РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ «НЕЙРОМОЛЬБЕРТ»

Мамонтов И.С. [1]*, Хорошко А.Е.* [2]*,*
Научный руководитель: Брагин А.Д. [3]
[1]*ТПУ ИШИТР ОИТ, гр. 8К21, e-mail: ism18@tpu.ru*
[2]*ТПУ ИШИТР ОИТ, гр. 8К21, e-mail: aek60@tpu.ru*
[3]*ТПУ ИШИТР ОИТ, старший преподаватель, e-mail: bragin@tpu.ru*

Аннотация

«Нейромольберт» – интерактивное веб-приложение для демонстрации возможностей генеративного ИИ, позволяющее создавать уникальные графические произведения посредством ручного рисования и автоматической генерации изображений. Разработано с использованием Vue.js, Flask ориентировано на образовательные и культурные мероприятия для популяризации инновационных технологий.

Ключевые слова: генеративный ИИ, веб-приложение, генерация изображений, Vue.js, Flask, stable diffusion.

Введение

В условиях активного развития искусственного интеллекта (ИИ) и растущего интереса к инновационным технологиям, необходимо предоставлять наглядные инструменты для демонстрации его возможностей широкой аудитории и вовлечения людей в изучение инновационных подходов к использованию генеративного искусственного интеллекта.

Основное назначение такого инструмента – интерактивная демонстрация современных возможностей генеративного искусственного интеллекта. Целевая аудитория - посетители различных мероприятий всех возрастных категорий.

Цель данного проекта – создать интерактивное веб-приложение, которое позволит пользователям рисовать эскизы вручную и получать на их основе художественные изображения, сгенерированные нейросетевой моделью. Такой подход способствует повышению уровня понимания возможностей современных технологий, в частности генеративного искусственного интеллекта, и их популяризации среди широкой аудитории.

Выбор стека разработки

Для реализации сервера обработки запросов используется легкий фреймворк Flask, который позволяет быстро настраивать маршруты и API [1]. За счет своей простоты Flask хорошо подходит для прототипирования и небольших/средних проектов. В данном случае он отвечает за загрузку конфигураций и обработку запросов (таких как получение списка стилей и генерация изображений).

Для развертывания используется WSGI-сервер Waitress, который позволяет запустить Flask-приложение в production-режиме. Сервер хорошо подходит для разрабатываемого приложения, так как он надежен и прост в использовании, а само приложение не подразумевает большое количество запросов со стороны клиента и очень серьезных нагрузок.

Для реализации клиентской части веб-приложения был выбран Vue.js – современный, прогрессивный фреймворк, позволяющий строить разделенные компонентные интерфейсы. Такой подход обеспечивает быструю реакцию веб-приложения, удобное управление состоянием на уровне компонентов и легкость дальнейших расширений [2]. Компонентный дизайн также позволяет изящно организовать взаимодействие с пользователем, реализовать поддержку динамического обновления и плавных анимационных переходов.

Генерация изображений осуществляется с помощью современных моделей (Stable Diffusion XL [3]) через библиотеку diffusers, что в паре с PyTorch дает возможность использовать GPU и выполнять вычисления в float16 для оптимизации потребления памяти. Библиотеки Pillow (PIL) и NumPy используются на этапе предобработки изображений.

Использование веб-стека для основы приложения позволяет обеспечить ряд преимуществ:

- Приложение доступно в браузере – пользователю не требуется устанавливать дополнительное ПО, также это дает возможность в дальнейшем масштабировании на несколько устройств.
- Современные веб-технологии позволяют создавать адаптивный и отзывчивый интерфейс с широким набором визуальных эффектов и ускоряют процесс разработки по сравнению с desktop-фреймворками.

Разработка сервера обработки запросов

Сервер принимает POST-запросы с эскизом изображения в формате base64 и параметрами генерации (такими как выбранный стиль, размер изображения). Изображение конвертируется и предварительно обрабатывается (инвертируются цвета, происходит масштабирование).

