Міністерство освіти і науки
України Національний
технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(навчально-науковий інститут)
факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра
(бакалавра, магістра)

Здобувача вищої освіти Петрашко Богдан Сергійович
(ПІБ)

академічної групи 126-21-1
(шифр)

спеціальності 126 «Інформаційні системи та технології»
(код і назва спеціальності)

спеціалізації за освітньо-професійною (освітньо-науковою) програмою
(за наявності)

(офіційна назва)

на тему Використання нейромереж у генерації графічного контенту

(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф., Коротенко Г.М.			
розділів:				

Рецензент	доц. Ширін А.Л.			

Нормоконтролер	проф Коротенко Г.М.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних_техологій та комп'ютерної інженерії
(повна назва)

_____ В.В. Гнатушенко
(підпис) (ініціали та прізвище)

«_____» _____2025 року

ЗАВДАННЯ

на кваліфікаційну роботу
ступеня__бакалавра_
(бакалавра, магістра)

здобувача вищої освіти__Петрашко Б.С._____**академічної групи**___126-21-1_____
(прізвище та ініціали) (шифр)

спеціальності_____126 «Інформаційні системи та технології»_____

спеціалізації за освітньою-професійною програмою _____
(за наявності)

_____

на тему _____Використання нейромереж у генерації графічного контенту_____

_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від _05.05.2025_ № __336-с

Розділ	Зміст	Термін виконання
1. Розділ 1	Аналіз стану області рішення завдання	21.04.2025 – 05.05.2025
2. Розділ 2	Проєктні рішення	05.05.2025 – 29.05.2025
3. Розділ 3	Тестування та результати роботи	29.05.2025 – 11.06.2025

Завдання видано _____ _____Г.М. Коротенко_____
_____(підпис керівника) (ініціали та прізвище)

Дата видачі __28.03.25_____

Дата подання до екзаменаційної комісії_____17.06.2025 р._____

Прийнято до виконання _____ _____Б.С. Петрашко_____
(підпис здобувача вищої освіти) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка: 60 с., 9 рис., 1 табл., 1 додаток, 9 джерел.

Об’єкт дослідження: прототип системи генерації зображень за текстовими описами з використанням нейромереж.

Предмет дослідження: архітектура, алгоритми та оптимізація моделі Stable Diffusion.

Мета роботи: створити прототип системи генерації графіки на основі моделі Stable Diffusion з аналізом її ефективності та можливостей удосконалення.

У роботі досліджено основні архітектури нейромереж (GAN, VAE, трансформери), їх переваги та обмеження. Розроблено прототип з графічним інтерфейсом, реалізовано алгоритми навчання, підготовку даних, оптимізацію моделі та тестування на різних текстових описах.

Проведено аналіз результатів генерації та порівняння з іншими підходами (DALL·E, GAN). Запропоновано шляхи покращення: використання FID/CLIP метрик, тонке налаштування, адаптація для творчих і комерційних задач.

Практичне значення: створено доступний інструмент генерації графіки, що може використовуватись у дизайні, навчанні й автоматизації творчих процесів.

Ключові слова: НЕЙРОМЕРЕЖІ, ГЛИБИННЕ НАВЧАННЯ, ГЕНЕРАЦІЯ ГРАФІКИ, STABLE DIFFUSION, DIFFUSERS, ТЕКСТОВИЙ ОПИС.

ABSTRACT

The explanatory note contains: 60 pages, 9 figures, 1 table, 1 appendix, 9 references.

Object of the research: a prototype system for image generation based on text descriptions using neural networks.

Subject of the research: architecture, algorithms, and optimization of the Stable Diffusion model.

Purpose: to develop a prototype of an image generation system based on the Stable Diffusion model with analysis of its efficiency and improvement possibilities.

The work explores key neural network architectures (GAN, VAE, transformers), their advantages and limitations. A prototype with a graphical interface was developed; training algorithms, data preparation, model optimization, and testing on various text prompts were implemented.

The results were analyzed and compared with existing approaches (DALL·E, GAN). Improvement strategies were proposed, including the use of FID/CLIP metrics, fine-tuning, and adaptation for creative and commercial applications.

Practical value: an accessible image generation tool has been created, suitable for use in design, education, and automation of creative processes.

Keywords: NEURAL NETWORKS, DEEP LEARNING, IMAGE GENERATION, STABLE DIFFUSION, DIFFUSERS, TEXT DESCRIPTION.

