimizer=optimizer,
num_warmup_steps=args.lr_warmup_steps, num_training_steps=max_train_steps)

    ddp_kwargs = DistributedDataParallelKwargs(find_unused_parameters=False)
```

```
    accelerator = Accelerator(
        mixed_precision=args.mixed_precision,
gradient_accumulation_steps=args.gradient_accumulation_steps,
        log_with=get_accelerator_log_with(args.use_clearml),
project_dir=os.path.join(args.output_dir, "logs"),
        kwargs_handlers=[ddp_kwargs],
    )

    transformer, optimizer, train_dataloader, lr_sched =
accelerator.prepare(transformer, optimizer, train_dataloader, lr_sched)

    if vae is not None:
        vae = accelerator.prepare_model(vae, evaluation_mode=True)
    if uni_model is not None:
        uni_model = accelerator.prepare_model(uni_model, evaluation_mode=True)
    noise_scheduler.alphas_cumprod =
noise_scheduler.alphas_cumprod.to(accelerator.device)

    init_clearml_tracker(accelerator, args)

    logger.info("=" * 60)
    logger.info("SART PROGRESSIVE TRAIN")
    logger.info("=" * 60)

    global_step = 0
    for epoch in range(args.num_epochs):
        transformer.train()
        progress_bar = tqdm(total=num_update_steps_per_epoch, disable=not
accelerator.is_local_main_process, desc=f"Epoch {epoch}")
        epoch_loss = 0.0

        dataloader_iterator = iter(train_dataloader)

        for step in range(len(train_dataloader)):
            batch = next(dataloader_iterator)

            with accelerator.accumulate(transformer):
                if args.use_precomputed:
                    loss = training_step_precomputed(batch, transformer,
noise_scheduler, vae_scale, vae_shift, args.uncond_prob, args.snr_gamma)
                else:
                    loss = training_step(batch, transformer, vae, vae_transform,
uni_model, uni_transform, noise_scheduler, vae_scale, vae_shift, args.image_size,
args.uncond_prob, args.snr_gamma)

                accelerator.backward(loss)
                if accelerator.sync_gradients:
                    accelerator.clip_grad_norm_(trainable_parameters,
args.max_grad_norm)
                optimizer.step()
                lr_sched.step()
                optimizer.zero_grad()

            if accelerator.sync_gradients:
                progress_bar.update(1)
                global_step += 1

            epoch_loss += loss.detach().item()
            logs = {"loss": loss.detach().item(), "lr": lr_sched.get_last_lr()[0],
"step": global_step}
            progress_bar.set_postfix(**logs)

            if global_step % args.log_every_n_steps == 0:
                log_metrics(accelerator, {"loss": logs["loss"], "lr": logs["lr"]},
step=global_step, prefix="train")

        avg_loss = epoch_loss / max(len(train_dataloader), 1)
        logger.info(f"Epoch {epoch} — avg loss: {avg_loss:.6f}")
        log_metrics(accelerator, {"epoch_avg_loss": avg_loss}, step=epoch,
prefix="train")
```

```
261
262         accelerator.wait_for_everyone()
263
264         if accelerator.is_main_process:
265             should_save = ((epoch + 1) % args.save_every_n_epochs == 0 or epoch ==
266 args.num_epochs - 1)
267
268             if should_save:
269                 unwrapped = accelerator.unwrap_model(transformer)
270                 mode_str = "precomputed" if args.use_precomputed else "onthefly"
271                 save_path = os.path.join(args.save_dir,
272 f"progressive_full_{args.image_size}_{mode_str}_epoch_{epoch}")
273                 os.makedirs(save_path, exist_ok=True)
274                 save_file(unwrapped.state_dict(), os.path.join(save_path,
275 "diffusion_pytorch_model.safetensors"))
276
277         progress_bar.close()
278
279     accelerator.end_training()
290     logger.info("Training complete!")
291
292 if __name__ == "__main__":
293     main()
294
```

Б2. Исходный код модуля для обучения низкоранговых адаптеров LoRA

```
 1 import os
 2 import logging
 3 import torch
 4 from accelerate import Accelerator, DistributedDataParallelKwargs
 5 from diffusers.optimization import get_scheduler
 6 from peft import LoraConfig
 7 from safetensors.torch import load_file
 8 from torch.utils.data import DataLoader
 9 from tqdm.auto import tqdm
10 import histo_diffusion.config.paths as paths
11 from histo_diffusion.config.configs import get_model_config
12 from histo_diffusion.dataset.efficient_datasets import CachedFeaturesDataset
13 from histo_diffusion.models.pixcell_lora_transformer import (
14     PixCellTransformer2DModelLoRA,
15 )
16 from histo_diffusion.utils.checkpointing import save_checkpoint
17 from histo_diffusion.utils.inp_args_processor import parse_args_lora
18 from histo_diffusion.utils.logging_utils import setup_global_logging
19 from histo_diffusion.utils.model_utils import (
20     compile_models,
21     compute_loss_with_timestep_bins,
22     restore_training_state_params,
23     get_model_optimizer,
24     load_train_scheduler,
25 )
26 from histo_diffusion.utils.tracking import (
27     get_accelerator_log_with,
28     init_clearml_tracker,
29     log_metrics,
30 )
31
32 logger = logging.getLogger(__name__)
33
34
35 def _diffusion_forward_lora(
36     vae_latents: torch.Tensor,
```

```
    uni_emb: torch.Tensor,
    lora_transformer,
    noise_scheduler,
    uncond_prob: float,
    snr_gamma: float = None,  # type: ignore
) -> tuple:
    """Shared diffusion forward for LoRA training.

