МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 31.00.00.000 ПЗ

Група ІПм-23-2

Лаврук Андрій

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Лаврук Андрій Миколайович

(прізвище, ім'я, по батькові)

УДК 004.4

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та алгоритми генеративного штучного інтелекту для побудови систем генерації зображень

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Лаврук А.М.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник проф. Лютак Ігор Зіновійович, д-р техн. наук., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Лавруку Андрію Миколайовичу
(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи **“**Моделі, методи та алгоритми генеративного штучного інтелекту для побудови систем генерації зображень**”**
керівник проекту (роботи) проф. Лютак Ігор Зіновійович, д.т.н., доцент
затверджені наказом закладу вищої освіти від **“** 22 ” листопада 2024 р. № 781/7

2. Строк подання студентом проекту (роботи) 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні дані про сучасні типи генеративних моделей, формальний опис та алгоритми функціонування систем генерації зображень

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Теоретичні відомості про штучний інтелект та сучасні методи генерації зображень
2. Аналіз існуючих генеративних моделей штучного інтелекту та методи їх інтеграції у системи генерації зображень
3. Розробка алгоритму та програмна реалізація системи генерації зображень на основі текстової підказки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Схема роботи GAN мережі (рис. 1.2., ст. 27)
2. Схема роботи дифузійної моделі. (рис. 1.4, ст. 31)
3. Схема простого варіаційного автокодувальника. (рис. 1.6., ст. 35)
4. Схема основних складових моделі трансформера. (рис. 1.7., ст. 39)
5. Use Case діаграма взаємодії користувача із системою (рис. 3.1., ст. 72)
6. Діаграма послідовності системи генерації зображень (рис. 3.2., ст. 73)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц. к.т.н. Вовк Р. Б.	

7. Дата видачі завдання 04 вересня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури	21.09.2024	виконано
2	Аналіз сучасних технологій генеративного штучного інтелекту	03.10.2024	виконано
3	Сучасні генеративні моделі штучного інтелекту для генерації зображень	11.10.2024	виконано
4	Дослідження якості генерації зображень сучасних генеративних моделей штучного інтелекту	23.10.2024	виконано
5	Формулювання вимог та алгоритмів роботи системи	01.11.2024	виконано
6	Програмна реалізація рішення	11.11.2024	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 97 с., 27 рис., 2 табл., 41 джерело.

Тема: Моделі, методи та алгоритми генеративного штучного інтелекту для побудови систем генерації зображень.

Об’єкт дослідження: моделі та алгоритми функціонування генеративного штучного інтелекту для генерації зображень.

Мета роботи: аналіз моделей і алгоритмів роботи генеративного штучного інтелекту та розробка алгоритму роботи системи для генерації зображень на основі текстової підказки.

Предмет дослідження: моделі генеративного штучного інтелекту та їх інтеграція в програмне забезпечення.

Результати дослідження:

Виконано аналіз існуючих моделей генеративного штучного інтелекту і на їх основі запропоновано власне архітектурне рішення та алгоритм роботи системи для генерації зображень.

Висновок:

В результаті досліджень було отримано власну систему генерації зображення для робочого столу на основі генеративного штучного інтелекту. Дана робота вирішує проблему інтеграції генеративних моделей штучного інтелекту в програмні рішення.

ГЕНЕРАТИВНИЙ ШТУЧНИЙ ІНТЕЛЕКТ, ГЕНЕРАТИВНА МОДЕЛЬ, ГЕНЕРАЦІЯ ЗОБРАЖЕНЬ, КОНСТРУЮВАННЯ ПІДКАЗОК, ІНТЕГРАЦІЯ, ХМАРНІ ОБЧИСЛЕННЯ, КОМП’ЮТЕРНИЙ ДОДАТОК.

ANNOTATION

Master's work: 97 p., 27 fig., 2 tab., 41 sources.

Topic: Models, methods and algorithms of generative artificial intelligence for building image generation systems.

Object of research: models and algorithms of generative artificial intelligence for image generation.

Purpose: to analyze models and algorithms of generative artificial intelligence and develop an algorithm for generating images based on text prompts.

Subject of research: models of generative artificial intelligence and their integration into software.

Research results:

The existing models of generative artificial intelligence are analyzed and, on their basis, the author proposes his own architectural solution and algorithm for the system for generating images.

Conclusion:

The research resulted in a proprietary desktop image generation system based on generative artificial intelligence. This paper solves the problem of integrating generative artificial intelligence models into software solutions.

GENERATIVE ARTIFICIAL INTELLIGENCE, GENERATIVE MODEL, IMAGE GENERATION, PROMPT ENGINEERING, INTEGRATION, CLOUD COMPUTING, COMPUTER APPLICATION.

ЗМІСТ

Стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 9

ВСТУП 10

РОЗДІЛ 1

1. ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ШТУЧНИЙ ІНТЕЛЕКТ ТА СУЧАСНІ МЕТОДИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

1.1. Теоретичні відомості про штучний інтелект 15

1.2. Машинне та глибоке навчання 20

1.3. Типи генеративних моделей штучного інтелекту 26

1.4. Висновки до розділу 40

РОЗДІЛ 2

2. АНАЛІЗ ІСНУЮЧИХ ГЕНЕРАТИВНИХ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ ТА МЕТОДИ ЇХ ІНТЕГРАЦІЇ У СИСТЕМИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

2.1. Сучасні генеративні моделі для генерації зображень 41

2.2. Основи побудови текстових запитів для генерації (prompt engineering) 53

2.3. Порівняльна оцінка результатів генерації зображень 56

2.4. Порівняння моделей для інтеграції в сторонні сервіси 63

2.5. Висновки до розділу 68

РОЗДІЛ 3

3. РОЗРОБКА АЛГОРИТМУ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ НА ОСНОВІ ТЕКСТОВОЇ ПІДКАЗКИ

3.1. Опис проблеми інтеграції генеративних моделей 69

3.2. Вимоги до програмного рішення розроблюваної системи 70

3.3. Розробка алгоритму роботи системи 72

3.4. Програмна реалізація системи генерації зображення 75

3.5. Тестування розробленого програмного продукту 86
3.6. Висновки до розділу 90
ВИСНОВКИ 91
ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 93
ДОДАТКИ 98

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI (ШІ) – Artificial intelligence, Штучний інтелект

API – Application Programming Interface, Прикладний програмний інтерфейс

CLIP – Contrastive Language-Image Pre-training, Порівняльне тренування мовних зображень

CNN – Convolutional Neural Networks, Згорткова нейронна мережа

DL – Deep learning, Глибоке навчання

GAN – Generative adversarial network, Генеративна змагальна мережа

GPT – Generative pre-trained transformer, Генеративний попередньо тренований трансформер

HTTP – HyperText Transfer Protocol, Протокол передачі гіпертекстових документів

IDE – Integrated Development Environment, Інтегроване середовище розробки

IT – Information Technology, Інформаційні технології

JSON – JavaScript Object Notation

LDM – Latent Diffusion Model, Модель латентної дифузії

LLM – Large language model, Велика мовна модель

ML – Machine learning, Машинне навчання

NLP – Natural language processing, Обробка природної мови

SSL – Self-supervised learning, Самокероване навчання

VAE – Variational autoencoder, Варіаційні автокодери

ПЗ – Програмне забезпечення

ВСТУП

Актуальність роботи

Сучасний світ надзвичайно стрімко розвивається, появляються різні нові технології та удосконалюються наявні. Штучний інтелект є одним із найбільш нашумілих технологічних напрямів на даний момент. Дана технологія розвивається та змінює стандартні підходи до обробки великих масивів даних для автоматизації процесів. Однією з ключових особливостей штучного інтелекту є його здатність оптимізувати завдання та економити час на їх виконанні [1]. Застосування штучного інтелекту дозволяє виконувати завдання не лише швидше, а й точніше. Штучний інтелект заполоняє дедалі більше різних галузей, навіть тих, які не пов’язані із IT сегментом. На даний момент використання ШІ можна зустріти в медицині для діагностування захворювань та розробки ліків, у транспортному секторі – це автономні транспортні засоби, безпека – розпізнавання обличь та підозрілих операцій у інтернеті. Також нейронні мережі застосовуються у робототехніці, де використовують декілька напрямі штучного інтелекту.

Однією із підкатегорій штучного інтелекту є генеративний штучний інтелект. Він призначений для безпосередньо генерації контенту. За допомогою генеративного штучного інтелекті відбувається значна кількість автоматизації процесів, це стосується: генерації зображень, відео, тексту, музики, графіки у відеоіграх та багато іншого. Завдяки цій автоматизації на даний момент стрімко зросла кількість контенту у всіх галузях. Це й не дивно, адже для створення зображення вручну, необхідне відповідне програмне забезпечення та навики в даній сфері, а за допомогою генеративного штучного інтелекту, все що необхідно – це написати текстовий опис бажаного зображення і натиснути кнопку згенерувати.

Як і люба технологія, штучний інтелект має свої недоліки. Для навчання моделі ШІ потрібно надзвичайно велика кількість даних. Також використання якісних моделей потребує значних апаратних ресурсів та електроенергії. Оскільки для навчання потрібна велика кількість даних, то значна кількість даних береться із відкритих джерел, а це спричиняє конфлікти, щодо дозволів на використання

авторських робіт. Не менш важливим є етичні ризики. На сьогоднішній день є чимало випадків маніпуляції даними та створення фейкових новин за допомогою ШІ. Також штучний інтелект здатний помилятися і видавати недостовірну інформацію під виглядом реальних фактів.

Актуальність даної теми зумовлена значним зростанням попиту на системи із інтеграцію генеративних штучних мереж, які виконують генерацію зображень швидко та якісно. Системи з кожним разом розвиваються і стають кращими та якіснішими. Раніше даного роду ШІ генерували зображення невисокої якості із значними артефактами, які міг побачити любий пересічний користувач інтернету. Вже на даний момент генеративні мережі досягли такого успіху, що для виявляння почерку штучного інтелекту приходиться залучати спеціалістів. Але для генерації якісного реалістичного зображення приходиться підібрати відповідну модель, якісний текстовий опис зображення, та тематику зображення. Як відомо, коли ШІ не знає конкретно, що генерувати, то він галюцинує та видає зображення з артефактами. Тому завжди потрібно підходити із професіоналізмом до написання текстового опису для того, щоб ШІ правильно зрозумів поставлене завдання.

На сьогоднішній день публіці представлено великий спектр моделей генеративного штучного інтелекту різної спеціалізації. Деякі є платними, а деякі з відкритим кодом програмного забезпечення. Тож для якісного вибору моделі генеративного штучного інтелекту варто порівняти деякі із наявних на ринку.

Порівняння роботи з відомими розв’язаннями проблеми

Роздуми про штучний інтелект, який здатен працювати подібно людському велися велись ще із античності. Багато фільмів та книг взяли за основу сюжету штучний інтелект. Та й самі філософи та дослідники на початку розвитку галузі вели дискусії щодо штучного розуму [2]. Наступним кроком розвитку стало використання ланцюга Маркова. Як тільки даний ланцюг навчиться на корпусі тексту, то його можна використати як імовірнісний генератор тексту [3]. На початку 70-х років 20-го століття Гарольд Коен виставляв генеративні роботи ШІ у галереях. Дані роботи були

створені комп’ютерною програмою AARON, що була створена Коеном для створення оригінальних картин. Спочатку за допомогою даної серії програм створювались абстрактні малюнки, які пізніше ускладнилися.

Стрімкий розвиток генеративного штучного інтелекту почався із 2014 року, коли було розроблено варіаційний автокодер та генеративну змагальну мережу. Саме за допомогою цих досягнень були створені перші кроки практичної реалізації глибоких нейронних мереж. Пізніше у 2017 році спеціалісти з компанії Google розробили покращену модель – Transformer [4]. Вона була кращою за попередні моделі. Згодом у 2018 році виходить перша версія ChatGPT, а у 2022 даний чат був вже представлений широкому загалу. Також у 2021 році було продемонстровано генеративну модель DALL-E для генерації зображень, а через деякий час було продемонстровано і інші відомі моделі, такі як, Stable Diffusion і Midjourney. Саме після виходу цих генеративних мереж почався бум згенерованих фотографій у інтернеті.

Також не менш важливим аспектом, пов’язаним із генеративним ШІ є конструювання підказок. Підказка – це текстовий опис того, що повинна згенерувати вибрана модель штучного інтелекту. Від підказки напряму залежить отриманий результат, тому підказка має чітко описувати, що вимагається він мережі.

На даний момент дослідження ідуть у сфері покращення вже існуючих моделей штучного інтелекту, регулювання етичних аспектів використання штучного інтелект та оптимізації використання ресурсів під час роботи даних мереж.

Мета і задачі дослідження

Метою магістерської роботи є аналіз моделей і алгоритмів роботи генеративного штучного інтелекту та розробка алгоритму роботи системи для генерації зображень на основі текстової підказки.

Досліджувана модель повинна використовувати генеративну модель штучного інтелекту за допомогою вибраного оптимально рішення для оптимізації обчислення та зменшення навантаження на стороні користувача. Система повинна

виконувати основне її призначення та працювати без перебоїв. Генерація зображень має бути простою та якісною.

Досягнення мети включало розв’язання таких задач:

1) огляд існуючих моделей генеративного ШІ;
2) дослідження сучасних підходів генерації зображень;
3) аналіз та порівняння існуючих моделей генеративного штучного інтелекту;
4) вибір релевантної генеративної моделі ШІ та обґрунтування доцільності її використання;
5) програмна реалізація системи генерації зображень на основі обраної моделі.

Об’єктом дослідження є моделі та алгоритми функціонування генеративного штучного інтелекту для генерації зображень.

Предметом дослідження є моделі генеративного штучного інтелекту та їх інтеграція в програмне забезпечення.

Методи дослідження

Для виконання дослідження було проведено аналіз відомих генеративних моделей ШІ. Для оцінки якості згенерованих зображень застосовано якісну оцінку результатів. За допомогою системного підходу було здійснено порівняння існуючих архітектурних рішень та їх застосування для практичної реалізації при побудові систем генерації зображень.

Наукова новизна одержаних результатів

Здійснено аналіз та порівняння існуючих моделей генеративного штучного інтелекту для генерації зображень, що у свою чергу дало змогу отримати нове програмне рішення та алгоритм роботи системи генерації зображень для робочого столу.

Практичне значення одержаних результатів

На основі проведеного дослідження було розроблено функціонуючу систему генерації зображень, яка генерує зображення за допомогою введеного користувачем текстового опису. Дана система може бути використана, як застосунок для генерації зображень для робочого столу користувача.

Особистий внесок

Основним результатом є:

1. Проведено порівняльну характеристику декількох моделей генеративного штучного інтелекту.

2. Розглянуто проблеми інтеграції генеративних моделей при розробці ПЗ.

3. Запропоновано алгоритм функціонування та програмну реалізацію системи генерації зображення з високим рівнем якості, яка виконує генерацію зображення за допомогою хмарних обчислень.

Публікації

Підготовлені тези доповіді на XVII міжнародній науково-практичній конференції «Інформаційні технології і автоматизація – 2024», яка відбулась 31 жовтня - 1 листопада 2024 року в м. Одеса.

Структура магістерської роботи.

Магістерська робота викладена на 97 сторінках друкованого тексту, який складається з вступу, трьох розділів, висновків, списку використаних джерел (41 найменування). Робота містить 27 рисунків, 2 таблиці та 1 додаток.

РОЗДІЛ 1
1. ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ШТУЧНИЙ ІНТЕЛЕКТ ТА СУЧАСНІ МЕТОДИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

1.1. Теоретичні відомості про штучний інтелект

1.1.1. Про штучний інтелект

Штучним інтелектом прийнято вважати інтелектуальні машини, що виконують поставлені завдання, які зазвичай потребують людського інтелекту. Це загальне поняття, яке об’єднує в собі всі підкатегорії штучного інтелекту. Зараз навіть компанії вставляють приставку AI у назву своїх продуктів, які так чи інакше мають відношення до використання штучного інтелекту. На даний момент застосування ШІ досягло небаченого росту. Із появою генеративної моделі ChatGPT, безліч нових стартапів почали вибудовувати собі шлях у сфері штучного інтелекту. Можна сказати, що випуск публічно доступної версії моделі ChatGPT дав поштовх до розвитку нової інноваційної сфери, яка розвивається по сьогоднішній день.

Робота штучного інтелекту полягає в тому, що система навчається на певному об’ємі даних і після цього здатна виконувати визначений для неї тип завдань, без прямого написання коду логіки вирішення проблеми. Тобто під час навчання система сама визначає для себе певні закономірності для вирішення завдання. Звичайно, ідеальних мереж не має і вони всі здатні помилятися. Тому кожну мережу після навчання перевіряють на тестових даних і отримуються результат у вигляді відсоткового співвідношення правильних рішень на поставлені завдання. Чим краще мережа навчена тим вищий відсоток правильних відповідей, але це не завжди так. Буває й так, що мережа має число правильних відповідей практично рівне 100%, проте мережа видає безліч хибних відповідей. Зазвичай дана мережа погано працює на нових даних, але демонструє хорошу точність на навчальних даних. Це свідчить про те, що замість того, щоб навчитися узагальнювати закономірності, модель занадто ретельно засвоїла особливості навчальної вибірки, включно з її шумом і

випадковими деталями. Таку мережу називають перенавченою.

1.1.2. Генеративний штучний інтелект

Генеративний штучний інтелект є підмножиною штучного інтелекту і призначений для генерації різного роду контенту (рис. 1.1.). Даного роду мережі із легкістю генерують текст, картинки, відео, музику та навіть пишуть код. Генеративний ШІ не обмежується генерацією різного роду творами та картинами, що копіюють стилі відомих митців. На даний момент вони знатні вести бесіду у вигляді текстового чату у реальному часі. Найвідомішим прикладом такого роду мережі є ChatGPT, що і задав тренд даних мереж. Його відповіді схожі на людські, хоча людина, яка активно спілкувалася з даним штучним розумом може вирізнити його нелюдські повадки. Але це не відміняє того, що тексти згенеровані такими мережами часто публікуються, як такі, що написані людиною.

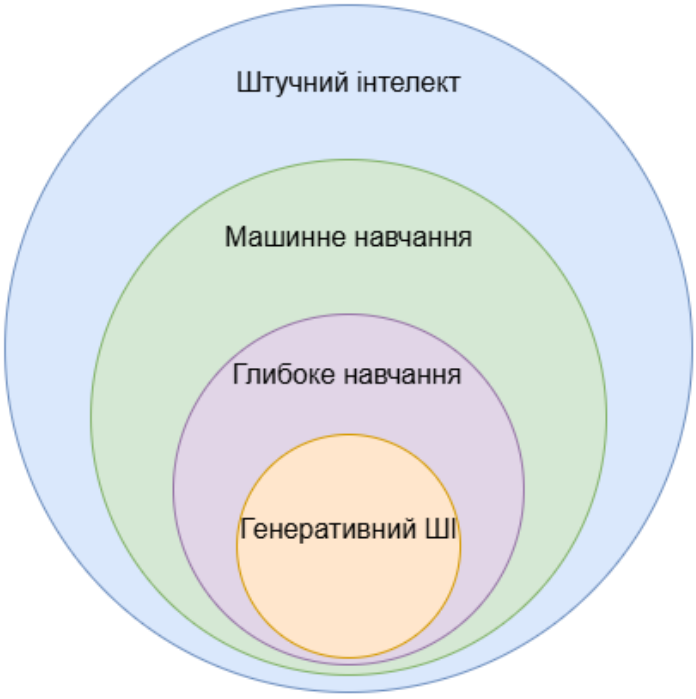

Рис. 1.1. Підгалузі штучного інтелекту

Штучний інтелект розвивався протягом значного періоду часу, перш ніж кінцевий користувач отримав генеративну мережу для вільного використання. Цей

процес зайняв декілька стадій розвитку штучного інтелекту. Одним із етапів розвитку штучного інтелекту є машинне навчання – це підтип штучного інтелекту, що здатен після навчання самостійно вирішувати певні завдання. Далій йде глибоке навчання – це підмножина машинного навчання, що знатна до самостійного виділення шаблонів у великих масивах даних. Тут йдеться не про просте навчання, а про більш осмислене навчання, де мережа безпосередньо виділяє для себе шаблони та структури даних для подальшої роботи. Після глибокого навчання йде генеративний штучний інтелект, який здатен, на основі великої кількості даних, генерувати дуже схожі дані, на яких він навчався [5].

Безсумнівно, генеративний ШІ здатен як на складні, так і на повсякденні завдання як-от генерація розкладу дня, кулінарного рецепту чи навіть поради щодо поставленого питання. Але не варто вірити штучному інтелекту на слово. Буває й таке, що він вигадує інформацію, замість того, щоб видати вірну відповідь [6]. Штучний інтелект здатен галюцинувати, а це значить, що будь-яка інформація, отримана за допомогою моделі генеративного ШІ може бути недостовірною. Інколи здається, що згенерована інформація є точною і достовірною, проте вона не ґрунтується на реальних фактах чи подіях. Оскільки даного роду мережі були створені для генерації схожих даних на основі великої кількості навчальних даних, то генеративні моделі часто намагаються згенерувати дані, які просто схожі на навчальні, тим самим будуючи правдоподібний текст з інформацією, яка хоч і здається правдоподібною але не має жодного підтвердження чи посилання на відомі факти [7].

Явище галюцинування поширене у випадках, коли користувач ввів недостатньо вірне запитання чи скористався занадто загальними термінами. В цьому випадку мережа просто вигадує та заповнює ці прогалини. На даний момент безліч студентів, журналісті та і звичайних користувачів звертаються до генеративних моделей за достовірною інформацією. Проте завжди варто перевіряти згенеровану інформацію перед її використанням. Одним із типових прикладів галюцинації мережі є плутання років. Хоч і штучний інтелект здатен шукати інформацію в мережі

інтернет, проте дану інформацію легко звірити, ввівши простий запит в інтернеті. Ще одним прикладом галюцинації є генерація різних результатів при зверненні до генеративної моделі. Мається на увазі, що при декількох тих самих, чи схожих запитах мережа може видавати кардинально різні результати. Також часто буває, що виданий результат є надто абстрактним і загальними поняттями. Чимало публікацій було згенеровано за допомогою штучного інтелекту, що в свою чергу може призвести до наповнення просторів інтернету недостовірними та абстрактними даними, серед яких пересічному користувачі буде все важче знайти якусь цінну та правдиву інформацію. Звичайно ШІ розвивається і дедалі більше видає конструктивної інформації.

Незважаючи на свої переваги, генеративний штучний інтелект має недоліки. Якщо в навчальних даних присутні помилкові твердження або суспільні упередження, навчена генеративна модель може бути упередженою та видавати помилкові дані, як вірні. Крім того, виникають етичні проблеми, коли генеративні технології використовуються для створення фальшивих текстів або зображень. Однак технологічний прогрес не стоїть на місці, створюючи нові можливості для використання генеративного ШІ в різних галузях.

На сьогоднішній день, генеративні моделі штучного інтелекту можуть допомогти з перекладом різних мов та підвищити рівень освіти серед учнів. За допомогою ШІ можна скласти хороше резюме. Крім того, генеративні моделі без проблем можуть автоматизувати введення даних та складання розпорядку дня, що підвищує продуктивність та заощаджує час [8].

Можна виділити ось такі позитивні сторони штучного інтелекту:

- автоматизація типових повторюваних процесів;
- можливість генерувати різного роду контент;
- розмиває межі мовних бар’єрів;
- сприяє навчанню;
- покращення сфери медицини;
- покращення голосових помічників;

- автопілоти для автомобільної галузі;
- покращення рекомендаційних систем;
- покращений спосіб пошуку інформації;

Негативні сторони штучного інтелекту можна представити у вигляді наступного списку:

- зниження попиту на працівників деяких професій;
- надлишкове перекладання обов’язків на ШІ, що призводить до зниження якості навиків працівників;
- навчання генеративних мереж вимагає надзвичайної кількості навчальних даних;
- упередженість і дискримінація, яка виникає із неперевірених даних для навчання;
- різного роду етичні проблеми;
- ризики пов’язані із конфіденційністю даних користувачів;
- системи генеративних мереж використовують значну кількість апаратних ресурсів;
- значне споживання електроенергії;

Хоч штучний інтелект має безліч негативних сторін, але його внесок у покращення багатьох сфер діяльності важко переоцінити.

1.1.3. Етичні аспекти ШІ

Хоча штучний інтелект є революційною технологією, проте через його вплив на суспільство та економіку викликає низка етичних питань, пов'язаних з його використанням. Відповідно ці питання потрібно вирішувати, щоб гарантувати відповідальне, справедливе і безпечне застосування штучного інтелекту.

Штучний інтелект є досить упередженим щодо певних соціальних, расових чи гендерних питань. Воно й не дивно, для навчання використовується велика кількість різної інформації, яка не завжди є конструктивною та толерантною до різних верств населення [9].

Ще одним аспектом є приватність та безпека персональних даних, які використовуються для навчання ШІ. Також не ясно хто несе відповідальність за скоєні збитки чи летальні випадки при генерації недостовірної інформації чи прийнятті рішення мережею. Як відомо, деякі компанії, як-от Tesla вирішили дане питання, ввівши правила для водіїв, які користуються їхнім автопілотом. Водіям заборонено забирати руки із керма чи пересідати на інше сидіння. Вся відповідальність на водієві транспорту, навіть якщо виникає дорожньо-транспортна пригода за помилки автопілота, адже водій зобов'язаний завжди бути за кермом та слідкувати за обстановкою на дорозі.

Варто також розглянути значну кількість скорочень місць, яка можлива із впровадженням штучного інтелекту в сфери, де зазвичай простої автоматизації було б недостатньо. Йдеться про дизайнерів, програмістів та інших спеціальностей, де досвід та вміння відіграють важливу роль у роботі. Зараз багато компаній вимагаються від працівників знань основних сервісів генеративного штучного інтелекту для його впровадження в роботі та оптимізації робочого процесу працівників.

