КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення
на тему:

РОЗРОБКА TELEGRAM-БОТА ДЛЯ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

Виконав студент 4-го курсу
Анатолій МАЛІБРОДА _____ (підпис)

Науковий керівник:
асистент, кандидат фіз.-мат. наук
Костянтин ЖЕРЕБ _____ (підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____ (підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«29» травня 2023 р.,
протокол №11

Завідувач кафедри
Олександр ПРОВОТАР _____ (підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 54 сторінок, 18 ілюстрацій, 27 джерел посилань.

ВАРІАЦІЙНІ АВТОКОДУВАЛЬНИКИ, ГЕНЕРАТИВНІ ЗМАГАЛЬНІ МЕРЕЖІ, ГЕНЕРАЦІЯ ЗОБРАЖЕНЬ, НЕЙРОННІ МЕРЕЖІ, PYTHON, STABLE DIFFUSION, TELEGRAM-БОТ.

Об'єктом дослідження та розробки в даній випускній кваліфікаційній роботі є генерація зображень за текстовим описом з використанням нейронних мереж. Предметом роботи є програмний засіб, що реалізовує генерацію зображень та взаємодію з користувачем за допомогою додатку Telegram та інтерфейсу чат-бота.

Основною метою даної роботи є проектування, реалізація та оцінка програмної системи, яка дозволяє користувачам генерувати зображення з використанням нейромережевих архітектур через зручний інтерфейс Telegram-бота.

Для досягнення цієї мети були використані різні методи та інструменти, зокрема мова програмування Python, бібліотека AIOGram, що реалізує зв'язок з Telegram Bot API, бібліотека Diffusers для реалізації та інтеграції нейронних мереж. Дослідження включало вивчення існуючої літератури з розробки Telegram-ботів та нейронних мереж для генерації зображень, а також аналіз суміжних робіт та рішень. Інформація про реалізацію бота, включаючи інструменти розробки, методи та код, детально представлена у роботі.

Ця робота пов'язана з існуючими дослідженнями в галузі нейронних мереж, генерації зображень та розробки Telegram-ботів і ґрунтується на них. Результати роботи можуть бути використані розробниками та дослідниками, зацікавленими у створенні подібних додатків, а також користувачами, які прагнуть генерувати зображення за допомогою нейронних мереж через зручний та доступний інтерфейс.

Сфера застосування цієї роботи включає генерацію зображень для художніх цілей, створення контенту та доповнення даних для задач машинного навчання. Значення цієї роботи полягає в тому, що вона може зробити генерацію зображень на основі нейронних мереж більш доступною для широкої аудиторії та надати уявлення про продуктивність сучасних архітектур нейронних мереж.

Подальша робота може бути зосереджена на розширенні можливостей бота, підвищенні ефективності та якості згенерованих зображень, а також дослідженні додаткових застосувань цієї технології.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

AI – Artificial Intelligence, штучний інтелект;
API – Application Programming Interface, прикладний програмний інтерфейс;
DM – Diffusion Model, дифузійна модель;
GAN – Generative Adversarial Network, генеративна змагальна мережа;
LDM – Latent Diffusion Model, модель прихованої дифузії;
ML – Machine Learning, машинне навчання;
UI – User Interface, інтерфейс користувача;
VAE – Variational Autoencoder, варіаційний автокодувальник.

ВСТУП

Оцінка поточного стану об'єкта дослідження або розробки. Стрімкий розвиток технологій штучного інтелекту (AI) та машинного навчання (ML) в останні роки призвів до значного прогресу в галузі генерації зображень за допомогою нейронних мереж [1]. З'явилося багато застосувань, починаючи від створення контенту і закінчуючи доповненням даних для навчання ML-моделей. Серед найпопулярніших нейромережевих архітектур для генерації зображень – генеративні змагальні мережі (GAN) [1] та варіаційні автокодери (VAE) [2].

Водночас платформи обміну повідомленнями, такі як Telegram, набули широкого поширення: у 2022 році Telegram повідомляв про понад 700 мільйонів щомісячних активних користувачів, що становить 40% зростання у порівнянні з 2021 р. [3]. Таке зростання робить доцільним розробку чат-ботів для надання користувачам різноманітних послуг і функцій, у тому числі пов'язаних зі штучним інтелектом і машинним навчанням.

Актуальність роботи та підстави для її виконання. Незважаючи на прогрес як у сфері генерації зображень, так і платформ для обміну повідомленнями, бракує доступних і зручних рішень, які б інтегрували ці технології. Розробляючи Telegram-бота для генерації зображень за допомогою нейронних мереж, ця робота має на меті заповнити цю прогалину та надати користувачам доступ до можливостей передових нейромережевих архітектур для генерації зображень за допомогою знайомої та широко використовуваної платформи. Крім того, ця робота сприяє більш широкому розумінню можливостей і потенційних застосувань нейронних мереж.

Мета та завдання роботи. Метою даної роботи є розробка, реалізація та оцінка програмної системи, яка дозволяє користувачам генерувати зображення з використанням сучасної нейромережевої архітектури за допомогою зручного інтерфейсу бота Telegram. Для досягнення цієї мети необхідно вирішити наступні завдання:

- провести огляд існуючої літератури з розробки Telegram-ботів та нейронних мереж для генерації зображень;
- проаналізувати функціональні та нефункціональні вимоги до Telegram-бота;
- розробити архітектуру системи та потік взаємодії з користувачем;
- реалізувати Telegram-бота та нейромережеві компоненти за допомогою

Python;

– оцінити продуктивність та ефективність системи.

Об’єкт, методи та засоби дослідження або розробки. Основним об’єктом цього дослідження є розробка Telegram-бота, здатного генерувати зображення за допомогою нейромережевих архітектур, зокрема Stable Diffusion. Будуть використані різні методи та інструменти, включаючи мову програмування Python, Telegram Bot API, бібліотеки AIOGram для реалізації бота та Diffusers для реалізації нейронних мереж.

Можливі сфери застосування. Розроблений Telegram-бот має потенційне застосування в різних галузях, зокрема:

– мистецтво: створення унікальних зображень, згенерованих штучним інтелектом, для художніх цілей;

– створення контенту: створення зображень для використання в маркетингу, соціальних мережах та інших цифрових платформах;

– розширення даних: покращення наборів даних для навчання ШІ шляхом створення додаткових, різноманітних зразків.

Зв’язок з іншими роботами. Ця випускна кваліфікаційна робота ґрунтується на існуючих дослідженнях у галузі нейронних мереж, генерації зображень та розробки Telegram-ботів. Вона також робить внесок у зростаючу кількість знань про інтеграцію технологій AI та ML в платформи обміну повідомленнями.

Інтегруючи досягнення в галузі генерації зображень за допомогою нейронних мереж зі зростаючою популярністю платформ обміну повідомленнями, таких як Telegram, ця робота має на меті надати нове, зручне для користувача рішення для генерації зображень на вимогу. Результати цієї роботи можуть надихнути на подальше вивчення потенційних застосувань технологій штучного інтелекту та машинного навчання в платформах обміну повідомленнями та за їх межами.

РОЗДІЛ 1 ОГЛЯД ЛІТЕРАТУРИ

1.1 Огляд розробки Telegram-ботів

Telegram – це хмарний сервіс обміну миттєвими повідомленнями та голосового зв'язку, запущений у 2013 році російським підприємцем Павлом Дуровим [4]. Він пропонує безпечну та швидку платформу для обміну повідомленнями з акцентом на конфіденційність та зручність користування. Однією з відмінних рис Telegram є підтримка ботів – програмних додатків, які виконують автоматизовані завдання і взаємодіють з користувачами через платформу обміну повідомленнями [5].

Боти в Telegram можуть виконувати різні завдання, наприклад, надавати інформацію, керувати групами, надавати клієнтську підтримку і навіть інтегруватися із зовнішніми сервісами. Telegram Bot API дозволяє розробникам створювати та керувати ботами, надаючи простий і потужний інтерфейс для взаємодії з платформою [6].

У цьому підрозділі буде надано огляд ключових концепцій та компонентів, задіяних у розробці Telegram-ботів, включаючи API ботів, процес створення ботів та обробку взаємодії з користувачами.

1.1.1 API Telegram-ботів

Telegram Bot API – це RESTful API, який дозволяє розробникам програмно взаємодіяти з платформою Telegram. Він надає методи для надсилання повідомлень, управління чатами, отримання оновлень та виконання різних інших дій від імені бота. Щоб використовувати API, розробники повинні отримати ключ API, створивши нового бота через BotFather – спеціальний Telegram-бот, який слугує інструментом для створення та управління ботами [6].

Отримавши ключ API, розробник може надсилати HTTP-запити на базову URL-адресу API `https://api.telegram.org/bot<token>/`, додаючи бажаний метод і параметри. API підтримує різні типи повідомлень, включаючи текст, зображення, відео та документи, а також вбудовані клавіатури та інші інтерактивні елементи. Для детального опису доступних методів і типів повідомлень розробники можуть звернутися до офіційної документації [7].

1.1.2 Процес створення бота

Створення Telegram-бота складається з наступних кроків [6]:

а) почати чат з BotFather, знайшовши його ім'я користувача (@BotFather) в Telegram;

б) надіслати команду `/newbot` боту BotFather;

в) дотримуватися інструкцій BotFather, щоб вказати ім'я для бота; Ім'я бота має бути унікальним і закінчуватися на bot (наприклад, "VisuoGenBot");

г) після успішного створення бота, BotFather надасть ключ API. Цей ключ необхідний для взаємодії з API бота і повинен зберігатися в безпеці.

1.1.3 Обробка взаємодії з користувачами

Боти можуть взаємодіяти з користувачами різними способами, наприклад, відповідати на повідомлення або ініціювати дії на основі вбудованих запитів. Для обробки взаємодії з користувачами розробники повинні реалізувати механізм отримання та обробки оновлень від платформи Telegram. Існує два основних методи отримання оновлень: "тривале опитування" та "веб-хуки" [7].

Тривале опитування (англ. "long polling") – цей метод передбачає багаторазове надсилання HTTP-запитів до методу `getUpdates` API бота, очікування відповіді та обробку отриманих оновлень. Цей підхід відносно простий у реалізації, але може спричинити затримку в обробці оновлень і використовувати більше ресурсів порівняно з веб-хуками.

Веб-хук – це URL-адреса зворотного виклику (англ. "callback"), на яку платформа Telegram надсилає оновлення, коли з'являються нові події, які потрібно обробити боту. Цей метод вимагає загальнодоступного сервера для отримання та обробки вхідних оновлень. Веб-хуки можуть запропонувати швидшу та ефективнішу обробку оновлень порівняно з тривалим опитуванням, але вони потребують додаткової інфраструктури та складнішого налаштування.

Після отримання оновлень бот повинен обробити їх і виконати відповідні дії, наприклад, надіслати відповідь, оновити стан чату або викликати зовнішні сервіси. Для полегшення розробки Telegram-ботів розробники можуть використовувати різні бібліотеки та фреймворки, наприклад, AIOGram для Python, Telegraf для Node.js або офіційні клієнтські бібліотеки Telegram Bot API для різних мов програмування [8].

Таким чином, розробка Telegram-бота включає в себе створення

бота, отримання API-ключа і налаштування механізму обробки взаємодії з користувачами через Telegram Bot API. Розробники можуть використовувати потужні можливості і простоту API ботів для створення ботів, які виконують різні завдання і безперешкодно взаємодіють з користувачами. Розуміючи ключові концепції та компоненти, що беруть участь у розробці Telegram-ботів, розробники можуть створювати ботів, які пропонують користувачам цінні послуги та функції, покращуючи їхній досвід роботи на платформі Telegram.

У контексті цієї випускної кваліфікаційної роботи процес розробки Telegram-бота буде включати проектування та реалізацію бота, який дозволяє користувачам генерувати зображення за допомогою нейронних мереж. Це вимагатиме інтеграції бота із зовнішніми бібліотеками та сервісами, пов'язаними з генерацією зображень на основі нейронних мереж, а також розробки зручного інтерфейсу та потоку взаємодії, що сприятиме залученню та задоволенню користувачів. У наступних розділах будуть розглянуті конкретні вимоги, проектування, деталі реалізації та результати оцінки розробленого Telegram-бота для генерації зображень за допомогою нейронних мереж.

1.1.4 Розробка Telegram-ботів на Python за допомогою бібліотеки AIOGram

Python – це універсальна мова програмування, яка широко використовується в різних галузях, включаючи AI, ML та розробку ботів. Бібліотека AIOGram – це популярна і добре підтримувана Python-обгортка для Telegram Bot API, яка спрощує процес створення та управління Telegram-ботами. Вона надає чистий і простий у використанні інтерфейс для взаємодії з платформою Telegram, абстрагуючись від складності створення HTTP-запитів і обробки відповідей API.

Бібліотека AIOGram пропонує декілька функцій, які спрощують процес розробки, а саме:

- високорівневий API зі зручними класами та методами для надсилання повідомлень, управління чатами та обробки оновлень;
- розширювана система обробки команд, що дозволяє розробникам визначати власні команди та відповідні обробники;
- вбудований механізм довготривалого опитування, що дозволяє легко отримувати оновлення без необхідності додаткової інфраструктури або

налаштування;

– підтримка розширених функцій ботів, таких як вбудовані клавіатури, вбудовані запити та запити зі зворотним викликом.