    All inputs must already be on the correct device and properly scaled.
    """
    device = vae_latents.device
    bs = vae_latents.shape[0]
    n_channels = vae_latents.shape[1]  # 16

    # Noise & timesteps
    noise = torch.randn_like(vae_latents)
    timesteps = torch.randint(
        0,
        noise_scheduler.config.num_train_timesteps,
        (bs,),
        device=device,
        dtype=torch.long,
    )
    noisy_latents = noise_scheduler.add_noise(vae_latents, noise, timesteps)

    # UNI dropout (classifier-free guidance)
    transformer_unwrapped = (
        lora_transformer.module
        if hasattr(lora_transformer, "module")
        else lora_transformer
    )
    if uncond_prob > 0:
        uncond =
transformer_unwrapped.caption_projection.uncond_embedding.clone().tile(
            bs, 1, 1
        )
        drop = (torch.rand((bs, 1, 1), device=device) < uncond_prob).float()
        uni_emb_input = (1 - drop) * uni emb + drop * uncond
    else:
        uni_emb_input = uni_emb

    # Forward pass
    epsilon_pred = lora_transformer(
        noisy_latents,
        encoder_hidden_states=uni_emb_input,
        timestep=timesteps,
        return_dict=False,
    )[0]

    # Loss compute
    model_pred = epsilon_pred[:, :n_channels]

    loss, bin_metrics = compute_loss_with_timestep_bins(
        noise_scheduler=noise_scheduler,
        model_pred=model_pred,
        noise=noise,
        timesteps=timesteps,
        latents=vae_latents,
        snr_gamma=snr_gamma,
        n_bins=4,
    )

    metrics = {
        "bin_metrics": bin_metrics,
    }

    return loss, metrics


def training_step_lora_precomputed(
```

```
    batch: dict,
    lora_transformer,
    noise_scheduler,
    vae_scale: float,
    vae_shift: float,
    uncond_prob: float,
    snr_gamma: float = None,
) -> tuple:
    """Pre-computed embeddings: reparameterization trick → diffusion forward."""
    # UNI embeddings (already computed)
    uni_emb = batch["uni"]

    # VAE reparameterization
    raw = batch["sd3_vae"]
    mean, std = raw[:, :16], raw[:, 16:]
    vae_latents = mean + std * torch.randn_like(mean)
    vae_latents = (vae_latents - vae_shift) * vae_scale

    loss, metrics = _diffusion_forward_lora(
        vae_latents,
        uni_emb,
        lora_transformer,
        noise_scheduler,
        uncond_prob,
        snr_gamma,
    )
    return loss, metrics


def main():
    args = parse_args_lora()

    os.makedirs(args.save_dir, exist_ok=True)
    os.makedirs(args.output_dir, exist_ok=True)

    setup_global_logging(
        log_file=f"{args.output_dir}/training_logs.log", level=logging.INFO
    )

    # Load models based on mode
    logger.info("Mode: PRE-COMPUTED embeddings (UNI and VAE will NOT be loaded)")
    uni_model, vae = None, None
    vae_scale = 1.5305
    vae_shift = 0.0609

    noise_scheduler = load_train_scheduler(args.path_to_pixcell)

    # Setup transformer with model config from configs.py
    config = get_model_config(args.image_size)
    lora_transformer = PixCellTransformer2DModelLoRA(**config)

    # Load base weights
    chkpt_path = os.path.join(
        args.path_to_pixcell, "transformer", "diffusion_pytorch_model.safetensors"
    )
    logger.info(f"Loading base weights from: {chkpt_path}")

    missing, unexpected = lora_transformer.load_state_dict(
        load_file(chkpt_path), strict=False
    )
    expected_missing = {"y_pos_embed.y_pos_embed"}
    real_missing = set(missing) - expected_missing
    if real_missing:
        logger.warning(f"Unexpected missing keys: {real_missing}")
    if unexpected:
        logger.warning(f"Unexpected keys in checkpoint: {unexpected}")
    if not real_missing and not unexpected:
        logger.info(
            f"Weights loaded OK (skipped computed buffers: {expected_missing &
set(missing)})"
```

```
        )

    # OPT: reduces memory consumption, but slows down learning
    if args.gradient_checkpointing:
        if hasattr(lora_transformer, "enable_gradient_checkpointing"):
            lora_transformer.enable_gradient_checkpointing()
        else:
            lora_transformer.gradient_checkpointing = True

    lora_transformer.requires_grad_(False)

    # Add LoRA adapters
    target_modules = [
        "attn2.add_k_proj",
        "attn2.add_q_proj",
        "attn2.add_v_proj",
        "attn2.to_add_out",
        "attn2.to_k",
        "attn2.to_out.0",
        "attn2.to_q",
        "attn2.to_v",
    ]

    transformer_lora_config = LoraConfig(
        r=args.lora_rank,
        lora_alpha=args.lora_alpha,
        lora_dropout=args.lora_dropout,
        init_lora_weights="gaussian",
        target_modules=target_modules,
    )
    lora_transformer.add_adapter(transformer_lora_config)