Регулювання контенту штучного інтелекту – це те, що зараз активно обговорюють різні країни. Для цього потрібно запровадити різного роду міжнародні та національні стандарти. Зараз багато країн вже обмежили використання ШІ для написаннях наукових робіт. Також заборона стосується використання штучного інтелекту в навчальних закладах, щоб запобігти використанню ШІ учнями та студентами у своїх роботах.

1.2. Машинне та глибоке навчання

1.2.1. Основні типи машинного навчання

Машинне навчання є підгалуззю штучного інтелекту і дозволяє комп'ютеру вчитися на даних. Машинне навчання поділяється на декілька основних типів в залежності від типу даних та методів навчання моделей. Машинне навчання

підпорядковує в собі системи, які здатні навчатися на розмічених чи нерозмічених даних, при цьому будуючи для себе алгоритм роботи системи без написання логіки у вигляді коду програми. Розрізняють такі основні типи машинного навчання:

- Навчання з учителем (Supervised Learning);
- Навчання без учителя (Unsupervised Learning);
- Навчання з підкріпленням (Reinforcement Learning);
- Напівконтрольоване навчання (Semi-Supervised Learning).

1.2.2. Навчання з учителем

Навчання з учителем (Supervised Learning) – принцип навчання даним підходом полягає у використання заздалегідь позначених даних. Даний спосіб ще називають контрольованим навчанням. Кожен зразок навчального набору даних містить вхідні дані та правильну відповідь. Тобто для кожного завдання мережа має завдання і готову відповідь. Мережа навчається на деякому наборі таких даних і після навчання, може прогнозувати правильну відповідь на основі тільки вхідних даних. Моделі навчені даним методом є досить легкими для переналаштування, адже можна легко змінити навчальні дані та правильні відповіді до них. Поширеними алгоритмами цього типу навчання є:

- лінійна регресія;
- наївний баєсів класифікатор;
- дерева рішень;
- алгоритм *k*-найближчих сусідів;
- нейронні мережі (багатошаровий перцептрон).

1.2.3. Навчання без учителя

Навчання без учителя (Unsupervised Learning) – це контрольоване навчання на добірці даних, яка не має позначок правильних відповідей. За даного типу навчання відбувається пошук закономірностей в даних, тобто ШІ сам визначає закономірності без втручання людини. Даний тип навчання є протиставлянням навчанню із учителем.

Найчастіше дане навчання використовуються для побудови систем кластеризації даних. Це може бути групування даних, рекомендаційні системи чи сегментація клієнтів. Також за допомогою даного типу навчання тренують системи для виявлення аномалій. Йдеться про виявлення різного роду нетипових операцій у банківській системі чи простого виявлення зразків, які значно відрізняються у певній вибірці даних. Навчання без учителя є дещо неконтрольованим, адже ми не знаємо, як саме мережа навчиться на неанотованих даних.

Можна сказати, що перевагою даного типу навчання є мінімальна підготовка даних, яка не потребує анотації відповідей. Також моделі можуть самостійно виявляти закономірності. Найкраще проявляється виділення закономірностей в багатовимірних структурах даних, які людині важко сприймати. Ще однією типовою задачею є зменшення розмірності структури даних. Даний тип навчання добре підходить для цієї задачі. Під час навчання відбувається маркування основних важливих змінних структури даних, а незначні змінні просто відкидаються. В результаті ми отримаємо узагальнену, зменшену у розмірі структуру даних.

Основним недоліком є відсутність контролю. В залежності від кількості даних, вибору алгоритму та метрик оцінювання результат може бути не завжди передбачуваним.

1.2.4. Навчання з підкріпленням

Навчання з підкріпленням (Reinforcement Learning) – це навчання за допомогою взаємодії із середовищем, модель отримує нагороди за правильні дії та покарання за хибні. Навчання з підкріпленням базується на спробах і помилках, що і є основою роботи даного типу навчання. Агент взаємодіє із середовищем, отримує зворотній зв'язок і перелаштовує свою поведінку відповідно до позитивної чи негативної оцінки результату своїх дій. В процесі навчання модель вчиться максимізувати нагороду. В такому методі навчання є:

- агент – це машина яка буде виконувати дії, при цьому даний агент не має жодної інформації про середовище;

- середовище – це об’єкт чи сукупність об’єктів, із якими може взаємодіяти агент;
- стан – певна ситуація, в якій агент знаходиться;
- дія – всі можливі способи взаємодії, які може робити агент;
- система винагород та покарання за вірні чи невірні прийняті рішення.

При дослідженні середовища модель в кожен дискретний момент часу виконує певну дію, яка може призвести до отримання винагороди, або до отримання покарання. Для позначення винагороди часто використовують числові значення, часто – це бали.

Коротко описати порядок дій моделі можна наступним чином. Спочатку модель отримує початковий стан, далі виконує дію, яка відповідає стратегії роботи системи. Після виконання дії, система отримує винагороду, що слугує зворотнім зв’язком. На основі позитивної чи негативної винагороди відбувається оновлення стратегії роботи системи для покращення результату роботи. Наступним кроком система переходить до нового стану і процес повторяється знову.

На основі навчання з підкріпленням будуються моделі, які використовуються для автономних систем управління роботами, а також їх використання можна зустріти у навчанні моделей на базі різних комп’ютерних чи настільних ігор.

Перевагою навчання з підкріпленням є здатність агента вчитися на основі нерозмічених даних у процесі взаємодії із середовищем. Недоліками є тривалий час навчання, чутливість до системи винагороди та розмитість рішень при навчанні.

1.2.5. Напівконтрольоване навчання

Напівконтрольоване навчання (Semi-Supervised Learning) – це навчання, яке поєднує в собі два типи навчання – навчання без учителя та навчання із учителем. Зазвичай дані моделі навчаються на невеличкій кількості анотованих даних та на великій кількості неанотованих даних. Це дозволяє моделі розуміти, що вона має робити на початку навчання, а вже потім використовуючи неанотовані дані, модель старається покращити свої результати.

Перевагою даного типу навчання є знатність до навчання на невеликих обсягах анотованих та великих обсягах неанотованих даних, що економить час та гроші на підборі даних для навчання. Недоліком є залежність від даних, адже потрібно правильно підібрати анотовані дані для навчання.

1.2.6. Проблеми при навчанні моделей штучного інтелекту

Безліч питань, щодо навчання ШІ виникає, ще до початку самого навчання певної моделі штучного інтелекту. Однією із проблем є вибір вірної моделі для завдання, яке необхідно вирішити. Наприклад, всім відомий алгоритм лінійної регресії підходить для простих завдань, як-от оцінка заробітної плати залежно від досвіду роботи та освіти. Проте даний алгоритм не підходить для завдань, де потрібно працювати з нелінійними залежностями. Для даних завдань підходить поліноміальна регресія, яка розширює можливості звичайної лінійної регресії.

Проблеми можуть виникнути також на етапі збору даних, адже для навчання необхідно велика кількість даних. При цьому для різних типів навчання можуть бути необхідні як неанотовані, так і анотовані дані. Зі свого боку, варто зазначити, що анотування даних вимагає значної професійної роботи, що не завжди можливо. Також дані для навчання можуть бути незбалансовані. Наприклад, у навчальному сеті є фотографії котів та собак, проте фотографії собак займають 73% від загальної кількості зображень. Це і є дисбалансом даних. Для збалансування даних можна наростити дані, яких є менше. Ще одним способом є ігнорування незбалансованих даних, проте, даний варіант підходить тільки у випадках, коли якась характеристика не є однією із ключових, або коли у навчальному наборі даних є ще багато інших характеристик, які є збалансованими. Також можна скористатися аугментацією. Аугментація – це додавання нових даних шляхом модифікації вже наявних даних із навчальних даних. Наприклад, з однієї фотографії роблять декілька, просто віддзеркаливши та поставивши фото під різними кутами.

Коли у навчальному наборі даних надзвичайно велика кількість характеристик, то навчання може бути неефективним через те, що деякі моделі не

знатні працювати при великій кількості параметрів. Також в даних бувають шуми та пусті дані. Тому перед навчанням дані ретельно перевіряють на наявність різного роду аномалій, при цьому їх видаляють, або заміняють на валідні дані.

Після формування всіх даних для навчання може виявитись, що для навчання просто недостатньо обчислювальних ресурсів. Для вирішення цієї проблеми використовуються розподілені обчислення, зменшуються розмірність даних та оптимізують алгоритми.

Також після навчання можна стикнутися із перенавчанням. Цей термін характеризує моделі, які добре запам’ятали навчальні дані, проте, використавши модель на нових даних вона показує погані результати. Незначні зміни характеристик нових даних відносно навчальних, сприймаються моделлю, як невірні.

На відміну від перенавчання, існує і термін недонавчання. Він характерний для моделей, що навчалися на малій вибірці даних, або на неякісних даних.

1.2.7. Глибоке навчання

Глибоке навчання (DL) – це підмножина машинного навчання, що використовує більш просунуті архітектурні рішення для вирішення завдань. В більшості рішень машинного навчання використовуються багатошарові нейронні мережі. Моделі побудовані на основі глибокого навчання здатні самостійно виокремлювати релевантні ознаки та шаблони з вхідних даних. Нейронні мережі складаються з декількох шарів нейронів, де кожен шар може виділяти певні ознаки з навчальних даних. Кількістю шарів характеризується глибина навчання і здатність мережі до ефективного виділення ознак та виконання поставлених завдань. Їхня ефективність досягається завдяки здатності обробляти величезні обсяги даних, але це зі свого боку вимагає значних обсягів навчальних даних і обчислювальної потужності. Глибоке навчання ефективно використовується в комп'ютерному зорі, обробці природної мови (NLP), рекомендаційних системах та конструюванні ліків. У медицині даний тип навчання дає змогу виявляти різного роду хвороби з високою точністю.

Також глибоке навчання посприяло розвитку наступного напряму штучного інтелекту – генеративному штучному інтелекту. На даний момент цей напрям активно розвивається та має попит серед звичайних користувачів.

1.3. Типи генеративних моделей штучного інтелекту

1.3.1. Генеративні змагальні мережі (GAN)

Одним з типів машинного навчання (ML) є генеративна змагальна мережа (GAN). Ян Гудфеллоу та його колеги надали натхнення для розробки цієї мережі. У 2017 році розпочалася робота над застосуванням GAN на людських обличчях [10-11].

Даний тип мережі здатен згенерувати зображення практично ідентичні навчальному набору та здатен генерувати зображення людей високого рівня реалістичності.

Основний принцип роботи – це робота двох мереж у парі. Перша – це непряме навчання через дискримінатор, друга – це мережа, що перевіряє зображення на те, наскільки згенероване фото є реалістичним. Можна сказати, що генератор зображення намагається обманути дискримінатор, надаючи йому все краще і краще зображення. А дискримінатор в свою чергу намагається відрізняти згенероване зображення від навчального зразка.

На практиці початковими навчальними даними дискримінатора є відомий набір даних. Дискримінатор навчають, надаючи йому вибірки з набору даних, поки він не досягне заданого рівня точності. Зазвичай спочатку на генератор надсилаються дані, вибрані випадковим чином (шуми), а потім дискримінатор оцінює зразки, створені генератором. На рисунку 1.2. зображено елементарну схему GAN мережі. Обидві мережі використовують метод зворотного поширення помилки, яка допомагає дискримінатору краще розпізнавати синтетичні зображення, одночасно покращуючи якість зображення генератора. Зазвичай, згорткова нейронна мережа слугує дискримінатором, а деконволюційна нейронна мережа – генератором [12].

GAN мережі застосовують для генерації фотореалістичних зображень різного

роду. Це можуть бути зображення людей, тварин, споруд, інтер'єрів одягу та споруд. За допомогою генеративних змагальних мереж збільшують роздільну здатність зображень, відновлюють пошкоджені зображення та використовують для розробки анімації. Також широкого застосування дані мережі зазнали у сфері кібербезпеки. Наприклад, ми можемо розглянути використати GAN для покращення систем виявлення фішингових листів. Мережа з кожним разом генерує все реалістичніший фішинговий лист, тим самим доповнює базу навчальних зразків для систем виявлення фішингових листів.

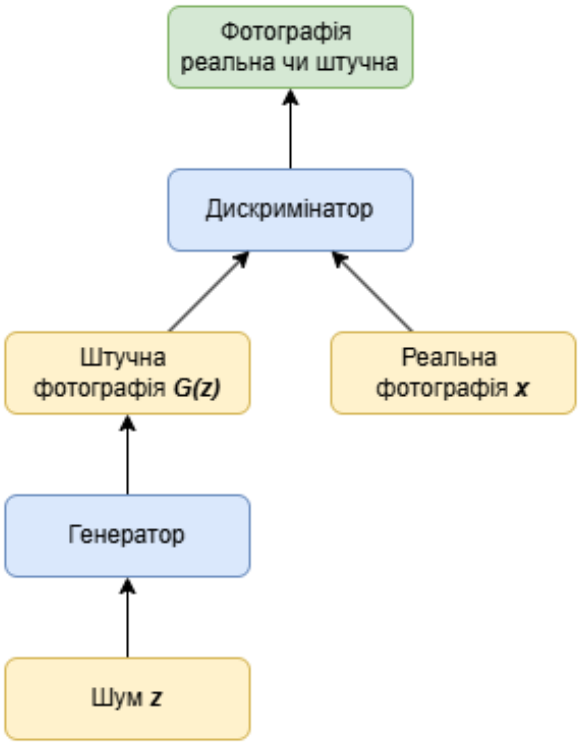

Рис. 1.2. Схема роботи GAN мережі

Досягнення цілей генерації зображення є складним процесом, тому для кращого розуміння роботи мережі, розглянемо її формулу. Генеративні змагальні мережі складаються з генератора G та дискримінатора D, які змагаються між собою, де генератор навчається генерувати реалістичні вибірки даних, в той час як дискримінатор намагається розрізнити реальні та згенеровані вибірки. Мета генератора – "обдурити" дискримінатор, створюючи вибірки, які максимально

наближені до реальних даних [13]. Механізм навчання в GAN – це гра двох гравців, які змагаються один проти хто з них кращий, кожен у своїй справі. На основі цього аналогу ми визначаємо оптимізаційну задачу, що характеризує взаємодію G і D:

$$\min_G \max_D V(D,G) = E_{x\sim p(x)}[log\, D(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))], \quad (1.1.)$$

де $p(\mathrm{x})$ – розподіл даних;

$p_z(\mathrm{z})$ – випадковий шум, який використовується для створення зразків;

$D(x)$ – ймовірність визначення x, як реальних даних;

$G(z)$ – згенеровані дані;

$D(G(z)))$ – ймовірність визначення синтетичних даних дискримінатором, які згенерував генератор як реальних.

Хоч і генеративні змагальні мережі підходять для генерації зображень, проте вони мають деякі проблеми. Однією із проблем є балансування генератора та дискримінатора під час навчання. Може виникнути ситуація, що дискримінатор занадто швидко визначить зображення, як реальне, що призведе до генерації зображення поганої якості. Також є проблеми пов’язані з генеруванням даних із конкретно зазначеними характеристиками та проблеми витрат обчислювальних ресурсів для якісного навчання.

Conditional GAN (cGAN) – генеративна змагальна мережа, яка побудована на основі базової мережі GAN, проте із покращенням. Покращенням слугує використання допоміжної інформації. Для цього використовують мітки класів для генератора і дискримінатора. Це дозволяє вирішити проблему однотипності генерації зображень, даючи можливість генерувати зображення різних типів.

CycleGAN – це мережа, яка здатна переносити характеристики одного зображення на інше. Вона навчається з непарних наборів даних. Архітектура даної мережі містить два генератори та два дискримінатори. Для зображень є дві області, які позначаються як X і Y . Генератор G приймає зображення з X як вхідні дані та

старається створити зображення в Y, яке має обманути дискримінатор D_X. Схожим чином генератор F генерує зображення у протилежному напрямку та намагається обдурити дискримінатор D_Y [14]. Елементарну схему роботи можна побачити на рисунку 1.3. Даний процес повторюється в декілька кроків. Дана модель роботи схожа на цикл, тому й дістала таку назву.

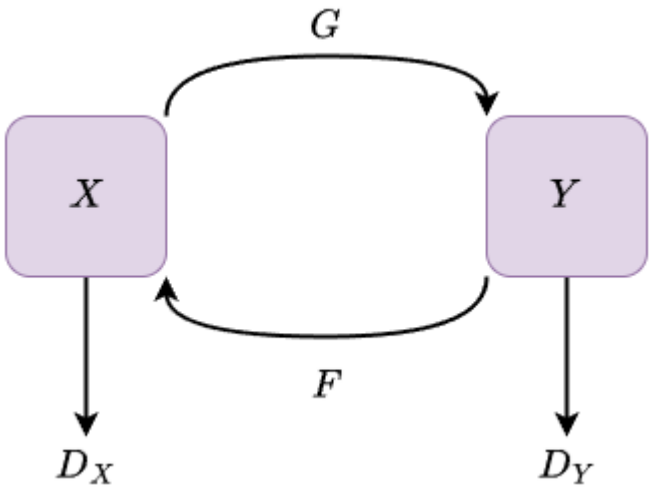

Рис. 1.3. Елементарна схема роботи CycleGAN.

Можна ще розглянути DCGAN (Deep Convolutional GAN). Даний тип генеративної мережі, як і попередній, призначений для покращення вже існуючої мережі GAN. В основі DCGAN, як в GAN лежать генератор та дискримінатор, однак до цієї зв’язки додається згорткова нейронна мережа. За допомогою додаткового компонента DCGAN мережа є стабільнішою під час навчання та дає кращі результати за звичайну GAN мережу. Deep Convolutional GAN покращує результати властивостей згенерованих зображень, і цьому слугують декілька причин. Перша, DCGAN мережа використовує згорткові мережі на дискримінаторі та генераторі для заміни шарів об'єднання. Друга причина – це використання алгоритму пакетної нормалізації для вирішення проблеми зникнення градієнта. Третя причина – це те, що в DCGAN мережах використовуються різні функції активації, та оптимізації, що значно покращує продуктивність та здатність GAN мережі генерувати якісніші зразки даних [15].

CNN часто застосовують через їхню здатність автоматично виявляти та аналізувати патерни в зображеннях. CNN ефективно зменшують розмірність вхідних даних, зберігаючи основну інформацію. Ця особливість робить згорткові мережі ідеальними для класифікації зображень [16]. CNN є потужним інструментів завдяки своїй ефективності та точності у машинному навчанні.

Існує й інша велика кількість GAN мереж, які покращують та доповнюють базову структурну модель. Серед них: Wasserstein GAN (WGAN), SRGAN (Super-Resolution GAN), BigGAN та інші.

1.3.2. Дифузійні моделі

Дифузійні моделі (Diffusion Models) – генеративні моделі, які продукують нові дані із шуму. Дані моделі були представлені у 2015 році, проте значний успіх дифузійних моделей припав на 13 квітня 2022 року, коли було анонсовано DALL-E 2 від компанії OpenAI, яка використовувала дану модель [17]. Дана модель може генерувати реалістичні зображення з випадкового шуму. На даний момент застосування дифузійних моделей є досить поширеним, адже вони здатні легко працювати з генерацією зображення, відео, графіки, також здатні прибирати шуми із зображення. Дифузійні моделі, як вже було сказано, входять до складу великих моделей, які складаються із кількох менших.

Елементарний принцип роботи дифузійних моделей для генерації зображень полягає у перетворенні текстового опису, так званої підказки, у зображення. Текстова підказка слугує списком вказівок, що має бути присутнім на зображенні. Далі спочатку створюється гаусівський шум, що проходить крізь згорткову нейронну мережу. Серед згорткових мереж часто застосовується U-Net, через її ефективність витягувати та відновлювати просторові ознаки із зашумлених даних. Прохід шуму через згорткову мережу відбувається за попередньо визначену кількість кроків. На кожному кроці згорткова нейромережа робить обчислення та прогнозує весь шум, що присутній на зображенні. Після виконання всіх ітерацій модель формує представлення згенерованого зображення. Далі для отримання зображення із

представлення використовують декодувальні моделі, такі як, варіаційний автокодувальник. За допомогою даного декодувальника ми формуємо зображення у звичних для нас пікселях, декодуючи його із латентного простору мережі [18]. На рисунку 1.4. представлено просту дифузійну модель.

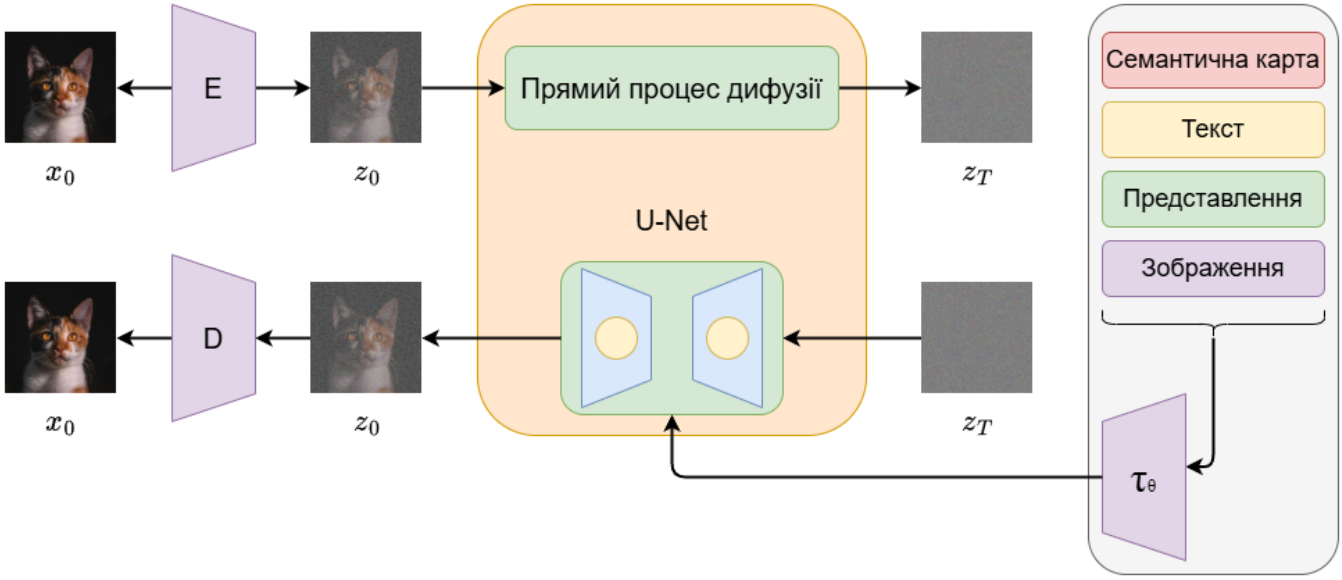

Рис. 1.4. Схема роботи дифузійної моделі.

Принцип роботи даних моделей полягає у поступовому додаванні гаусівських шумів до зображення, допоки зображення повністю не буде складатися із шуму. Після цього модель намагається поступово із шумів отримати зображення. Модель генерації зображення починається з випадкового шуму. Після навчання реверсу процесу дифузії на природних зображеннях, модель зможе генерувати нові реалістичні зображення високої якості. Коротше кажучи, модель перетворює дані на шум, а потім пробує їх максимально точно відтворити (рис. 1.5.). Додавання шуму не відбувається миттєво, даний процес займає декілька ітерацій, при цьому зменшення шуму також відбувається з ітераціями, поки реконструюються дані.

Дані моделі добре підходять для генерації зображень та відео у хорошій якості, добре справляються з усунення шумів із зображень. Також можуть генерувати текст та аудіо. Популярними комплексними моделями, які використовують дифузійну модель є Stable Diffusion та DALL-E 2, однак такі генеративні моделі зазвичай

використовують не одну модель, а декілька у зв’язці.

Рис. 1.5. Процес додавання та усунення шуму із зображення.

Тепер розглянемо формули роботи дифузійних моделей. Дифузійна модель визначається двома процесами. Перший – прямий процес, що поступово наповнює дані шумом $x_0 \sim \mathrm{q}(\mathrm{x}_0)$ протягом T тактів [19]:

$$q(x_t|x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t \mathbf{I}\right), \quad (1.2.)$$

де $t \in \{1,2,\ldots,T\}$ – номер кроку;

x_t– це стан даних на кроці t;

β_t – дисперсія, яка визначає рівень шуму на кроці t;

N – нормальний розподіл;

I – одинична матриця.

Другий – це зворотний процес:

$$p_\theta(x_{t-1}|x_t) = N\left(x_{t-1}; \mu_\theta(x_t,t), \sum_\theta(x_t,t)\right), \quad (1.3.)$$

де: $\mu_\theta(x_t,t)$ – середнє значення;

$\sum_\theta(x_t,t)$ – дисперсія;

θ – параметри моделі.

Принцип роботи дифузійних моделей полягає у поетапному зашумленні вхідного зображення (його позначають, як x_0), яке відбувається з певним кроком, допоки зображення не перетвориться на суцільний шум. Це називають прямим процесом. Протягом процесу додавання шумів ми пов'язуємо кожну ітерацію зашумлення із наступною, використовуючи стани x_{t-1} – для вхідного стану та x_t – для вихідного стану. Тобто ми створюємо своєрідний ланцюг, який з'єднує всі ітерації процесу додавання шуму для зображення. Це формує марківський ланцюг, де кожен стан залежить лише від попереднього [20]. Даний алгоритм можна представити наступною формулою:

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1.4.}$$

Тепер розглянемо зворотній процес. Він схожий на попередні, але вхідними даними є шум. Робота починається з найбільш зашумленого зображення, яке береться із випадкового нормального розподілу з нульовим середнім і одиничною дисперсією.. Береться зашумлене зображення, з якого поступово прибираємо шум, ітеративно відновлюючи зображення на кожному кроці через умовний розподіл $p_\theta(x_{t-1}|x_t)$, доки не отримаємо чисте зображення. Формула виглядає наступним чином:

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{1.5.}$$

Процедура зворотного зв'язку позначена, як p_θ. Формула моделює, як перейти від зашумленого стану x_t до менш зашумленого стану x_{t-1}, водночас поступово зменшуючи шум, щоб відновити чисте зображення. Умовний розподіл відображає, як зменшити шум на кожному кроці, щоб поступово відновити чисте зображення, де за формулою 1.3., вхідними параметрами даного розподілу є середнє значення та дисперсія.