С использованием библиотеки diffusers и PyTorch сервер загружает модели (Stable Diffusion XL, ControlNet, VAE) с оптимизациями для работы на GPU (float16, очистка кэша). Вызывается метод генерации изображения, в котором применяется выбранный стиль и вычислительные параметры.

Сгенерированное изображение сохраняется во временный буфер (с использованием BytesIO) и отправляется клиенту в формате PNG через Flask-функцию send_file().

Основные параметры сервера (такие как список стилей и настройки модели) вынесены в JSON-файлы конфигурации, что облегчает обновление и масштабирование решения.

Разработка интерфейса веб-приложения

Интерфейс разбит на независимые компоненты – холст для рисования, панель инструментов, панель выбора стилей и переключатель языка. Такой подход повышает модульность, упрощает тестирование и облегчает дальнейшее расширение функциональности.

Пользовательский интерфейс организован с использованием динамических средств обновления: графические элементы реагируют на действия пользователя, например, изменение размера кисти, выбор инструментов, переключение между режимами рисования. Для плавной анимации элементов применяются переходы, обеспечивающие более приятное восприятие.

Компонент холста реализует функционал рисования с поддержкой различных событий указателя. Он обеспечивает сохранение и восстановление состояний (undo/redo), а также масштабируется динамически при изменении размеров, сохраняя текущее содержимое.

При нажатии кнопки «генерировать изображение» на серверную часть отправляется запрос на генерации изображений. В процессе обработки отображается анимация загрузки, а по окончании работы результат выводится на экран. Асинхронный подход позволяет отделить вычисления от пользовательского интерфейса и обеспечить плавность работы приложения.

Компоненты, отвечающие за переключение языка и выбор стилей, обеспечивают динамическое изменение текстов и визуальных настроек. Благодаря использованию событий и передачи данных через свойства, интерфейс мгновенно реагирует на изменения предпочтений пользователя.

Снимок экрана интерфейса приложения представлен на рисунке 1.

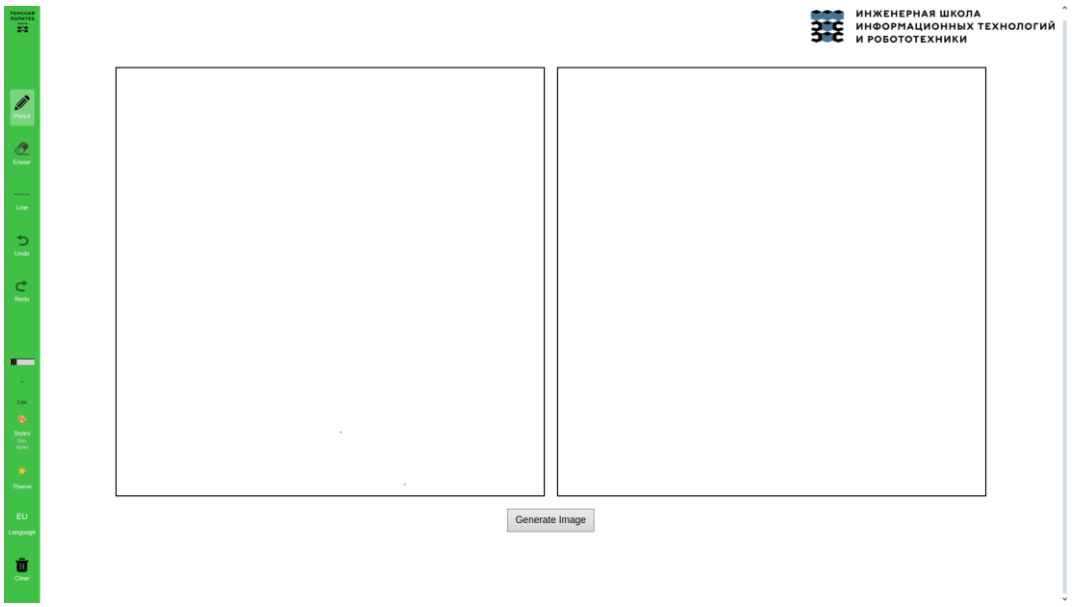

Рис. 1. Интерфейс веб-приложения

Описанное выше решение позволяет создать гибкий, отзывчивый и легко поддерживаемый пользовательский интерфейс, который обеспечивает доступное взаимодействие с функционалом генерации изображений и рисования.