    # HACK: optimization (for modern GPU)
    if args.enable_xformers:
        lora_transformer.enable_xformers_memory_efficient_attention()

    # HACK: optimization
    if args.compile_model:
        lora_transformer = compile_models(models=[lora_transformer], mode="default")

    # Prepare dataset
    datasets = (
        [paths.ds_by_name(d) for d in args.dataset_names]
        if args.dataset_names is not None
        else paths.ds_by_resolution(args.image_size)
    )

    # Init datasets from HDF5 base
    # logger.info(f"Loading precomputed FeaturesDataset: {ds_names}")
    ds_names = [ds.name for ds in datasets]
    logger.info(f"CachedFeaturesDataset: datasets={datasets}")
    total_dataset = CachedFeaturesDataset(
        root=paths.DATASETS_ROOT,
        dataset_names=ds_names,
        features_subdir="features",
        use_fp16=True if args.mixed_precision == "fp16" else False,
        indices=None,
    )

    logger.info(f"Dataset size: {len(total_dataset)} samples")

    # For Cached datasets use prefetch_factor=None, persistent_workers=False,
num_workers=1
    train_dataloader = DataLoader(
        total_dataset,
        batch_size=args.train_batch_size,
        shuffle=True,
        num_workers=args.num_workers,
        pin_memory=True,
        prefetch_factor=None,  # 2 if args.num_workers > 0 else None,
```

```
        persistent_workers=False,  # args.num_workers > 0,
    )

    # Optimizer
    lora_parameters = list(
        filter(lambda p: p.requires_grad, lora_transformer.parameters())
    )

    optimizer_kwargs = {"lr": args.learning_rate}
    optimizer_kwargs["weight_decay"] = 1e-2
    optimizer_kwargs["betas"] = (0.9, 0.999)
    optimizer_kwargs["eps"] = 1e-8

    optimizer = get_model_optimizer(
        o_type=args.optimizer, model_params=lora_parameters, **optimizer_kwargs
    )

    # LR scheduler
    lr_sched = get_scheduler(
        args.lr_scheduler,
        optimizer=optimizer,
        num_warmup_steps=args.lr_warmup_steps,
        num_training_steps=None,
    )

    # Accelerator
    ddp_kwargs = DistributedDataParallelKwargs(find_unused_parameters=True)
    accelerator = Accelerator(
        mixed_precision=args.mixed_precision,
        gradient_accumulation_steps=args.gradient_accumulation_steps,
        log_with=get_accelerator_log_with(args.use_clearml),
        project_dir=os.path.join(args.output_dir, "logs"),
        kwargs_handlers=[ddp_kwargs],
    )

    lora_transformer, optimizer, train_dataloader, lr_sched = accelerator.prepare(
        lora_transformer, optimizer, train_dataloader, lr_sched
    )

    if vae is not None:
        vae = accelerator.prepare_model(vae, evaluation_mode=True)
    if uni_model is not None:
        uni_model = accelerator.prepare_model(uni_model, evaluation_mode=True)

    # Move scheduler tensors to GPU (avoid CPU→GPU transfer every step)
    noise_scheduler.alphas_cumprod = noise_scheduler.alphas_cumprod.to(
        accelerator.device
    )

    init_clearml_tracker(accelerator, args)

    global_step = 0
    start_epoch = 0
    if args.resume_from:
        start_epoch, global_step, optimizer_cfg, lr_scheduler_cfg = (
            restore_training_state_params(checkpoint_path=args.resume_from)
        )
        optimizer.load_state_dict(optimizer_cfg)
        lr_sched.load_state_dict(lr_scheduler_cfg)

    for epoch in range(start_epoch, args.num_epochs):
        lora_transformer.train()
        progress_bar = tqdm(
            total=len(train_dataloader),
            disable=not accelerator.is_local_main_process,
            desc=f"Epoch {epoch}",
        )
        epoch_loss = 0.0

        for step, batch in enumerate(train_dataloader):
```

```
            with accelerator.accumulate(lora_transformer):
                loss, metrics = training_step_lora_precomputed(
                    batch=batch,
                    lora_transformer=lora_transformer,
                    noise_scheduler=noise_scheduler,
                    vae_scale=vae_scale,
                    vae_shift=vae_shift,
                    uncond_prob=args.uncond_prob,
                    snr_gamma=args.snr_gamma,
                )

                accelerator.backward(loss)
                if accelerator.sync_gradients:
                    accelerator.clip_grad_norm_(lora_parameters, args.max_grad_norm)
                optimizer.step()
                lr_sched.step()
                optimizer.zero_grad()

            if accelerator.sync_gradients:
                progress_bar.update(1)
                global_step += 1

            epoch_loss += loss.detach().item()
            logs = {
                "loss": loss.detach().item(),
                "lr": lr_sched.get_last_lr()[0],
                "step": global_step,
            }
            progress_bar.set_postfix(**logs)

            if global_step % args.log_every_n_steps == 0:
                log_metrics(
                    accelerator,
                    {"loss": logs["loss"], "lr": logs["lr"]},
                    step=global_step,
                    prefix="train",
                )

            if global_step % (args.log_every_n_steps * 2) == 0:
                log_metrics(
                    accelerator,
                    metrics["bin_metrics"],
                    step=global_step,
                    prefix="train",
                )