Щоб використовувати бібліотеку AIOGram, можна встановити її через пакетний менеджер pip:

```
pip install aiogram
```

Після встановлення є можливість імпортувати необхідні класи та методи з бібліотеки і почати створювати свого Telegram-бота. Наприклад, простий бот, який повторює повідомлення користувача, може бути реалізований наступним чином:

```
from aiogram import Bot, Dispatcher, executor, types

API_TOKEN = 'BOT TOKEN HERE'

bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)

@dp.message_handler(commands=['start', 'help'])
async def send_welcome(message: types.Message):
    await message.reply("Hi!\nI'm EchoBot!\nPowered by aiogram.")

@dp.message_handler()
async def echo(message: types.Message):
    await message.answer(message.text)

if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```

У даній роботі бібліотека AIOGram буде використана для розробки Telegram-бота для генерації зображень за допомогою нейронних мереж. Ця бібліотека пропонує потужний та зручний інтерфейс для взаємодії з платформою Telegram, що дозволяє ефективно розробляти та керувати ботом. Крім того, сумісність бібліотеки з іншими бібліотеками та фреймворками Python полегшує інтеграцію з необхідними нейромережевими компонентами, забезпечуючи цілісну та надійну програмну систему.

1.2 Нейронні мережі для генерації зображень

Поява штучного інтелекту (AI) і машинного навчання (ML) відкрила нові горизонти в області генерації зображень. Традиційні методи, такі як процедурна генерація та ручна розробка алгоритмів, мають свої обмеження, коли мова йде про створення реалістичних та різноманітних зображень. Глибоке навчання, підмножина ML, продемонструвало величезний потенціал у цій галузі завдяки використанню нейронних мереж, здатних вивчати складні патерни та репрезентації з великих наборів даних.

Нейронні мережі – це обчислювальні моделі, натхненні структурою і функціями біологічних нейронних мереж. Вони складаються з взаємопов'язаних вузлів, або нейронів, організованих пошарово, причому кожен нейрон отримує вхідні дані від інших нейронів, обробляє їх і передає вихідні дані іншим нейронам. Нейронні мережі можуть навчитися апроксимувати функції шляхом регулювання вагових коефіцієнтів зв'язків між нейронами в процесі навчання. Цей процес навчання, як правило, контролюється, тобто мережа навчається на основі маркованих даних, змінюючи свої ваги, щоб мінімізувати похибку між її виходом та істиною [9].

У контексті генерації зображень нейронні мережі можуть вивчати основну структуру і закономірності, присутні в навчальних зображеннях, що дозволяє їм генерувати нові зображення зі схожими характеристиками. Два основних типи нейронних мереж, що використовуються для генерації зображень – це генеративні змагальні мережі (GAN) та варіаційні автокодери (VAE). Ці нейронні мережі мають різні підходи до створення зображень, але показали чудові результати у створенні високоякісних, реалістичних зображень.

1.2.1 Генеративні змагальні мережі (GAN)

Генеративні змагальні мережі, запропоновані Яном Гудфеллоу (Ian J. Goodfellow) та його колегами в 2014 році – це клас моделей глибокого навчання, які привернули значну увагу в галузі генерації зображень завдяки своїй здатності створювати високоякісні, реалістичні зображення. GAN складаються з двох нейронних мереж, генератора та дискримінатора, які навчаються одночасно в процесі, відомому як змагальне навчання. Авторами змагальне навчання було визначене як гра в теорії ігор [1].

1.2.2 Архітектура GAN

Генератор – це нейронна мережа, яка генерує зображення, приймаючи на вхід випадковий шум і перетворюючи його на зображення за допомогою низки шарів згортки та дискретизації. Метою генератора є створення зображень, які неможливо відрізнити від реальних зображень у навчальному наборі даних. Генератор не має прямого доступу до реальних зображень; натомість він вчиться генерувати реалістичні зображення за допомогою зворотного зв'язку від дискримінатора.

Дискримінатор – це ще одна нейронна мережа, яка діє як класифікатор, визначаючи, чи є дане зображення справжнім (з навчального набору даних) або підробленим (згенерованим генератором). Дискримінатор навчається як на справжніх, так і на фальшивих зображеннях, навчаючись розрізняти їх за допомогою вагових коефіцієнтів у процесі навчання.

Взаємодія цих компонентів та зв'язок з навчальними даними добре ілюстровані на рис. 1.1.

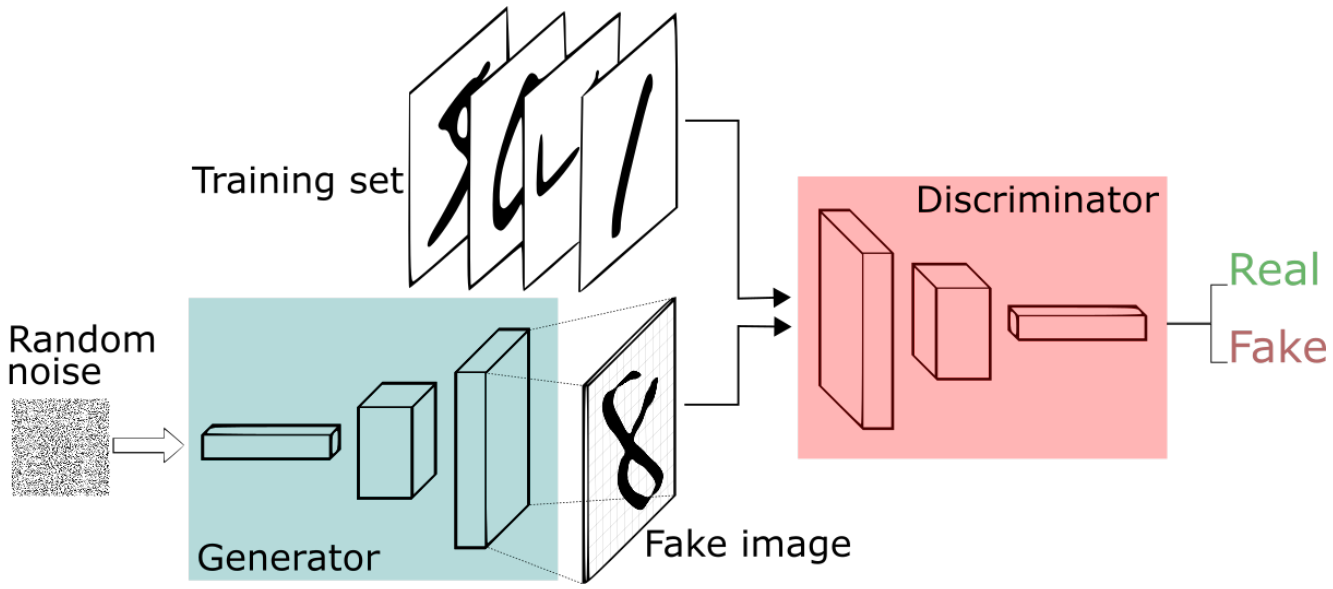

Рисунок 1.1 – Компоненти GAN [10]

1.2.3 Тренування GAN

Під час змагального навчання генератор і дискримінатор тренуються одночасно в мінімаксній грі для двох гравців, де генератор намагається обдурити дискримінатор, генеруючи реалістичні зображення, а дискримінатор намагається правильно класифікувати справжні та фальшиві зображення. Мета генератора –

максимізувати ймовірність помилки дискримінатора, в той час як дискримінатор намагається мінімізувати цю ймовірність. Цей змагальний процес триває до тих пір, поки генератор не навчиться створювати реалістичні зображення, а дискримінатор більше не зможе відрізняти справжні зображення від фальшивих.

Процес навчання GAN може бути складним, оскільки він часто передбачає збалансування швидкості навчання і архітектури генератора і дискримінатора, щоб не дати одному з них перевершити іншого. За умови успішного навчання, GAN можуть досягати вражаючих результатів, генеруючи високореалістичні зображення, які важко відрізнити від реальних.

1.2.4 Варіаційні автокодери (VAE)

Варіаційні автокодери, представлені Дідеріком Кінгмою (Diederik P. Kingma) та Максом Веллінгом (Max Welling) у 2013 році, є ще одним класом моделей глибокого навчання, що використовуються для генерації зображень. VAE – це тип генеративної моделі, яка поєднує глибоке навчання з імовірнісним моделюванням, що дозволяє їм генерувати різноманітні та реалістичні зображення. На відміну від GAN, які використовують змагальне навчання, VAE використовують інший підхід, заснований на автокодерах і варіаційному виведенні. [2]

1.2.5 Архітектура VAE

VAE складається з двох основних компонентів: кодера та декодера. Кодер – це нейронна мережа, яка приймає вхідне зображення і стискає його в низьковимірне латентне представлення, фіксуючи основні характеристики вхідного зображення. Кодер призначений для вивчення ймовірнісного відображення вхідного зображення в латентний простір, моделюючи латентне представлення як розподіл ймовірностей, а не як єдину точку.

Декодер – це ще одна нейронна мережа, яка бере вибірку з латентного простору і реконструює вхідне зображення. Декодер вчиться генерувати зображення, подібні до вхідних зображень, дотримуючись імовірнісної структури, заданої кодером. Вибираючи різні точки з латентного простору, декодер може генерувати широкий спектр зображень, які поділяють базову структуру та характеристики навчальних даних.

Принцип перетворення даних цими компонентами зображений на рис. 1.2.

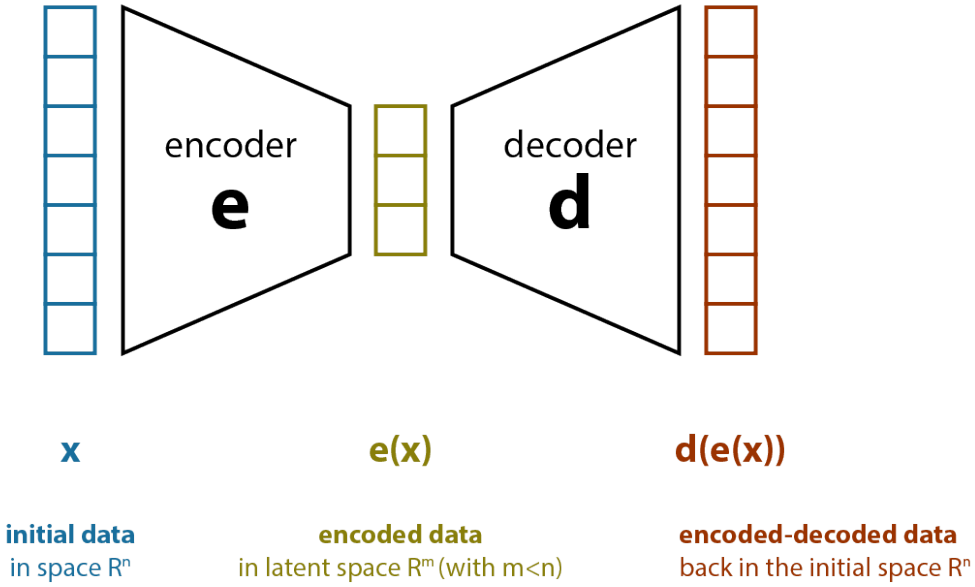

Рисунок 1.2 – Компоненти VAE [11]

1.2.6 Навчання VAE

VAE навчаються за допомогою комбінації двох функцій втрат: втрат при реконструкції та втрат при регуляризації. Втрати на реконструкцію вимірюють різницю між вхідним зображенням і реконструйованим зображенням, згенерованим декодером, заохочуючи VAE навчитися вірно представляти вхідні дані. Втрати регуляризації є мірою розбіжності між вивченим прихованим розподілом і попереднім розподілом, як правило, стандартним гаусівським розподілом. Цей термін регуляризації заохочує VAE знаходити гладкий і безперервний латентний простір, полегшуючи генерацію різноманітних і реалістичних зображень.

Під час навчання VAE вчиться знаходити компроміс між втратами при реконструкції і втратами при регуляризації, в результаті чого створюється компактний і добре структурований латентний простір, який можна використовувати для генерації нових зображень. Вибираючи точки з латентного простору і пропускаючи їх через декодер, VAE може генерувати зображення, які поділяють властивості навчальних даних, демонструючи при цьому варіативність і новизну.

Таким чином, як GAN, так і VAE виявилися ефективними методами генерації зображень, кожен з яких має свої переваги та недоліки. GAN відомі тим, що створюють високоякісні, реалістичні зображення, але процес їхнього навчання

може бути складним і нестабільним. З іншого боку, VAE пропонують більш стабільний процес навчання і добре структурований латентний простір, але згенеровані зображення іноді можуть бути менш чіткими або деталізованими порівняно з тими, що генеруються GAN. Вибір відповідної архітектури нейронної мережі для генерації зображень залежить від конкретних вимог проекту, таких як бажана якість зображення, різноманітність і стабільність навчання.

1.3 Суміжні роботи та існуючі рішення

У цьому підрозділі будуть розглянуті різні пов'язані роботи та існуючі рішення в галузі генерації зображень за допомогою нейронних мереж, зосередившись на використанні GAN і VAE, а також інтеграції цих методів з чат-ботами. Вивчаючи ці роботи, ми можемо визначити сучасні підходи, їхні обмеження та можливості для вдосконалення, що допоможе в розробці Telegram-бота для генерації зображень за допомогою нейронних мереж.

1.3.1 Рішення для генерації зображень на основі GAN

DCGAN (Алек Редфорд (Alec Radford) та ін., 2016, [12]) – Deep Convolutional Generative Adversarial Networks, є розширенням оригінальної архітектури GAN, яка включає згорткові шари як в генераторі, так і в дискримінаторі. DCGAN продемонстрували здатність генерувати високоякісні, реалістичні зображення, навчившись фіксувати ієрархічні представлення навчальних даних. DCGAN були використані в різних додатках, включаючи синтез зображень, передачу стилю і розфарбовування зображень.