Выбор генеративной модели

Для генерации изображений был проведен анализ современных моделей, среди которых выделялась архитектура Stable Diffusion за ее высокое качество и гибкость в управлении стилями, а также удовлетворительные требования к аппаратной части [3]. Модель позволяет работать с текстовыми запросами, а также интегрироваться с ControlNet для дополнительного управления результатом. Архитектура модели представлена на рисунке 2.

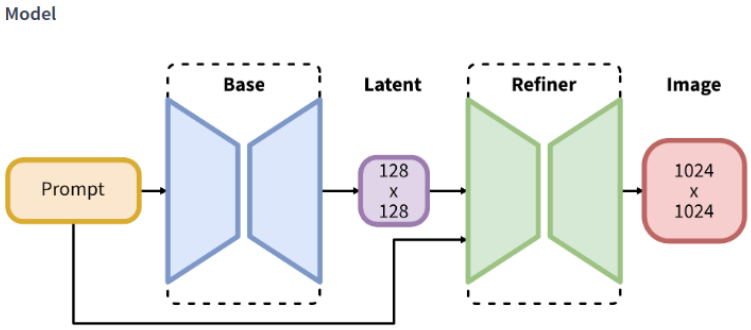

Рис. 2. Архитектура модели Stable Diffusion XL

Модель оптимизирована для работы на GPU с использованием вычислений в float16, что снижает расход памяти и ускоряет процесс генерации. Итоговая интеграция модели происходит через библиотеку diffusers и PyTorch, что обеспечивает стабильное выполнение вычислительно-тяжелых операций через веб-интерфейс.

Разработка стилей для генерации

В каждом запросе (prompt) задается определенный стиль для генерации изображений. Например, "Legendary Adventure" и "Epic Saga" направляют модель на создание эпических, героических, мифологических сцен, добавляя ключевые слова вроде "epic, mythical, grand, heroic, detailed".

Каждый такой запрос делится на две составляющих – позитивную и негативную. Такое разделение позволяет более гибко управлять конечным результатом генерации.

Позитивные ключевые слова помогают моделям фокусироваться на желаемых характеристиках. В "Epic Saga" упоминаются "vibrant, detailed, magical, adventure, epic scale", что задает тон и настроение изображения.

Негативные ключевые слова исключают нежелательные характеристики. Например, "boring, mundane, contemporary, plain, realistic" в "Legendary Adventure" помогают избежать скучных или реалистичных элементов, которые не соответствуют стилю, а также помогают ограничить нежелательные сцены, так как веб-приложение ориентированно в том числе на школьную аудиторию. Такой формат prompt позволяет легко изменять стиль изображения, просто выбрав другой шаблон или добавив новые параметры. На рисунке 3 представлен фрагмент JSON-файла, содержащего стили для генерации изображений.