        # End of epoch
        avg_loss = epoch_loss / max(len(train_dataloader), 1)
        logger.info(f"Epoch {epoch} — avg loss: {avg_loss:.6f}")
        log_metrics(
            accelerator, {"epoch_avg_loss": avg_loss}, step=epoch, prefix="train"
        )

        # Save checkpoint
        accelerator.wait_for_everyone()
        if accelerator.is_main_process:
            save_checkpoint(
                model=lora_transformer,
                optimizer=optimizer,
                lr_scheduler=lr_sched,
                epoch=epoch,
                global_step=global_step,
                metric_value=avg_loss,
                checkpoint_dir=args.save_dir,
                prefix=f"lora_{args.image_size}",
                n_top=5,
                mode="min",
            )
        progress_bar.close()

```

```
    accelerator.end_training()
    logger.info("Training complete!")


if __name__ == "__main__":
    main()
```

Приложение В

Исходный код модуля контекстно-ориентированного встраивания объектов

В исходном коде модуля представлена реализация трех этапов алгоритма контекстно-ориентированного встраивания объектов:

- пространственное и хроматическое выравнивание;
- семантически обусловленная диффузионная генерация;
- морфологическая коррекция границ на основе карты воздействия.

```
import os
import sys
import gc
import json
import random
import cv2
import torch
import numpy as np
from tqdm import tqdm
from PIL import Image
from skimage import color
import torchvision.transforms as T
from safetensors.torch import load_file

import histo_diffusion.config.paths as paths
from histo_diffusion.config.configs import get_model_config
from histo_diffusion.models.pixcell_lora_transformer import
PixCellTransformer2DModelLoRA
from histo_diffusion.utils.model_utils import load_vae, load_uni_model,
extract_uni_embeddings
from diffusers import DiffusionPipeline

DEVICE = "cuda:0" if torch.cuda.is_available() else "cpu"
IMAGE_SIZE = 1024
DTYPE = torch.bfloat16

TARGET_COUNT = 2607
CFG_SCALE = 2.5
STRENGTH = 0.55
WALL_SIGMA = 15.0
STEPS = 50

def get_mask_for_img(img_path, mask_folder):
    name_no_ext = os.path.splitext(os.path.basename(img_path))[0]
    for ext in [".png", ".jpg", ".jpeg", ".tif"]:
        p = os.path.join(mask_folder, name_no_ext + ext)
        if os.path.exists(p):
            return p
    return None

def get_smart_coordinates(target_path, donor_mask_path,
                          canvas_size=1024, max_attempts=100):
    mask = cv2.imread(donor_mask_path, cv2.IMREAD_GRAYSCALE)
    if mask is None: return canvas_size // 2, canvas_size // 2

    coords = cv2.findNonZero(mask)
```

```
    if coords is not None:
        _, _, w, h = cv2.boundingRect(coords)
        vessel_size = int(max(w, h) * 1.41)
    else:
        vessel_size = 256

    vessel_size = min(vessel_size, canvas_size - 10)
    half_size = vessel_size // 2
    target_bgr = cv2.imread(target_path)
    target_resized = cv2.resize(target_bgr, (canvas_size, canvas_size))
    gray = cv2.cvtColor(target_resized, cv2.COLOR_BGR2GRAY)
    tissue_mask = gray < 180

    for _ in range(max_attempts):
        cx = random.randint(half_size, canvas_size - half_size)
        cy = random.randint(half_size, canvas_size - half_size)
        patch = tissue_mask[cy - half_size : cy + half_size,
                            cx - half_size : cx + half_size]
        if np.mean(patch) > 0.90: return cx, cy

    for _ in range(max_attempts):
        cx = random.randint(half_size, canvas_size - half_size)
        cy = random.randint(half_size, canvas_size - half_size)
        patch = tissue_mask[cy - half_size : cy + half_size,
                            cx - half_size : cx + half_size]
        if np.mean(patch) > 0.50: return cx, cy

    return canvas_size // 2, canvas_size // 2

def get_dominant_nuclei_angle(image_bgr, mask_gray=None, resize_dim=512,
                              debug=False, debug_save_path=None):
    if image_bgr is None: return 0.0
    h, w = image_bgr.shape[:2]
    scale = resize_dim / max(h, w)
    new_w, new_h = int(w * scale), int(h * scale)
    img_resized = cv2.resize(image_bgr, (new_w, new_h))
    rgb_img = cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB)
    hed = color.rgb2hed(rgb_img)
    h_channel = hed[:, :, 0]
    h_norm = cv2.normalize(h_channel, None, 0, 255, cv2.NORM_MINMAX,
                           dtype=cv2.CV_8U)
    _, binary = cv2.threshold(h_norm, 0, 255, cv2.THRESH_BINARY +
                              cv2.THRESH_OTSU)