StyleGAN (Теро Каррас (Tero Karras) та ін., 2019, [13]) – це вдосконалена архітектура GAN, яка представляє концепцію архітектури генератора на основі стилю, що дозволяє тонко контролювати зовнішній вигляд згенерованих зображень. StyleGAN широко використовується для створення високореалістичних і різноманітних зображень, таких як людські обличчя з високою роздільною здатністю, а також застосовується для таких завдань, як передача стилю, доповнення даних і адаптація до домену.

1.3.2 Рішення для генерації зображень на основі VAE

DRAW (Карол Грегор (Karol Gregor) та ін., 2015, [14]) – Deep Recurrent Attentive Writer, це модель на основі VAE, яка вводить рекурентну нейронну

мережу (RNN) як в кодер, так і в декодер, що дозволяє моделі генерувати зображення в послідовному порядку. DRAW здатна генерувати зображення зі складною структурою і була застосована для таких завдань, як синтез почерку, доповнення сцен і стиснення даних.

Beta-VAE (Ірина Хіггінс (Irina Higgins) та ін., 2017, [15]) – це модифікація оригінальної архітектури VAE, яка вводить гіперпараметр β для контролю балансу між втратами при реконструкції та втратами при регуляризації. Регулюючи β, Beta-VAE може вивчати більш розрізнені та інтерпретовані латентні репрезентації, які можуть бути використані для створення різноманітних і змістовних зображень. Beta-VAE застосовуються для різних завдань, включаючи синтез зображень, маніпулювання атрибутами та неконтрольоване навчання ознак.

1.3.3 Застосунки для генерації зображень

Midjourney – це програма та сервіс штучного інтелекту, створена та розміщена незалежною дослідницькою лабораторією Midjourney Inc. з Сан-Франциско. Midjourney генерує зображення з описів природної мови, які називаються “підказками” (англ. “prompts”).

На момент написання роботи інструмент перебуває у відкритій бета-версії, в яку він вийшов 12 липня 2022 р. [16] Команду Midjourney очолює Девід Хольц (David Holz). У серпні 2022 р. Хольц повідомив The Register, що компанія вже є прибутковою [17]. Користувачі створюють твори мистецтва за допомогою Midjourney, використовуючи команди бота Discord.

Компанія працює над вдосконаленням своїх алгоритмів, випускаючи нові версії моделей кожні кілька місяців. Версія 2 їхнього алгоритму була запущена у квітні 2022 року, а версія 3 – 25 липня того ж року. 5 листопада 2022 року користувачам була доступна альфа-версія версії 4, а 15 березня 2023 року – альфа-версія версії 5.

Наразі Midjourney доступний лише через бота Discord на їхньому офіційному сервері Discord, шляхом прямого повідомлення боту або запрошення бота на сервер третьої сторони. Щоб згенерувати зображення, користувачі використовують команду `/imagine` і вводять запит, а бот повертає набір з чотирьох зображень. Користувачі можуть вибрати, які з них вони хочуть покращити. Midjourney також працює над веб-інтерфейсом.

Stable Diffusion – це модель глибокого навчання для перетворення тексту

в зображення, випущена у 2022 році. Вона в першу чергу використовується для створення детальних зображень на основі текстових описів, хоча також може бути застосована для інших завдань, таких як зафарбовування, розфарбовування та створення нового зображення на основі текстової підказки та іншого зображення. Її розробив стартап Stability AI у співпраці з низкою академічних дослідників та некомерційних організацій [18].

Код і ваги моделі були опубліковані у відкритому доступі, і вона може працювати на більшості споживчого обладнання, оснащеного графічним процесором середньої потужності. Це відрізняє модель від інших аналогів, котрі працюють на пропрієтарних закритих технологіях та лише як хмарні сервіси.

Stable Diffusion використовує різновид дифузійної моделі (DM), що називається моделлю прихованої дифузії (LDM), розроблену групою CompVis в Мюнхенському університеті Людвіга—Максиміліана. Представлені в 2015 році, дифузійні моделі навчаються з метою видалення послідовного застосування гауссівського шуму на навчальних зображеннях. Як видно з рис. 1.3, архітектура цієї моделі є суттєво складнішою за VAE – вона є надбудовою над класичними VAE. Їх можна розглядати як послідовність автокодерів зі згладжуванням. Stable Diffusion складається з 3-ох частин: варіаційного автокодера, U-Net і додаткового текстового кодера. Кодер VAE стискає зображення з піксельного простору до латентного простору меншої розмірності, захоплюючи більш фундаментальне семантичне значення зображення. Гауссів шум ітеративно застосовується до стисненого латентного представлення під час прямої дифузії. Блок U-Net, що складається з магістралі ResNet, знешумлює вихідні дані прямої дифузії у зворотному напрямку, отримуючи латентне подання. Нарешті, декодер VAE генерує остаточне зображення, перетворюючи представлення назад у піксельний простір. Крок знешумлення може бути гнучко обумовлений текстом, зображенням або іншою формою опису ознак. Закодовані дані ознак піддаються впливу знешумлюючих U-Net мереж через механізм перехресної уваги. Для виділення ознак з тексту використовується фіксований, попередньо навчений текстовий кодер CLIP ViT-L/14 для перетворення текстових підказок у простір вбудовування. Дослідники вказують на підвищену обчислювальну ефективність для навчання та генерації як на перевагу LDM [19].

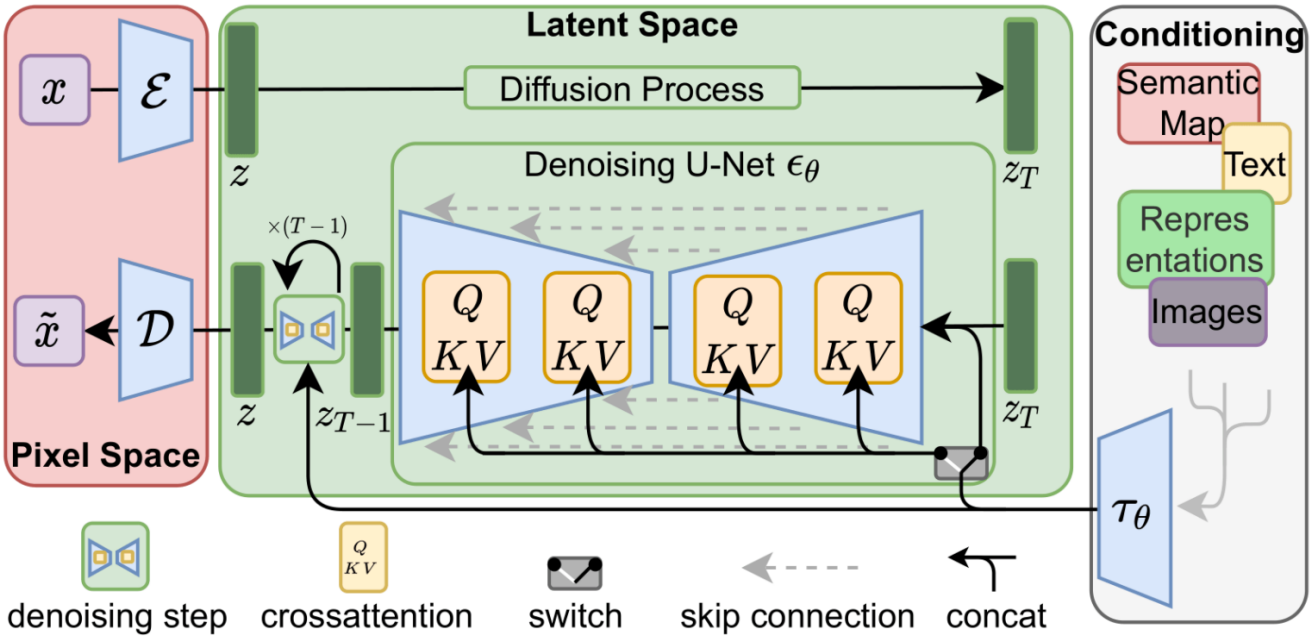

Рисунок 1.3 – Архітектура Stable Diffusion [19]

РОЗДІЛ 2 АНАЛІЗ ВИМОГ

2.1 Функціональні вимоги

Перед реалізацією Telegram-бота, призначеного для генерації зображень за допомогою нейронних мереж, треба окреслити функціональні вимоги до нього. Ці вимоги визначають основні функції та можливості, які бот повинен надавати користувачам. Функціональні вимоги до запропонованого Telegram-бота включають:

а) **автентифікація користувача:** бот повинен мати можливість ідентифікувати користувачів через їхні Telegram-акаунти. Це дозволить боту зберігати специфічні для користувача налаштування та історію згенерованих зображень;

б) **інтерфейс користувача**: бот повинен надавати інтуїтивно зрозумілий і зручний інтерфейс для взаємодії користувачів із системою генерації зображень. Користувачі повинні мати можливість надсилати текстові команди або використовувати кнопки та вбудовані клавіатури для навігації по функціях та можливостях бота;

в) **генерація зображень**: бот повинен вміти генерувати різноманітні та реалістичні зображення на основі даних, введених користувачем. Користувачі повинні мати можливість вказати бажані характеристики згенерованих зображень, такі як предмет, стиль або тема;

г) **експорт зображень**: бот повинен дозволяти користувачам завантажувати створені зображення. Користувачі також повинні мати можливість ділитися створеними зображеннями з іншими користувачами Telegram;

д) **оновлення та обслуговування**: бот повинен бути розроблений таким чином, щоб полегшити регулярне оновлення та обслуговування, гарантуючи, що моделі генерації зображень будуть відповідати останнім дослідженням і досягненням у галузі нейронних мереж.

Задовольняючи ці функціональні вимоги, запропонований Telegram-бот надасть користувачам універсальну та цікаву платформу для створення зображень за допомогою передових нейромережевих моделей. Бот не тільки продемонструє можливості AI для створення зображень, але й послужить цінним інструментом для вивчення викликів і можливостей, пов’язаних з інтеграцією цих технологій в платформу обміну повідомленнями, таку як Telegram.

2.2 Нефункціональні вимоги

На додаток до функціональних вимог, Telegram-бот для генерації зображень повинен також відповідати ряду нефункціональних вимог. Ці вимоги стосуються загальної продуктивності, надійності та зручності використання системи, які мають вирішальне значення для забезпечення позитивного користувацького досвіду. До нефункціональних вимог до запропонованого Telegram-бота належать:

а) **продуктивність**: бот повинен генерувати зображення швидко та ефективно, мінімізуючи час очікування для користувачів. Система повинна бути оптимізована для обробки декількох одночасних запитів без значного зниження продуктивності. Генерація зображень є тривалим процесом, а обробка запитів є асинхронною, тому час відповіді не буде миттєвим. Проте він не повинен бути надто довгим, щоб можна було дочекатися результату в рамках однієї сесії в Telegram. Тому розумною вимогою буде час відповіді до двох хвилин;

б) **масштабованість**: бот повинен бути спроектований таким чином, щоб пристосуватися до зростаючої кількості користувачів і запитів. Базова інфраструктура повинна мати можливість горизонтального масштабування, що дозволяє додавати більше обчислювальних ресурсів за потреби;

в) **надійність**: бот повинен забезпечувати надійний і стабільний користувацький досвід. Система повинна бути розроблена з урахуванням відмовостійкості та механізмів обробки помилок, щоб гарантувати, що вона залишатиметься працездатною і швидко реагуватиме навіть у разі збоїв у роботі апаратного чи програмного забезпечення;

г) **безпека**: бот повинен забезпечувати конфіденційність і безпеку даних користувача, включаючи інформацію про автентифікацію користувача та згенеровані зображення. Система повинна бути спроектована таким чином, щоб запобігти несанкціонованому доступу, витоку даних та іншим загрозам безпеці;

д) **зручність використання**: бот повинен бути простим у використанні та навігації, з чітким та інтуїтивно зрозумілим користувацьким інтерфейсом. Система повинна надавати корисні інструкції, щоб допомогти користувачам максимально ефективно використовувати доступні функції та можливості;