Здатність отримувати зображення із шуму – це головна частина процесу навчання дифузійних моделей, яка забезпечує їх здатність працювати у зворотному напрямку та відновлювати дані із шуму, які були попередньо зашумлені. За всіма вище згаданими формулами можна побудувати систему, яка в результаті зможе генерувати зображення з шумів.

Як вже було сказано вище, дифузійні моделі добре підходять для генерації зображення, проте, вони також застосовуються для розфарбовування, редагування та сегментації зображень [21].

Перевагами дифузійних моделей є висока якість згенерованих зображень, стабільність навчання, а до недоліків варто віднести вимоги до обчислювальних ресурсів та тривалий процес генерації.

1.3.3. Автокодувальники

Автокодувальник – це некерована нейромережева модель машинного навчання, яка була створена для зниження розмірності. Автокодувальники також здатні виконувати кластеризацію, виявляти аномалій, крім того, їх використовують для побудови рекомендаційних систем [22]. Звичайно базова модель не здатна справитися із всіма зазначеними завданнями, проте автокодувальники, як і GAN мережі, мають модифіковані версії, які підходять для тих чи інших завдань. Простий автокодувальник складається з декількох частин. Він містить кодувальник (кодер) – компонент, що стискає вхідні дані у латентний простір, а частина, яка намагається відновити стиснені дані – це декодувальник (декодер). Також є вхідний та вихідний шари, які призначенні для отримання даних на вході та створення схожого результату на виході відповідно. Між кодером та декодером є латентний простір, в якому містяться ключові характеристики вхідних даних у вигляді вектора.

Значної популярності у моделях генерації зображень зазнали варіаційні автокодувальники (VAE). Їх будова схожа на звичайні автокодувальники, проте архітектура варіаційного автокодувальника представляє кожну латентну змінну як два вектори, що подають вектор середніх значень та вектор стандартних відхилень.

На рисунку 1.6 продемонстровано просту схему варіаційного кодувальника. Стандартні автокодувальники призначені для точного відновлення вхідних даних, в той час як VAE мають на меті згенерувати нові зразки на основі схожих даних. Варіаційні автокодувальники застосовуються в інших генераційних моделях для покращення результатів їх роботи. Прикладом використанн VAE є дифузійна модель Stable Diffusion [23].

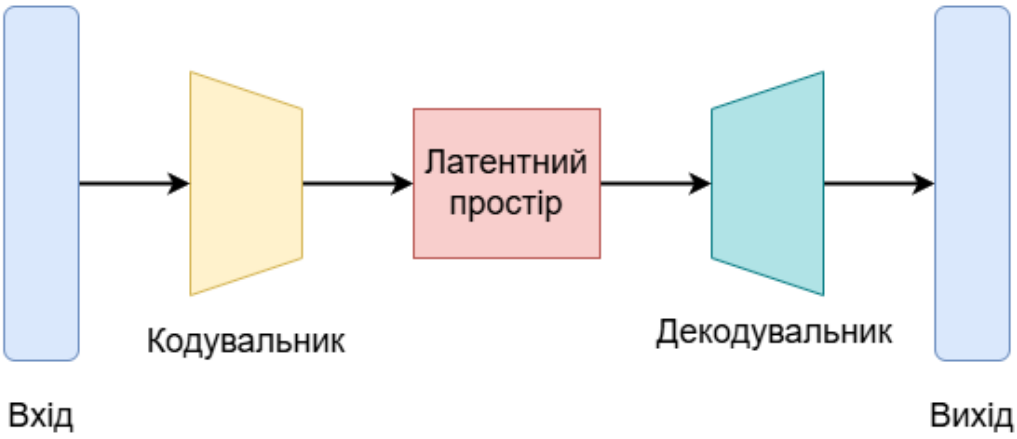

Рис. 1.6. Схема простого варіаційного автокодувальника.

До переваг варіаційного автокодувальника можна віднести здатність генерувати нові екземпляри даних. Серед недоліків наявні проблеми із продуктивністю та присутність розмитості згенерованих екземплярів відносно оригіналу – це так звані втрати при реконструкції.

1.3.4. Трансформери

Архітектура трансформерів – це нейромережева архітектура, яка використовує принципи механізму уваги. Ідею роботи трансформера можна порівняти з уважним книжковим критиком, який розбирає книгу на складові частини, як от розділи та абзаци тексту. При цьому критик зосереджується на ключових моментах, щоб визначити найбільш важливі для розуміння елементи книги. Такий принцип підходить і для зображень, де зображення розкладається на складові, які поодинці

можна оцінити, щоб скласти загальну картину розуміння композиції. Для даного типу мереж моделювання залежностей у великих масивах даних є досить ефективним, адже мережа з легкістю справляється з цим. Трансформери допомагають нам ефективно сегментувати зображення на ключові частини і розпізнати технічні та стилістичні аспекти твору. Це досягається за допомогою механізму уваги, коли мережа одночасно утримує фокус на різних елементах, беручи до уваги контекст [24].

Тепер розглянемо основні формули, які необхідні для розуміння уваги та багатоголової уваги. Формула 1.6. описує обчислення уваги:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (1.6.)$$

де Q – матриця запитів (query);

K – матриця ключів (key);

V – матриця значень (value);

d_k – розмірність ключів;

softmax – функція, яка перетворює ваги в імовірності, що вказують на важливість елементів.

У формулах 1.7. та 1.8. описано обчислення багатоголової уваги:

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right), \quad (1.7.)$$

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O, \quad (1.8.)$$

де $head_i$ – кожна "голова" має свої параметрами W_i^Q, W_i^K, W_i^V;

W^O – кінцева проєкційна матриця, що об'єднує виходи всіх голів у загальний вектор.

Трансформери використовують самокерованого навчання (SSL). Даний тип навчання є серединою між навчанням з учителем і без учителя. Особливістю даного

методу навчання є опрацювання великої кількості немаркованих даних, при цьому мережа сама для себе створює відповідні мітки. Можна сказати, що модель сама для себе розставляє мітки та навчається на цих промаркованих даних, а вже потім навчається на невеликому обсязі даних промаркованих людиною. Після цього мережу налаштовують для виконання певних завдань. Даний підхід добре підходить для навчання трансформерів.

Тепер розглянемо принцип роботи трансформерів. Архітектура трансформера складається із двох елементів: кодувальника та декодувальника. При цьому дані елементи складаються із менших елементів.

Спочатку розглянемо кодувальник та його допоміжні елементи. Перед входом до кодувальника відбувається перетворення вхідних токенів (тексту) у числові вектори за допомогою шару вкладення. Хоча архітектура трансформера призначена для роботи з послідовностями, проте розташування кожного фрагмента в послідовності явно не враховується. Щоб обійти це обмеження, використовують позиційне кодування. Завдяки позиційному кодуванню модель здатна витягувати просторову інформацію із зображення та враховувати відносне розташування фрагментів [25].

Далі переходимо безпосередньо до кодувальника. Він також складається з двох елементів: багатоголової уваги та нейронної мережі прямого поширення. Тож, багатоголовий механізм уваги дозволяє пов’язувати слова, які мають смисловий зв’язок. Показники уваги обчислюються на основі запиту, ключа та значення. Після того, як попередньо задані вектори пройшли через лінійний рівень, шляхом множення матриці скалярного добутку між запитами та ключами створюється матриця оцінок. Матриця балів відображає ступінь зв’язку слів по відношенню до інших слів. Вищий бал є індикатором високої ймовірності використання.

Наступний крок – це зменшення величини оцінки уваги простим діленням їх на квадратний корінь розмірності векторів ключа. Ця процедура допомагає отримати стабільніші градієнти. Тепер йде застосування функції *softmax* для отримання вагових коефіцієнтів уваги, які представлені в вигляді ймовірностей у діапазоні від 0 до 1. При

цьому дана функції підвищує високі бали, а низькі знижує. Це допомагає підвищити результативність виявлення слів, яким необхідно виділити більше уваги [26]. Після цього відбувається множення, раніше отриманих ваг, на вектор значень. В результаті вийде вектор виходу. За допомогою цих маніпуляцій ми отримаємо лише дані з високими оцінками, які далі подаються на лінійний шар. Багатоголова увага дістала свою назву через те, що вона складається із декількох шарів, де замість одного виконання механізму уваги, виконується декілька функції уваги паралельно [27]. Під час цього процесу кожна стадія працює незалежно, створюючи вихідний вектор.

Далі відбуваються кроки нормалізації та додаються залишкові з'єднання до кожного підшару, включаючи нейронну мережу прямого поширення. Всі ці дії зменшують проблему зникнення градієнта. Виходом кінцевого рівня кодувальника є набір векторів.

Тепер розглянемо декодувальник з його допоміжними складовими. Як і в кодувальника на вході приймаються вхідні дані, ділі відбувається позиційне кодування. Далі перейдемо до багатоголової уваги, адже тут є деякі ключові відмінності від кодувальника. Особливістю є обмеження переходу позиції токена до наступного токена послідовності. За допомогою даної особливості регулюється контроль у механізмі уваги, що, до прикладу, дозволяє генеративним моделям забезпечувати послідовний процес перекладу тексту.

Також не менш важливою особливістю є те, що кодувальник з'єднується із компонентами всередині декодувальника. На рисунку 1.7. можна побачити дане з'єднання та й загалом всі складові компоненти стандартного трансформера. Це допомагає вирівнюванню входу кодувальника і декодувальника, що в результаті дає змогу декодувальнику розпізнавати і виділяти найбільш відповідні частини вхідного сигналу кодувальника. Далі принцип роботи нейронної мережі прямого поширення знову стає схожим на принцип роботи кодувальника. Після цього є деякі відмінності, дані проходять через лінійний рівень, який виконує формування набору оцінок для кожного можливого токена, а за допомогою функції *softmax* – відбувається розподіл та нормалізація оцінок на ймовірності, як і в попередньому використанні функції

softmax. По суті кінцевим результатом декодувальника є результат функції *softmax*, що представляє словники з ймовірнісними коефіцієнтами використання слів.

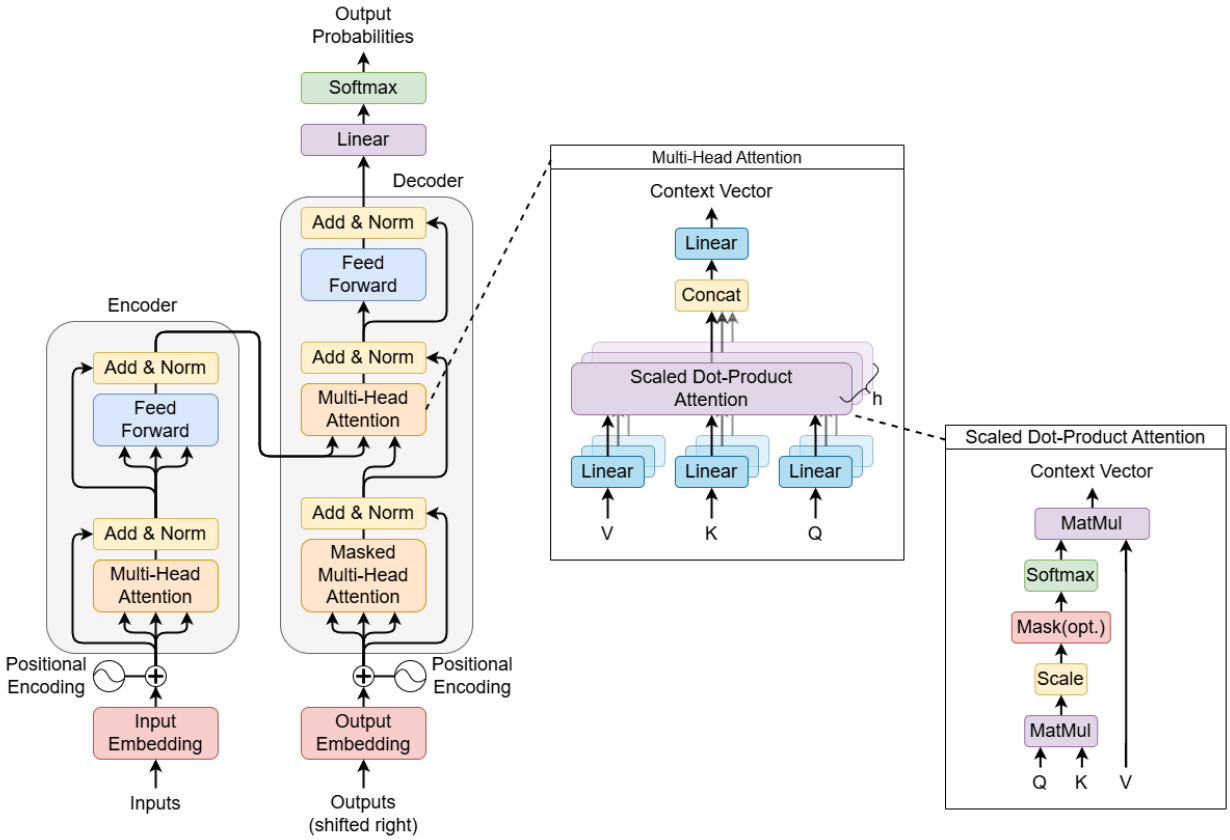

Рис. 1.7. Схема основних складових моделі трансформера.

Декодувальник трансформера генерує токени один за одним, використовуючи раніше згенеровані токени, як вхідні дані декодувальника для шару вкладення. Цей весь процес повторюється, доки в результаті ми не отримаємо згенерований результат, тобто текст.

Варто відзначити що кодувальник та декодувальник може складатися із N шарів. Структура із N шарів дозволяє моделі ефективно обробляти складні залежності та диверсифікувати фокус уваги в кожній із голів. Шари з’єднуються послідовно, тобто на вхід шар отримує вихідні дані із попереднього шару. Ця N-шарова стратегія значно підвищує здатність моделі до прогнозування, оскільки вона отримує більш досконале розуміння різних комбінацій уваги.

Застосування трансформерів є досить широким. Воно включає вирішення

завдань у сфері обробки природньої мови (NLP) та перекладу [28]. Трансформери також добре справляються із написанням коду різними мовами програмування, справляються із задачами комп’ютерного зору та використовуються як рекомендаційні системи. Трансформери відіграють важливу роль для генерації зображень. Вони чудово справляються з інтерпретацією тексту у інструкції процесу генерації зображень для інших генеративних моделей ШІ.

Трансформери, як і люба система має свої переваги та недоліки. До переваг можна віднести паралельні обчислення, що значно впливає на швидкість навчання. Трансформери є досить гнучкими для виконання ними різного розу завдань, а їх механізм уваги позволяє ефективно обробляють послідовності будь-якої довжини. Також варто відзначити їх архітектуру, що дозволяє моделі легко масштабуватися. А серед недоліків є такі: великі вимоги до апаратної частини та значні витрати електроенергії при роботі системи. Також недоліком є складність інтерпретування прийняття тих чи інших рішень моделлю та залежність від великої кількості навчальних даних.

1.4. Висновки до розділу

В даному розділі було розглянуто основні теоретичні аспекти штучного інтелекту. Визначено ключові поняття та методи навчання штучного інтелекту. Розглянуто основні проблеми при навчанні моделей штучного інтелекту. Виконано опис основних типів генеративних моделей штучного інтелекту, розглянуто їхнє значення у генерації зображень та проаналізовано їх переваги та недоліки.

Окремо було розглянуто різні етичні аспекти використання ШІ, зокрема ризики пов’язані з генерацією даних та впливом цих даних на суспільство. Цим аналізом підкреслено важливість відповідального використання ШІ в різних сферах життя.

Отже, цей розділ закладає теоретичну основу для подальшого дослідження та розробки системи генерації зображень.

РОЗДІЛ 2
2. АНАЛІЗ ІСНУЮЧИХ ГЕНЕРАТИВНИХ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ ТА МЕТОДИ ЇХ ІНТЕГРАЦІЇ У СИСТЕМИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

В цьому розділі буде розглянуто сучасні існуючі генеративні моделі штучного інтелекту для генерації зображень та проведено їх аналіз. Дослідження в даному напрямку є досить перспективним, адже все більше сфер, в яких відбувається застосування генеративного штучного інтелекту, зокрема і для генерації зображень. Буде досліджено способи інтеграції даних генеративних моделей в сторонні сервіси. Також буде розглянуто кожну модель детально та описано її характеристики, як-от цінова політика, якість згенерованих зображень, можливість використання у комерційних цілях та архітектура моделей. Всі ці питання буде детально розглянуто та запропоновано оптимальну модель для інтеграції у системи генерації зображень.

2.1. Сучасні генеративні моделі для генерації зображень

2.1.1 DALL-E

DALL-E – це генеративна модель штучного інтелекту, яка розроблена компанією OpenAI. Назва походить від двох назв. Перша – це назва одного із ключових персонажів мультфільму ВОЛЛ-І (WALL-E). Друга – це прізвище відомого іспанського художника Сальвадора Далі. Дана комбінація слів і утворила назву даної моделі [29].

У січні 2021 року було вперше представлено першу модель DALL-E, при цьому розробка була профінансована компанією Microsoft у вигляді гранду на суму 1 мільярд доларів [30]. Після цього, у квітні 2022 року була анонсована наступна версія – DALL-E-2. Дана версія не була публічною, доступ надавався за запрошенням обмеженій кількості людей. Ймовірно, таким чином компанія хотіла отримати відгук від користувачів та провести перевірку продуктивності системи в реальних умовах.

Дана модель могла генерувати реалістичні зображення за допомогою введеного тексту. Також модель мала функціонал для редагування зображення, які вже були згенеровані. Пізніше, у вересні 2022 року, доступ до моделі отримали всі охочі. Зараз останньою версією генеративної моделі компанії OpenAI є DALL-E-3, яка була представлена у вересні 2023 році.

Перша версія DALL-E була розроблена на основі генеративного попередньо тренованого трансформера (GPT). GPT – це велика мовна модель (LLM), яка здатна до обробки природної мови. Будова DALL-E складається з таких основних компонентів як:

- дискретний VAE;
- трансформер;
- декілька CLIP кодувальників зображення та тексту.

Дискретний варіаційний автокодувальник призначений для перетворетворення зображення у дискретне представлення. За принципом роботи він схожий на варіаційний автокодувальник. Їх головною відмінністю є те, що дискретний VAE дозволяє будувати представлення на основі списку існуючих токенів, замість безперервного генерування представлення, як у звичайних VAE. Тобто це дозволяє параметризувати очікуваний результат зображення. Це необхідно через те, що текстові запити, введені користувачем, теж представленні у вигляді дискретного представлення, тобто у вигляді токенів. Представлення запиту користувача у вигляді токенів відбувається за допомогою трансформера. Його основним завданням є переклад людського текстового запиту у токени, які в подальшому будуть використовуватися для генерації зображення.

Окремої уваги заслуговує такий компонент, як CLIP (порівняльне тренування мовних зображень). За допомогою даної моделі відбувалося тренування моделі GPT. Призначення максимально просте та зрозуміле – це розставлення маркерів на вибірці тренувальних даних, мається на увазі що кожне зображення отримає деякі мітки із перелічених об’єктів на зображенні. Навчання моделі CLIP відбувалося на наборі зображень із інтернету, які мали текстові описи. Навчання відбувалося за допомогою

самокерованого навчання, про яке також згадується у першому розділі під час опису архітектури трансформерів. Під час навчання створюються векторні представлення зображення та тексту у латентному просторі, що в подальшому дозволяє генерувати зображення, яке максимально відповідає текстовому опису. Під час роботи DALL-E даний компонент відповідає за зв'язування згенерованого зображення із текстовим описом, тим самим виконує оцінювання наскільки зображення відповідає текстовому опису.

Наступна версія генеративної моделі отримала назву DALL-E 2, зазнавши деяких архітектурних змін. Ключова відмінність, яка відрізняє нову версію від попередньої – це використання дифузійної моделі замість авторегресійного трансформера. Дане нововведення значно покращило генеративні можливості нової версії.

Основним завдання DALL-E є генерування зображення різних стилів. Це можуть бути реалістичні зображення чи відтворення реальних витворів художнього мистецтва. Чудовою функціональною можливістю є здатність до редагування чи доповнення вже наявних згенерованих зображень.

Генерація зображень є досить простим процесом, адже все що потрібно зробити користувачі – це ввести текстовий опис і модель сама розуміє, що їй далі робити. Модель розуміє, що малювати та де розставляти елементи. Мається на увазі, коли користувач вводить запит про неіснуючу істоту чи об'єкт, який тримає щось чи взаємодіє із світом, то генеративна модель чудово розуміє, що і де потрібно розставити. Модель чудово справляється із розстановкою голови, тіла та інших компонентів, щоб вони виглядали максимально органічно. Також моделі добре справляються із домальовуванням прогалин, тобто елементів оточення чи інтер'єру, які користувач безпосередньо не вказав. Наприклад, коли попросити мережу згенерувати осінній ліс, вона спокійно зможе згенерувати ліс, при цьому листя дерев будуть жовтавих та жовтогарячих відтінків, що притаманно даній порі року.

З появою DALL-E 2 також з'явилася можливість створювати різні варіації зображень на основі згенеровано зображення, при цьому за допомогою контекстної

підказки можна уточнювати, як саме наступне фото буде відрізнятися від оригіналу.

На даний момент остання версія генеративної мережі – DALL-E 3. Вона є покращеною версією попередника, здатна обробляти складні текстові описи та генерувати якісніші зображення. Також варто зазначити, що версії DALL-E 2 та DALL-E 3 доступні через API для розробок із інтеграцією у програмні продукти. Ціни варіюються в залежності від версій моделей, а також якості та роздільної здатності зображень.

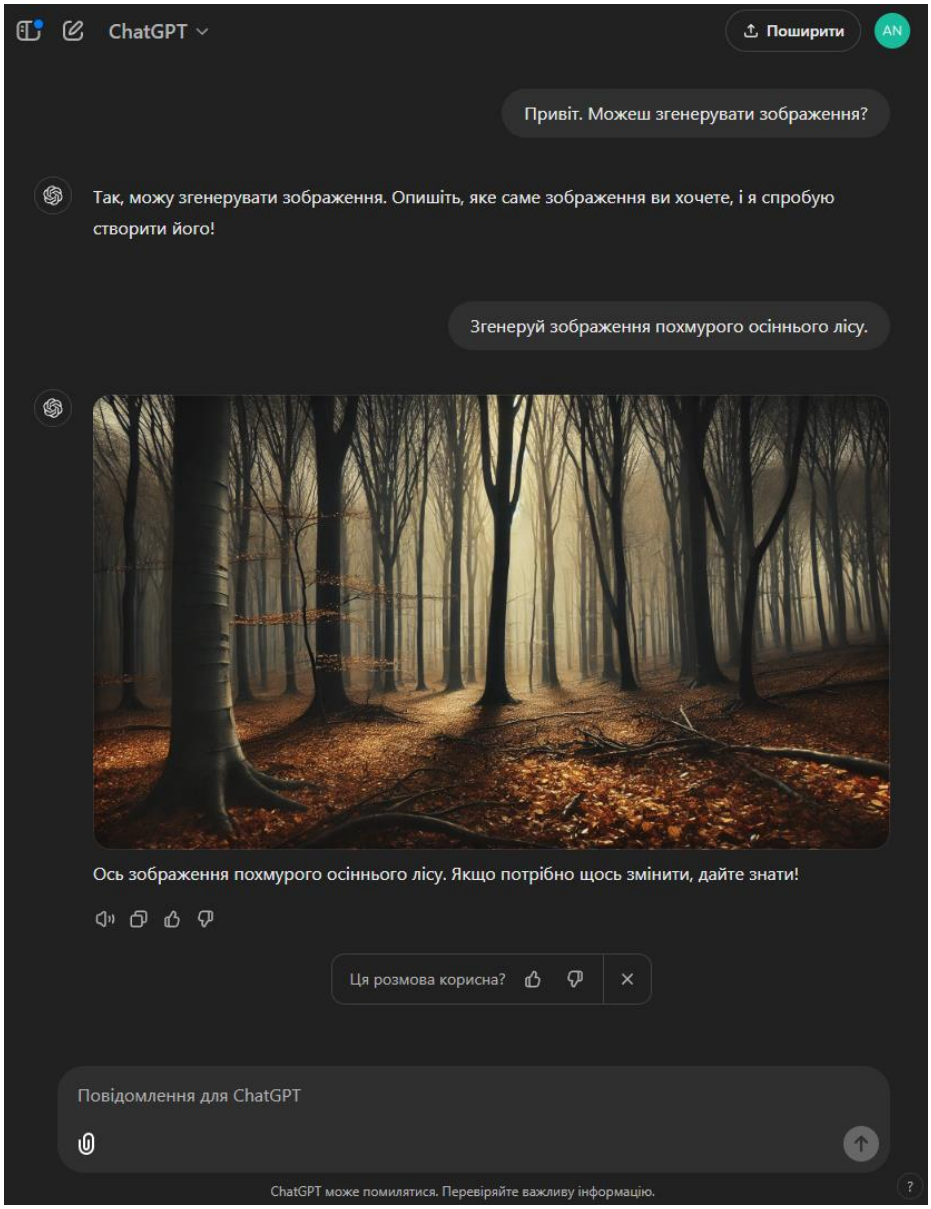

Рис. 2.1. Генерація зображення за допомогою DALL-E 3

Проаналізувавши основні особливості DALL-E, можна сказати, що він є досить простим, адже даний сервіс доступний для звичайних користувачів у вигляді чату (рис. 2.1.). Користувач просто вводить текстовий опис і отримує відповідне зображення. Генеративна модель інтегрована в ChatGPT, що і робить її такою зручною. ChatGPT є мультимодальною системою, що дозволяє працювати із різними типами вхідних даних, тим самим полегшуючи роботу з генерацією зображень. Вхідними даними можуть слугувати: текст, зображення, введення голосом. Це робить систему максимально зручною для звичайного користувача, дозволяючи взаємодіяти з нею, у любий доступний спосіб. Чудовою особливістю є те, що сервіс розуміє українську мову. Він автоматично перекладає текстову підказку та доповнює її для кращого результату.