    if mask_gray is not None:
        mask_resized = cv2.resize(mask_gray, (new_w, new_h),
                                  interpolation=cv2.INTER_NEAREST)
        binary = cv2.bitwise_and(binary, binary, mask=mask_resized)

    contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL,
                                   cv2.CHAIN_APPROX_SIMPLE)
    angles = []
    for cnt in contours:
        if len(cnt) >= 5:
            area = cv2.contourArea(cnt)
            if 10 < area < 800:
                (_, _), (_, _), angle = cv2.fitEllipse(cnt)
                angles.append(angle)
    if not angles: return 0.0
    median_angle = float(np.median(angles))
    return median_angle

def transform_donor_and_mask(donor_pil, mask_pil, angle,
                             center_x, center_y, canvas_size=1024):
    mask_arr = np.array(mask_pil)
    coords = cv2.findNonZero(mask_arr)
    if coords is None: return donor_pil, mask_pil
    x, y, w, h = cv2.boundingRect(coords)
    donor_cropped = donor_pil.crop((x, y, x + w, y + h))
    mask_cropped = mask_pil.crop((x, y, x + w, y + h))
```

```
    donor_rotated = donor_cropped.rotate(angle, expand=True,
                                         resample=Image.BICUBIC)
    mask_rotated = mask_cropped.rotate(angle, expand=True,
                                       resample=Image.NEAREST)

    new_donor = Image.new("RGB", (canvas_size, canvas_size), (0, 0, 0))
    new_mask = Image.new("L", (canvas_size, canvas_size), 0)

    rw, rh = donor_rotated.size
    paste_x = int(center_x - rw / 2)
    paste_y = int(center_y - rh / 2)

    new_donor.paste(donor_rotated, (paste_x, paste_y), mask=mask_rotated)
    new_mask.paste(mask_rotated, (paste_x, paste_y), mask=mask_rotated)
    return new_donor, new_mask

def smart_color_match_v2(donor_pil, target_pil, mask_pil,
                         blend_strength=0.85):
    don_arr = np.clip(np.array(donor_pil).astype(np.float32)/255.0, 0, 1)
    tgt_arr = np.clip(np.array(target_pil).astype(np.float32)/255.0, 0, 1)
    mask_arr = np.array(mask_pil) > 127

    tgt_gray = np.mean(tgt_arr, axis=2)
    threshold = np.percentile(tgt_gray, 85)
    tgt_tissue_mask = tgt_gray < threshold

    if np.sum(tgt_tissue_mask) < 100:
        tgt_tissue_mask = np.ones_like(tgt_gray, dtype=bool)

    don_gray = np.mean(don_arr, axis=2)
    don_tissue_mask = mask_arr & (don_gray < 0.85)

    don_lab = color.rgb2lab(don_arr)
    tgt_lab = color.rgb2lab(tgt_arr)

    for i in range(3):
        don_pixels = don_lab[..., i][don_tissue_mask]
        tgt_pixels = tgt_lab[..., i][tgt_tissue_mask]

        if len(don_pixels) > 10 and len(tgt_pixels) > 10:
            don_mean, don_std = don_pixels.mean(), don_pixels.std()
            tgt_mean, tgt_std = tgt_pixels.mean(), tgt_pixels.std()

            if don_std > 1e-5:
                std_ratio = np.clip(tgt_std / don_std, 0.5, 2.0)
                matched_pixels = (don_pixels - don_mean) * std_ratio +
                                 tgt_mean
                final_pixels = matched_pixels * blend_strength +
                               don_pixels * (1.0 - blend_strength)
                don_lab[..., i][don_tissue_mask] = final_pixels

    res_rgb = color.lab2rgb(don_lab)
    res_rgb = (res_rgb * 255).clip(0, 255).astype(np.uint8)
    return Image.fromarray(res_rgb)

def prep_models():
    vae, vae_transform = load_vae(str(paths.SD3_VAE_DIR))
    vae = vae.to(device=DEVICE, dtype=DTYPE).eval()
    config = get_model_config(IMAGE_SIZE)
    transformer = PixCellTransformer2DModelLoRA(**config)
    state_dict = load_file(CHECKPOINT_PATH)
    transformer.load_state_dict(state_dict, strict=True)
    transformer = transformer.to(device=DEVICE, dtype=DTYPE).eval()
    uni_model, uni_transform = load_uni_model(str(paths.UNI2H_CHECKPOINT))
    uni_model = uni_model.to(device=DEVICE, dtype=DTYPE).eval()
    pipeline_path = os.path.join(project_root, "pipeline.py")
    pipeline = DiffusionPipeline.from_pretrained(
                                       str(paths.pixcell_dir(IMAGE_SIZE)),
                                       vae=vae,
                                       custom_pipeline=pipeline_path,
```