е) **зручність обслуговування**: бот повинен бути розроблений з урахуванням модульності та зручності обслуговування, що дозволяє легко оновлювати, виправляти помилки та вдосконалювати систему;

ж) **розширюваність**: бот повинен бути розроблений з урахуванням розширюваності, що дозволяє додавати нові можливості, функції та нейромережеві моделі в майбутньому. Система повинна бути модульною і гнучкою, що дозволить безперешкодно інтегрувати її з новими досягненнями в галузі генерації зображень і нейронних мереж.

Враховуючи ці нефункціональні вимоги, запропонований Telegram-бот не лише забезпечить надійну та цікаву платформу для генерації зображень за допомогою нейронних мереж, але й гарантуватиме високий рівень продуктивності, надійності та зручності використання.

2.3 Варіанти використання та історії користувачів

Для більш детальної розробки та кращого розуміння вимог буде корисно описати кілька варіантів використання та історій користувачів, які демонструють, як запропонований Telegram-бот для генерації зображень за допомогою нейронних мереж може бути використаний різними користувачами. Опишу кілька прикладів, які надають цінну інформацію про користувацький досвід:

– **створення власного зображення**. Історія користувача: Аліса хоче створити унікальне зображення футуристичного міського пейзажу для обкладинки своєї науково-фантастичної книги. Вона взаємодіє з Telegram-ботом, вказуючи бажані характеристики зображення. Бот генерує зображення, яке Аліса може завантажити та використати будь-де;

– **ділитися згенерованими зображеннями з друзями**. Історія користувача: Дейв створює зображення за допомогою Telegram-бота, яке йому здається особливо кумедним. Він вирішує поділитися зображенням зі своїми друзями в Telegram та інших соціальних мережах. Бот дозволяє Дейву легко завантажувати та ділитися створеним зображенням.

Ці приклади використання та історії користувачів показують потенційні можливості застосування запропонованого Telegram-бота для генерації зображень за допомогою нейронних мереж. Вивчивши потреби та очікування різних користувачів, можна краще зрозуміти, що є найбільш важливим у продукті та який функціонал потрібно впровадити.

РОЗДІЛ 3 АРХІТЕКТУРА ТА ДИЗАЙН СИСТЕМИ

3.1 Високорівнева архітектура

Високорівнева архітектура Telegram-бота для генерації зображень за допомогою нейронних мереж складається з декількох ключових компонентів, які працюють разом, щоб забезпечити безперебійну та ефективну роботу користувача. Основними компонентами архітектури системи є:

- **Telegram Bot API** – слугує основним інтерфейсом між ботом та платформою Telegram. Бот використовуватиме API для отримання повідомлень користувачів, надсилання відповідей та управління користувацькими даними, такими як налаштування та історія;
- **рівень інтерфейсу користувача (UI)** – відповідає за забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для взаємодії користувачів з ботом. Цей рівень обробляє вхідні дані користувача, відображає згенеровані зображення та надає опції навігації, такі як кнопки та вбудовані клавіатури;
- **компонент бота** – відповідає за обробку запитів користувачів, управління моделями нейронної мережі та генерацію зображень. Цей компонент буде побудований з використанням Python і використовуватиме бібліотеку AIOGram для взаємодії з Telegram Bot API;
- **компонент нейронних мереж** – як і компонент бота, є складовою серверної частини. За допомогою мови Python та бібліотеки Diffusers буде реалізовувати генерацію зображень нейронними мережами;
- **сховище** – частина серверів Telegram, в ньому зберігаються дані користувача та згенеровані ним зображення;
- **безпека та автентифікація** – відповідає за автентифікацію користувачів, шифрування даних та інші заходи безпеки для захисту конфіденційності користувачів.

Високорівнева архітектура (рис. 3.1) розроблена таким чином, щоб бути модульною і масштабованою, що дозволяє легко оновлювати, обслуговувати і додавати нові функції або моделі. Поділяючи систему на окремі компоненти, кожен з яких має певні обов'язки, архітектура гарантує, що система залишається надійною, ремонтопридатною та ефективною.

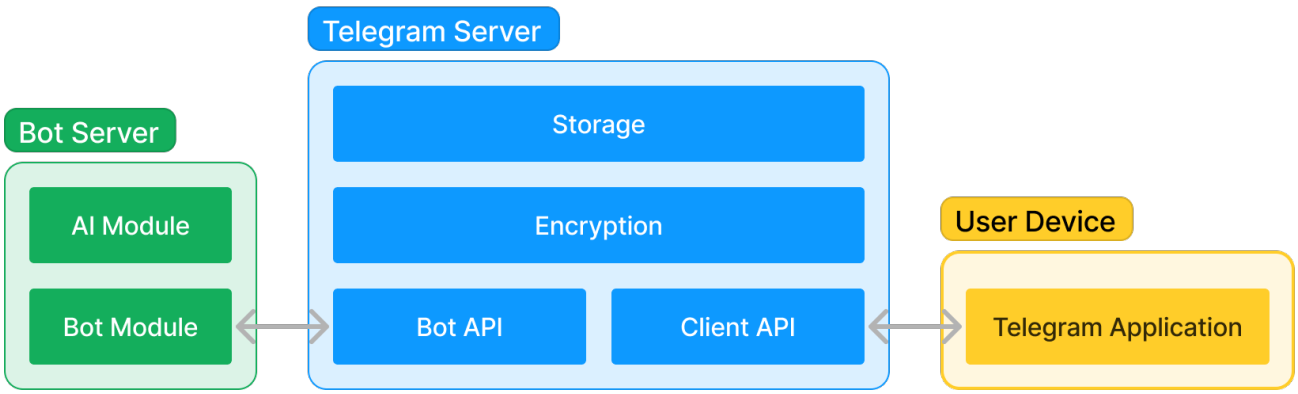

Рисунок 3.1 – Високорівнева архітектура Telegram-бота для генерації зображень за допомогою нейронних мереж

3.2 Проектування компонентів

3.2.1 Компонент Telegram-бота

Компонент Telegram-бота слугує основним інтерфейсом між користувачами та системою генерації зображень. Його основні обов'язки включають обробку користувацьких команд, зв'язок з модулем нейронних мереж та відправка відповідей користувачам. У цьому підрозділі буде описано дизайн компонента Telegram-бота, зосередивши увагу на його структурі, ключових функціях та взаємодії з іншими компонентами системи.

Структура. Компонент Telegram-бота буде реалізовано за допомогою бібліотеки AIOGram, яка надає зручний та високорівневий API для взаємодії з платформою Telegram. Структура бота складатиметься з наступних елементів:

– обробники команд: відповідатимуть за обробку команд користувача, таких як `/start`, `/help`, `/txt2img` тощо. Кожен обробник буде пов'язаний з певною функцією, яка виконуватиме відповідну дію;
– обробники повідомлень: оброблятимуть текстовий ввід користувача, який не відповідає жодній конкретній команді. Ці обробники можуть бути використані для розбору вводу користувача під час багатокрокових процесів, наприклад, для надання параметрів генерації зображень;
– стани розмови: бот використовуватиме функцію `StatesGroup`, що надається бібліотекою AIOGram, для управління багатокроковими процесами, такими як отримання від користувача параметрів генерації зображень.

Основні функції. Компонент Telegram-бота надаватиме кілька ключових функцій для полегшення взаємодії з користувачем та генерації зображень. Вони

включають:

– запуск та довідка – забезпечує можливістю користувачам вступити до бота, інструкціями з використання та інформацією про доступні команди;
– генерація зображень – надає користувачу можливість ввести текстову підказку та отримати зображення, згенероване нейромережею на її основі;
– генерація анімації – надає користувачу можливість отримати анімацію створення зображення на основі текстової підказки.

Взаємодія з іншими компонентами. Компонент Telegram-бота повинен взаємодіяти з іншими компонентами під час своєї роботи. Основні види взаємодії включають:

– Telegram Bot API – взаємодія для отримування запитів користувачів та відповіді на них. Буде забезпечуватися за допомогою бібліотеки AIOGram;
– компонент нейронної мережі – взаємодія для ініціації генерації зображень на основі користувацьких даних та отримання результатів у відповідному форматі.

Розробивши надійний і зручний компонент Telegram-бота, ми можемо гарантувати, що користувачі матимуть безперебійний і приємний досвід взаємодії з системою генерації зображень. Цей компонент відіграватиме вирішальну роль у виконанні функціональних і нефункціональних вимог, а також забезпечить основу для майбутніх удосконалень і розширень системи.

Послідовність обробки команд. Для кращого розуміння послідовності дій користувача та простішої реалізації її обробки буде корисно розробити модель бота, яка наглядно показуватиме, які потрібно буде реалізувати обробники команд та як організувати переходи між ними. Бот міститиме набір станів, між якими буде реалізоване перемикання з різних обробників команд. Стани, що забезпечать просту взаємодію з користувачем (рис. 3.2):

– стан за замовчуванням – у цьому стані користувач може активувати будь-яку з команд, наприклад виклик довідки або генерацію зображення;
– очікування текстової підказки для генерації зображення – стан, в який користувач може перейти шляхом активації команди `/txt2img`, тобто команди для генерації зображення;
– очікування текстової підказки для генерації анімації – аналогічний до попереднього стан, проте вже для команди `/txt2gif` та генерації анімації;
– процес генерації – виставляється ботом після отримання текстової

підказки та початку генерації. У цьому стані користувач не може надсилати інші команди або перейти в інші стани. Після завершення генерації виставляється стан за замовчуванням і бот очікує на нові команди.

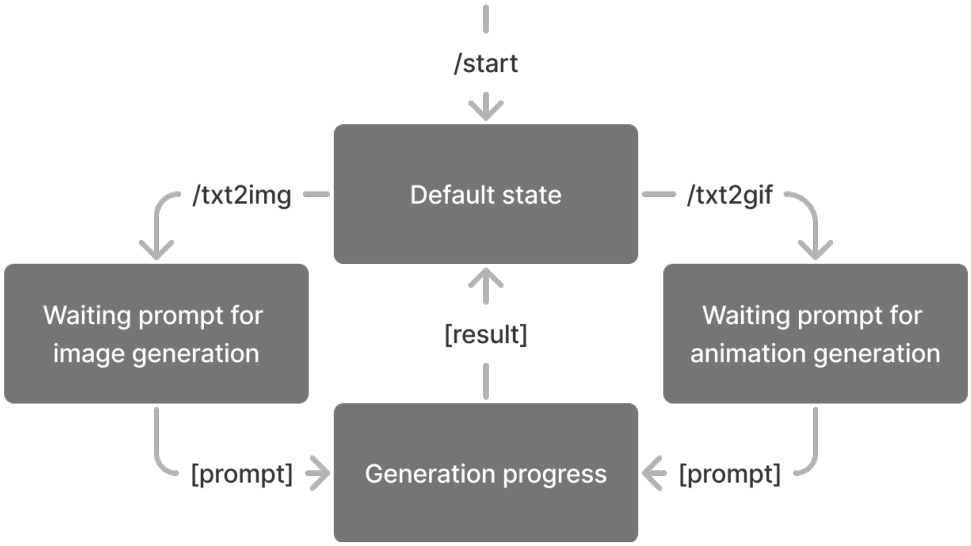

Рисунок 3.2 – Структура та процес обробки команд

3.2.2 Компонент нейронної мережі

Компонент нейронної мережі є ядром системи генерації зображень, що відповідає за створення різноманітних і реалістичних зображень на основі введених користувачем даних і бажаних характеристик. У цьому підрозділі буде описано дизайн нейромережевого компонента, зосередившись на його структурі, ключових функціях та взаємодії з іншими компонентами системи.

Структура. Нейромережевий компонент буде реалізовано з використанням Diffusers, популярної бібліотеки, яка надає високорівневий, зручний API для створення, навчання та розгортання нейромережевих моделей. Для компонента нейронної мережі в якості моделі для генерації зображення було обрано Stable Diffusion, як одну з найпопулярніших та найсучасніших моделей, котра є відкритою та надає досить якісні результати генерації. Компонент складатиметься з наступних елементів:

– модель для генерації – бібліотека Diffusers за допомогою високорівневого API надає можливість взаємодіяти з набором нейромережевих моделей для багатокрокової генерації зображень;

– функції для генерації даних – використовують модель для генерації, задаючи для неї текстову підказку, яку отримують як вхідні параметри;