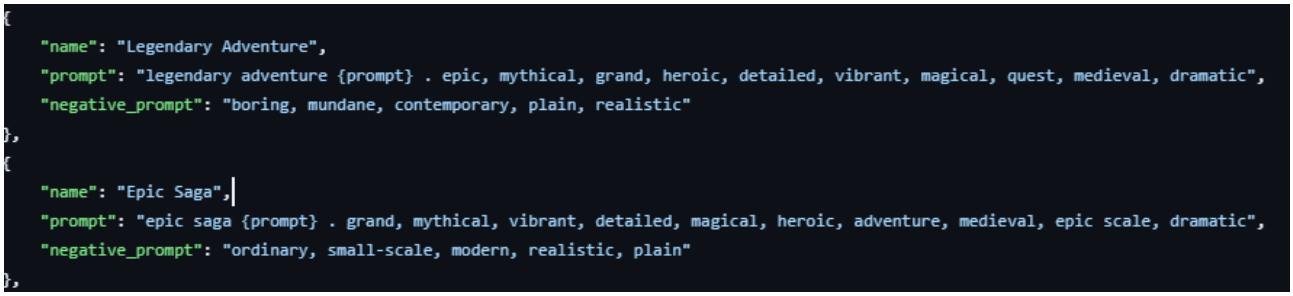

Рис. 3. Запрос для генерации изображения в определенном стиле

Интеграция генеративной модели в веб-приложение

С помощью Flask создается специальный маршрут (/generate), который принимает POST-запросы с данными (изображение в формате base64 и параметры генерации, такие как выбранный стиль).

В серверном коде изображение декодируется из base64, конвертируется в нужный формат, выполняется масштабирование и корректировка (например, замена белого фона на прозрачный), чтобы подготовить данные для модели.

После подготовки данных происходит вызов генеративной модели, интегрированной через библиотеку diffusers и PyTorch. Переобученная модель, настроенная для работы с заданными стилями, генерирует новое изображение с использованием оптимизированных вычислений (float16, GPU) [4].

Сгенерированное изображение сохраняется во временный буфер и отправляется обратно клиенту в виде PNG-файла с помощью Flask-функции send_file(). Это обеспечивает быструю доставку результата в веб-интерфейс. Фронтенд, реализованный на Vue.js, отправляет запрос на сервер при нажатии кнопки генерации, отображает анимацию загрузки и, по завершении обработки, получает сгенерированное изображение для отображения пользователю.

Результаты

Пользователи могут напрямую взаимодействовать с веб-приложением, выполнять рисование на холсте, настраивать инструменты и выбирать стили генерации через удобную панель инструментов.

При отправке изображения с выбранными параметрами сервер обрабатывает запрос, применяет генеративную модель (на базе Stable Diffusion XL с дообучением и оптимизациями для работы на GPU) и возвращает сгенерированное изображение в формате PNG. Это обеспечивает быстрый и наглядный результат.

Разделение логики на серверную и клиентскую части, использование внешних JSON-конфигураций и настройка через API позволяют легко обновлять и расширять функционал, не затрагивая всю систему целиком.

Итоговый продукт сочетает вычислительно-тяжелые операции глубокого обучения с динамичным веб-интерфейсом, что обеспечивает комфортное взаимодействие для пользователей – от художников до любителей экспериментов с генеративным искусством.

Для исполнения изначальной цели проекта, то есть популяризации генеративного ИИ, была собрана установка, состоящая из монитора Samsung с сенсорным экраном, системного блока с видеокартой NVidia RTX4070 12gb, для представления проекта на выставках.