```
                                    local_files_only=True,
                                    trust_remote_code=True,
            ).to(DEVICE)
    pipe.transformer = transformer
    pipe.set_progress_bar_config(disable=True)
    return pipeline

def generate_single_image(donor_path, target_path,
                          donor_mask_path, angle, pos_x, pos_y):
    resize_transform = T.Resize((IMAGE_SIZE, IMAGE_SIZE),
                                interpolation=T.InterpolationMode.BILINEAR,
                                antialias=True)
    raw_donor_img = resize_transform(Image.open(donor_path).convert("RGB"))
    raw_donor_mask = Image.open(donor_mask_path).convert("L")
    raw_donor_mask = raw_donor_mask.resize((IMAGE_SIZE, IMAGE_SIZE),
                                           resample=Image.NEAREST)
    target_img = resize_transform(Image.open(target_path).convert("RGB"))

    donor_img, donor_mask = transform_donor_and_mask(raw_donor_img,
                                                     raw_donor_mask,
                                                     angle, pos_x,
                                                     pos_y, IMAGE_SIZE)
    color_matched_donor = smart_color_match_v2(donor_img, target_img,
                                               donor_mask,
                                               blend_strength=0.85)
    don_t = T.ToTensor()(color_matched_donor).to(device=DEVICE, dtype=DTYPE)
    tgt_t = T.ToTensor()(target_img).to(device=DEVICE, dtype=DTYPE)
    mask_t = T.ToTensor()(donor_mask).to(device=DEVICE, dtype=DTYPE)
    mask_bin = (mask_t > 0.1).float()
    raw_comp = tgt_t * (1.0 - mask_bin) + don_t * mask_bin
    raw_comp_pil = T.ToPILImage()(raw_comp.squeeze().cpu().clamp(0, 1))

    mask_np = mask_bin.squeeze().cpu().numpy().astype(np.uint8)
    inside_dist = cv2.distanceTransform(mask_np, cv2.DIST_L2, 5)
    outside_dist = cv2.distanceTransform(1 - mask_np, cv2.DIST_L2, 5)

    sigma = WALL_SIGMA
    wall_np = np.exp(-(outside_dist ** 2) / (2 * sigma ** 2)) * \
              np.exp(-(inside_dist ** 2) / (2 * sigma ** 2))

    wall_np = wall_np / (wall_np.max() + 1e-8)
    ai_influence = torch.tensor(wall_np, device=DEVICE,
                                dtype=DTYPE).unsqueeze(0)
    with torch.no_grad(), torch.autocast(device_type='cuda', dtype=DTYPE):
        uni_emb = extract_uni_embeddings(raw_comp.unsqueeze(0),
                                         uni_model,uni_transform,
                                         DEVICE, IMAGE_SIZE)
        neg_emb = pipe.get_unconditional_embedding(batch_size=1)

        generated = pipe(
            image=raw_comp_pil,
            strength=STRENGTH,
            uni_embeds=uni_emb,
            negative_uni_embeds=neg_emb,
            guidance_scale=CFG_SCALE,
            num_inference_steps=STEPS,
            height=IMAGE_SIZE,
            width=IMAGE_SIZE,
        ).images[0]

        gen_t = T.ToTensor()(generated).to(device=DEVICE, dtype=DTYPE)
        final_tensor = raw_comp * (1.0 - ai_influence) +
                       gen_t * ai_influence
        final_img = T.ToPILImage()(final_tensor.squeeze().cpu().clamp(0, 1))

    final_mask_pil = donor_mask.point(lambda p: 255 if p > 127
                                      else 0).convert('L')
    return final_img, final_mask_pil

if __name__ == "__main__":
```

```
    MASK_DIR = "./data/masks"
    JSON_PATH = "./data/target_to_donor_mapping_v2.json"
    OUTPUT_DIR = "./data/target_to_donor_out"
    TILES_DIR = os.path.join(OUTPUT_DIR, "tiles")
    MASKS_DIR = os.path.join(OUTPUT_DIR, "masks")
    CHECKPOINT_PATH = "./data/checkpints/diffusion_model.safetensors"

    with open(JSON_PATH, "r", encoding="utf-8") as f:
    target_to_donor_map = json.load(f)
    pairs = list(target_to_donor_map.items())

    actual_target_count = min(TARGET_COUNT, len(pairs))

    gen_cnt = 0
    pbar = tqdm(total=actual_target_count, desc="Генерация", unit="img")

    for target_path, donor_path in pairs:
        if gen_cnt >= actual_target_count:
            break
        donor_mask_path = get_mask_for_img(donor_path, MASK_DIR)
        if not donor_mask_path:
            continue
        try:
            tgt_bgr = cv2.imread(target_path)
            don_bgr = cv2.imread(donor_path)
            don_mask_gray = cv2.imread(donor_mask_path,
                                       cv2.IMREAD_GRAYSCALE)

            tgt_angle = get_dominant_nuclei_angle(tgt_bgr)
            don_angle = get_dominant_nuclei_angle(don_bgr, don_mask_gray)
            optimal_rotation = (don_angle - tgt_angle)

            if random.choice([True, False]):
                optimal_rotation += 180

            TARGET_ANGLE = optimal_rotation % 360

            pos_x, pos_y = get_smart_coordinates(target_path,
                                                 donor_mask_path,
                                                 IMAGE_SIZE)

            final_image, final_mask = generate_single_image(
                donor_path=donor_path,
                target_path=target_path,
                donor_mask_path=donor_mask_path,
                angle=TARGET_ANGLE,
                pos_x=pos_x,
                pos_y=pos_y
            )

            tgt_name = os.path.splitext(os.path.basename(target_path))[0]
            dnr_name = os.path.splitext(os.path.basename(donor_path))[0]
            save_name = f"gen_{gen_cnt}_tgt-{tgt_name}_don-{dnr_name}.png"