Серед недоліків можна відмітити нижчу якість згенерованих зображень порівняно із конкурентами. Також надто простий функціонал може слугувати недоліком для користувачів, які прагнуть ширшого функціоналу.

2.1.2 Stable Diffusion

Stable Diffusion – це модель генеративного штучного інтелекту, призначена для перетворення текстового опису у зображення. Це дифузійна модель, яка була випущена у 2022 році і є головним продуктом компанії Stability AI.

Stable Diffusion зазвичай використовується для генерації високоякісних зображень, проте, як і інші генеративні моделі на ринку, не обмежується тільки цим. За допомогою даної моделі так само можна редагувати вже існуючі зображення. Модель дає змогу генерувати серію зображень за раз, що дає змогу краще оцінити згенерований матеріал та вибрати кращий результат.

Перша архітектура моделі була розроблена на основі моделі латентної дифузії (LDM). Моделі на основі латентної дифузії виконують процес дифузії в латентному просторі, при цьому відбувається економія витрат на навчанні та отриманні результату генерації [31]. Загалом Stable Diffusion складається з трьох основних компонентів. Першим компонентом є варіаційний автокодувальник, який

призначений для зменшення розмірності піксельного зображення та представлення його у латентному прості. Цей процес відбувається за допомогою поступового додавання гаусівського шуму до стисненого зображення у латентному просторі під час процесу прямої дифузії. Перетворення зображенні в латентний простір є необхідним, оскільки обчислювальні витрати значно знижуються. У латентному просторі зберігається лише ключова інформацію про структуру зображення, що значно зменшує вагу представлення у порівнянні із його піксельним представленням. Другий компонент – згорткова нейронна мережа U-Net. Вона призначена для зворотного процесу, а саме ітераційного отримання представлення із латентного простору, при цьому очищуючи наявний шум із зображення. І останній ключовий компонент – це текстовий кодувальник, що забезпечує перетворення тексту на представлення, що слугує інструкціями для процесу керування генерацією.

За допомогою механізму перехресної уваги та мережі U-Net відбувається знешумлення кондиціонованих даних. Під кондиціонованими даними, ідеться про проекційний шар, який об’єднує текст і зображення в один простір. Дане з’єднання забезпечує зв’язок між текстовим описом і зображенням. У моделі Stable Diffusion для цього застосовується CLIP мережа – це фіксований попередньо навчений текстовий кодувальник, призначений для створення вкладень слів (word embedding), які керують процесом генерації зображень. Коротко принцип роботи можна описати так, спочатку вхідний текст перетворюється на векторне представлення у латентному просторі завдяки текстовому кодувальнику CLIP. Після цього, в наслідок роботи блоку U-Net, через ітеративний зворотний процес дифузії відбувається процедура генерації зображення. За допомогою механізму перехресної уваги забезпечується уточнення процесу генерації зображення шляхом використання текстових підказок користувача. В результаті, після очищення декодувальником латентного представлення від шумів, ми отримаємо готове зображення.

Наступні версії мали мінорні оновлення аж до версії XL. Дана версія є максимально схожою до попередніх версій, проте є й нововведення. Нова модель отримала покращення мережі U-Net, додатковий текстовий кодувальник та

покращення перехресної уваги.

Stable Diffusion 3.0 отримала значні зміни в архітектурі. Вона базується на дифузійному трансформаторі. Дана модель є мультимодальною, тобто вона здатна окремо опрацьовує зображення та текст. Також дана модель навчилася краще розуміти підказки і одночасно концентруватися на багатьох об’єкта.

Остання актуальна версія це 3.5. Вона включає декілька різних типів цієї моделі:

- Stable Image Ultra – флагманська модель, яка поєднує в собі потужність Stable Diffusion 3.5 Large із вдосконаленими робочими процесами для отримання фотореалістичних зображень найвищої якості;
- Stable Diffusion 3.5 Large – має 8 мільярдів параметрів, ідеальна для професійного використання з високою якістю генерації зображення;
- Stable Diffusion 3.5 Large Turbo – спрощена та прискорена версія Stable Diffusion 3.5 Large;
- Stable Diffusion 3.5 Medium – має 2,5 мільярда параметрів, це баланс між якістю та простотою налаштування. Оптимальна версія для встановлення на користувацькі системи.

Stable Diffusion є open souce проектом, що дозволяє різними дослідникам та розробникам використовувати напрацювання компанії Stability AI у своїх інтересах. Завдяки відкритості моделей Stable Diffusion, вони й набули значної популярності. Це дозволяє користувачам з усього світ розробляти застосунки та покращення для даних моделей. В додачу до інтерфейсу користувача StableStudio, існує ще безліч популярних користувацьких інтерфейсів. Найпопулярнішим є AUTOMATIC1111 Stable Diffusion Web UI [32]. Вона надає доступ до функцій, які не доступні у звичайні версії Stable Diffusion та використовує деякі оптимізації для роботи системи.

Ще однією перевагою даної моделі є здатність розширювати навчальні набори даних, що дає можливість генерувати зображення інших напрямів, які не були доступні для звичайної версії. Однак проблемою є значні системні вимоги, адже для додаткового навчання знадобиться потужний комп’ютер та додатковий час. Також

проблемою є те, що новий набір може взагалі не покращити генеративні характеристики оновленої моделі, а в деяких випадках й погіршити.

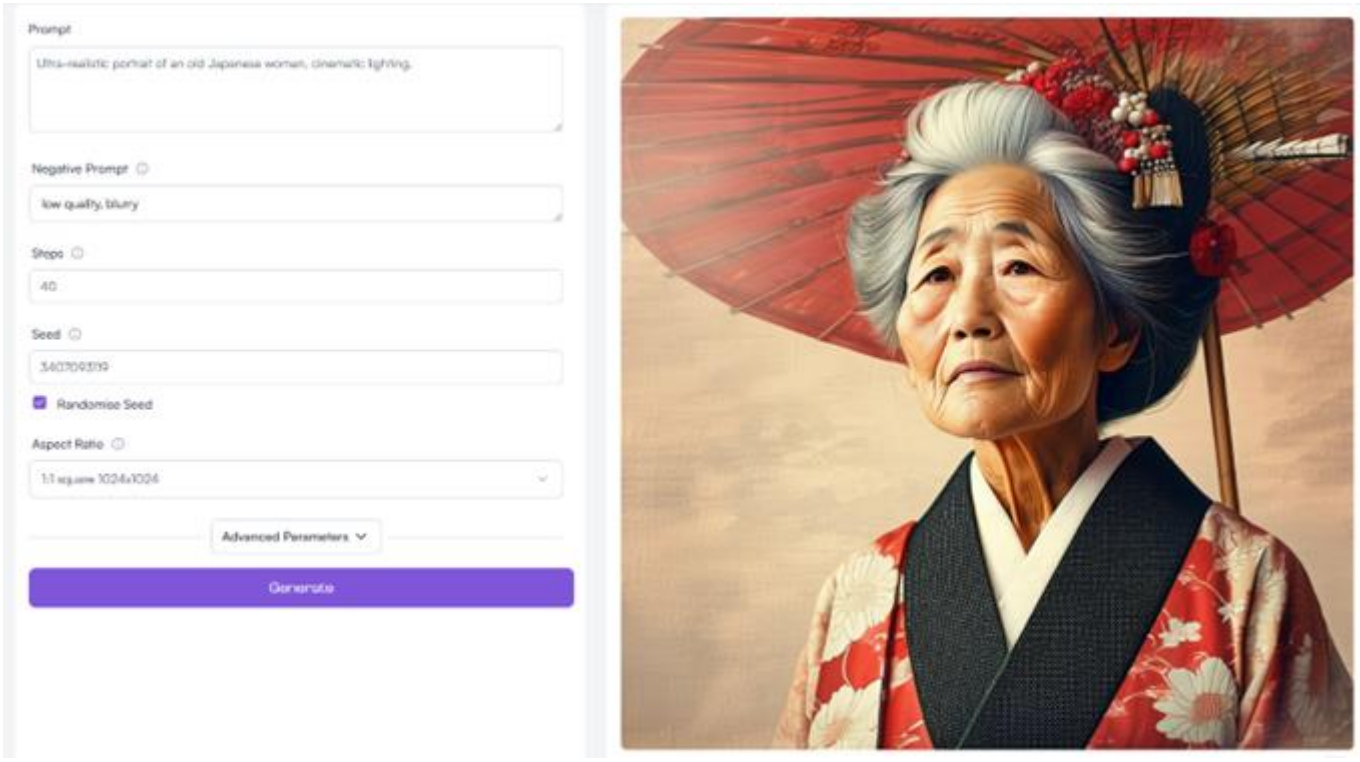

Рис. 2.2. Генерація зображення за допомогою сервісу Segmind та моделі Stable Diffusion 3.5 Large

Великою перевагою Stable Diffusion є здатність встановити його на локальний комп’ютер. Однак, як зазначалось раніше, системні вимоги вимагають потужного комп’ютера, що не завжди доступний у рядового користувача. Здебільшого, користувачі використовують сторонні сервіси, які розміщають дану модель на хмарному сервісі. Дані сервіси дають можливість користувачам генерувати зображення за невеличку плату. Компанія Stability AI надає доступ на використання їхнього API. Користувачі отримують токени, які можуть використати на доступні генеративні моделі. В залежності від якісних характеристик, кількість токенів на генерацію зображення може варіюватися. Серед популярних сервісів є DreamStudio від компанії Stability AI, який дозволяє користувачам генерувати безліч зображень використовуючи браузер, надаючи їм декілька стартових токенів. Я скористався

сервісом Segmind для генерації зображень, який дозволяє використовувати різні моделі Stable Diffusion та надає стартовий рахунок для тестування роботи системи. На рисунку 2.2. продемонстровано процес генерації зображення за допомогою даного сервісу та моделі Stable Diffusion 3.5 Large. Також даний сервіс надає достатню кількість налаштувань доступних для генерації зображень.

Також варто відзначити, що Stable Diffusion поширюється по двох ліцензіях для розміщення на серверах. Перша призначена для дослідників, розробників та творців. Вона має обмежену кількість доступних моделей та є безкоштовною для осіб та підприємців із доходами менше 1 мільйона доларів на рік. Друга – призначена для осіб з доходами в понад 1 мільйон доларів на рік та має спеціальне ціноутворення.

Проаналізувавши дану генеративну модель, можна сказати, що Stable Diffusion є універсальним інструментом і чудово підходить для генерації зображення у різних стилях. Дає можливість використання як для звичайних користувачів так і для дослідників. Однак варто зауважити, що для користування Stable Diffusion прийдеться регулярно платити, або користуватися сторонніми безкоштовними сервісами, яка надаються доступ до моделей Stable Diffusion. Це пов’язано з тим, що компанія Stability AI себе позиціонує більше, як компанію, яка надає доступ до своїх генеративних моделей по API для великих бізнес-клієнтів.

2.1.3 Генеративні моделі компанії Leonardo.AI

Leonardo.AI – це потужний сервіс для роботи з генерацією зображень, який надає широкий спектр інструментів та функціональних можливостей для звичайних користувачів та для людей, які на професійному рівні займаються творчістю. Leonardo.AI використовує машинне навчання для аналізу зображень та відео, комбінуючи це із сучасними підходами генерації зображень [33]. Застосування Leonardo.AI можливе в багатьох сферах, включаючи навчальні заклади, де вчителі можуть генерувати зображення для покращення та спрощення навчального процесу.

Leonardo.AI – це Сіднейська компанія, заснована у 2022 році в Австралії, яка була придбана компанією Canva за неоголошену суму. На даний момент, Canva і

Leonardo.AI працюють, як два окремі сервісні рішення.

Компанія надає зручний базовий тарифний план у якому доступно 150 кредитів на день. Використання кредитів залежить від різних характеристик зображення, які можна вибрати при генерації. Також базовий тариф включає до 200 дій для Realtime Canvas та до 200 дій для Realtime Generation. Однак, всі згенеровані зображення в базовому тарифі є публічними і доступні іншим користувачам.

В системі є можливість вибору моделі генерації, при цьому є безліч користувацьких моделей. Кожна із цих моделей має свою невеличку сторінку з описом та прикладами робіт згенерованими цією моделлю. Також є можливість одночасного генерування декількох екземплярів зображень до введеного текстового опису. Окрім цього, є можливість вибору співвідношення сторін, роздільної здатності та стилю зображення. При виборі кожної із функцій сумарно підраховується вартість генерації зображення у кредитах. Вона відображається на кнопці генерації зображення (рис. 2.3.).

Для користувачів, які не знають як правильно написати текстову підказку (prompt) для генеративної мережі є дві чудові функції. Перша – це випадкова генерація текстової підказки, що дозволяє згенерувати підказу з хорошим формуванням та можливо підкинути ідею користувачу для генерації зображення. Друга чудова функція – це генерування текстової підказки на основі вже наявної текстової підказки чи набору слів. Це дозволяє придумувати складніші ідеї на основі простої підказки та покращувати формування вже наявної підказки. Наприклад, якщо користувач введете підказку з одним словом “Кіт” і скористається функцією генерації підказки, то отримаєте підказку із ширшим описом зображення, яке буде включати розширену інформацію про кота.

Після формування текстової підказки можна згенерувати зображення. Це займає певний час, після чого можна завантажити дане зображення чи відредагувати його. Для редагування є чудова функція, яка дозволяє редагування зображення у реальному часі – це Realtime Canvas. Даний функціонал дозволяє вибрати наявне зображення. Після цього, користувач за допомогою зручного інтерфейсу, має

можливість малювати по зображенню, тим самим, у боковому вікні в режимі реального часу користувач може побачити різницю між оригінальним і згенерованим зображенням на основі правок. Якщо ж зображення має певні дефекти чи галюцинації, можна скористатися кнопкою вдосконалення зображення, щоб покращити його.

Ще цікавими функціями є Canvas Editor та Realtime Generation. Перша функція надає можливість виділяти конкретну область на зображенні та доповнювати її згенерованим контентом, відповідно до введеної текстової підказки. Друга ж генерує зображення в реальному часі відповідно до текстового опису та дозволяє регулювати декілька повзунків, які відповідають за певні стилі. Скажімо, можна перетягнути повзунок Coloring Book, що дозволить нам буквально за мить побачити, як зображення перебудовується. В результаті ми отримаємо рисунок, який на вигляд як рисунок із книжки. Дана функція має декілька повзунків, які можна комбінувати між собою.

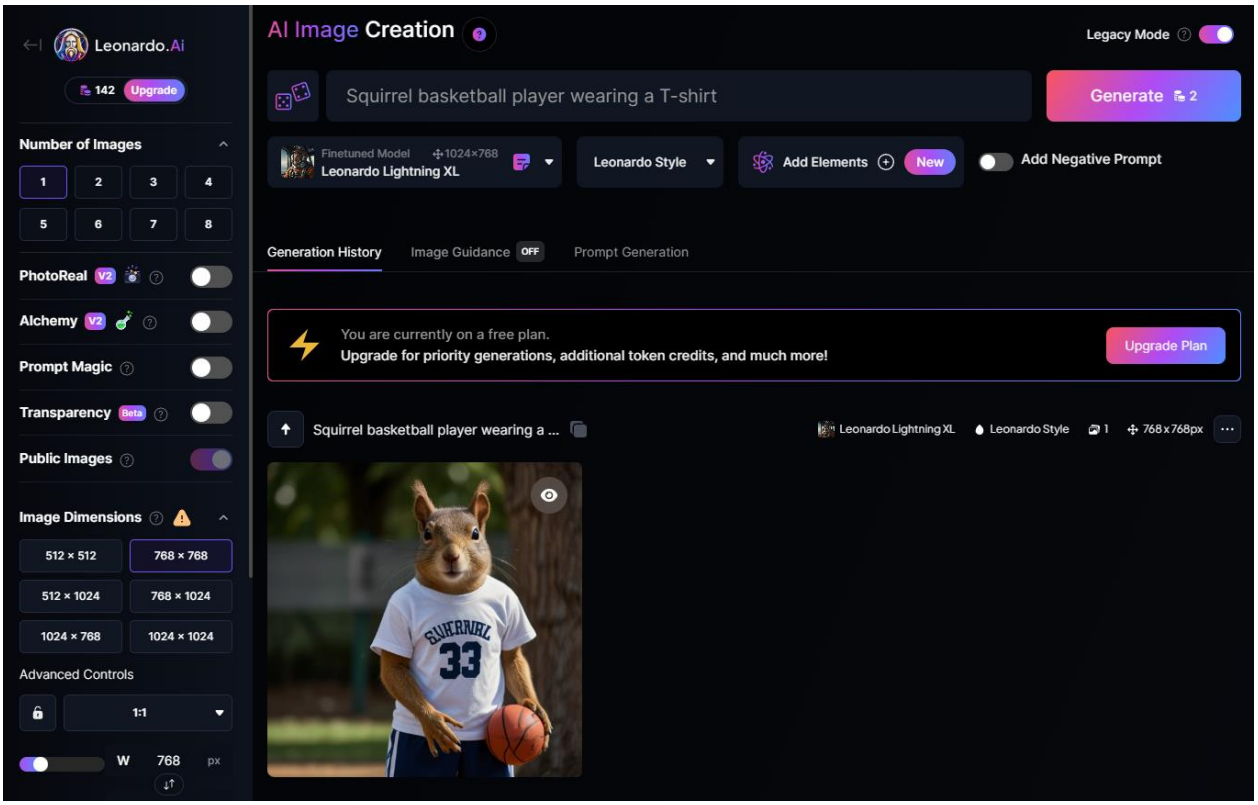

Рис. 2.3. Генерація зображення за допомогою сервісу Leonardo.AI та моделі Leonardo Lightning XL

На рисунку 2.3. продемонстровано інтерфейс та процес генерації зображення за допомогою моделі Leonardo Lightning XL. При виставлених параметрах генерації, що можна побачити на рисунку, генерація одного зображення вартуватиме 2 кредити.

Проаналізувавши функціональні можливості та аспекти використання Leonardo.AI, можна оцінити його переваги та недоліки. Серед переваг можна відмітити широкий спектр інструментів для створення та редагування контенту, що робить його універсальною платформою для різних потреб. Серед функціоналу доступні різні генеративні моделі та художні стилі, які можна активно застосовувати. Leonardo.AI має сучасний інтерфейс, що робить його привабливим для нових користувачів. Однак, я вважаю, що інтерфейс є не настільки простим, щоб ним могли користуватися нові користувачі, хоча на платформі є інструкції з використання сервісу. Скориставшись декількома функціями даної платформи, можу сказати, що Leonardo.AI може генерувати зображення за лічені секунди, що, безсумнівно, є чудовим вибором для швидкого створення контенту. Водночас, при швидкій генерації, Leonardo.AI генерує реалістичні та високоякісні зображення, що безсумнівно є великим плюсом. Також наявний базовий вартісний план, що є чудовим доповненням до плюсів системи. Це дозволяє новим користувачам ознайомитись із базовим функціоналом системи. Для професіоналів доступні декілька тарифних планів, що дає можливість вибрати план, який підходить як фінансово, так і функціонально.

Серед виявлених недоліків можна виділити вартість. Хоча платформа дозволяє згенерувати декілька зображень в базовому плані, проте, для активних користувачів цього буде не достатньо. Платні підписки можуть бути дорогими для деяких користувачів. Багато функцій та як-от: генерація високої якості, висока роздільна здатність чи привітність згенерованих зображень є доступною лише для платних планів. Leonardo.AI – це продукт, який орієнтується на професіоналів, тому для звичайних користувачів може здатися дещо складним у користуванні. Це хороший вибір для професіоналів, адже їм доступний широкий спектр функцій, що дає можливість використовувати генерацію зображень максимально якісно.

Ще одним недоліком є відсутність локалізації інтерфейсу користувача. Доступна тільки англійська мова, водночас введення текстових підказок теж обмежено суто англійською мовою.

2.2. Основи побудови текстових запитів для генерації (prompt engineering)

Підказка (prompt) – це набір інструкцій призначений для генеративного штучного інтелекту та написаний природною мовою [34].

Конструювання підказок (prompt engineering) – це техніка побудови оптимальних текстових запитів для генеративних мереж [35].

Конструювання підказок є важливою складовою генерації зображення. Вона безпосередньо впливає на процес генерації зображення та слугує інструкцією для детального пояснення генеративній моделі, що повинно бути виконано. Конструювання підказок важлива для всіх генеративних мереж. Від структурованості викладеного пояснення завдання залежать результати для генерації тексту, зображення чи програмного коду. При цьому конструювання підказок не обмежується текстом. Для мультимодальних систем таких, як ChatGPT доступні декілька типів введення. З цього можна зробити висновок, що формування належної підказки для генеративної мережі може бути й у вигляді голосового повідомлення чи зображення. Проте більшість систем орієнтовані на текстові підказки.

Структура ефективного запиту повинна бути правильно сформульованою. Інструкції повинні бути чіткими, з максимальною деталізацією та з використанням ключових слів. Якість результату безпосередньо залежить від якості складання текстової підказки. Добре сформованих запит дозволяє належним чином пояснити генеративній мереж, що від неї вимагається та відкрити її генеративний потенціал на повну.

Існують різні методики для кращого формування підказок. Більшість методів призначені для LLM моделей, адже вони працюють в контексті розмови (чату) та

враховують попередні результати. Ефективними методиками є: ланцюжок думок, підказування від найменшого до найбільшого, декодування узгодженості, самовдосконалення та інші. Проте для генеративних мереж призначених для генерації зображень дані методики не підходять. Дані моделі зазвичай не розуміють складні структури речень, тому для побудови підказок використовують простіші запити із використанням відповідних методик.

Для кращого результату при формуванні підказки для генерації зображення, використовують такі характеристики: яскравість, контраст, тип освітлення, стиль зображення та інші. При формулюванні підказки описують зображення простими словами, включаючи дрібні деталі для більш релевантного результату, однак, без складних синтаксичних конструкцій. Також на результат може впливати порядок слів у реченні. У моделях Stable Diffusion 3.5 large та Leonardo Lightning XL є можливість використати ще й негативну підказку. Негативна підказка слугує списком речей, об'єктів чи стилів, які не потрібно використовувати при генерації зображення [36]. Негативна підказка зазвичай виділена в окреме текстове поле. Це зроблено через те, що деякі моделі погано розуміють заперечення у звичайному реченні. А от для моделей DALL-E непотрібне використання додаткових полів, бо дані моделі працюють у парі з великою мовною моделлю ChatGPT, яка прекрасно розуміє семантику речень.

Ще одним із способі отримати хорошу підказку є використання сервісів, де є приклади готових підказок, або ж використати інший генеративний штучний інтелект для генерації підказки. На сервісі Leonardo.AI є можливість згенерувати випадкову або покращити вже існуючу підказку, що значно покращує якість та простоту генерації зображень. Однак, на даний момент треба враховувати і те що, генеративні моделі мають безліч додаткових параметрів, які значно впливають на результат генерації зображення, тому не варто спиратися тільки на добре написану текстову підказку.

Крім того, варто згадати про етичний аспект. Сучасні нейронні мережі мають безліч вбудованих фільтрів безпеки, які не дадуть вам можливості згенерувати

образливий, шкідливий чи чутливий контент.

Для того, щоб отримувати кращі результати від генеративних мереж можна слідувати наступним крокам:

1) Напишіть максимально коротко та ясно, що ви хочете побачити в кінцевому результаті, водночас уникайте складних формулювань. За потреби використайте негативну підказку.

2) Налаштуйте параметри моделі під потреби вашого зображення.

3) Згенеруйте зображення та оцініть відповідність вашим очікуванням. Якщо зображення не відповідає вашим очікуванням, то скоригуйте підказку та повторіть спробу.

Крім людей, які намагаються покращувати результати генерації за допомогою текстових підказок, існують і зловмисники, які використовують так звані підказкові ін’єкції. Це вид атак, який маніпулює генеративними моделями ШІ, змушуючи їх виконувати дії зловмисника [37]. Такі дії зловмисників спрямовані на отриманні даних в обхід заборонам та фільтрам генеративної моделі. Дані атаки в більшості спрямовані на LLM, адже там є більше можливостей обманути модель в контексті цілого чату. Однак для генеративних мереж, які призначені для генерації зображень також є свої вразливості. Однією з атак є пошук підказки схожої за семантикою, як до звичайної підказки, так і до підказки, яка перетинає межу фільтра безпеки [38]. Це відбувається через те, що схожі заборонені слова зберігаються у вигляді певних областей. Класифікатор безпеки визначає чи слова із підказки є забороненими, ділячи дані межею прийняття рішень. Оскільки нейронні мережі не здатні максимально точно класифікувати певні слова чи словосполучення, то виникають ситуації, де певних слів немає в списку заборонених. При цьому дані слова максимально близько знаходяться до межі прийняття рішення і максимально схожі за смислом до заборонених слів, що дозволяє зловмисникам генерувати заборонений контент в обхід фільтрам безпеки.

Для протидії маніпуляції через підказкові ін’єкції використовують різні підходи для унеможливлення шкідливих дій. Для цього навіть можуть

застосовуватися додаткові мережі, які призначені для перевірки написаної підказки на наявність підказкових ін’єкцій.

Отже, можна сказати, що конструювання підказок допомагає користувачам покращити очікуваний результат при генерації зображень. Однак, використання текстових підказок не завжди використовується у правомірних діях, проте розробники стараються покращувати методи розпізнавання та захисту від такого роду атак.

2.3. Порівняльна оцінка результатів генерації зображень

Метою даного підрозділу є порівняння декількох різних генеративних моделей штучного інтелекту у здатності їх до генерації якісних зображень. У даному порівняні було використано три моделі штучного інтелекту: DALL-E 3, Stable Diffusion 3.5 Large та Leonardo Lightning XL.