– модуль для обробки зображень – забезпечує можливість конвертувати отримані результати генерації у зображення в загальноприйнятих форматах.

Ключові функції. Компонент нейронної мережі надаватиме кілька ключових функцій:

– генерація зображень – створення зображення на основі текстової підказки за допомогою нейромережевої моделі та конвертація у файл відповідного формату;
– генерація анімацій – створення та конвертація анімацій процесу генерації зображення;
– асинхронна генерація – можливість викликати процес генерації асинхронно, що забезпечить кращу продуктивність, не блокуючи обробку команд інших користувачів.

Взаємодія з іншими компонентами. Компонент нейронної мережі буде взаємодіяти лише з компонентом Telegram-бота, надаючи йому результати генерації у відповідь на ініціацію в обробниках команд.

Розробивши надійний та ефективний нейромережевий компонент, ми зможемо гарантувати, що система генерації зображень зможе генерувати різноманітні та реалістичні зображення на основі вхідних даних користувача та бажаних характеристик.

3.3 Дизайн інтерфейсу

Дизайн інтерфейсу відіграє життєво важливу роль у наданні користувачам інтуїтивно зрозумілого та безперешкодного досвіду взаємодії з системою генерації зображень. Оскільки основним користувацьким інтерфейсом для нашої системи є Telegram-бот, важливою є розробка простого у використанні інтерфейсу в рамках обмежень платформи Telegram. У цьому підрозділі буде окреслено ключові аспекти дизайну інтерфейсу, включаючи навігацію та взаємодію з користувачем.

Візуальну частину інтерфейсу бота забезпечує клієнтський застосунок Telegram. Тому бот повинен використовувати можливості платформи для налаштування підказок, що допоможуть користувачам орієнтуватися в ньому. Ключові елементи інтерфейсу, що забезпечать досвід користувача включають:

– вітальне повідомлення – після початку розмови з ботом користувачі будуть зустрінуті вітальним повідомленням, яке містить короткий вступ до бота та його можливостей;
– список команд – користувачам буде показано список доступних команд та їх опис, що дозволить їм легко зорієнтуватися та отримати доступ до потрібних функцій;
– інформаційні повідомлення – слугуватимуть для індикації прогресу та стану бота, надаючи користувачеві вичерпну інформацію та інструкції.

Навігація в інтерфейсі Telegram-бота має бути простою та зручною для користувача. Користувачі повинні мати можливість швидко отримувати доступ до потрібних функцій. Платформа надає різні можливості для навігації між станами та функціями бота, одним із найпростіших способів є команди. Користувач матиме доступ до усіх команд та зможе легко і швидко отримати доступ до потрібних функцій або довідки.

Взаємодія з користувачем також будується засобами клієнтських застосунків Telegram. Користувач має можливість вводити текст та надсилати повідомлення для керування ботом, а той у свою чергу надсилає відповіді з інформаційними повідомленнями або згенерованими медіафайлами.

Розробивши зручний та візуально привабливий інтерфейс для Telegram-бота, можна гарантувати, що користувачі матимуть позитивний досвід взаємодії з системою генерації зображень.

3.4 Потік та обробка даних

Розуміння потоку та обробки даних у системі створення зображень має вирішальне значення для забезпечення ефективної та точної роботи. Корисним є опис потоку даних між компонентами системи та процесів, пов’язаних з обробкою вхідних даних користувача, створенням зображень і наданням результатів користувачам.

Введення даних користувачем. Потік даних починається з введення користувачем даних у вигляді команд або текстових повідомлень. Цей ввід обробляється відповідними обробниками команд у компоненті Telegram-бота.

Взаємодія з компонентом нейронних мереж. звертається до функцій генерації зображень, надаючи їм відповідні дані. Внутрішній сервер відповідає за обробку запитів та управління взаємодією між компонентом Telegram-бота,

компонентом нейронної мережі та компонентом сховища.

Взаємодія з нейромережевим компонентом. Після обробки користувацького введення компонент Telegram-бота безпосередньо взаємодіє з компонентом нейронної мережі, щоб генерувати зображення на основі вхідних даних користувача та бажаних характеристик. Це передбачає виклик відповідної функції з передачею параметрів, необхідних для генерації.

Генерація зображення. Компонент нейронної мережі обробляє вхідні дані і генерує зображення, використовуючи відповідну модель. Згенероване зображення конвертується в необхідний формат і повертається Telegram-боту, який відповідає також і за відправку вихідних даних користувачеві.

Доставка вихідних даних. Після того, як згенероване зображення отримано Telegram-ботом, воно надсилається назад до користувача через Telegram Bot API. Після чого зображення зберігається на сервері Telegram та відображається користувачеві в інтерфейсі чату. Користувачі можуть переглядати, зберігати або ділитися згенерованими зображеннями за бажанням.

Розуміючи та оптимізуючи потік даних та обробку в системі генерації зображень, можна забезпечити ефективну роботу та безперебійний користувацький досвід.

РОЗДІЛ 4 РЕАЛІЗАЦІЯ TELEGRAM-БОТА

4.1 Середовище та інструменти розробки

Для ефективної реалізації компонента Telegram-бота системи генерації зображень важливо вибрати відповідні інструменти розробки та налаштувати відповідне середовище розробки. У цьому підрозділі буде описано інструменти та середовище, які використовуються для розробки компонента Telegram-бота, що забезпечить безперебійний та ефективний процес розробки.

Мова програмування. Завдяки своїй простоті, читабельності та широкій бібліотечній підтримці, Python є найбільш підходящою мовою програмування для компонента Telegram-бота. Вона широко використовується в галузі штучного інтелекту і пропонує безліч ресурсів для роботи з Telegram-ботами та нейронними мережами.

Інтегроване середовище розробки (IDE). Для розробки компоненту Telegram-бота було обрано Visual Studio Code. Visual Studio Code – це легке, крос-платформове та легко налаштовуване середовище розробки, яке пропонує чудову підтримку для розробки на Python, включаючи підсвічування синтаксису, завершення коду, налагодження та інтеграцію контролю версій.

Бібліотеки та фреймворки. AIOGram – ця бібліотека надає зручну та просту у використанні обгортку навколо API Telegram Bot, що дозволяє розробникам швидко створювати Telegram-ботів за допомогою Python. Серед інших функцій бібліотека надає підтримку для обробки оновлень, надсилання повідомлень та управління вбудованими клавіатурами.

Контроль версій. Для контролю версій буде використовуватися Git – широко розповсюджена система, що дозволяє розробникам відстежувати зміни, співпрацювати над проектом та керувати різними версіями кодової бази. Репозиторій розміщуватиметься на такій платформі, як GitHub, забезпечуючи централізоване місце для управління проектом та його ресурсами.

Налаштувавши відповідне середовище розробки та підібравши відповідні інструменти, ми можемо спростити процес реалізації компонента Telegram-бота, забезпечивши ефективну розробку та високу якість кінцевого продукту.

4.2 Створення та налаштування бота

Щоб створити нового Telegram-бота, потрібно написати іншому боту BotFather (`https://t.me/BotFather`), який спеціально призначений для цього. Зробити це може будь-який користувач Telegram абсолютно безкоштовно.

Після початку діалогу, як зображено на рис. 4.1, BotFather надсилає повідомлення з описом доступних команд для налаштування не тільки ботів, а й інших видів застосунків, інтегрованих в Telegram.

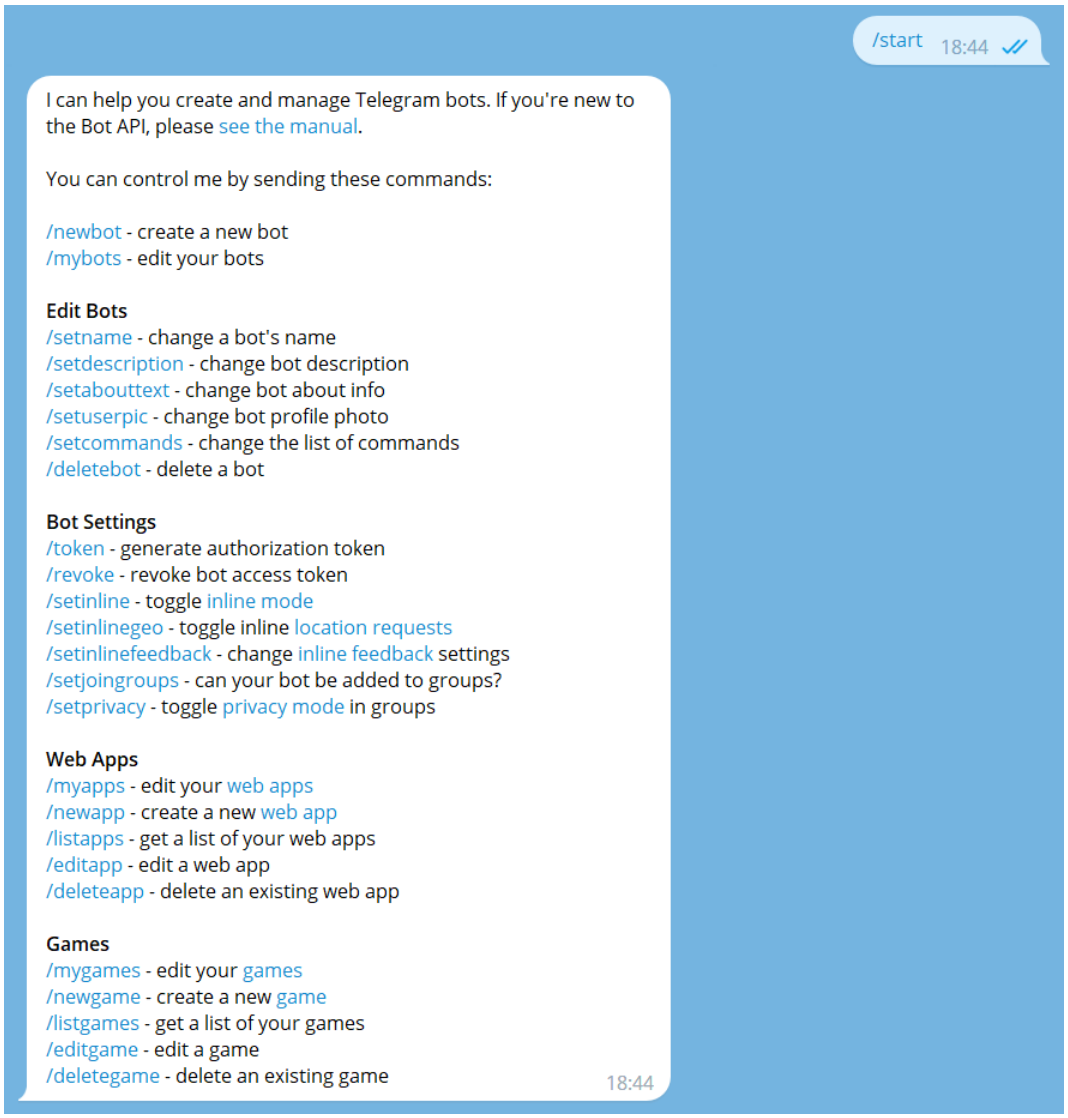

Рисунок 4.1 – Початок діалогу з BotFather

Бачимо, що за допомогою BotFather користувачі можуть створювати та видаляти своїх ботів, налаштовувати різні параметри. Серед них:

– ім’я бота;

– текстовий опис бота;

– фото профіля для бота;
– список команд;
– доступ до режимів використання в чатах.

Сам процес створення нового бота є простим. Він складається з наступних етапів (рис. 4.2):

а) потрібно надіслати команду `/newbot` до BotFather;
б) далі вводиться відображуване ім'я для бота;
в) останнім етапом є задання імені користувача для бота, воно і буде використовуватися для знаходження бота і посилання на нього;
г) на цьому етапі бот уже створений, отримано так званий ключ для доступу до нього. Його потрібно зберігати у безпечному та захищеному місці, адже будь-хто може контролювати бота за допомогою цього ключа.

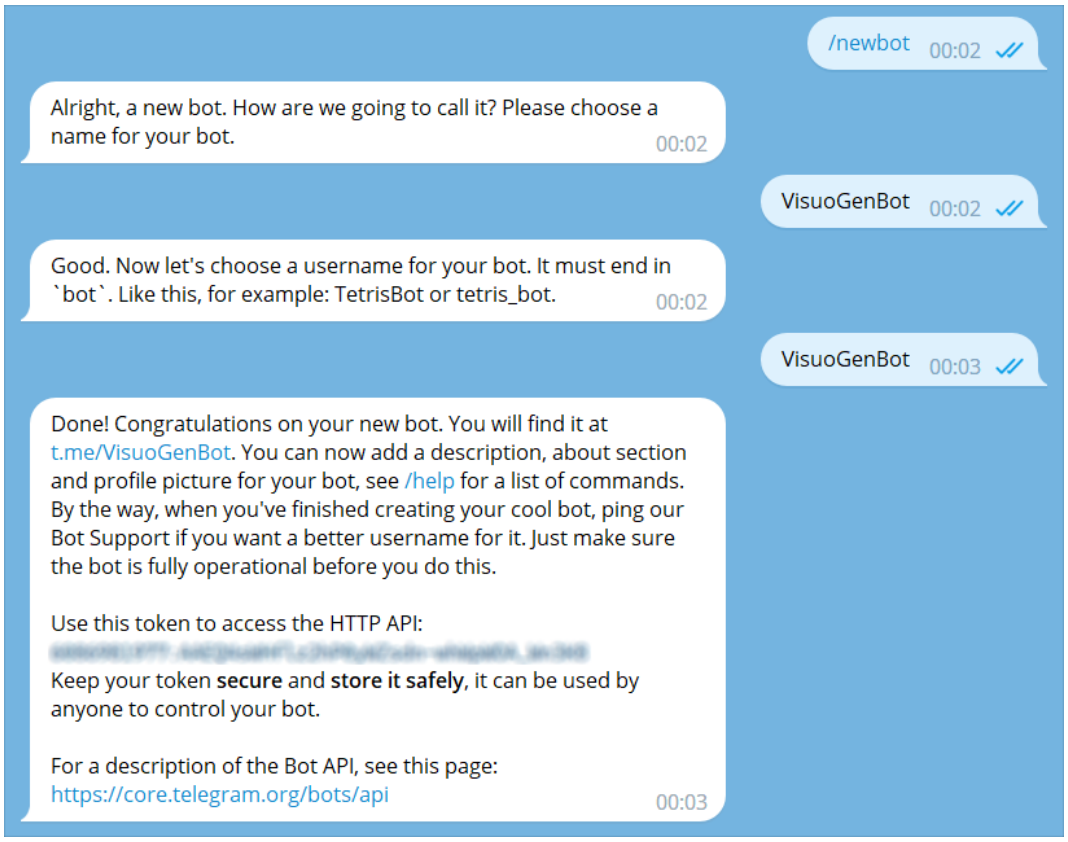

Рисунок 4.2 – Створення нового Telegram-бота

Після створення бота, за допомогою команди `/mybots` та інтерфейсу BotFather можна налаштувати різноманітні параметри (рис. 4.3). Зображення користувача для бота було налаштоване з використанням одного зі згенерованих зображень.

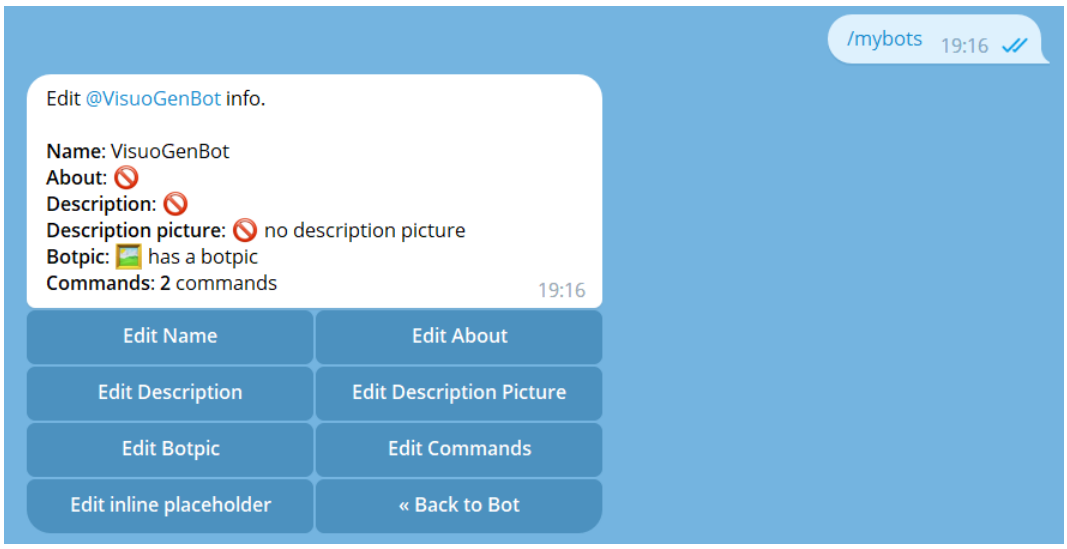

Рисунок 4.3 – Інтерфейс налаштування бота

Натиснувши на кнопку “Edit Botpic”, можна надіслати фото для зображення профіля (рис. 4.4).

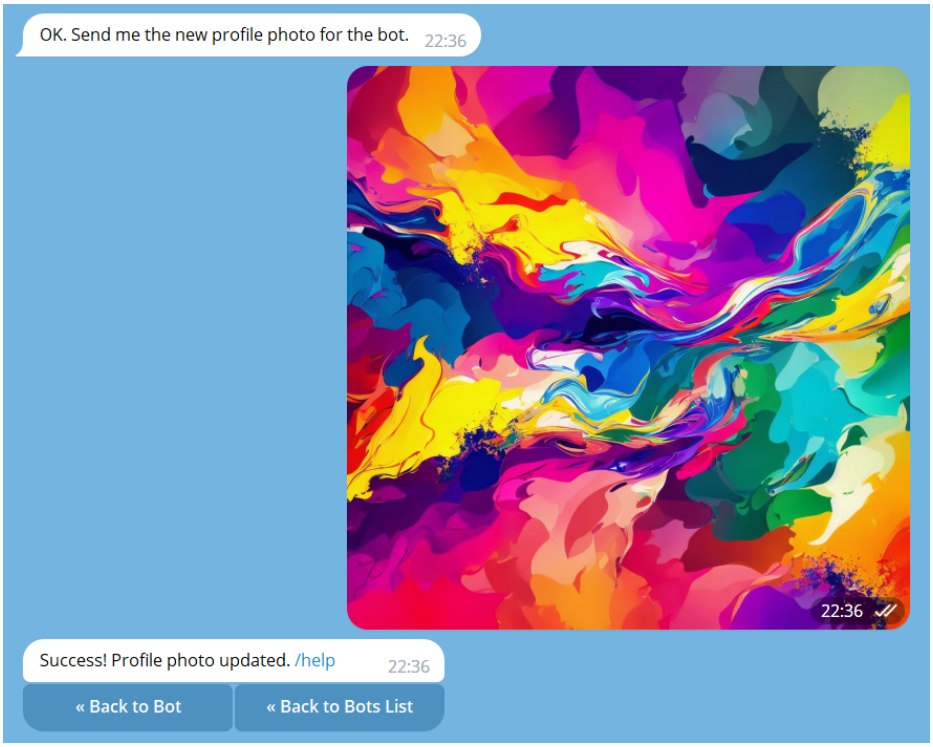

Рисунок 4.4 – Процес зміни зображення профіля

На цьому етапі поки можна завершити налаштування бота та перейти до реалізації обробки повідомлень від користувачів.

4.3 Обробка команд і взаємодія з користувачем

Як основну бібліотеку для реалізації обробки команд було використано просту та повністю асинхронну бібліотеку AIOGram. Вона побудована на механізмі створення асинхронних програм asyncio та фреймворці aiohttp, що реалізовує клієнтську та серверну частини протоколу HTTP. [20]

Щоб почати використовувати бібліотеку AIOGram, потрібно її спочатку встановити за допомогою вбудованого в Python пакетного менеджера.