            final_image.save(os.path.join(TILES_DIR, save_name))
            final_mask.save(os.path.join(MASKS_DIR, save_name))
            gen_cnt += 1
            pbar.update(1)
            if gen_cnt % 10 == 0:

        except Exception as e:
            print(f"\n Error: {os.path.basename(donor_path)}: {e}")
        continue

    pbar.close()
    print(f"\nFinish {gen_cnt} successfully generated.")
```

Приложение Г

Исходный код модуля для интеграции механизма KV-компрессии в слои самовнимания (self-attention)

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from diffusers.models.attention_processor import AttnProcessor2_0
# spatial key/value compression: O(N^2) -> O(N^2 / R^2)
class KVCompressAttnProcessor(AttnProcessor2_0):
    def __init__(self, hidden_size, sr_ratio=2):
        super().__init__()
        self.sr_ratio = sr_ratio
        if sr_ratio > 1:
            # depthwise conv initialized as average pooling (conv-avg-init)
            self.sr = nn.Conv2d(hidden_size, hidden_size,
                                groups=hidden_size,
                                kernel_size=sr_ratio, stride=sr_ratio)
            self.sr.weight.data.fill_(1.0 / sr_ratio ** 2)
            self.sr.bias.data.zero_()
            self.norm = nn.LayerNorm(hidden_size)

    def _downsample(self, tokens, H, W):
        # (B, N, C) -> spatial conv with stride R -> (B, N/R^2, C)
        if self.sr_ratio <= 1:
            return tokens
        B, _, C = tokens.shape
        x = tokens.permute(0, 2, 1).view(B, C, H, W)
        x = self.sr(x).flatten(2).transpose(1, 2)
        return self.norm(x)

    def forward(self, attn, hidden_states, encoder_hidden_states=None,
                attention_mask=None, **kwargs):
        # compression only for self-attention
        # delegate cross-attention to the base processor
        if encoder_hidden_states is not None or self.sr_ratio <= 1:
            return super().__call__(
                                attn,
                                hidden_states,
                                encoder_hidden_states=encoder_hidden_states,
                                attention_mask=attention_mask, **kwargs)
        B, N, _ = hidden_states.shape
        H = W = int(N ** 0.5)
        q, k, v = attn.to_q(hidden_states),
                  attn.to_k(hidden_states),
                  attn.to_v(hidden_states)
        # optional q/k normalization if present in the model
        if getattr(attn, "norm_q", None) is not None:
            q = attn.norm_q(q)
        if getattr(attn, "norm_k", None) is not None:
            k = attn.norm_k(k)
        # compress keys and values spatially
        k, v = self._downsample(k, H, W), self._downsample(v, H, W)
        d = k.shape[-1] // attn.heads
        q = q.view(B, N, attn.heads, d).transpose(1, 2)
        k = k.view(B, -1, attn.heads, d).transpose(1, 2)
        v = v.view(B, -1, attn.heads, d).transpose(1, 2)
        # memory-efficient exact attention
        out = F.scaled_dot_product_attention(q, k, v).transpose(1, 2)
        out = out.reshape(B, N, -1).to(q.dtype)
        out = attn.to_out[1](attn.to_out[0](out))   # projection + dropout
        return out
```