Кожна із моделей була налаштована під генерацію зображень у роздільній здатності 1024x1024 пікселів на дюйм. Для DALL-E 3 більше ніяких додаткових налаштування не виконувалося, оскільки в форматі чату немає налаштування даної моделі. Налаштування роздільної здатності було змінено, просто ввівши підказку, яка інформувала модель, що наступні згенеровані зображення повинні бути з вказаною роздільною здатністю.

Для Stable Diffusion 3.5 Large доступний значно ширший список налаштувань. На сервісі Segmind виставив такі налаштування:

- Negative Prompt (негативна підказка) – “low quality, blurry”, щоб зменшити кількість неякісних зображень. Даний параметр був заповнений за промовчуванням, нічого не редагував;
- Steps (кількість кроків для генерації зображення) – 40 кроків;
- Seed (випадкове число, початкове значення для створення зображення) – поставив галочку для випадкової генерації;
- Aspect Ratio (співвідношення сторін зображення) – 1:1 квадрат, 1024x1024;

- Guidance Scale (вплив підказки на результат генерації) – 4.5;
- Sampler (метод вибірки для формування зображення) – euler;
- Scheduler (планувальник усунення шуму) – sgm_uniform;
- With (ширина зображення в пікселях) – 1024;
- Height (висота зображення в пікселях) – 1024;
- Output Format (формат виводу) – jpeg;
- Image Quality (якість зображення) – 95.

На сервісі Leonardo.AI вибрав модель Leonardo Lightning XL, вимкнув стиль Leonardo Style. Також не використовувала негативна підказка, оскільки тут вона необов'язкова. Ось список налаштувань, з яким проводилась генерація зображень на даній платформі:

- PhotoReal (для надзвичайно реалістичних результатів) – вимкнуто;
- Alchemy (покращення генерації) – вимкнуто;
- Prompt Magic (краща відповідність зображення написаній підказці) – вимкнуто;
- Transparency (розмиття фону) – вимкнуто;
- Image Dimensions (роздільна здатність) – 1024x1024;
- Guidance Scale (вплив підказки на результат генерації) – 7;
- Tilling (повторювання текстур фону) – вимкнуто;
- Recommended sizes (автоматичне встановлення розмірів) – увімкнуто;
- Use fixed seed (випадкове число, початкове значення для створення зображення) – вимкнуто;
- Scheduler (планувальник усунення шуму) – Leonardo.

Загалом підсумовуючи, використав моделі з параметрами в більшості за промовчуванням, а підказку писав англійською мовою. Для сервісу Leonardo.AI вимкнув повзунки, які покращують зображення у якомусь певному стилі та ті, які вимагають великої кількості токенів.

Дане оцінювання є якісними аналізом, в якому буде проведено оцінку візуальної привабливості кожного із зображень, оцінку відповідності текстовій

підказці, буде розглянуто якість деталізації та оцінено наявність артефактів на зображеннях. Це порівняння базується на складанні суб'єктивного судження, щодо кожної із моделей генерації зображень.

2.3.1. Підказка №1

Метою порівняння за допомогою першої текстової підказки є визначення якості генерації зображення портрету людей за допомогою трьох різних генеративних моделей штучного інтелекту.

Згенеровані зображення відповідають підказці "Реалістичний портрет молодої дівчини" і їх можна побачити на рисунку 2.4. Розглядаючи зображення згенеровані моделлю DALL-E 3 можна однозначно сказати, що вони є згенеровані штучною мережею. Вони відмічаються своєю синтетичною.

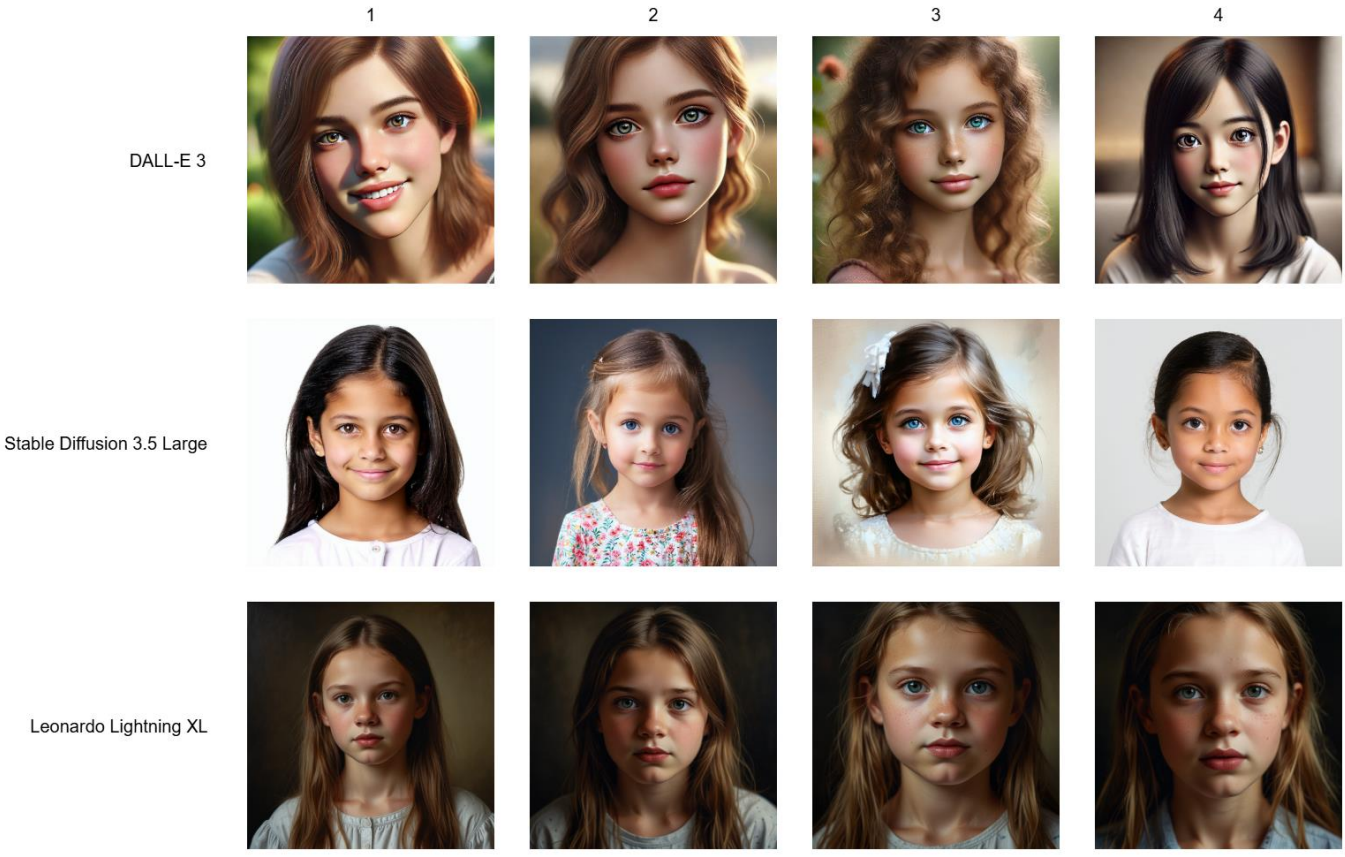

Рис. 2.4. Зображення, згенеровані генеративними моделями у відповідь на підказку "Realistic portrait of a young girl"

Тільки 3-й варіант має схожість із природним зображенням людини, однак придивившись детально все одно можна розпізнати штучність у зображенні. Щодо деталізації, всі зображення згенеровані з достатньою кількістю деталей. Серед артефактів, помітні незначні неточності в зображенні очей на 1, 3 та 4-му варіантах.

Генерація за допомогою моделей Stable Diffusion 3.5 Large та Leonardo Lightning XL значно випереджає модель DALL-E 3 як в деталізації, так і в реалістичності. Тільки 3-й варіант зображення моделі Stable Diffusion 3.5 Large виглядає штучно, проте досить високої якості. А от зображення згенеровані за допомогою Leonardo Lightning XL хоч і є дуже високої якості, однак мають явну присутність великої кількості різних фільтрів, які надають зображенням кіношного стилю. Також на 1-му зображення моделі Leonardo Lightning XL видно артефакти генерації очей.

В підсумку можна сказати, що всі моделі відповідають текстовій підказці. На мою думку, модель Stable Diffusion 3.5 Large показала себе найкраще, бо вона показала чудові результати, хоч і один із екземплярів є явно неприродним зображенням. Друге місце посідає Leonardo Lightning XL своєю здатністю фільтрування зображення до кіношної якості. Третє місце займає DALL-E 3, оскільки більшість згенерованих зображень є не настільки хорошої якості, як в його конкурентів.

2.3.2. Підказка №2

Використання даної підказки спрямоване для оцінки якості генеративних моделей для генерації зображень природи. Для цього була використана підказка “Похмурий, засніжений зимовий ліс”, а результати генерації для кожної з моделей можна побачити на рисунку 2.5.

Оцінюючи зображення згенеровані за допомогою DALL-E 3 можна сказати, що з природою дана мережа справляється значно краще і видає хороші результати, проте на зображеннях все ще видно штучність. Деталізація усіх чотирьох варіантів дуже висока.

Модель Stable Diffusion 3.5 Large цього разу згенерувала досить посередні результати. Зображення загалом непогані, однак досить багато артефактів, які кидаються в очі. Це і золотисте підсвічування біля підніжжя дерев, яке добре видно на 1-му рисунку, це і нетипова жовтава трава присутня на кожному екземплярі.

Оцінюючи результати генерації моделі Leonardo Lightning XL можна сказати, що вона найкраще справилася із цим завданням. Згенеровані зображення деталізовані, реалістичні та без явних артефактів. Згенеровані зображення природи виглядають наче це реальні фотографії.

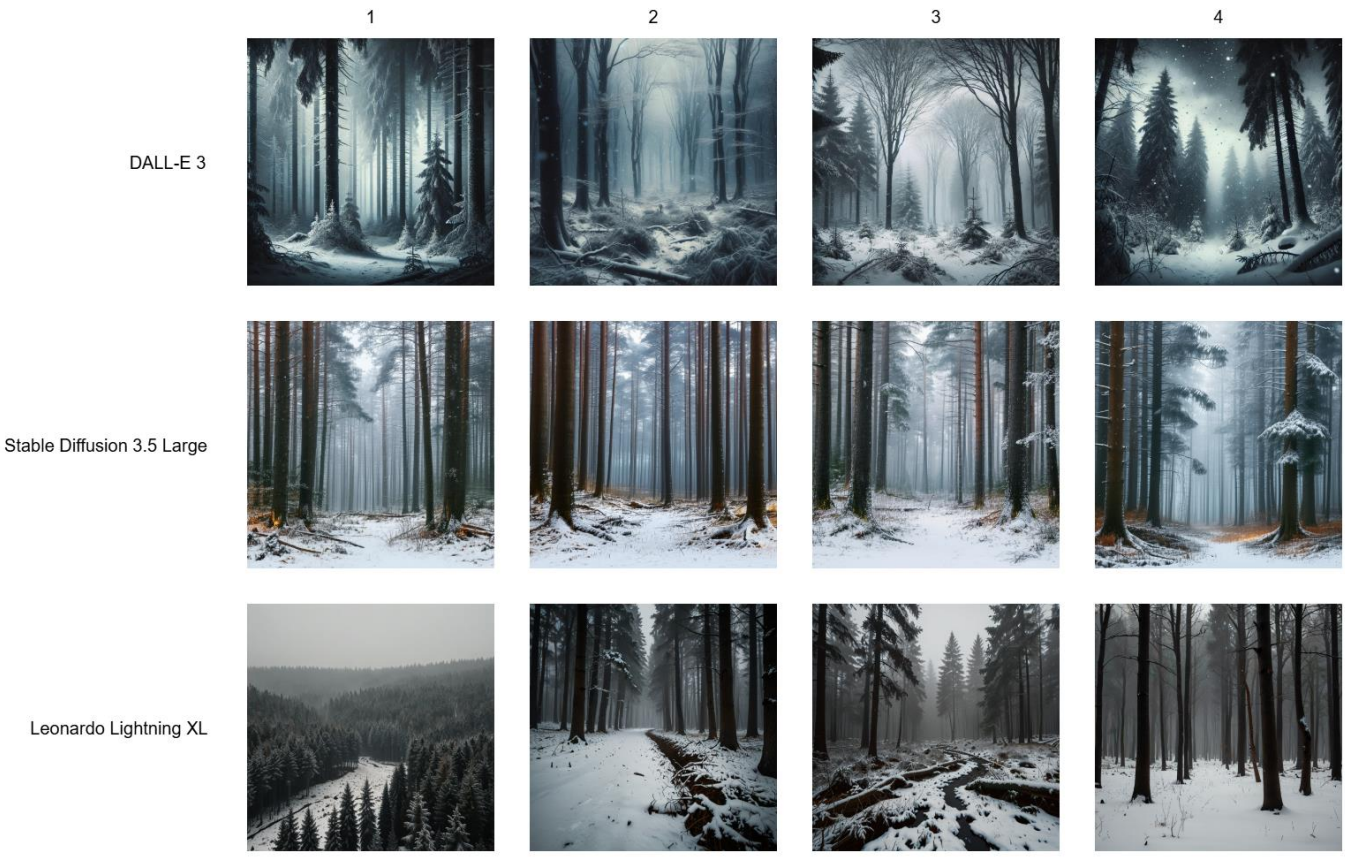

Рис. 2.5. Зображення, згенеровані генеративними моделями у відповідь на підказку “A gloomy, snowy winter forest”

Загалом кожне згенероване зображення відповідає текстовій підказці. В даному порівнянні Leonardo Lightning XL займає перше місце, а друге і третє місця – DALL-E 3 та Stable Diffusion 3.5 Large відповідно.

2.3.3. Підказка №3

Останньою текстовою підказкою є “Європейське село, люди йдуть вулицею, яскравий літній день”. Всі згенеровані зображення зображені на рисунку 2.6. Дане порівняння призначене отримати результати при складних умовах, де багато різних деталей.

Починаючи з DALL-E 3 зразу ж можна помітити, що зображення є штучними і в них є артефакти. Найбільш помітними є змазування обрисів обличь і кінцівок, нерівні віконні ставні та незрозумілі надписи.

Stable Diffusion 3.5 Large та Leonardo Lightning XL теж мають такі самі недоліки, як і в попередній моделі, проте якість зображень значно краща. Також варто відмітити, що зображення згенеровані моделлю Stable Diffusion 3.5 Large вирізняються натуральністю кольорів, в той час, як Leonardo Lightning XL генерує зображення з більш насиченими кольорами.

Рис. 2.6. Зображення, згенеровані генеративними моделями у відповідь на підказку “A European village, people walking down the street, a bright summer day”

Stable Diffusion 3.5 Large та Leonardo Lightning XL мають досить схожі результати, однак мені все ж більше до вподоби зображення згенеровані за допомогою моделі Stable Diffusion 3.5 Large. DALL-E 3 вочевидь відстає від свої конкурентів, проте має певний шарм.

Результати даного аналізу представлено у вигляді стислого короткого опису у вигляді таблиці (табл. 2.1.). Дані результати демонструють оцінку кожної із моделей за конкретним набором характеристик.

Таблиця 2.1.

Якісна оцінка генеративних моделей штучного інтелекту за результатами згенерованих зображень

Характеристика	**DALL-E 3**	**Stable Diffusion 3.5 Large**	**Leonardo Lightning XL**
Штучність зображення	сильно помітна	практично не помітна, за винятком артефактів	практично не помітна, за винятком артефактів та накладених фільтрів
Привабливість зображення	середня	висока	дуже висока
Наявність фільтрів	відсутні	відсутні	кіношний ефект
Відповідність підказці	висока	висока	висока
Рівень деталізації	середня	високий	високий
Наявність артефактів	незначні артефакти для підказки №1 та помітна кількість для підказки №3	невелика кількість для підказки №2 та помітна кількість для підказки №3	загалом мала кількість, помітна кількість для підказ №3

Підсумовуючи можна сказати, що Leonardo Lightning XL надав найкращі результати, у порівнянні з суперниками. Stable Diffusion 3.5 Large надає чудові результати, які схожі на реальні, однак присутні незначні неточності у генерації. DALL-E 3 зайняла третє місце через явну неприродність згенерованих зображень, однак, використання цієї моделі можливе для генерації більш синтетичних зображень, які чудово можна застосувати у сферах, де не потрібна надзвичайна реалістичність отриманих результатів.

2.4. Порівняння моделей для інтеграції в сторонні сервіси

Далі розгляну кожну із генеративних моделей, які розглядав у підпункті 2.1., у контексті використання їх функціоналу генерації зображень для розробки власних рішень та коротко зображу основні їх особливості у таблиці 2.2.

Для генерації зображень використовував сервіси OpenAI для DALL-E 3, Leonardo.AI для Leonardo Lightning XL. Для Stable Diffusion 3.5 Large скористався сервісом Segmind замість сервісу офіційного розробника (рис. 2.4.). Причиною стала невелика кількість стартових токенів для ознайомлення на сервісі Stability AI. А от на сервісі Segmind – я отримав стартовий капітал, який дає можливість згенерувати значно більшу кількість зображень для тієї ж генеративної версії моделі. До того ж, сервіс Segmind дає доступ до генерації зображень за допомогою інших генеративних моделей, як-от Flux чи Llama.

Спочатку розглянемо архітектурні особливості кожної із моделей. В основі DALL-E 3 лежать трансформатор та дифузійна модель. Наступним представником є модель Stable Diffusion 3.5 Large. Судячи із назви, як можна здогадатися, вона базуються на дифузійних моделях. А от про архітектури моделей Leonardo.AI не вдалося знайти достовірну інформацію. Судячи із активного використання дифузійних моделей в багатьох сучасних моделях, можна зробити припущення, що Leonardo.AI теж використовує дану технологію, адже сучасні дифузійні моделі дозволяють отримувати зображення відмінної якості.

Швидкість генерації зображень в кожному із сервісів займала від декількох секунд до однієї хвилини, при цьому якість зображень була достатньо хорошою. Найбільше часу на генерацію зображень потребував сервіс Segmind із моделлю Stable Diffusion 3.5 Large. Більш детально про якість генерації кожної із моделей було розглянуто та описано в підрозділі 2.3.

Кожен із наявних сервісів має API, який доступний для розробників. Водночас кожен із сервісів надає детальну документацію по роботі із API, демонструючи шматки коду, як приклади використання. Із трьох кандидатів, тільки OpenAI надає бібліотеку для взаємодії із сервісом, що полегшує процес розробки. Всі сервіси дають можливість використовувати API за допомогою HTTP запитів.

За можливістю налаштування параметрів генерації зображення OpenAI надає дуже скудний список налаштувань для своєї генеративної моделі DALL-E 3. Серед них є: текстова підказка, версія моделі, якість зображення, кількість зображень для генерації, формат відповіді та стиль згенерованого зображення. Конкуренти пропонують дещо ширший спектр налаштувань. До прикладу, Segmind для версії Stable Diffusion 3.5 Large надає ще доступ до таких функцій як-от: негативна підказка, число випадковості покоління, шкала строгості виконання підказки для отримання результату та інші. Найбільше налаштувань надає сервіс компанії Leonardo.AI. Він надає такі можливості, як-от: контраст, покращення підказки, стиль зображення, режим алхімії та багато інших функцій для покращення зображення.

Інфраструктура всіх сервісів використовує хмарний підхід, оптимізований для масштабованості та балансування навантаження. З поміж усіх представників, модель Stable Diffusion 3.5 Large можна встановлювати локально. Це забезпечує гнучкість, особливо для організацій з власною інфраструктурою та власними потребами щодо використання генеративних моделей. Однак, варто враховувати те, що використання локальної моделі для малих проектів буде зовсім не вигідним, адже для обслуговування серверів необхідні матеріальні та фінансові ресурси. Для малих проектів використання API є більш вигідним. Ще одним способом використання є безпосереднє використання генеративної моделі на пристроях користувачів, яке

інтегроване в додаток. Проте, даний спосіб стикається з проблемою обмеженої продуктивності кінцевих користувачів.

Усі сервіси мають різну цінову політику. Одні виставляють ціну генерації зображення за певну кількість токенів чи кредитив, а у інших фіксована ціна за кожне зображення. Ціна генерації зображення за допомогою моделей DALL-E 3 рахується у доларах за готове зображення, а для моделі Leonardo Lightning XL – ціна у токенах, відповідно до використаних функцій генерації. Для Stable Diffusion 3.5 Large у сервісі Segmind ціна формується на основі затраченого часу під час генерації зображення. Для OpenAI для користування, просто потрібно поповнити баланс грошима, а у Leonardo.AI та Segmind потрібно купувати один із планів підписки. Перевагою Segmind над конкурентами є те, що даних сервіс надає базовий баланс та доступ для користування їхнім API.

Також всі сервіси надають доступ для комерційного використання згенерованих зображень, однак забороняють їхнє використання у цілях, які можуть зашкодити іншим людям. Йдеться про шахрайство, спам, викривлення інформації, шокуючий контент та інше. Вся юридична відповідальність лягає на користувачів, які використовують дані зображення.

Провівши підрахунки тарифів та цін, в середньому DALL-E 3 дає змогу згенерувати 8 – 25 зображень на 1$, в залежності від якості зображення. Модель Stable Diffusion 3.5 Large дає змогу згенерувати приблизно 15 зображень на 1$, Leonardo Lightning XL із параметрами, з якими відбувалась оцінка якості зображень у попередньому підрозділі – 129 зображень на 1$. В даному плані найкращим варіантом є використання сервісу Leonardo.AI та моделі Leonardo Lightning XL. Але також варто врахувати, що розрахунки для Leonardo Lightning XL можуть бути значно вищими в залежності від використання додаткових функцій. Також розглянув сторонній сервіс, який надає доступ до моделі Stable Diffusion 3.5 Large. Згідно із заявленими характеристиками сервісу та за власними підрахунками, зображення генерується приблизно від 18 до 60 секунд. Цінова політика сервісу – це ціна за секунду роботи, 0.001$/секунду. Провівши додаткові розрахунки, виходить в межах

від 16 до 55 зображень на 1$. Тобто, вигідніше використовувати сторонні сервіси для моделі Stable Diffusion 3.5 Large.

Таблиця 2.2.

Порівняння генеративних моделей для інтеграції в сторонні системи

Характеристика	**DALL-E 3**	**Stable Diffusion 3.5 Large**	**Leonardo Lightning XL**
Архітектура моделі	трансформатор та дифузійна модель	дифузійна модель	--
Наявність додаткових налаштувань	мінімальна кількість налаштувань	достатня кількість налаштувань	розширена кількість налаштувань
Спосіб інтеграції в інші системи	API	API та локальне встановлення	API
Сервіси, які надають до моделей	OpenAI	Stability AI та інші сторонні сервіси	Leonardo.AI
Цінова політика для інтеграції за допомогою API	від 0.040 до 0.120 дол./зобр.	покупка кредитів, 1 кред. = 0.01$ (ціна на сайті Stability AI), 6.5 кред./зобр.	тарифні плани, від 9 до 299 дол./міс.
Підтримка різних мов вводу підказки	Так	Ні	Ні
Вільний доступ на користування зображеннями	Так	Так	Так
Приватність згенерованих зображень	Так	Так	Так

Щодо локалізації, то тільки DALL-E 3 підтримує українську мову, це досягається завдяки тісній інтеграції у систему ChatGPT, якає підтримує велику кількість мов. Моделі Stable Diffusion 3.5 Large та Leonardo Lightning XL не підтримуються українську мову.

Використання генеративних моделей штучного інтелекту може значно покращити продуктивність певних професійних та не тільки груп користувачів, які під час роботи обробляють велику кількість зображень. Також застосування генеративних мереж може добре інтегруватися в освітні процеси, дозволяючи студентам на основі згенерованих концепції розробляти проекти, у такий спосіб полегшуючи процес навчання [39-40].

Отже можна сказати, що DALL-E 3 – це модель, яка є дуже простим та зручним інструментом для звичайних користувачів. Він не вражає великою кількістю параметрів та реалістичністю зображень, однак досить простий у користуванні. Модель Stable Diffusion 3.5 Large орієнтована на використання розробниками та дослідниками. Користувачам доступна можливість встановлювати дану модель на локальний сервер та експериментувати з моделлю. Говорячи про Leonardo Lightning XL можна сказати, що дана модель чудово підійде художникам, архітекторам, дизайнерам чи іншим спеціалістам, які працюють із стилізованими зображеннями та потребують гнучких налаштувань стилів при генерації зображень.

За результатами порівняння моделей у таблиці 2.2. можна сказати, що оптимальною моделлю для розробок та інтеграції у сторонні сервіси є модель Stable Diffusion 3.5 Large завдяки гнучкій ціновій політиці та можливості встановлювати на локальний сервер. Для використання Stable Diffusion 3.5 Large по API краще використовувати сторонні сервіси, які надають більш помірну цінову політику, або скористатися іншими генеративними моделями, як-от Leonardo Lightning XL. Leonardo.AI дає чудову цінову політику для використання їхнього API, що робить використання даного сервісу чудовими рішення для інтеграції генеративних моделей саме через використання API. Інтеграції DALL-E 3 є найпростішою, адже компанія

OpenAI надає мінімальну кількість налаштувань, що спрощує інтеграцію даної генеративної мережі.

2.5. Висновки до розділу

В цьому розділі детально описано три сучасні моделі генеративного штучного інтелекту – DALL-E 3, Stable Diffusion 3.5 Large та Leonardo Lightning XL. Розглянуто їх функціональні можливості, переваги та недоліки. Також розглянуто сервісні платформи на яких розміщені дані моделі.