```
pip install aiogram
```

Далі створюється бот з використанням ключа, отриманого від BotFather. Замість того, щоб вказувати цей ключ прямо в код, будемо його задавати як змінну оточення, тому що він має бути прихований з метою захисту контролю бота.

```
import aiogram
from os import getenv
from aiogram.contrib.fsm_storage.memory import MemoryStorage

bot = aiogram.Bot(getenv("BOT_TOKEN"))
storage = MemoryStorage()
dp = aiogram.Dispatcher(bot, storage=storage)
```

Наведений код показує створення бота та обробника команд зі збереженням даних в пам’яті.

Далі створимо набір станів для бота, щоб обробляти багатокрокові дії:

```
from aiogram.dispatcher.filters.state import StatesGroup, State

class BotState(StatesGroup):
    txt2img_prompt_wait = State()
    txt2gif_prompt_wait = State()
    generation_in_progress = State()
```

Ці стани відображають архітектуру та послідовність подій в боті:

- `txt2img_prompt_wait` – стан очікування підказки для генерації зображення;
- `txt2gif_prompt_wait` – стан очікування підказки для генерації анімації;
- `generation_in_progress` – стан генерації результату, з цього стану користувач не може самостійно перейти у будь-який інший;
- початковий стан не потрібно задавати окремо, так як бот в ньому знаходиться за замовчуванням, якщо не було переходу в якийсь інший.

AIOGram самостійно контролює перехід між станами для кожного користувача, тому не потрібно вручну зберігати стани для різних користувачів.

Обробники для конкретних команд створюються за допомогою спеціальних декораторів. Команда для простого переходу з початкового стану в стан очікування підказки для генерації може виглядати наступним чином:

```
from aiogram.types import Message

@dp.message_handler(commands=['txt2img'])
async def txt2img_handler(message: Message):
    await message.answer('Enter prompt')
    await BotState.txt2img_prompt_wait.set()
```

Цей код демонструє створення функції для обробки команди `/txt2img`. Після її отримання від користувача, бот переходить у стан `txt2img_prompt_wait` і надсилає користувачу прохання ввести підказку.

Далі реалізуємо функцію для отримання підказки та створення зображення:

```
from aiogram.dispatcher import FSMContext

@dp.message_handler(state=BotState.txt2img_prompt_wait)
async def txt2img_prompt_handler(message: Message, state:
   FSMContext):
    try:
        await BotState.generation_in_progress.set()
        await message.answer('Ok, image generation in progress')
        await message.answer_photo(await txt2img_async(message.text
   ))
    except Exception as ex:
        await message.answer(f'Error occured during generation, try
    later:\n{str(ex)}')
    finally:
        await state.reset_state()
```

Передавши у декоратор певний стан, вказується, що даний обробник повинен викликатися тільки з цього стану. Всередині виставляється стан генерації результату, користувачу надсилається інформаційне повідомлення для очікування. Використаємо функцію `txt2img_async`, яку буде реалізовано пізніше. Вона прийматиме підказку від користувача, та на її основі буде генерувати зображення. Після отримання результату бот надсилає його користувачу. Якщо під час цього процесу виникла помилка, відображається повідомлення про це.

Незалежно від результату, в кінці обробника бот переходить в початковий стан.

Аналогічним чином реалізовується обробник, що створює анімацію генерації зображення. Тільки замість `message.answer_photo(result)` викликається `message.answer_animation(result)`.

Також реалізуємо обробник, що сигналізує про активний процес генерації, якщо користувач в цей момент надішле боту якесь повідомлення.

```
@dp.message_handler(state=BotState.generation_in_progress)
async def wait_handler(message: Message):
    await message.answer("Generation in progress, please wait")
```

Задля покращення користувацького досвіду, було написано функцію, що при запуску буде задавати список можливих команд для бота. Для цього потрібно викликати необхідну функцію з API та передати в неї список команд та їх описів.

```
async def set_commands():
    await bot.set_my_commands([
        BotCommand('txt2img', 'Generate image from text'),
        BotCommand('txt2gif', 'Create gif of image generation
   progress'),
    ])
```

Після виконання, в додатку Telegram буде відображатися меню, зображене на рис. 4.5.

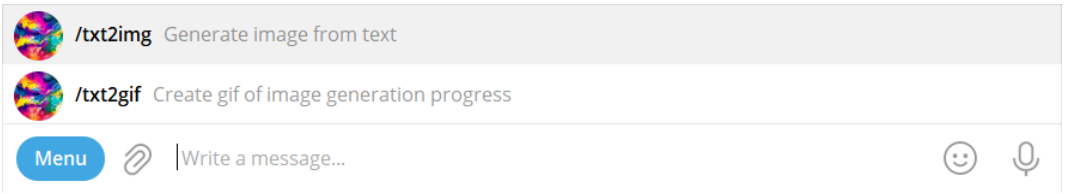

Рисунок 4.5 – Меню зі списком команд бота

Щоб усе зібрати та запустити потрібно використати наступний код:

```
from aiogram import executor

executor.start(dp, set_commands())
executor.start_polling(dp, skip_updates=True)
```

Після запуску бот почне довге опитування для отримання оновлень від користувачів. Усі повідомлення будуть викликати відповідні обробники команд, причому асинхронно, тобто різні користувачі зможуть користуватися ботом, не блокуючи один одного.

РОЗДІЛ 5 РЕАЛІЗАЦІЯ НЕЙРОННОЇ МЕРЕЖІ

5.1 Середовище та інструменти розробки

Для ефективної та зручної реалізації компонента нейронної мережі використаю підхід, що забезпечує просту інтеграцію з Telegram-ботом. Усі інструменти використані для розробки бота будуть використані і для нейронної мережі. Мова програмування Python добре підходить для машинного навчання, тому що має дуже багато користувацьких бібліотек для цього. Функції для генерації зображень будуть реалізовані як окремий модуль в проекті. Тож віртуальне середовище та IDE будуть використані одні і для бота, і для нейронної мережі. Обидва компонента будуть знаходитися в одному репозиторії.

Бібліотеки та фреймворки. Для роботи з нейронними мережами буде використана відкрита бібліотека Diffusers. Вона реалізує типові архітектури нейронних мереж для генерації медіа контенту. Diffusers також реалізовує модель Stable Diffusion та підтримує гнучке налаштування кожного з компонентів. [21]

Бібліотеку Diffusers можна встановити за допомогою пакетного менеджера Pip виконавши наступну команду:

```
pip install --upgrade diffusers
```

Diffusers всередині використовує іншу бібліотеку PyTorch, створену для машинного навчання на GPU та CPU, тому необхідно встановити і її. Щоб мати можливість запускати генерацію зображень на графічному прискорювачі, потрібно встановити версію бібліотеки, скомпільовану з підтримкою CUDA. Команду для встановлення правильної конфігурації можна створити на офіційному веб-сайті бібліотеки [22]:

```
pip3 install torch torchvision torchaudio --index-url https://
   download.pytorch.org/whl/cu117
```

Як було зазначено, потрібно також мати встановлену CUDA SDK – набір інструментів, що дозволяють запускати підпрограми на графічних прискорювачах Nvidia. Завантажити версію CUDA SDK, сумісну з встановленою версією PyTorch можна з офіційного веб-сайту [23].

Для обробки та збереження зображень використовується бібліотека Pillow, яка дозволяє просто конвертувати, редагувати, завантажувати та зберігати зображення різних форматів [24].

5.2 Вибір та реалізація моделі

Для компонента нейронної мережі було вибрано модель Stable Diffusion, оскільки вона є відкритою та безкоштовною. Крім того, забезпечує високу якість зображень при відносно невеликому обчислювальному навантаженні.

Як було зазначено, бібліотека Diffusers містить реалізацію усіх компонентів Stable Diffusion. Також в бібліотеці присутній клас `StableDiffusionPipeline`, котрий реалізує роботу усіх компонентів разом для генерації зображень [25]. Об'єкти цього класу конструюються із моделей-складових, серед них:

- `vae` (`AutoencoderKL`) – модель варіаційного автокодера (VAE) для кодування та декодування зображень до та з латентних представлень;
- `text_encoder` (`CLIPTextModel`) – кодер тексту. Stable Diffusion використовує текстову частину `CLIP`, зокрема варіант `clip-vit-large-patch14`;
- `tokenizer` (`CLIPTokenizer`) – токенізатор класу `CLIPTokenizer`;
- `unet` (`UNet2DConditionModel`) – умовна архітектура U-Net для знешумлення закодованого латентного простору зображення;
- `scheduler` (`SchedulerMixin`) – планувальник, який використовується у поєднанні з `unet` для знешумлення. Може бути одним із `DDIMScheduler`, `LMSDiscreteScheduler` або `PNDMScheduler`;
- `safety_checker` (`StableDiffusionSafetyChecker`) – модуль класифікації, який оцінює, чи можна вважати згенеровані зображення образливими або шкідливими;
- `feature_extractor` (`CLIPImageProcessor`) – модель, яка визначає ознаки зі згенерованих зображень, щоб використовувати їх як вхідні дані для `safety_checker`.

Об'єкт `StableDiffusionPipeline` можна також створити по користувацьких конфігураціях, розміщених на веб-ресурсі `https://huggingface.co/models`. Користувачі можуть створювати власні налаштування моделей та тренувати їх, після чого можна розмістити створену конфігурацію на платформі, щоб нею могли користуватися інші. Бібліотека Diffusers дозволяє автоматично завантажувати моделі з цього ресурсу та використовувати у своїх програмах. Для генерації зображень було обрано модель `https://huggingface.co/runwayml/stable-diffusion-v1-5`. Вона є найпопулярнішою на момент написання роботи.

Код для генерації зображень за допомогою бібліотеки Diffusers може виглядати наступним чином:

```
from diffusers import StableDiffusionPipeline

model_id = 'runwayml/stable-diffusion-v1-5'
pipe = StableDiffusionPipeline.from_pretrained(model_id,
   torch_dtype=torch.float16)
pipe = pipe.to('cuda')

image = pipe('lion riding bicycle',
             width=512, height=512,
             num_inference_steps=50).images[0]

image.save('./img/leo.png')
```

Даний приклад на основі текстової підказки “лев на велосипеді” генерує зображення розміром 512 на 512 пікселів за 50 кроків знешумлення. На рис. 5.1 наведено кілька результатів генерації.

Рисунок 5.1 – “Леви на велосипедах”

5.3 Інтеграція з ботом

Перше що потрібно зробити – це реалізувати функцію, яка, приймаючи текстову підказку, буде генерувати і повертати зображення.