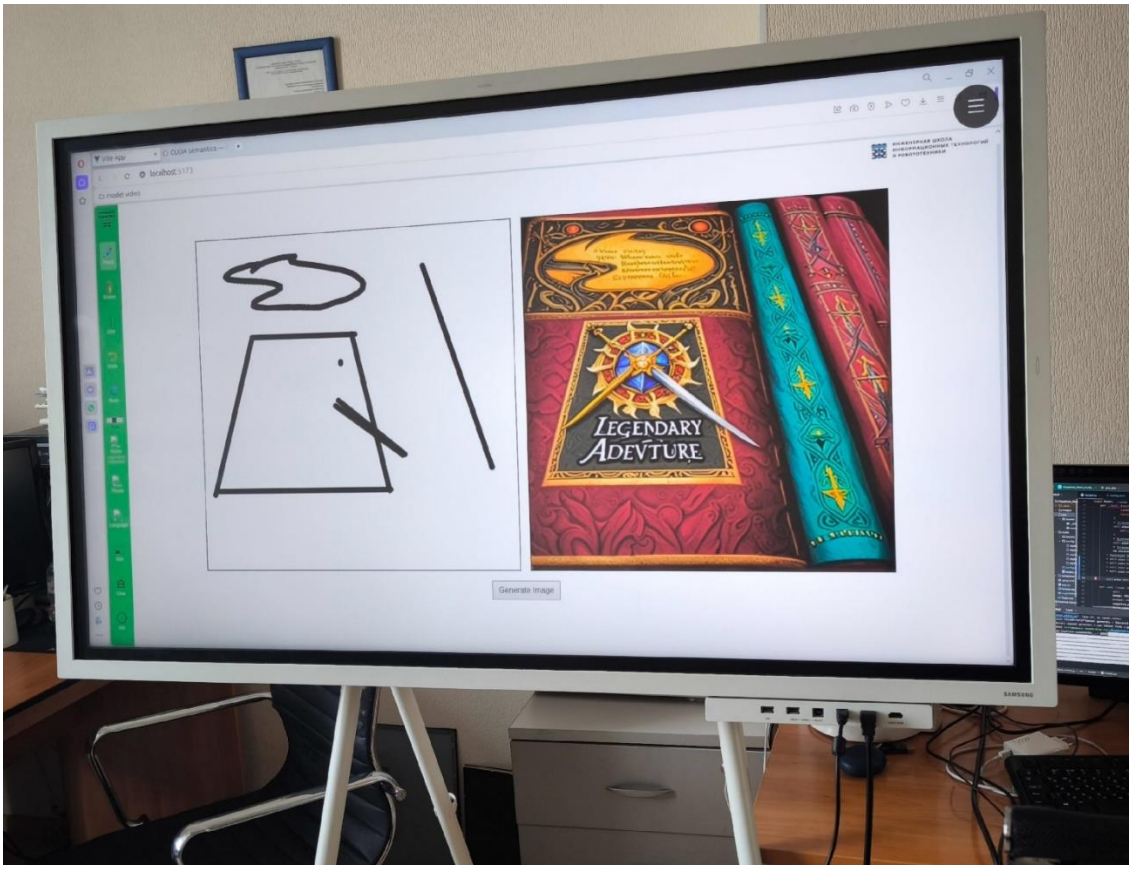

Рис. 4. Стенд для демонстрации веб-приложения «Нейромольберт»

Заключение

Разработанное приложение в сочетании с панелью для рисования позволяют наглядно демонстрировать широкой публике возможности современных генеративных технологий. Эта установка была успешно опробована на нескольких общественных мероприятиях и подарила сотням пользователей увлекательный опыт. «Нейромольберта» представляет собой комплексный процесс, охватывающий проектирование, создание прототипа, интеграцию генеративной модели, оптимизацию и масштабирование системы. Этот детальный подход обеспечивает создание продукта, который не только демонстрирует современные возможности генеративного ИИ, но и позволяет пользователям активно участвовать в творческом процессе.

Список использованной литературы

1. Flask // Документация: сайт. – 2025. – [Электронный ресурс]. – URL: flask.palletsprojects.com/en/stable/ (дата обращения: 09.04.2025).

2. Vue.js // Руководство: сайт. – 2025. – [Электронный ресурс]. – URL: vuejs-doc-ru.vercel.app/guide/introduction.html (дата обращения: 09.04.2025).

3. Stable Diffusion XL Base 1.0 // Hugging Face: сайт. – 2025. – [Электронный ресурс]. – URL: huggingface.co/stabilityai/stable-diffusion-xl-base-1.0 (дата обращения: 09.04.2025).

4. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis // ArXiv: научная статья. – 2025. – [Электронный ресурс]. – URL: arxiv.org/html/2307.01952 (дата обращения: 09.04.2025).