Окремо було розглянуто про конструювання підказок та їх вплив на результат генерації зображень. Відповідно було описано про загрози пов’язані із підказковою ін’єкцією. Крім того, проведено аналіз якості генерації зображень для кожної із трьох генеративних моделей та порівняно їхні особливості при інтеграції в сторонні сервіси. За результатами аналізу Stable Diffusion 3.5 Large є оптимальним варіантом для розробок та досліджень, адже є продуктом з відкритим кодом. Компанія надає можливість використання API та дозволяє розміщати дану генеративну модель на локальному сховищі.

За допомогою цих результатів можна обрати оптимальну генеративну модель штучного інтелекту відповідно до вимог розроблювальної системи та потреб користування.

РОЗДІЛ 3
3. РОЗРОБКА АЛГОРИТМУ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ НА ОСНОВІ ТЕКСТОВОЇ ПІДКАЗКИ

В даному розділі буде проведе процес розробки програмного забезпечення включаючи інтеграцію генеративної моделі штучного інтелекту для генерації зображень. Під час розробки буде продемонстровано та пояснено ключові фрагменти коду. Також буде розроблено алгоритм роботи системи та її архітектуру.

Це практичний розділ, метою якого є узагальнити всі знання обговорені у попередніх розділах та використати їх для реалізації системи генерації зображень.

3.1. Опис проблеми інтеграції генеративних моделей

Із появою штучного інтелекту, все більше компаній стараються використовувати його у своїх програмних продукти та сервісах. Штучний інтелект є трендом, який є не просто технологією заради технології, а значно полегшує взаємодію користувачів із системою, оптимізуючи процеси рядового користувача.

Однією з проблем є складність розробки власної моделі ШІ. Це дуже тривалий та високовартісний процес. Водночас, виникають різні проблеми, які необхідно вирішити при використанні різного роду генеративних моделей в програмному забезпеченні. Однією з них є нездатність користувацьких систем справлятися із великими генеративними моделями. Більшість сучасних генеративних моделей потребують значних ресурсів системи, що унеможливлює їхню пряму інтеграцію у комп’ютерні чи мобільні додатки. Однак компанії, які розробляють генеративні моделі надають широкі можливості їх застосування за допомогою API. Використання API вирішує проблему складних обчислень на користувацьких пристроях. Всі обчислення відбуваються за допомогою хмарних сервісів. Також можна забути про витрати на розробку власної моделі. Сучасні сервіси надають безліч тарифних планів,

які підійдуть як дрібним компаніям, так і великим гравцям на ринку.

Ще однією з можливих проблем, які необхідно вирішити – це максимальна інтуїтивність та простота інтерфейсу користувача. Багато сучасних сервісів, які надають послуги генерації зображень, викладають велику кількість налаштувань на звичайного користувача. Тому рішенням може бути, виставлення стандартних чи оптимальних параметрів вибраної генеративної мережі та приховування всіх складних та не потрібних звичайному користувачу функціональних можливостей.

3.2. Вимоги до програмного рішення розроблюваної системи

Для демонстрації інтеграції генеративних моделей буде розроблено систему для генерації зображень для робочого столу. Дана система повинна надавати користувачу вводити текстову підказку у відповідному полі. Оскільки процес генерації зображення може займати певний час, то повинно бути реалізовано відображення того, що в даний момент відбувається процес генерації зображення і користувачу слід зачекати. В цей час користувач не має мати можливості повторно натиснути кнопку генерації, допоки не закінчиться попередній процес. Оскільки це застосунок для генерації зображень для робочого столу, то має й бути можливість встановлення згенеровано зображення як фону робочого столу. Також не менш важливим аспектом, який впливає на зручність роботи із застосунком є відображення вже згенерованого зображення у певній області, яка для цього виділена. Всі згенеровані зображення повинні зберігатися в певній директорії проекту. Крім того, користувач повинен мати можливість вибрати вже існуюче зображення чи попередньо згенероване. До того ж варто зазначити, що інтерфейс користувача моє бути прости та інтуїтивно зрозумілим. Не повинні використовуватися дуже яскраві та токсичні кольори. Елементи інтерфейсу повинні бути із підписами, що відповідають їхньому функціоналу. Також система повинна обробляти помилки, які виникають під час генерації зображення.

Основною вимогою до розроблюваного ПЗ є те, що система повинна на основі

текстової підказки генерувати якісні зображення. Для досягнення високого рівня якості та швидкості генерації має бути застосовано хмарний сервіс генерації зображень. Цим сервісом може виступити Segmind, тому що він пропонує вигідне рішення щодо використання власного API для інтеграції у ПЗ. За допомогою Segmind можна легко інтегрувати відомі генеративні моделі штучного інтелекту у систему любої складності. Відповідно до порівнянь у другому розділі, оптимальним варіантом для досліджень та розробок є модель Stable Diffusion 3.5 Large, яка дозволяє генерувати зображення хорошої якості. Segmind надає доступ до даної моделі, що дасть нам можливість інтегрувати її у систему генерації зображень. Також варто відмітити, що згенеровані зображення за допомогою даного сервісу та моделі можна використовувати як у побутових, так і в комерційних цілях.

На сьогоднішній день використання хмарних обчислень є дуже популярним та вигідним, тому архітектура даної системи генерації буде являти собою взаємодію клієнтської частини та хмарного сервісу Segmind. В результаті має бути розроблено тонкий клієнт, який являє собою простий клієнтський додаток, що виконує генерацію зображення за допомогою ресурсів хмарного рішення. По суті, клієнтська частина є свого роду візуальним представлення функціоналу генерації зображень, який доступний за допомогою API із хмари. В додатку реалізовується функціонал для конкретного сценарію використання генерації, в даному випадку – це генерація зображень для робочого столу. Оскільки всі обчислення відбуваються у хмарі, то клієнтська частина займає мінімальну кількість пам’яті і не потребує значних ресурсів для роботи.

Серед недоліків даного підходу є те, що потрібне постій з’єднання з інтернетом. Хоча проблема доступності інтернету у 21-му столітті не має виникати. Також проблемою можуть бути технічні неполадки у хмарному сервісі, що може призвести до загальних збоїв у роботі всіх користувачів [41]. Однак, дана проблема виникає дуже зрідка. До переваг, як вже зазначалося, можна віднести перенесення обчислення на хмару, що позитивно впливає на зменшення вимог клієнтської частини. Плюсом також є простота інтеграції хмарного сервісу за допомогою API, що

позволяє інтегруватися практично у любий додаток та любий пристрій, який має підключення до інтернету.

3.3. Розробка алгоритму роботи системи

3.3.1. Сценарії використання системи генерації зображення

Роботу системи можна поділити на два сценарії роботи, де основним буде – генерація зображення для робочого столу, а другорядним – використання програми для вибору зображення із директорії та встановлення як зображення робочого столу. Use Case діаграму взаємодії користувача із функціоналом системи продемонстровано на рисунку 3.1. Також на даному рисунку зображено, як функції системи взаємодіють між собою.

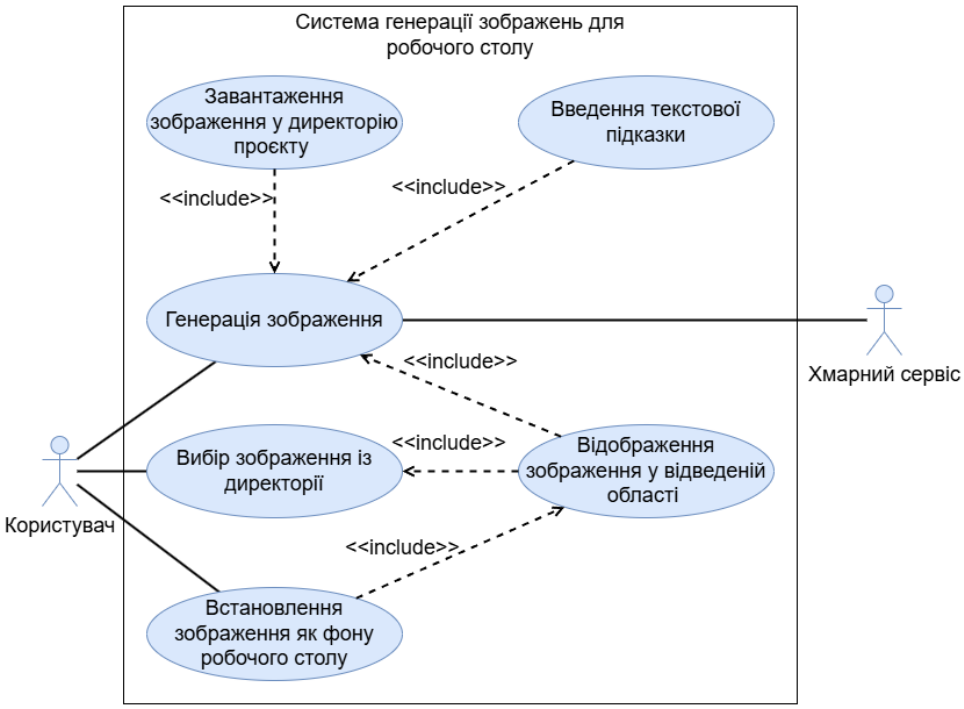

Рис. 3.1. Use Case діаграма взаємодії користувача із системою

Користувач має три функції, які може використати – це згенерувати зображення, встановити зображення як фон робочого столу та має можливість вибрати зображення із директорії. При генерації зображення користувач обов’язково

повинен вказати текстову підказку, а після генерації зображення збережеться в директорію проєкту та відобразиться у відведеній області, після чого користувач може встановити дане зображення як шпалери робочого столу комп’ютера. Також користувач має можливість простого вибору зображення із каталогу та може встановити вибране зображення як фон робочого столу.

3.3.2. Алгоритми роботи системи під час генерації зображення

Після зображення роботи системи у вигляді Use Case діаграми, можна перейти до покрокового опису роботи системи під час генерації зображення. Початок роботи користувача починається із відкриття програмного додатку, де він серед доступного функціоналу може вибрати ту чи іншу функцію, в залежності від поставленої мети.

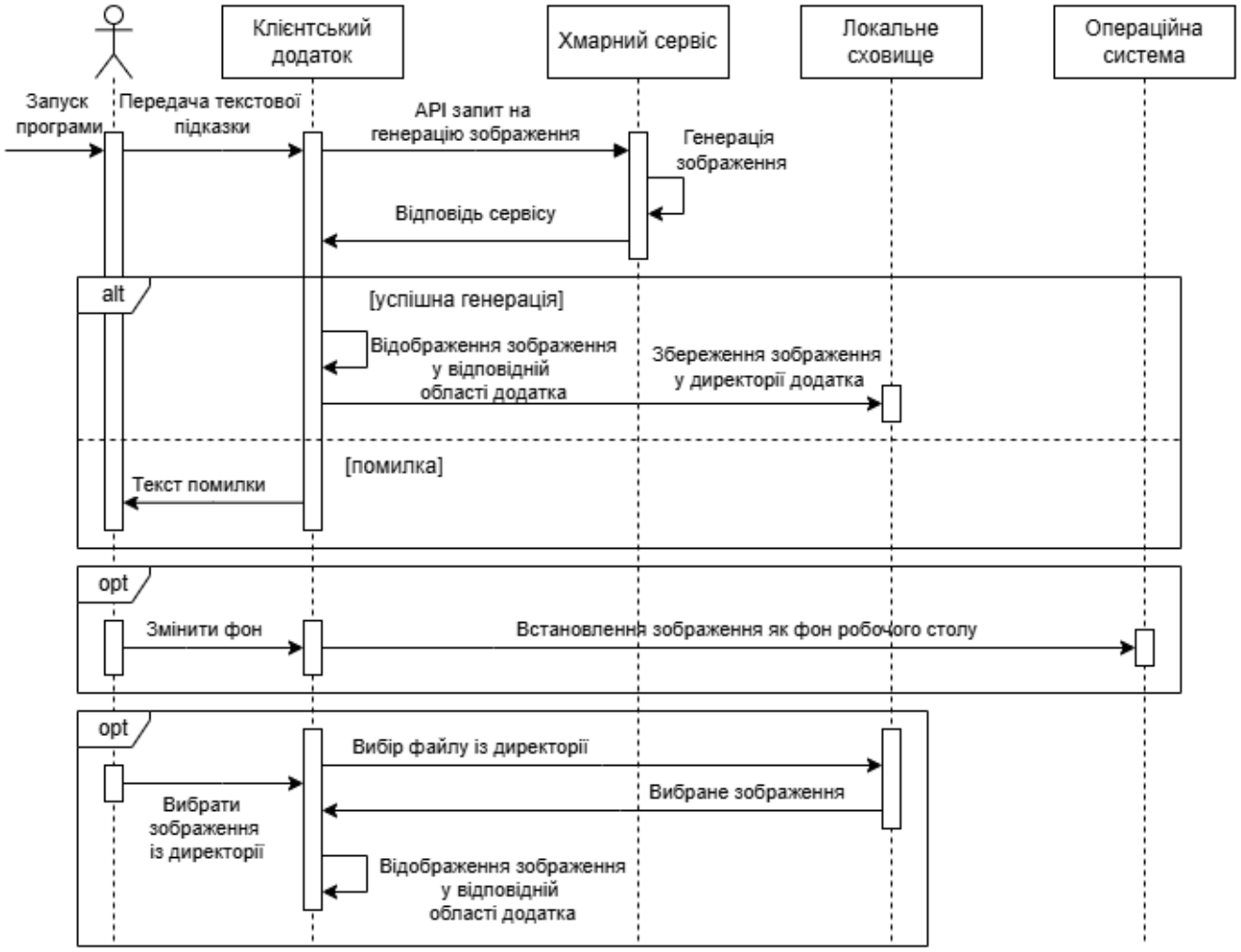

Рис. 3.2. Діаграма послідовності системи генерації зображень

На рисунку 3.2. зображено покроковий процес роботи програми враховуючи кроки які відбуваються на стороні клієнтського додатка, хмарного сервісу, локального сховища та операційної системи. Відкривши програмний застосунок користувач може почати процес генерації зображень. На даній діаграмі зображено декілька послідовностей дій.

Ось приклад першої послідовності дій:

1. Користувач вводить текстову підказку і натискає кнопку генерації зображення.
2. З клієнтської частини відбувається запит до хмарного сервісу за допомогою API.
3. У хмарному сервісі відбувається процес генерації зображення відповідно до вказаної текстової підказки.
4. Після закінчення генерації на клієнтську частину повертається відповідь із хмарного сервісу.
5. На клієнтській частині перевіряється чи відповідь є успішною, тобто чи успішно відбулася генерація зображення. Якщо виникла помилка, то виводиться відповідне повідомлення. Якщо ж все добре, то зображення зберігається у директорії додатка та виводиться у відповідну область для відображення користувачу.

Наступна послідовність відповідає за зміну фону робочого столу. Користувач натискає змінити фон, після чого, відбувається заміна шпалер робочого столу. Дана послідовність не є строго обов’язковою, це означає, що користувач може декілька разів поспіль генерувати зображення, доки не досягне бажаного результату. Після цього він зможе використати згенероване зображення та зробити його фоном робочого столу.

Ще однією необов’язковою послідовністю дій є вибір зображення із директорії. В цій послідовності відбувається вибір зображення із локального сховища. Дана послідовність демонструє додатковий функціонал системи, щоб не прив’язуватись тільки до згенерованих зображень.

3.4. Програмна реалізація системи генерації зображення

3.4.1. Реєстрація на та отримання ключа доступу для API

Щоб мати можливість генерувати зображення, необхідно зареєструватися на сервісі Segmind та отримати API ключ, який дасть доступ до використання API у розроблюваній системі генерації зображень для робочого столу (рис. 3.3.).

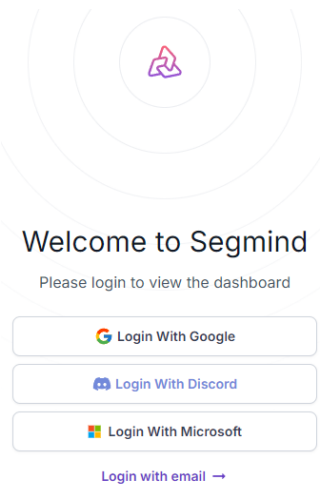

Рис. 3.3. Отримання доступу до сервісу Segmind

Наступний крок це безпосередньо сторінка з ключами, де можна створювати та отримувати нові ключі доступу (рис. 3.4.).

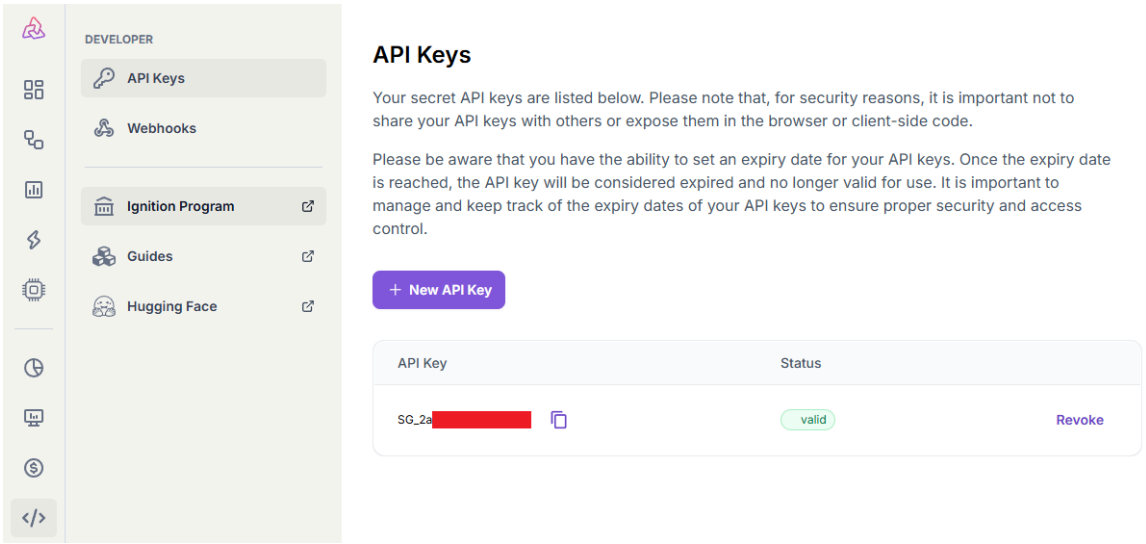

Рис. 3.4. Сторінка API Keys

Після отримання ключа доступу можна перейти на сторінку генеративної моделі Stable Diffusion 3.5 Large та отримати необхідну інформацію для впровадження даної моделі у систему генерації зображень (рис. 3.5.).

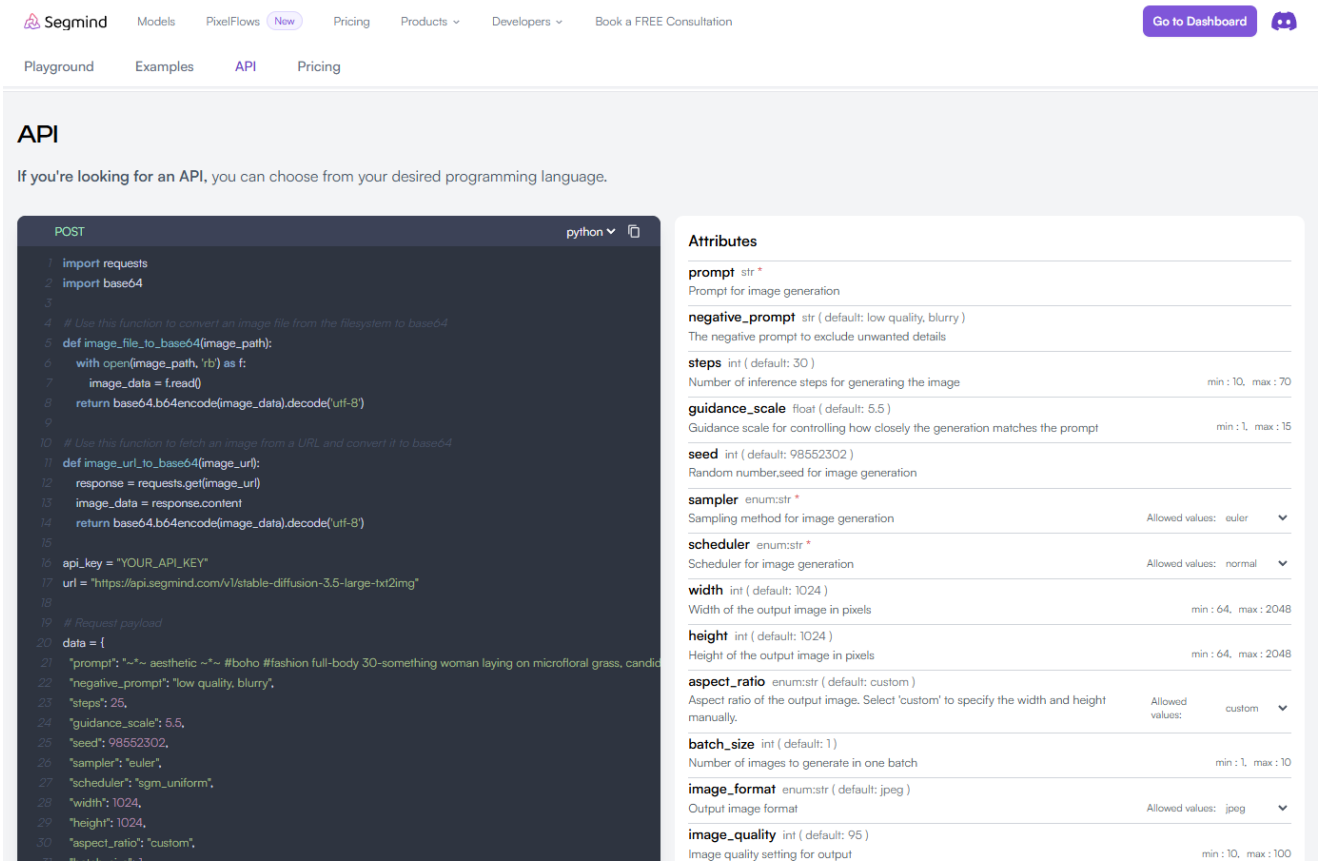

Рис. 3.5. Приклад коду для інтеграції Stable Diffusion 3.5 Large

Приклади доступні для Python, Node.js та Curl. Хоч приклади тільки три, проте, звернення до сервісу виконуються у вигляді HTTP запитів з тілом у JSON форматі. Це дозволяє використовувати даний сервіс за допомогою будь-якої мови програмування.

3.4.2. Розробка системи

Для розробки системи було вибрано мову програмування C#. Дана мова є чудовим інструментом, який пропонує високу швидкість роботи та приємні особливості коду. Також причиною вибору даної мови є те, що вона від компанії

Microsoft та чудово інтегрована для розробки під сімейство операційних систем Windows. Додаток буде використовувати Windows Forms інтерфейс, який є досить легким при розробці та підходить для проєктів малої та середньої складності.

Для цього комплекту чудово підходить така IDE система, як Microsoft Visual Studio Community 2022 (рис. 3.6.). За допомогою цього інтегрованого середовища й буде відбуватися розробка системи. Це середовище є досить зручним та пропонує широкий спектр доступних функцій для розробників.

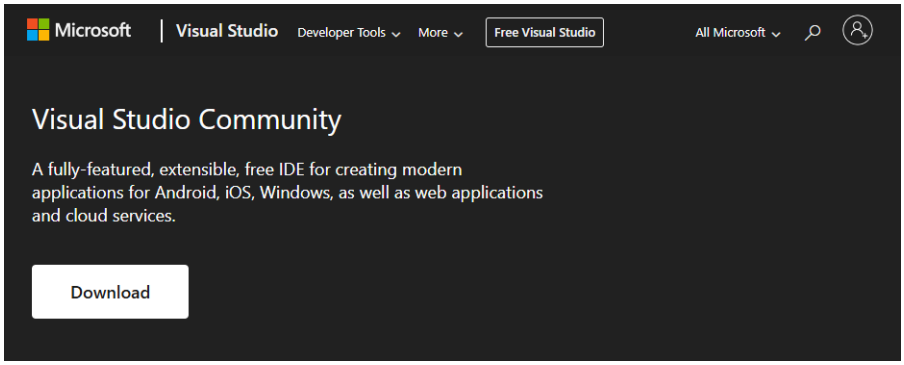

Рис. 3.6. IDE Microsoft Visual Studio Community 2022

Під час установки даної IDE, встановлюються всі додаткові компоненти необхідні для роботи мови програмування C#, при умові, що при установці було виставлено галочку установки необхідних пакетів для .NET desktop development. Наступним кроком є створення проєкту та налаштування робочої області. Після цього можна приступати до безпосередньої розробки проєкту. Для початку продемонструю всі встановлені NuGet пакети (рис. 3.7.).

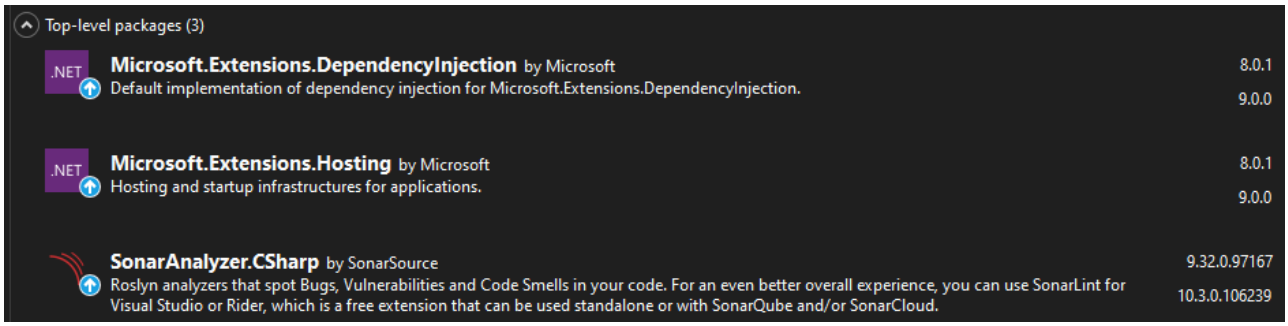

Рис. 3.7. Використані NuGet пакети

Дані пакети призначені для налаштування конфігурації, впровадження залежностей та аналізу коду.