```
from diffusers import StableDiffusionPipeline,
from img_utils import img2bytes
import io

def txt2img(prompt: str) -> io.BytesIO:
    model_id = 'runwayml/stable-diffusion-v1-5'
    pipe = StableDiffusionPipeline.from_pretrained(model_id,
   torch_dtype=torch.float16)
```

```
    pipe = pipe.to("cuda")
    image = pipe(prompt, 512, 512, 50).images[0]
    return img2bytes(image)
```

По коду вище видно як конструюється `StableDiffusionPipeline` з обраної моделі та потім використовується для генерації зображення за вхідною текстовою підказкою.

Результатом генерації є зображення у вигляді об'єкту `PIL.Image` з бібліотеки Pillow. Бібліотека AIOGram та Telegram Bot API потребують, щоб зображення було передане у формі даних у відповідному форматі, наприклад JPEG. Тому потрібно також реалізувати функцію для конвертації об'єкту PIL.Image в буфер io.BytesIO, де зберігатиметься зображення в форматі JPEG:

```
import io
import PIL as pil

def img2bytes(img: pil.Image) -> io.BytesIO:
    bytes = io.BytesIO()
    img.save(bytes, 'JPEG')
    bytes.seek(0)
    return bytes
```

Створення зображення та відправка його ботом реалізована, проте сам процес генерації є "тяжкою" операцією і виконується досить тривалий час. Було виміряно час генерації на наступних складових:

- процесор – 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz;
- графічна карта – NVIDIA GeForce RTX 3060 Laptop GPU з 6GB виділеної пам'яті формату GDDR6;
- оперативна пам'ять – встановлено 32GB пам'яті формату DDR4;

На такому пристрої функція генерації зображення `txt2img` відпрацьовує в середньому за 14 секунд. Якщо цю функцію викликати прямо з обробника команди то увесь бот буде блокуватися на час генерації і не буде відповідати на повідомлення інших користувачів. Виходом із цієї ситуації було б використання того факту, що бібліотека AIOGram є асинхронною, тобто в обробниках команд можна асинхронно викликати корутини, що не будуть блокувати виконання та зупиняти отримання оновлень. Існує спосіб перетворити довгу блокуючу операцію в асинхронну корутину, для цього потрібно передати обчислення функції виділеному виконавцеві (англ. "executor"):

```
async def txt2img_async(prompt: str) -> io.BytesIO:
    loop = asyncio.get_running_loop()
    return await loop.run_in_executor(None, txt2img, prompt)
```

Цей код відправляє виконання функції `txt2img` з параметром `prompt` до виділеного виконавця, який є першим параметром методу `run_in_executor`. У даному випадку вказано `None` – це означає, що буде використано виконавця за замовчуванням, тобто `ThreadPoolExecutor`. Виконання функції буде направлено у системний потік, відмінний від головного, в якому крутиться цикл обробки подій.

Написавши таку обгортку, ми змогли реалізувати асинхронну генерацію зображень для різних користувачів, таким чином забезпечивши кращий користувацький досвід та підвищивши продуктивність бота.

Однією з функції бота також є створення анімації процесу генерації зображення. Для цього буде використано один із параметрів запуску генерації об'єктом `StableDiffusionPipeline` – `callback`. В генерацію можна передати функцію зворотнього виклику, що буде викликатися кожну ітерацію під номером `callback_steps`. Сама функція повинна приймати три параметри:

– step (int) – номер ітерації, на котрій викликано функцію;
– timestep (int) – число, що вказує на поточний етап генерації;
– latents (torch.FloatTensor) – масив з даними в латентному просторі, які потрібно декодувати автокодером, щоб отримати поточний стан зображення.

Тоді функція, що генерує зображення та створює анімацію цього процесу виглядатиме наступним чином:

```
def txt2gif(prompt: str) -> tuple[io.BytesIO, io.BytesIO]:
    model_id = 'runwayml/stable-diffusion-v1-5'
    pipe = StableDiffusionPipeline.from_pretrained(model_id,
  torch_dtype=torch.float16)
    pipe = pipe.to("cuda")
    images = []

    def save_decoded_latent(step: int, timestep: int, tensor: torch
  .FloatTensor):
        images.append(pipe.numpy_to_pil(pipe.decode_latents(tensor)
  )[0])

    pipe(prompt, 512, 512, 50, callback=save_decoded_latent,
  callback_steps=5)
```

```
    return make_gif(images), img2bytes(images[-1])
```

Щоб згенерувати анімацію, ми декодуємо дані з латентного простору на кожній п'ятій ітерації та зберігаємо їх в масив зображень. Щоб Telegram-бот міг відправити користувачу анімацію, набір зображень потрібно перетворити у дані формату GIF. Функція `make_gif` виглядає так:

```
def make_gif(images: list[pil.Image]) -> io.BytesIO:
    bytes = io.BytesIO()
    images[0].save(bytes, format='GIF', save_all=True, optimize=
  True, append_images=images[1:], loop=1, duration=300)
    bytes.name = 'image.gif'
    bytes.seek(0)
    return bytes
```

Потрібно також в боті створити обробник команди для генерації анімації. Щоб викликати `txt2gif` асинхронно, потрібно написати для неї обгортку, аналогічно з `txt2img_async`. Тоді обробник команди буде виглядати наступним чином:

```
@dp.message_handler(state=BotState.txt2gif_prompt_wait,
  content_types=aiogram.types.ContentType.TEXT)
async def txt2gif_prompt_handler(message: Message, state:
  FSMContext):
    try:
        await BotState.generation_in_progress.set()
        await message.answer('Ok, animation generation in progress'
  )
        gif, img = await txt2gif_async(message.text)
        await message.answer_photo(img)
        await message.answer_animation(gif)
    except Exception as ex:
        await message.answer(f'Error occured during generation, try
   later:\n{str(ex)}')
    finally:
        await state.reset_state()
```

РОЗДІЛ 6 АНАЛІЗ РЕЗУЛЬТАТІВ

6.1 Порівняння бота з аналогами

Більшість рішень, що надають користувачам можливість генерувати зображення за допомогою нейронних мереж, є платними та вимагають щомісячної підписки. На момент написання роботи на платформі Telegram також не було безкоштовного бота для генерації зображень. Тому будуть розглянуті деякі аналоги з веб інтерфейсом.

Stable Diffusion Online [26] – сервіс, що надає доступ до генерації зображень через веб інтерфейс. Він також використовує Stable Diffusion для своєї роботи, тому порівняння з реалізованим ботом є цілком доречним.

Отримати доступ до сервісу можна перейшовши за посиланням `https://stablediffusionweb.com/`. На головній сторінці відразу видно інтерфейс для взаємодії з нейронною мережею (рис. 6.1).

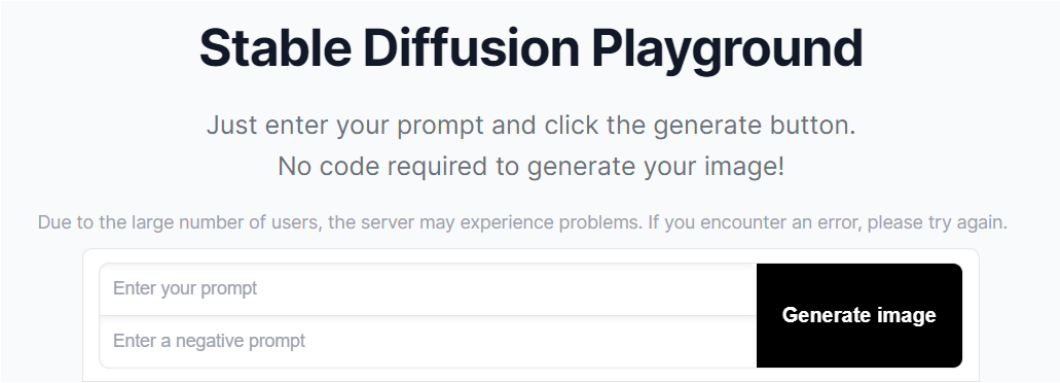

Рисунок 6.1 – Інтерфейс Stable Diffusion Online

В інтерфейсі присутні поля для вводу підказки для генерації зображення та для вводу заперечувальної підказки, котра вказує на ознаки, які повинні бути відсутніми на вихідному зображенні. Можливість ввести негативну вказівку для генерації є перевагою перед реалізованим Telegram-ботом.

Проте Stable Diffusion Online має і недоліки у порівнянні з реалізованим ботом:

– відсутня можливість створення анімації процесу генерації зображення;
– сервіс не має функціоналу для збереження та керування історією згенерованих зображень.

Реалізований Telegram-бот для збереження історії взаємодії використовує засоби самої платформи для обміну повідомленнями: зображення та вся історія зберігаються у діалозі та можуть бути завантажені в будь-який момент часу. Користувач має зручний доступ до даних в діалозі через інтерфейс налаштування чату (рис. 6.2). Окрім цього, зображення та анімації розділені в різні категорії для зручності пошуку.

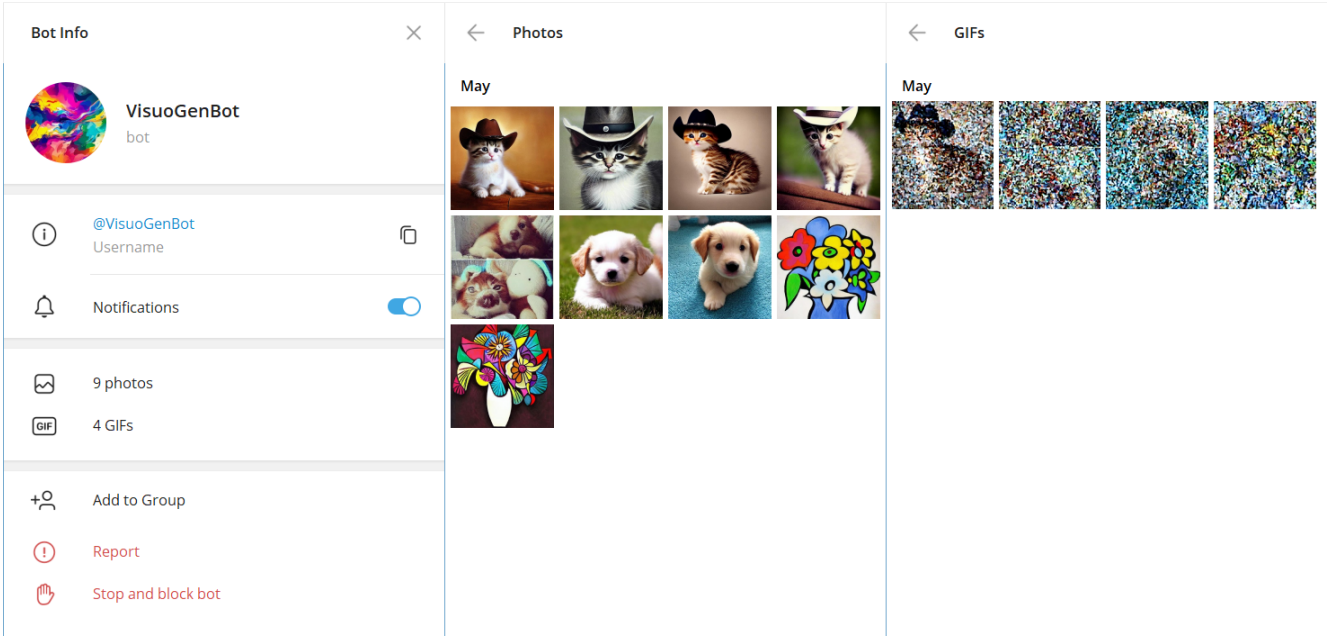

Рисунок 6.2 – Інтерфейс перегляду історії бота

DALL-E – пропрієтарна модель для генерації зображень від компанії OpenAI. Доступ до неї можна отримати через зручний веб-інтерфейс за посиланням `https://labs.openai.com/`. Проте для користування потрібна реєстрація, а на безкоштовній основі можна зробити тільки 15 спроб на місяць. Цього достатньо для порівняння з реалізованим ботом. DALL-E також побудована на основі VAE, проте офіційну реалізацію опубліковано не було [27].

На головній веб-сторінці для використання моделі присутній інтерфейс, що дозволяє лише ввести текстову підказку та згенерувати зображення (рис. 6.3). Також для простого випробовування можна отримати випадкову текстову підказку для генерації.

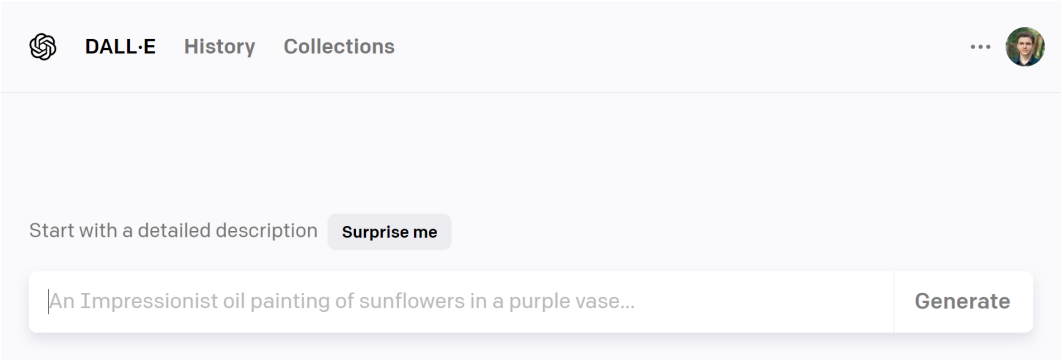

Рисунок 6.3 – Інтерфейс DALL-E

Як і реалізований бот, ресурс від OpenAI також має можливість переглядати історію генерації зображень і завантажувати їх. Проте Telegram-бот надає можливість легко пересилати результати іншим користувачам натисканням всього однієї кнопки. У DALL-E є ще кілька недоліків у порівнянні з реалізованим ботом:

– обмежена кількість генерації на безкоштовній основі;
– відсутність можливості створити анімацію.

6.2 Аналіз якості та порівняння результатів генерації із аналогами

Було порівняно якість згенерованих зображень на наступних категоріях та текстових підказках:

– портрети – “portrait of man, moustaches, brown cowboy hat, green t-shirt”;
– тварини – “little kitten, blue fur, high grass, sunny day”;
– пейзажі – “fancy landscape, mountains, river, castle, night”;
– натюрморти – “yellow chair, metal table, apples on table”.

Щоб порівняти передачу стилю, для кожної категорії було застосовано також додаткові підказки для стилю:

– “high quality photo”;
– “oil painted style”.

Було оцінено наступні критерії:

– якість передачі ознак – наскільки збережені ознаки, наведені в текстовій підказці;
– якість зображення – чіткість вихідного зображення та відсутність артефактів;
– якість передачі стилю – збереження стилістичних ознак, наведених в текстовій підказці.

Порівняння проводилося шляхом генерації по 4 зображення для кожної категорії та кожного стилю.

Рисунок 6.4 – Порвіняння згенерованих портретів

Як видно з рис. 6.4, зі збереженням ознак в портретах впоралися усі моделі, проте більш стабільні в цьому сенсі результати дав реалізований Telegram-бот. Загальна якість зображення, як і передача стилю, найвищою є у моделі DALL-E.

Рисунок 6.5 – Порівняння згенерованих тварин

З рис. 6.5 видно, що з категорією тварин найгірше впорався сервіс Stable Diffusion Online і програє конкурентам по всіх критеріях. У збереженні ознак VisuoGenBot і DALL-E показали себе однаково добре. Однак сервіс від OpenAI не зовсім добре впорався із передачею художнього стилю.

Рисунок 6.6 – Порівняння згенерованих пейзажів

При генерації зображень пейзажів (рис. 6.6), коректний час доби, наявність гір, річки та замку повністю передав лише VisuoGenBot. Окрім цього, у реалізованого бота дуже вдало вийшло передати художній стиль у порівнянні з конкурентами. Також у цій категорії у VisuoGenBot менше присутніх артефактів на зображеннях.

Рисунок 6.7 – Порівняння згенерованих натюрмортів

Категорія натюрмортів – найскладніша для усіх моделей (рис. 6.7). Найкращим результатом є генерація у фотореалістичному стилі від моделі DALL-E. Проте вона показала і найгірший результат у художньому стилі, ці зображення вийшли зовсім не чіткими та містять найбільше артефактів. Можна сказати, що найстабільніший і досить якісний результат отриманий за допомогою VisuoGenBot.

На основі результатів оцінювання генерацій в різних категоріях, можна зробити висновок, що реалізований бот надає результат, часто якісніший від конкурентів, який можна використовувати в різних цілях, як для генерації контенту, так і для особистого використання.

ВИСНОВКИ

Протягом цієї роботи було успішно досягнуто основної мети – розробити Telegram-бота, здатного генерувати зображення за допомогою нейронних мереж. Це стало можливим завдяки всебічному розумінню суміжних теорій і технологій, включаючи розробку Telegram-ботів, генеративні змагальні мережі (GAN) та варіаційні автокодери (VAE).

У процесі розробки дотримувалися суворих практик програмної інженерії. Процес розпочався з ретельного аналізу вимог, який привів до чіткого розуміння очікуваної поведінки системи. На етапі проектування було розроблено високорівневу архітектуру та детальні проекти компонентів, які слугували основою для реалізації.

На етапі реалізації було успішно розроблено Telegram-бота та інтегровано модель для генерації зображень.

Хоча поточна реалізація Telegram-бота забезпечує надійну платформу для генерації зображень, є місце для майбутніх удосконалень та покращень:

- додавання більшої кількості нейромережевих моделей: поточна реалізація підтримує лише одну попередньо навчену модель Stable Diffusion. У майбутньому можна буде інтегрувати інші типи нейронних мереж або попередньо навчені моделі, щоб запропонувати ширший спектр методів генерації зображень;
- налаштування користувача: можливості налаштування користувача можуть бути розширені, щоб дозволити користувачам налаштовувати більше параметрів процесу генерації зображень;
- масштабування та оптимізація продуктивності: зі збільшенням кількості користувачів може виникнути потреба в оптимізації або масштабуванні системи, щоб впоратися зі зростаючим попитом;
- новий функціонал: в майбутньому можна буде розширити функціонал бота, надавши користувачеві можливість генерувати зображення на основі іншого, редагувати зображення.

Успішне завершення роботи демонструє потенціал інтеграції передових методів штучного інтелекту в такі доступні платформи, як Telegram. Розроблений Telegram-бот надає зручний інтерфейс для складних процесів генерації зображень, відкриваючи можливості нейронних мереж для ширшої аудиторії. Незважаючи на труднощі, з якими довелося зіткнутися, результати підтверджують доцільність та

ефективність обраного підходу.

Ця випускна кваліфікаційна робота також підкреслює важливість принципів програмної інженерії у створенні надійного, легкого в обслуговуванні та зручного для користувача програмного забезпечення. Від аналізу вимог і проектування системи до реалізації – кожен етап процесу розробки програмного забезпечення відіграє вирішальну роль в успіху.

У підсумку, ця робота не лише створила функціонального Telegram-бота для генерації зображень, але й зробила внесок у ширшу сферу програмної інженерії та штучного інтелекту, продемонструвавши практичне застосування нейронних мереж у широко використовуваній платформі для обміну повідомленнями.

Код реалізованого Telegram-бота знаходиться в репозиторії на платформі GitHub за посиланням `https://github.com/antl-m/visuo-gen-bot`.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Generative Adversarial Nets [Текст] / Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. і Bengio, Y. // Advances in Neural Information Processing Systems / під ред. Ghahramani, Z., Welling, M., Cortes, C. [та ін.]. — [S. l.] : Curran Associates, Inc. — 2014. — Т. 27. — Режим доступу: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.
2. Kingma, D. P. Auto-Encoding Variational Bayes [Текст] / Kingma, D. P. і Welling, M. — [S. l. : s. n.]. — 2022. — arXiv : stat.ML/1312.6114.
3. 700 million users and telegram premium [Текст] // Telegram. — 2022. — Режим доступу: https://telegram.org/blog/700-million-and-premium.
4. Russia's Zuckerberg Launches Telegram, a new Instant Messenger Service [Текст] // Reuters. — 2013. — Режим доступу: https://www.reuters.com/article/idUS74722569420130830.
5. Telegram FAQ [Електронний ресурс]. — Telegram. — Режим доступу: https://telegram.org/faq (дата звернення: 18.04.2023).
6. Bots: An introduction for developers [Електронний ресурс]. — Telegram APIs. — Режим доступу: https://core.telegram.org/bots (дата звернення: 18.04.2023).
7. Telegram Bot API [Електронний ресурс]. — Telegram APIs. — Режим доступу: https://core.telegram.org/bots/api (дата звернення: 18.04.2023).
8. Bot API library examples [Електронний ресурс]. — Telegram APIs. — Режим доступу: https://core.telegram.org/bots/samples (дата звернення: 18.04.2023).
9. Lawrence, J. Introduction to Neural Networks: Design, Theory, and Applications [Текст] / Lawrence, J. і Luedeking, S. — [S. l.] : California Scientific Software, 1994. — ISBN: 9781883157005. — Режим доступу: https://books.google.pl/books?id=XVpaAAAAYAAJ.
10. Silva, T. S. A Short Introduction to Generative Adversarial Networks [Текст] // https://sthalles.github.io. — 2017. — Режим доступу: https://sthalles.github.io/intro-to-gans/.
11. Joseph Rocca, B. R. Understanding variational autoencoders (VAES) [Текст] // Medium. — 2021. — Режим доступу: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

12. Radford, A. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks [Текст] / Radford, A., Metz, L. і Chintala, S. — [S. l. : s. n.]. — 2016. — arXiv : cs.LG/1511.06434.
13. Karras, T. A Style-Based Generator Architecture for Generative Adversarial Networks [Текст] / Karras, T., Laine, S. і Aila, T. — [S. l. : s. n.]. — 2019. — arXiv : cs.NE/1812.04948.
14. DRAW: A Recurrent Neural Network For Image Generation [Текст] / Gregor, K., Danihelka, I., Graves, A., Rezende, D. J. і Wierstra, D. — [S. l. : s. n.]. — 2015. — arXiv : cs.CV/1502.04623.
15. Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework [Текст] / Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S. і Lerchner, A. // International Conference on Learning Representations. — [S. l. : s. n.]. — 2017. — Режим доступу: https://openreview.net/forum?id=Sy2fzU9gl.
16. Midjourney. We're officially moving to open-beta! [Електронний ресурс]. — Twitter. — 2022. — Режим доступу: https://twitter.com/midjourney/status/1547108864788553729.
17. Claburn, T. Holz, founder of AI Art Service midjourney, on future images [Текст] // The Register® - Biting the hand that feeds IT. — 2022. — Режим доступу: https://www.theregister.com/2022/08/01/david_holz_midjourney/.
18. Mostaque, E. Stable diffusion launch announcement [Текст] // Stability AI. — 2022. — Режим доступу: https://stability.ai/blog/stable-diffusion-announcement.
19. Alammar, J. The Illustrated Stable Diffusion [Текст]. — 2022. — Режим доступу: https://jalammar.github.io/illustrated-stable-diffusion/.
20. Welcome to AIOGRAM's documentation! [Електронний ресурс]. — aiogram 2.25.1 documentation. — Режим доступу: https://docs.aiogram.dev/en/latest/index.html (дата звернення: 09.05.2023).
21. Huggingface. Huggingface/Diffusers: State-of-the-art diffusion models for image and audio generation in pytorch [Електронний ресурс]. — GitHub. — Режим доступу: https://github.com/huggingface/diffusers (дата звернення: 09.05.2023).
22. PyTorch [Електронний ресурс]. — PyTorch. — Режим доступу: https://pytorch.org/ (дата звернення: 09.05.2023).

23. Nvidia. Cuda Toolkit 11.7 downloads [Електронний ресурс]. — NVIDIA Developer. — 2022. — Режим доступу: https://developer.nvidia.com/cuda-11-7-0-download-archive.
24. Pillow [Електронний ресурс]. — Pillow (PIL Fork). — Режим доступу: https://pillow.readthedocs.io/en/stable/ (дата звернення: 09.05.2023).
25. Huggingface. Diffusers [Електронний ресурс]. — Hugging Face – The AI community building the future. — Режим доступу: https://huggingface.co/docs/diffusers/ (дата звернення: 09.05.2023).
26. Stable Diffusion Online [Електронний ресурс]. — Stable Diffusion Online. — Режим доступу: https://stablediffusionweb.com/ (дата звернення: 12.05.2023).
27. O'Connor, R. How DALL-E 2 actually works [Текст] // AssemblyAI - News, Tutorials, AI Research. — 2023. — Режим доступу: https://www.assemblyai.com/blog/how-dall-e-2-actually-works/.