Приложение Д

Исходный код модуля для предварительного вычисления и кэширования эмбеддингов

```
import gc
import os
import warnings
from concurrent.futures import ThreadPoolExecutor, as_completed
from pathlib import Path

import h5py
import numpy as np
import torch
from torch.utils.data import Dataset

from histo_diffusion.dataset.datasets import BaseH5Dataset

# offline: store vae posterior parameters
#          (mean | std, 32 channels), not a sample
@torch.inference_mode()
def precompute_features(image, vae, vae_transform, uni_model,
                        uni_transform, out_dir, stem):
    posterior = vae.encode(vae_transform(image)).latent_dist
    mean_std = torch.cat([posterior.mean, posterior.std], dim=1)
    np.save(out_dir / f"{stem}_sd3_vae.npy", mean_std.half().cpu().numpy())
    #store uni2-h visual condition
    uni = extract_uni_embeddings(image, uni_model, uni_transform,
                                 image.device, 1024)
    np.save(out_dir / f"{stem}_uni.npy", uni.half().cpu().numpy())


# return available system memory in gib
def _available_ram_gb():
    try:
        import psutil
        return psutil.virtual_memory().available / 1024 ** 3
    except ImportError:
        with open("/proc/meminfo") as f:
            for line in f:
                if line.startswith("MemAvailable:"):
                    return int(line.split()[1]) / 1024 ** 2
    return float("inf")

# shared ram-cache infrastructure on top of the hdf5-indexed base dataset
class _CachedBase(BaseH5Dataset):
    _FILES_PER_SAMPLE = 2
    def _collect_all_paths(self):
        with h5py.File(self.h5_path, "r") as hf:
            decoded = {ds: [s.decode("utf-8") for s in hf[ds][:]]
                       for ds in self.dataset_names}
        n_pos = len(self.indices) if self.indices is not None
                                  else self.total_len
        result = []
        for pos in range(n_pos):
            global_idx = self.indices[pos] if self.indices is not None
                                           else pos
            ds_name, local_idx = self._resolve(global_idx)
            result.append((decoded[ds_name][local_idx], ds_name))
        return result

    # estimate cache size from the first 8 files and verify available ram
```

```
    def _check_memory(self, n_samples, sample_paths, safety_gb=20.0):
        total_bytes, counted = 0, 0
        for img_path, ds_name in sample_paths[:8]:
            stem = Path(img_path).stem
            features_dir = self.root / ds_name / getattr(self,
                                                         "features_subdir",
                                                         "features")
            for suffix in ("_uni.npy", "_sd3_vae.npy"):
                p = features_dir / f"{stem}{suffix}"
                if p.exists():
                    total_bytes += p.stat().st_size
                    counted += 1
        avg_bytes = (total_bytes / counted) if counted else 600_000
        estimated = (avg_bytes * self._FILES_PER_SAMPLE * n_samples) /
                    1024 ** 3
        if estimated + safety_gb > _available_ram_gb():
            raise MemoryError(f"not enough ram: need ~{estimated:.1f} gb")
        return estimated

    # load every sample concurrently; args[0] is the output position
    def _run_parallel(self, fn, args_list, n_workers):
        results, failed = [None] * len(args_list), []
        with ThreadPoolExecutor(max_workers=n_workers) as executor:
            future_to_pos = {executor.submit(fn, a): a[0]
                             for a in args_list}
            for future in as_completed(future_to_pos):
                pos = future_to_pos[future]
                try:
                    results[pos] = future.result()
                except Exception as exc:
                    failed.append((pos, str(exc)))
                    warnings.warn(f"load failed at pos={pos}: {exc}")
        return results, failed

    # move cached tensors to page-locked memory
    def _pin_list(self, lst):
        return [t.pin_memory() if t is not None else None for t in lst]

    def free_cache(self):
        # release the ram cache explicitly
        for attr in ("_vae_cache", "_uni_cache", "_mask_cache", "_names"):
            if hasattr(self, attr):
                delattr(self, attr)
        gc.collect()

# load one sample's (uni, vae) latents; np.load releases the gil
def _load_features_worker(args):
    pos, img_path, features_dir, dtype = args
    stem = Path(img_path).stem
    uni = torch.from_numpy(np.load(features_dir /
                                   f"{stem}_uni.npy")).to(dtype)
    vae = torch.from_numpy(np.load(features_dir /
                                   f"{stem}_sd3_vae.npy")).to(dtype)
    return pos, uni, vae

class CachedFeaturesDataset(_CachedBase):
    _FILES_PER_SAMPLE = 2

    def __init__(self, root, dataset_names, features_subdir="features",
                 use_fp16=True, pin_memory=False, num_load_workers=None,
                 safety_margin_gb=20.0, **kwargs):
        super().__init__(root=root, dataset_names=dataset_names, **kwargs)
        self.features_subdir = features_subdir
        n_total = super().__len__()
        n_workers = num_load_workers or min(32, os.cpu_count() or 4)
        dtype = torch.float16 if use_fp16 else torch.float32

        # read all feature paths from hdf5, verify ram, load in parallel
        all_paths = self._collect_all_paths()
        self._check_memory(n_total, all_paths, safety_margin_gb)
```

```
        args_list = [
            (pos, img_path, self.root / ds_name / features_subdir, dtype)
            for pos, (img_path, ds_name) in enumerate(all_paths)
        ]
        results, failed = self._run_parallel(_load_features_worker,
                                             args_list, n_workers)
        # assemble caches
        self._uni_cache = [r[1] if r else None for r in results]
        self._vae_cache = [r[2] if r else None for r in results]
        self._names = [p for p, _ in all_paths]

        # drop positions that failed to load
        if failed:
            bad = {pos for pos, _ in failed}
            self._uni_cache = [t for i, t in enumerate(self._uni_cache)
                               if i not in bad]
            self._vae_cache = [t for i, t in enumerate(self._vae_cache)
                               if i not in bad]
            self._names = [n for i, n in enumerate(self._names)
                           if i not in bad]
            warnings.warn(f"skipped {len(failed)} samples load errors")

        # optional page-locked memory for faster dma to gpu
        if pin_memory:
            self._uni_cache = self._pin_list(self._uni_cache)
            self._vae_cache = self._pin_list(self._vae_cache)

    # before __init__ finishes the cache is not built yet
    # -> delegate to parent.
    # afterwards the cache is the source of truth
    # (accounts for failed loads)
    def __len__(self):
        if hasattr(self, "_vae_cache"):
            return len(self._vae_cache)
        return super().__len__()

    def __getitem__(self, idx):
        # o(1) read from ram
        return {
            "uni": self._uni_cache[idx],       # (N, 1536)
            "sd3_vae": self._vae_cache[idx],   # (32, H, W) -> mean | std
            "name": self._names[idx],
        }


def sample_latent(mean_std, shift, scale):
    # training-time reparameterization
    # z = mu + sigma * eps (preserves stochasticity)
    mean, std = mean_std[:, :16], mean_std[:, 16:]
    z = mean + std * torch.randn_like(mean)
    return (z - shift) * scal
```