Для початку розглянемо файл Program.cs. Це головний файл програми, з якого відбувається ініціювання всіх компонентів системи. Код даного класу наведено у лістингу 3.1.

Лістинг 3.1.

```
using AIWallpaperApp.Services;
using Microsoft.Extensions.DependencyInjection;

namespace AIWallpaperApp;

internal static class Program
{
    public static IServiceProvider? ServiceProvider { get;
private set; }

    [STAThread]
    static void Main()
    {
        var host =
HostBuilderService.CreateHostBuilder().Build();
        ServiceProvider = host.Services;

        Application.EnableVisualStyles();

Application.Run(ServiceProvider.GetRequiredService<Form1>());
    }
}
```

Далі необхідно створити файл appsettings.json, у якому будуть зберігатися ключ доступу до сервісу та посилання на модель Stable Diffusion 3.5 Large (лістинг 3.2.). Поле “ApiKey” не містить справжнього ключа через те, що він зберігається у окремому файлі secrets.json. Це зроблено для того, щоб при копіюванні проєкту чи використанні Git системи, не відбувалася публікація ключа. Все це зроблено з міркувань безпеки, щоб унеможливити витік ключів доступу у публічний простір. Зазвичай для цього використовуються окремі сервісні рішення, які є значно зручнішими.

Лістинг 3.2.

```
{
  "Segmind": {
    "ApiKey": "Secret-API-key",
    "Url": "https://api.segmind.com/v1/stable-diffusion-3.5-
large-txt2img"
  }
}
```

У лістингу 3.3. продемонстровано клас, який є моделлю для зчитування даних із файлу appsettings.json.

Лістинг 3.3.

```
namespace AIWallpaperApp.Models;

public class AppSettings
{
    public string ApiKey { get; set; } = string.Empty;
    public string Url {  get; set; } = string.Empty;
}
```

Наступним важливим класом є HostBuilderService.cs (лістинг 3.4.). Даний клас призначений для додавання залежностей сервісів та зчитування конфігурації з файлу appsettings.json. Як можна побачити, в цьому класі підключений сервісі для роботи з зображеннями.

Лістинг 3.4.

```
using AIWallpaperApp.Models;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System.Reflection;

namespace AIWallpaperApp.Services;

public static class HostBuilderService
{
    public static IHostBuilder CreateHostBuilder()
    {
        var config = new
ConfigurationBuilder().SetBasePath(AppContext.BaseDirectory)
            .AddJsonFile("appsettings.json", false, true)
```

```
                    .AddUserSecrets(Assembly.GetExecutingAssembly(),
true)
                    .AddEnvironmentVariables()
                    .Build();

                var hostBuilder = Host.CreateDefaultBuilder()
                    .ConfigureServices((context, services) =>
                    {

services.Configure<AppSettings>(config.GetSection("Segmind"));
                        services.AddTransient<IImageService,
ImageService>();
                        services.AddSingleton<Form1>();
                    });

                return hostBuilder;
            }
        }
```

Наступний код демонструє інтерфейс IImageService, який буде реалізовано в класі ImageService (лістинг 3.5.).

Лістинг 3.5.

```
namespace AIWallpaperApp.Services;


public interface IImageService
{
    Task<HttpResponseMessage> GenerateImageAsync(string prompt);
    int SetWallpaper(string imagePath);
}
```

У лістингу 3.6. продемонстровано класі ImageService, що реалізує інтерфейс з лістингу 3.5. У цьому класі є два методи, які є важливими для роботи системи. Метод GenerateImageAsync призначений для генерації зображення за допомогою стороннього API і приймає аргументом текстову підказку. Змінна data представляє собою json файл, який буде передано в тіло запиту. Дана змінна являє собою набір налаштувань моделі Stable Diffusion 3.5 Large. Всі ці параметри було розглянуто в попередньому розділі, під час аналізу якості зображень декількох генеративних моделей. Також було додано генератор натуральних випадкових чисел для

урізноманітнення генерації, тобто, щоб при повторному застосуванні підказки змінювався результат генерації. Після формування тіла запиту відбувається безпосереднє надсилання запиту на сервер. При цьому в заголовках вказується ключ доступу API та додаються параметри для генерації. Без ключа, ми отримаємо повідомлення, що користувач не авторизований. Даний метод повертає тип даних HttpResponseMessage.

Лістинг 3.6.

```
        public async Task<HttpResponseMessage>
GenerateImageAsync(string prompt)
        {
            var random = new Random();
            var randomSeed = random.Next(0, int.MaxValue);

            var data = new
            {
                prompt = prompt!,
                negative_prompt = "low quality, blurry",
                steps = 25,
                guidance_scale = 5.5,
                seed = randomSeed,
                sampler = "euler",
                scheduler = "sgm_uniform",
                width = 1792,
                height = 1024,
                aspect_ratio = "custom",
                batch_size = 1,
                image_format = "jpeg",
                image_quality = 95,
                base64 = false
            };

            using (HttpClient httpClient = new HttpClient())
            {
                var request = new
HttpRequestMessage(HttpMethod.Post, new Uri(_settings.Url));
                request.Content = JsonContent.Create(data);
                request.Headers.Add("x-api-key", _settings.ApiKey);

                return await httpClient.SendAsync(request);
            }

        }
```

Другий метод SetWallpaper призначений для встановлення зображення як фону робочого столу для вже згенерованого чи вибраного зображення (лістинг 3.7.). Він приймає шлях до зображення як аргумент. Три константні змінні в методі означають зміну шпалер системи. Імпортована функція API SystemParametersInfo призначена для зміни шпалер робочого столу за допомогою шляху до зображення та параметрів зміни шпалер.

Лістинг 3.7.

```
        public int SetWallpaper(string imagePath)
        {
             const int SPI_SETDESKWALLPAPER = 20;
             const int SPIF_UPDATEINFILE = 1;
             const int SPIF_SENDCHANGE = 2;

            [DllImport("user32.dll", CharSet = CharSet.Auto,
SetLastError = true)]
            static extern int SystemParametersInfo(int uAction, int
uParam, string lpvParam, int fuWinIni);

            return SystemParametersInfo(SPI_SETDESKWALLPAPER, 0,
imagePath, SPIF_UPDATEINFILE | SPIF_SENDCHANGE);
        }
    }
```

Останнім важливим класом є Form1. В ньому відбувається з’єднання функціоналу системи із її візуальним представленням (лістинг 3.8.). Метод btnGenerate_Click при натисканні запускає процес генерування зображення. При цьому в даному методі перевіряється чи поле підкази є порожнім. Якщо поле порожнє, то кнопка генерації неактивна, крім того, кори запущено процес генерації, користувач не може ще раз натиснути на кнопку згенерувати. Він повинен дочекатися завершення процесу генерації зображення. В цьому методі реалізовано перевірку на успішність відповіді. При будь-якій помилці під час запиту до API, ми отримаємо повідомлення із зазначенням помилки. Також в даному методі відбувається видобування зображення із відповіді API сервісу. В результаті ми побачимо зображення у виділеній для цього області. При цьому варто зазначити, що всі згенеровані зображення зберігаються у відповідній директорії проєкту та можуть бути повторно використані.

Лістинг 3.8.

```
        private async void btnGenerate_Click(object sender,
EventArgs e)
        {
            SetUIStateAfter_btnGenerate_Click();

            var prompt = textBoxPrompt.Text;
            var response = await
_imageService.GenerateImageAsync(prompt);

            RestoreUIStateAfter_btnGenerate_Click();

            if (response.IsSuccessStatusCode)
            {
                using (var ms = new MemoryStream(await
response.Content.ReadAsByteArrayAsync()))
                using (var image = Image.FromStream(ms))
                {
                    var imageName = @"img_" + Guid.NewGuid() +
".jpg";
                    var directory =
Directory.GetParent(Environment.CurrentDirectory)?.Parent?.Parent?.Ful
lName;
                    var moveToDirectory = directory + "\\Images\\" +
imageName;

                    image.Save(imageName, ImageFormat.Jpeg);
                    var imageFileInfo = new FileInfo(imageName);
                    imageFileInfo.MoveTo(moveToDirectory);

                    picBox.ImageLocation = moveToDirectory;
                    lblImageTip.Visible = false;
                }
            }
            else
            {
                MessageBox.Show(response.StatusCode.ToString());
            }
        }
```

У лістингу 3.9. наведені методи, які призначені для перевірки підказкового поля на наявність символі, встановлення анімації для панелі процесу. Також наявний метод під назвою btnSetWallpaper_Click, який призначений для встановлення згенерованого чи вибраного зображення як шпалери робочого столу. Даний метод приймає шлях зображення із області згенерованого чи вибраного зображення.

Лістинг 3.9.

```
        private void btnSetWallpaper_Click(object sender, EventArgs
e)
        {
            _imageService.SetWallpaper(picBox.ImageLocation);
        }

        private void textBoxPrompt_TextChanged(object sender,
EventArgs e)
        {
            btnGenerate.Enabled =
!string.IsNullOrWhiteSpace(textBoxPrompt.Text);
        }

        private void SetUIStateAfter_btnGenerate_Click()
        {
            btnGenerate.Enabled = false;
            prgImageGeneration.Visible = true;
            prgImageGeneration.MarqueeAnimationSpeed = 10;
        }

        private void RestoreUIStateAfter_btnGenerate_Click()
        {
            prgImageGeneration.MarqueeAnimationSpeed = 0;
            prgImageGeneration.Visible = false;
            btnGenerate.Enabled = true;
        }
```

Лістинг 3.10. демонструє метод btnSelecFile_Click, який дозволяє вибирати зображення із вибраної користувачем директорії. За промовчування вибрана директорія, де зберігаються всі згенеровані раніше зображення. Принцип роботи простий, при натисканні на кнопку вибору файлу, відкриється діалогове вікно де користувач може перейти до іншої папки або ж вибрати наявне зображення із цієї директорії. За допомогою фільтра відфільтровані формати файлів, які можна вибрати. В даному випадку – це популярні формати зображень.

Лістинг 3.10.

```
        private void btnSelecFile_Click(object sender, EventArgs e)
        {
            using (var fileDialog = new OpenFileDialog())
            {
                var directory =
Directory.GetParent(Environment.CurrentDirectory)?.Parent?.Parent?.Ful
lName;
```

```
                    fileDialog.InitialDirectory = directory +
"\\Images";
                    fileDialog.Filter = "Image files(*.jpeg; *.jpg;
*.png;)|*.jpeg; *.jpg; *.png;";

                    if (fileDialog.ShowDialog() == DialogResult.OK)
                    {
                        picBox.ImageLocation = fileDialog.FileName;
                    }
                }

                lblImageTip.Visible = false;
            }
        }
```

На рисунку 3.8. продемонстровано процес розробки, а саме побудова інтерфейсу користувача.

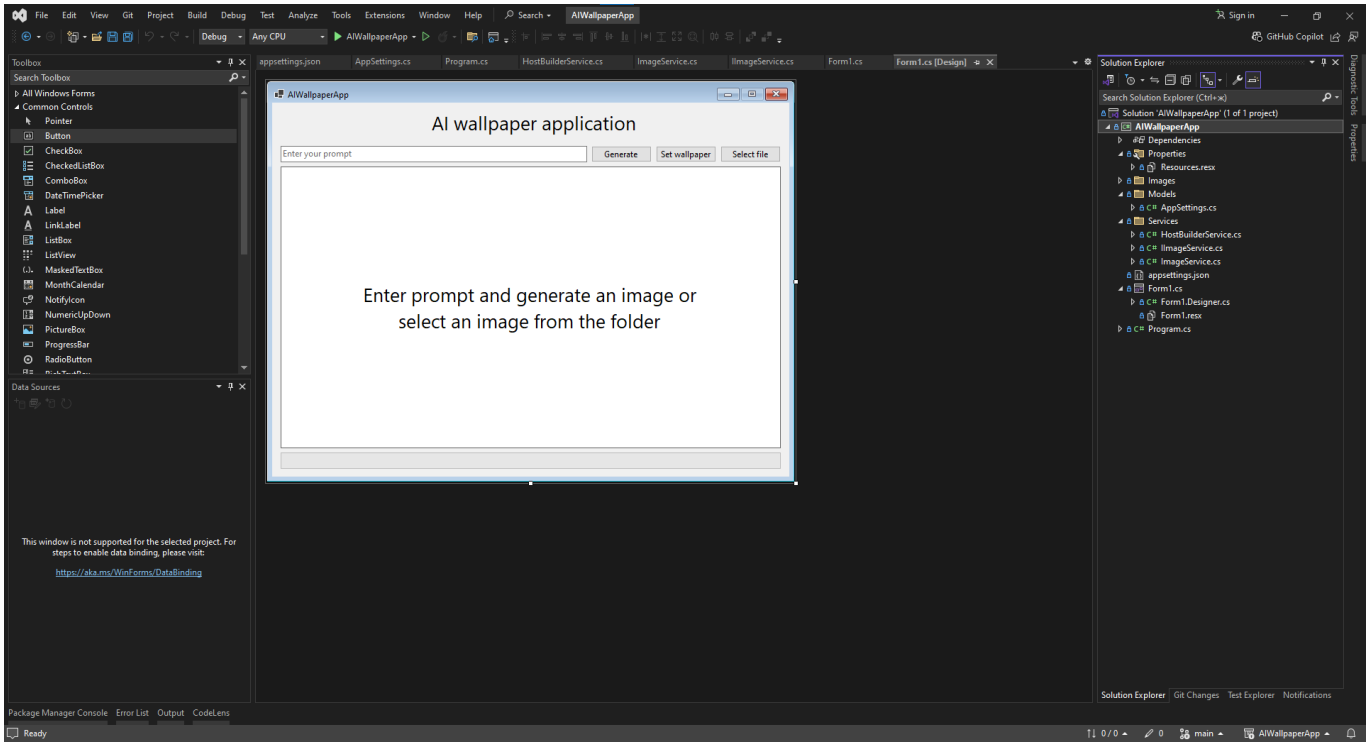

Рис. 3.8. Вигляд процесу розробки у середовищі Visual Studio Community 2022

Робота у даному IDE є досить простою у користуванні. За допомогою Windows Forms можна легко розробляти прості прототипи програм не вдаючись до використання складних архітектурних рішень.

3.5. Тестування розробленого програмного продукту

Після програмної реалізації системи генерації зображень, необхідно провести її тестування. Необхідно перевірити чи система належним чином виконує всі функції, включаючи генерацію зображень, встановлення згенеровано зображення як фон робочого столу та можливість вибору вже існуючого зображення із папки.

Для початку відкриваємо дану програму і побачимо її інтерфейс (рис. 3.9.). Ми можемо побачити чотири елементи, із якими користувач може активно взаємодіяти – це поле вводу підказки, кнопка генерації зображення, кнопка встановлення зображення як фону робочого столу та кнопка вибору вже існуючого зображення із папки на комп'ютері. Також можна побачити область для відображення згенерованого зображення.

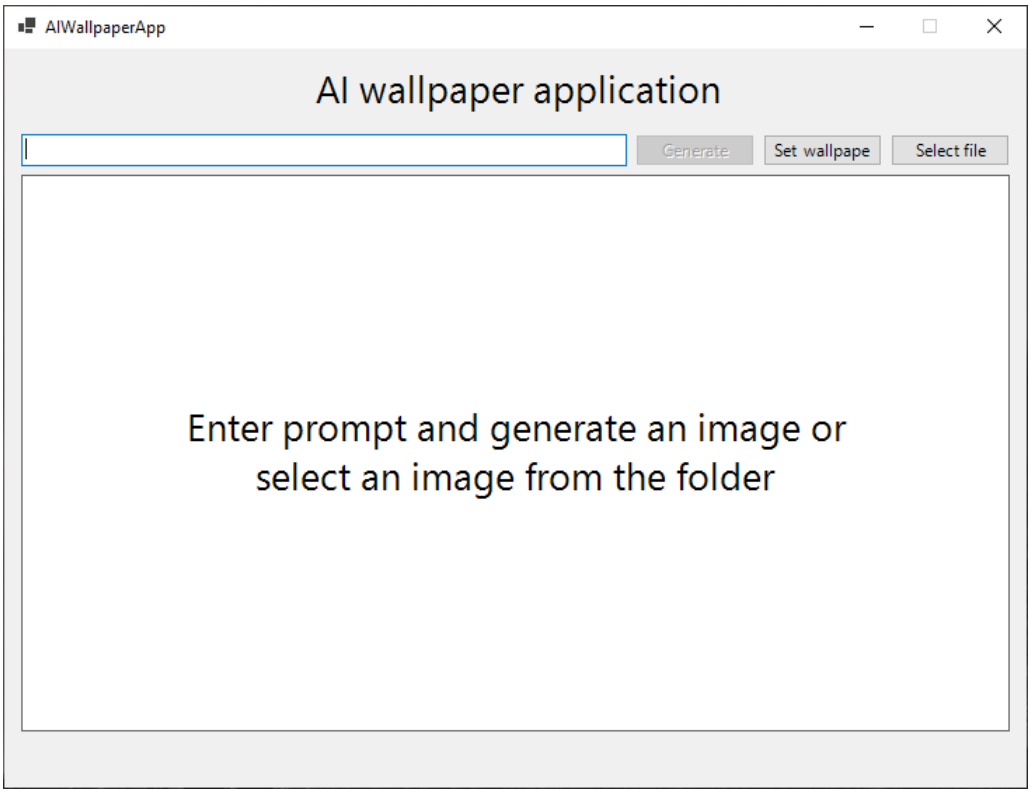

Рис. 3.9. Вигляд системи генерації зображень для робочого столу

Наступним кроком потрібно написати текстову підказку, враховуючи всі тонкості написання, які детально було описано у другому розділі. Для написання

підказки, використав підхід із використання другої генеративної мережі для генерації підказки. Для цього використав велику мовну модель ChatGPT. Підказку сформовано англійською мовою, адже Stable Diffusion 3.5 Large не підтримує українську мову. Підказка виглядає наступним чином: “A serene winter landscape with snow-covered pine trees, a frozen river winding through the scene, and distant mountains bathed in soft morning light. The ground is blanketed with fresh, untouched snow that sparkles under the pale winter sun. The sky is a crisp, clear blue with a few wispy clouds. The atmosphere is tranquil and cold, capturing the peaceful essence of winter. Photorealistic, high detail, vibrant yet natural colors, and smooth textures.”

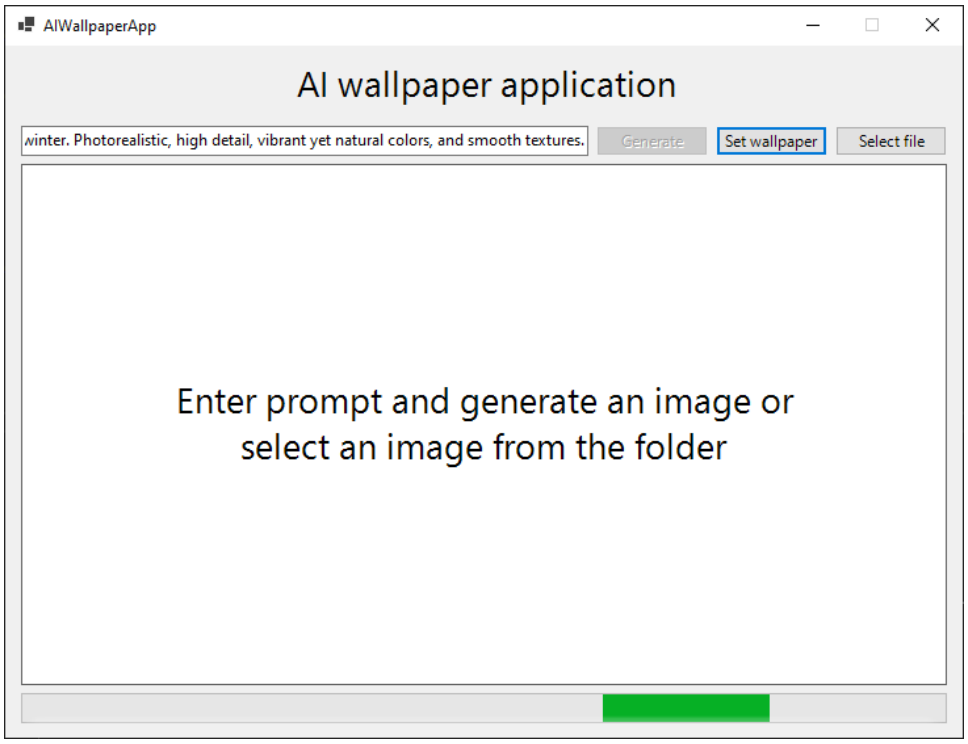

Рис. 3.10. Процес генерації зображення

Переклад даної підказки виглядає так: “Спокійний зимовий пейзаж із засніженими соснами, замерзлою річкою, що звивається крізь сцену, і далекими горами, осяяними м'яким ранковим світлом. Земля вкрита свіжим, незайманим снігом, який виблискує під блідим зимовим сонцем. Небо – чисте, прозоре, блакитне з кількома легкими хмаринками. Атмосфера спокійна і холодна, передаючи мирну сутність зими. Фотореалістичність, висока деталізація, яскраві, але природні кольори

та плавні текстури.” Після формування підказки вставляю її в поле для підказки та натискаю кнопку для генерації зображення. В результаті відбувається процес генерації, що помітно по анімації панелі процесу нижче області для зображення (рис. 3.10.). Процес генерації займає певний час, тому кнопка генерації буде неактивна. Після завершення генерації, користувач побачить згенероване зображення у відповідній області (рис. 3.11.).

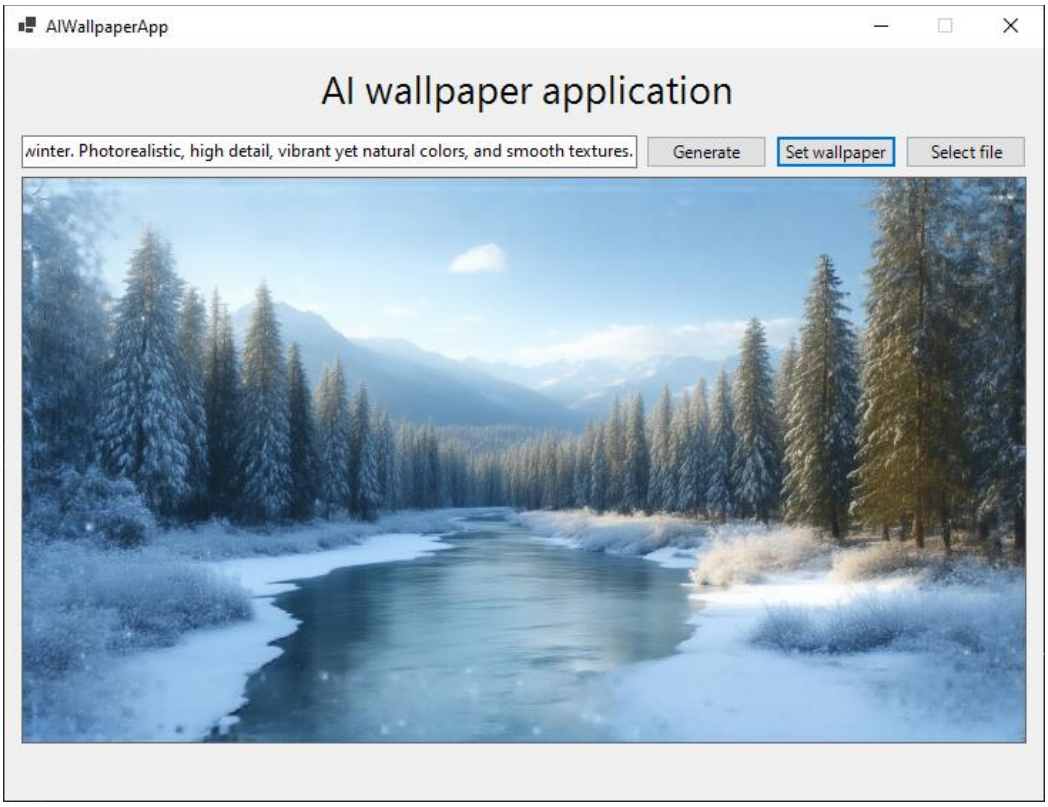

Рис. 3.11. Результат генерації зображення

Після того, як згенеровано зображення – можна скористатися функціоналом встановлення зображення як фону робочого столу. В результаті зображення робочого столу буде замінено на згенероване зображення (рис. 3.12.). Як видно із даного рисунку, згенероване зображення у програмі відповідає фону робочого стола. Відповідно до функціоналу, користувач також може вибрати вже існуюче зображення із довільної директорії, проте за промовчування відкривається директорія проекту, де зберігаються всі згенеровані зображення (рис. 3.13.). На рисунку 3.14. продемонстровано вікно відображення зображення, де після вибору зображення із

директорії з’явилося вибране зображення.

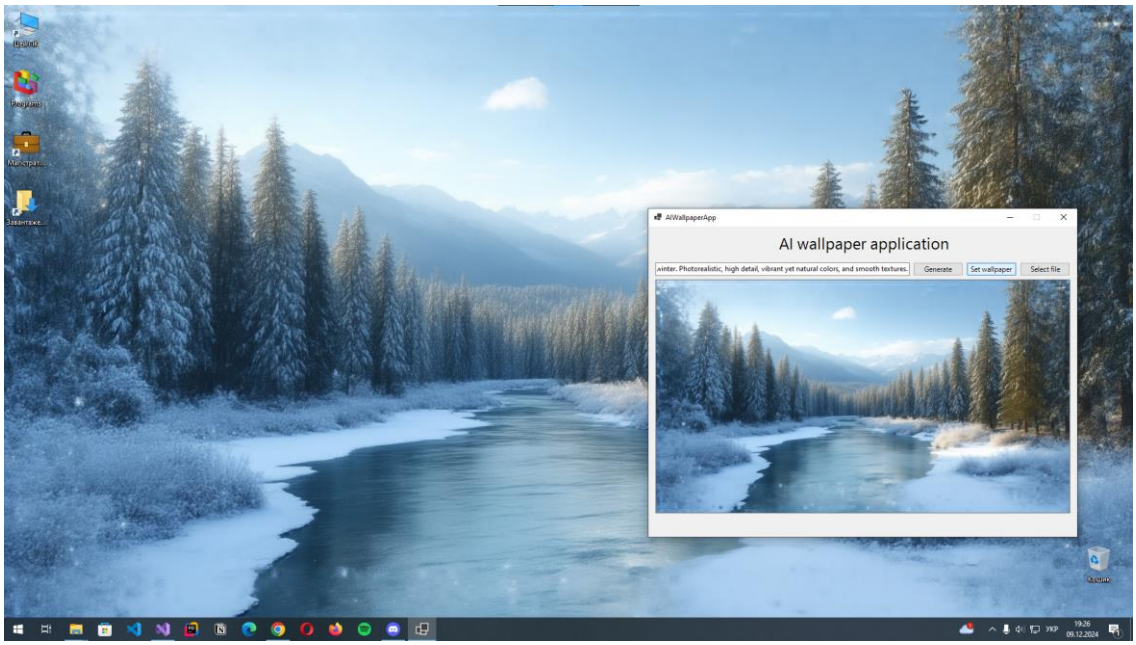

Рис. 3.12. Результат заміни фону робочого столу

На рисунку 3.13. можна побачити, що при використанні функції вибору зображення із директорії, відкривається базова директорія за промовчуванням, де зберігаються всі раніше згенеровані зображення.

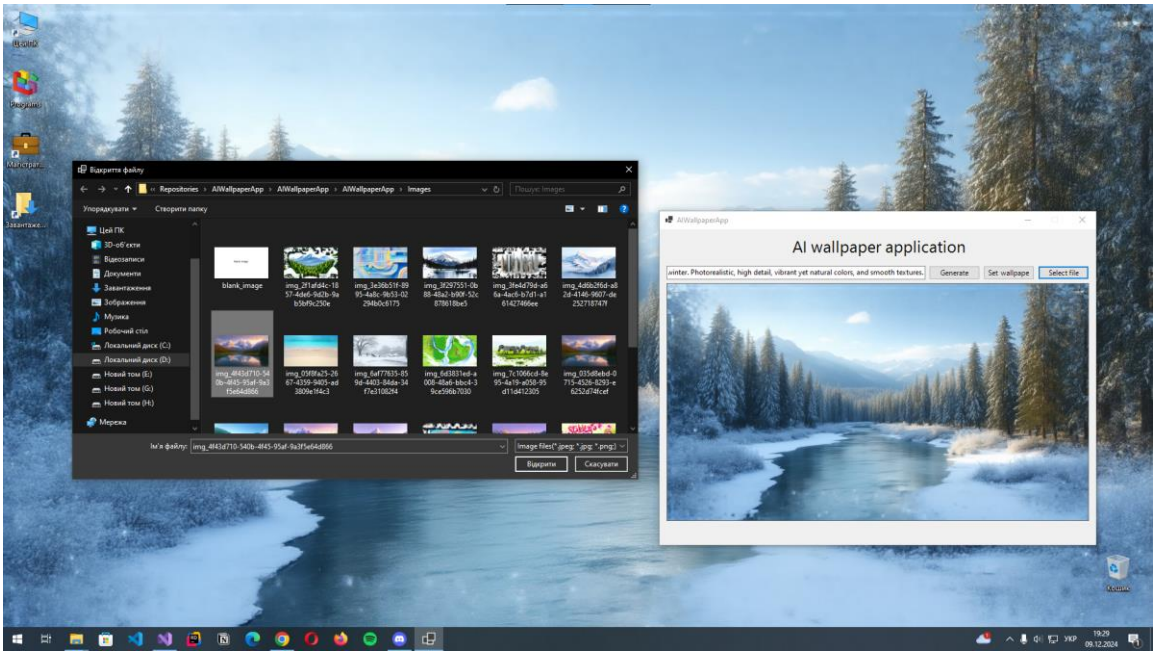

Рис. 3.13. Вибір зображення із директорії

Після вибору зображення із директорії, користувач може встановити дане зображення як фон робочого столу, або ж згенерувати чи вибрати інше зображення із директорії.

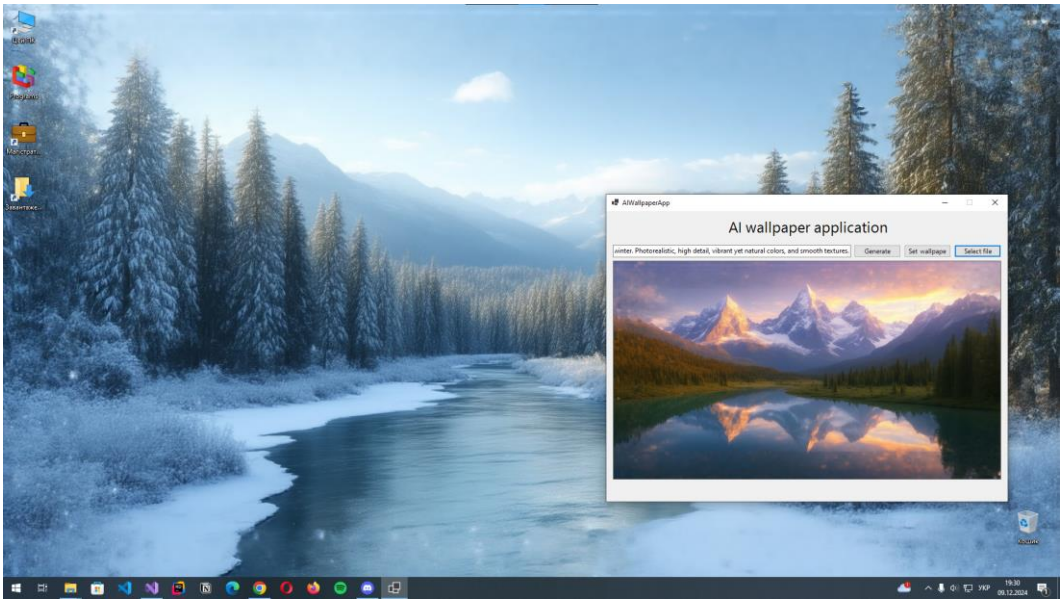

Рис. 3.14. Результат вибору зображення із директорії

В результаті вийшла чудова система, яка дозволяє користувачам легко змінювати фон робочого столу на зображення, яке можна створити за допомогою текстової підказки.

3.6. Висновки до розділу

В даному розділі було розглянуто проблему інтеграції генеративних мереж в програмні застосунки. Було визначено ключові вимоги для реалізації системи на основі моделей штучного інтелекту, розглянуто архітектуру розроблюваної системи та оцінено її переваги та недоліки. До того ж було докладно описано алгоритми та сценарії роботи системи для генерації зображень робочого столу.

Результатом даного розділу є програмна реалізація системи для генерації зображень робочого столу із інтеграцією генеративної моделі штучного інтелекту Stable Diffusion 3.5 Large.

ВИСНОВКИ

В даній магістерській роботі було розглянуто проблему інтеграції генеративних моделей штучного інтелекту у програмні застосунки, які призначені для генерації зображень. Було описано про штучний та генеративний інтелекти. Також було розглянуто поняття машинного та глибокого навчання. Крім цього, розглянуто основні типи машинного навчання та їхні особливості. Етичні аспекти також стали проблемо, яку було розглянуто у даній роботі, зокрема аспекти використання штучного інтелекту та як він впливає на суспільство.

На основі декількох типів генеративних моделей штучного інтелекту було розглянуто будову та принципи роботи генеративних мереж. Зокрема було розглянуто генеративні змагальні мережі, дифузійні моделі, автокодувальники та трансформери. Описав будову, переваги та недоліки кожного із цих типів моделей.

Для вирішення проблеми інтеграції генеративних моделей у різного роду системи було проведено аналіз вже існуючих сучасних рішень на ринку генеративних мереж. Серед відомих моделей розглянув DALL-E 3, Stable Diffusion 3.5 Large та Leonardo Lightning XL. Відповідно провів порівняння якості результатів генерації зображення. Дослідження проводилось для трьох різних текстових підказок, де перша – генерація портрету людини, друга – генерація зображення природного краєвиду та третє – сільська місцевість. За результатами цього дослідження можна сказати, що Leonardo Lightning XL випередив своїх конкурентів по якості згенерованих зображень. Stable Diffusion 3.5 Large та DALL-E 3 зайняли друге та третє місця відповідно.

Також було проведено порівняльний аналіз для вищезгаданих моделей та їхніх сервісів, де вони розглядаються в контексті доступності для інтеграції у сторонні сервіси. В результаті порівняння було визначено, що оптимальним рішення для інтеграції у сторонні сервіси є Stable Diffusion 3.5 Large. Компанія Stability AI надає можливість встановлення даної моделі на локальні комп’ютери та використовувати у своїх дослідницьких роботах чи для розробки власних рішень. Також було визначено,

що для інтеграції суто через API, хорошим вибором є продукти компанії Leonardo.AI, адже вони пропонують вигідніші тарифи.

Після проведених досліджень було складено вимоги до розроблюваної системи та запропоновано алгоритми і сценарії роботи системи генерації зображень. Було обговорено вибір сервісу Segmind та моделі Stable Diffusion 3.5 Large для інтеграції у систему. У результаті було прийнято рішення побудови системи на основі архітектури, яка складається із тонкого клієнта та хмарного сервісу, до якого клієнтський додаток звертається за допомогою API. Після формування цілісної картини розроблюваного додатка, було програмно реалізовано систему генерації зображень для робочого столу із інтеграцією генеративної моделі штучного інтелекту. Дана система цілком виконує всі свої функції та була протестована після її реалізації.

Таким чином, за допомогою даної роботи в результаті було розроблено систему, яка призначена спростити підбір фону робочого столу для звичайних користувачів за допомогою генерації зображення, шляхом інтеграції генеративної моделі штучного інтелекту в дану систему.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Геренко, С. (2024). Штучний інтелект у графічному дизайні: кейс генеративних нейромереж. *Деміург: ідеї, технології, перспективи дизайну*, *7*(1), 78–91. https://doi.org/10.31866/2617-7951.7.1.2024.300924
2. Mijwel, M. M. (2015). History of Artificial Intelligence Yapay Zekânın T arihi. Computer Science,(April 2015), 3-4.
3. Fine, S., Singer, Y. & Tishby, N. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning* **32**, 41–62 (1998). https://doi.org/10.1023/A:1007469218079
4. Cao, Yihan; Li, Siyu; Liu, Yixin; Yan, Zhiling; Dai, Yutong; Yu, Philip S.; Sun, Lichao (March 7, 2023). "A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT". arXiv preprint arXiv:2303.04226.
5. Banh, L., Strobel, G. Generative artificial intelligence. Electron Markets 33, 63 (2023). https://doi.org/10.1007/s12525-023-00680-1
6. Feuerriegel, S., Hartmann, J., Janiesch, C. et al. Generative AI. Bus Inf Syst Eng 66, 111–126 (2024). https://doi.org/10.1007/s12599-023-00834-7
7. Занько Н. В. Вплив штучного інтелекту на освітній процес: аналіз змін / Н. В. Занько, П. А. Глуховецький // Поліграфічні, мультимедійні та web-технології. Інновації та розвиток: монографія. – Харків: ТОВ «Друкарня Мадрид», 2024. – С. 124-134.
8. Mannuru, N. R., Shahriar, S., Teel, Z. A., Wang, T., Lund, B. D., Tijani, S., Pohboon, C. O., Agbaji, D., Alhassan, J., Galley, J., Kousari, R., Ogbadu-Oladapo, L., Saurav, S. K., Srivastava, A., Tummuru, S. P., Uppala, S., & Vaidya, P. (2023). Artificial intelligence in developing countries: The impact of generative artificial intelligence (AI) technologies for development. Information Development, 0(0). https://doi.org/10.1177/02666669231200628
9. Wang, W., & Siau, K. (2018). Ethical and moral issues with AI.

10. "This Person Does Not Exist: Neither Will Anything Eventually with AI". March 20, 2019. (дата звернення: 29.11.2024) Режим доступу: https://alagraphy.medium.com/this-person-does-not-exist-neither-will-anything-if-artificial-intelligence-keeps-learning-1a9fcba728f
11. "ARTificial Intelligence enters the History of Art". December 28, 2018. (дата звернення: 29.11.2024) Режим доступу: https://www.issuewire.com/artificial-intelligence-enters-the-history-of-art-1620667772563815
12. Генеративна змагальна мережа. *Вікіпедія*. (дата звернення: 27.11.2024) Режим доступу: https://uk.wikipedia.org/wiki/Генеративна_змагальна_мережа#cite_note-OpenAI_com-8
13. Gan, Z., Chen, L., Wang, W., Pu, Y., Zhang, Y., Liu, H., ... & Carin, L. (2017). Triangle generative adversarial networks. Advances in neural information processing systems, 30.
14. Wang, T., & Lin, Y. (2024). CycleGAN with Better Cycles. arXiv preprint arXiv:2408.15374.
15. Dewi, C., Chen, RC., Liu, YT. et al. Synthetic Data generation using DCGAN for improved traffic sign recognition. Neural Comput & Applic 34, 21465–21480 (2022). https://doi.org/10.1007/s00521-021-05982-z
16. Purwono, P., Ma'arif, A., Rahmaniar, W., Fathurrahman, H., Frisky, A., & Haq, Q. (2023). Understanding of Convolutional Neural Network (CNN): A Review. *International Journal of Robotics and Control Systems, 2*(4), 739-748. doi:https://doi.org/10.31763/ijrcs.v2i4.888
17. Ramesh, Aditya; Dhariwal, Prafulla; Nichol, Alex; Chu, Casey; Chen, Mark (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents. arXiv:2204.06125.
18. Л. Р. Кулик і О. Б. Мокін, «МЕТОДИ ЗАБЕЗПЕЧЕННЯ КОНСИСТЕНТНОСТІ ГЕНЕРАЦІЇ В ДИФУЗІЙНИХ МОДЕЛЯХ», *Вісник ВПІ*, вип. 4, с. 75–85, Серп. 2024.
19. Ho, J., Saharia, C., Chan, W., Fleet, D.J., Norouzi, M., & Salimans, T. (2021).

Cascaded Diffusion Models for High Fidelity Image Generation. J. Mach. Learn. Res., 23, 47:1-47:33. arXiv.2106.15282

20. Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. arXiv.2006.11239
21. Croitoru, F.-A., Hondru, V., Ionescu, R. T., & Shah, M. (2023). Diffusion Models in Vision: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(9), 10850–10869. arXiv.2209.04747
22. Bank, D., Koenigstein, N., & Giryes, R. (2021). *Autoencoders*. arxiv.org/abs/2003.05991
23. Sadat, S., Buhmann, J., Bradley, D., Hilliges, O., & Weber, R. M. (2024). *LiteVAE: Lightweight and Efficient Variational Autoencoders for Latent Diffusion Models*. arXiv.2405.14477
24. Санніков, Є. (2024). ПОРІВНЯННЯ МОЖЛИВОСТЕЙ АРХІТЕКТУР НЕЙРОННИХ МЕРЕЖ У ДОСЛІДЖЕННЯХ ТВОРІВ МИСТЕЦТВА. *Збірник наукових праць «Українська академія мистецтва»*, (35), 221-229.
25. Козлов, С. Л. ., & Колесницький, О. К. (2024). ЗАСТОСУВАННЯ АРХІТЕКТУРИ ТРАНСФОРМЕР ДО ЗАДАЧІ SUPER-RESOLUTION. *Наукові праці Вінницького національного технічного університету*, (1).
26. Ferrer, J. (2024) How transformers work: A detailed exploration of Transformer architecture, DataCamp. (дата звернення: 30.11.2024) Режим доступу: https://www.datacamp.com/tutorial/how-transformers-work
27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention Is All You Need. arXiv.1706.03762v7
28. Islam, S., Elmekki, H., Elsebai, A., Bentahar, J., Drawel, N., Rjoub, G., & Pedrycz, W. (2024). A comprehensive survey on applications of transformers for deep learning tasks. Expert Systems with Applications, 241, 122666.
29. Johnson, Khari (5 січня 2021). OpenAI debuts DALL-E for generating images from text. VentureBeat. (дата звернення: 03.12.2024) Режим доступу:

https://venturebeat.com/2021/01/05/openai-debuts-dall-e-for-generating-images-from-text/

30. Microsoft Invests In and Partners with OpenAI to Support Us Building Beneficial AGI. OpenAI (22 червня 2019). (дата звернення: 03.12.2024) Режим доступу: https://openai.com/index/microsoft-invests-in-and-partners-with-openai/
31. Valvano, G., Agostino, A., De Magistris, G., Graziano, A., & Veneri, G. (2024). Controllable Image Synthesis of Industrial Data using Stable Diffusion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 5354-5363).
32. Kabir, A. I., Mahomud, L., Al Fahad, A., & Ahmed, R. (2024). Empowering Local Image Generation: Harnessing Stable Diffusion for Machine Learning and AI. Informatica Economica, 28(1).
33. Ruiz-Rojas, L. I., Acosta-Vargas, P., De-Moreta-Llovet, J., & Gonzalez-Rodriguez, M. (2023). Empowering education with generative artificial intelligence tools: Approach with an instructional design matrix. *Sustainability*, *15*(15), 11524. https://doi.org/10.3390/su151511524
34. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. https://arxiv.org/abs/2302.11382
35. Ye, Q., Axmed, M., Pryzant, R., & Khani, F. (2024). *Prompt Engineering a Prompt Engineer*. arxiv.org/abs/2311.05661
36. Ban, Y., Wang, R., Zhou, T., Cheng, M., Gong, B., & Hsieh, C.-J. (2024). *Understanding the Impact of Negative Prompts: When and How Do They Take Effect?* arxiv.org/abs/2406.02965
37. Liu, Y., Deng, G., Li, Y., Wang, K., Wang, Z., Wang, X., Zhang, T., Liu, Y., Wang, H., Zheng, Y., & Liu, Y. (2024). *Prompt Injection attack against LLM-integrated Applications*. arxiv.org/abs/2306.05499
38. Yang, Y., Hui, B., Yuan, H., Gong, N., & Cao, Y. (2023). SneakyPrompt: Jailbreaking Text-to-image Generative Models. arxiv.org/abs/2305.12082

39. Derevyanko, N., & Zalevska, O. (2023). Comparative analysis of neural networks Midjourney, Stable Diffusion, a nd D ALL-E and ways of their implementation in the educational process of students of design specialities. Scientific B ulletin o f M ukachevo State University. Series “Pedagogy and Psychology”, 9(3), 36-44. https://doi.org/10.52534/msu-pp3.2023.36
40. YILDIRIM, E. (2023). COMPARATIVE ANALYSIS OF LEONARDO AI, MIDJOURNEY, AND DALL-E: AI'S PERSPECTIVE ON FUTURE CITIES. *URBANIZM: Journal of Urban Planning & Sustainable Development*, (28).
41. Qian, L., Luo, Z., Du, Y., Guo, L. (2009). Cloud Computing: An Overview. In: Jaatun, M.G., Zhao, G., Rong, C. (eds) Cloud Computing. CloudCom 2009. Lecture Notes in Computer Science, vol 5931. Springer, Berlin, Heidelberg.

ДОДАТКИ

Додаток А

Фрагменти програмних лістингів

Program.cs

```
using AIWallpaperApp.Services;
using Microsoft.Extensions.DependencyInjection;

namespace AIWallpaperApp;

internal static class Program
{
    public static IServiceProvider? ServiceProvider { get; private
set; }

    [STAThread]
    static void Main()
    {
        var host = HostBuilderService.CreateHostBuilder().Build();
        ServiceProvider = host.Services;

        Application.EnableVisualStyles();
        Application.Run(ServiceProvider.GetRequiredService<Form1>());
    }
}
```

appsettings.json

```
{
  "Segmind": {
    "ApiKey": "Secret-API-key",
    "Url": "https://api.segmind.com/v1/stable-diffusion-3.5-large-
txt2img"
  }
}
```

AppSettings.cs

```
namespace AIWallpaperApp.Models;

public class AppSettings
{
    public string ApiKey { get; set; } = string.Empty;
    public string Url {  get; set; } = string.Empty;
}
```

```
HostBuilderService.cs
using AIWallpaperApp.Models;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System.Reflection;

namespace AIWallpaperApp.Services;

public static class HostBuilderService
{
    public static IHostBuilder CreateHostBuilder()
    {
        var config = new
ConfigurationBuilder().SetBasePath(AppContext.BaseDirectory)
            .AddJsonFile("appsettings.json", false, true)
            .AddUserSecrets(Assembly.GetExecutingAssembly(), true)
            .AddEnvironmentVariables()
            .Build();

        var hostBuilder = Host.CreateDefaultBuilder()
            .ConfigureServices((context, services) =>
            {

services.Configure<AppSettings>(config.GetSection("Segmind"));
                services.AddTransient<IImageService, ImageService>();
                services.AddSingleton<Form1>();
            });

        return hostBuilder;
    }
}

IImageService.cs
namespace AIWallpaperApp.Services;

public interface IImageService
{
    Task<HttpResponseMessage> GenerateImageAsync(string prompt);
    int SetWallpaper(string imagePath);
}

ImageService.cs
using AIWallpaperApp.Models;
using Microsoft.Extensions.Options;
using System.Net.Http.Json;
using System.Runtime.InteropServices;

namespace AIWallpaperApp.Services;
```

```
public class ImageService : IImageService
{
    private readonly AppSettings _settings;

    public ImageService(IOptions<AppSettings> settings)
    {
        _settings = settings.Value;
    }

    public async Task<HttpResponseMessage> GenerateImageAsync(string
prompt)
    {
        var random = new Random();
        var randomSeed = random.Next(0, int.MaxValue);

        var data = new
        {
            prompt = prompt!,
            negative_prompt = "low quality, blurry",
            steps = 25,
            guidance_scale = 5.5,
            seed = randomSeed,
            sampler = "euler",
            scheduler = "sgm_uniform",
            width = 1792,
            height = 1024,
            aspect_ratio = "custom",
            batch_size = 1,
            image_format = "jpeg",
            image_quality = 95,
            base64 = false
        };

        using (HttpClient httpClient = new HttpClient())
        {
            var request = new HttpRequestMessage(HttpMethod.Post, new
Uri(_settings.Url));
            request.Content = JsonContent.Create(data);
            request.Headers.Add("x-api-key", _settings.ApiKey);

            return await httpClient.SendAsync(request);
        }

    }

    public int SetWallpaper(string imagePath)
    {
         const int SPI_SETDESKWALLPAPER = 20;
         const int SPIF_UPDATEINFILE = 1;
         const int SPIF_SENDCHANGE = 2;
```

```
        [DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError
= true)]
        static extern int SystemParametersInfo(int uAction, int
uParam, string lpvParam, int fuWinIni);

        return SystemParametersInfo(SPI_SETDESKWALLPAPER, 0,
imagePath, SPIF_UPDATEINFILE | SPIF_SENDCHANGE);
    }
}

Form1.cs
using AIWallpaperApp.Services;
using System.Drawing.Imaging;

namespace AIWallpaperApp;

public partial class Form1 : Form
{
    private readonly IImageService _imageService;

    public Form1(IImageService imageService)
    {
        InitializeComponent();

        _imageService = imageService;
    }

    private async void btnGenerate_Click(object sender, EventArgs e)
    {
        SetUIStateAfter_btnGenerate_Click();

        var prompt = textBoxPrompt.Text;
        var response = await _imageService.GenerateImageAsync(prompt);

        RestoreUIStateAfter_btnGenerate_Click();

        if (response.IsSuccessStatusCode)
        {
            using (var ms = new MemoryStream(await
response.Content.ReadAsByteArrayAsync()))
            using (var image = Image.FromStream(ms))
            {
                var imageName = @"img_" + Guid.NewGuid() + ".jpg";
                var directory =
Directory.GetParent(Environment.CurrentDirectory)?.Parent?.Parent?.Ful
lName;
                var moveToDirectory = directory + "\\Images\\" +
imageName;
```

Продовження додатку А

```
                image.Save(imageName, ImageFormat.Jpeg);
                var imageFileInfo = new FileInfo(imageName);
                imageFileInfo.MoveTo(moveToDirectory);

                picBox.ImageLocation = moveToDirectory;
                lblImageTip.Visible = false;
            }
        }

        else
        {
            MessageBox.Show(response.StatusCode.ToString());
        }
    }

    private void btnSetWallpaper_Click(object sender, EventArgs e)
    {
        _imageService.SetWallpaper(picBox.ImageLocation);
    }

    private void textBoxPrompt_TextChanged(object sender, EventArgs e)
    {
        btnGenerate.Enabled =
!string.IsNullOrWhiteSpace(textBoxPrompt.Text);
    }

    private void SetUIStateAfter_btnGenerate_Click()
    {
        btnGenerate.Enabled = false;
        prgImageGeneration.Visible = true;
        prgImageGeneration.MarqueeAnimationSpeed = 10;
    }

    private void RestoreUIStateAfter_btnGenerate_Click()
    {
        prgImageGeneration.MarqueeAnimationSpeed = 0;
        prgImageGeneration.Visible = false;
        btnGenerate.Enabled = true;
    }

    private void btnSelecFile_Click(object sender, EventArgs e)
    {
        using (var fileDialog = new OpenFileDialog())
        {
            var directory =
Directory.GetParent(Environment.CurrentDirectory)?.Parent?.Parent?.Ful
lName;

            fileDialog.InitialDirectory = directory + "\\Images";
```

Продовження додатку А

```
            fileDialog.Filter = "Image files(*.jpeg; *.jpg;
*.png;)|*.jpeg; *.jpg; *.png;";

            if (fileDialog.ShowDialog() == DialogResult.OK)
            {
                picBox.ImageLocation = fileDialog.FileName;
            }
        }

        lblImageTip.Visible = false;
    }
}
```