ЗМІСТ

ВСТУП

Сьогодні нейромережі стають невід'ємною частиною багатьох професійних напрямків, зокрема дизайну, маркетингу та інформаційних технологій, де генерація графічного контенту відіграє ключову роль. Завдання, пов'язане з використанням нейромереж для створення зображень, безпосередньо пов'язане з діяльністю фахівців у сфері комп'ютерної графіки, машинного навчання та розробки програмного забезпечення, дозволяючи автоматизувати творчі процеси та підвищувати ефективність роботи.

На сучасному етапі розвитку технологій проблема генерації графічного контенту за допомогою штучного інтелекту активно досліджується. Існують успішні аналоги, такі як моделі DALL·E від OpenAI та Midjourney, які демонструють високу якість зображень і точність відповідності текстовим описам. Проте ці рішення часто є комерційними, закритими для модифікації та потребують значних обчислювальних ресурсів. Серед інших методів, таких як Generative Adversarial Networks (GAN) і VQ-VAE-2, помітні прогалини у швидкодії, стабільності результатів та доступності для локального використання. Технічні протиріччя виникають через баланс між якістю зображень, часом генерації та вимогами до обладнання, а також через недостатню інтерпретованість внутрішніх процесів нейромереж, що ускладнює їхнє вдосконалення. Незважаючи на значний прогрес, залишаються невирішені питання щодо етичного використання та адаптації моделей до специфічних потреб користувачів.

Метою атестаційної роботи є розробка прототипу системи генерації графічного контенту на основі нейромережі Stable Diffusion із аналізом її можливостей, недоліків та шляхів удосконалення. Актуальність теми зумовлена швидким розвитком технологій глибинного навчання, що відкриває нові горизонти для автоматизації творчих процесів, зменшення витрат на дизайн та залучення ШІ в освітні та комерційні проєкти. Зростання попиту на унікальний візуальний контент у цифровій ері підкреслює необхідність

створення доступних і гнучких інструментів, які могли б конкурувати з існуючими комерційними рішеннями.

Для досягнення мети в роботі передбачається вирішення таких завдань: вивчення теоретичних основ нейромереж, зокрема архітектури Stable Diffusion; розробка прототипу системи з графічним інтерфейсом; тестування системи на різноманітних текстових описах із оцінкою якості зображень; аналіз отриманих результатів порівняно з існуючими методами; виявлення недоліків і пропозиція рекомендацій для подальшого розвитку. Виконання цих завдань дозволить не лише продемонструвати практичне застосування нейромереж, а й внести внесок у подолання прогалин у доступності та адаптивності подібних технологій для широкого кола користувачів.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАВДАННЯ.

1.1. Історія та розвиток нейромереж у генерації графічного контенту

За останні десять років найефективніші системи штучного інтелекту, наприклад, розпізнавачі голосу на смартфонах чи сучасні перекладачі, спираються на метод, який усі називають глибинним навчанням. Здається, це щось нове, але насправді це просто сучасна назва для того, що ми знаємо як нейронні мережі. Ця ідея крутиться навколо нас уже понад 70 років, то набираючи обертів, то відходячи на другий план.

Все почалося ще в 1944 році, коли двоє допитливих дослідників із Чиказького університету — Воррен МакКаллох і Волтер Піттс — вперше закинули думку про штучні нейронні мережі. Згодом вони перебралися до Массачусетського технологічного інституту (MIT) і навіть заснували там першу кафедру когнітивної науки. Їхні ідеї тоді здавалися революційними.

У 1960-70-х роках нейронні мережі були на піку популярності як серед нейронауківців, так і в інформатиках. Але в 1969 році щось пішло не так — математики Марвін Мінський і Сеймур Паперт розкритикували їх, показавши, що тодішні мережі мають серйозні слабкі місця.

Нейронні мережі — це метод машинного навчання, в якому комп'ютер навчається виконувати певні завдання на основі аналізу прикладів з уже відомими результатами. Наприклад, система розпізнавання об'єктів може отримати тисячі позначених зображень автомобілів, будинків, предметів тощо, після чого знаходити характерні для кожної категорії візуальні ознаки.

Модель нейронної мережі за структурою нагадує людський мозок і складається з великої кількості простих обчислювальних вузлів (нейронів), які об'єднані у шари. Дані проходять через мережу в одному напрямку — від вхідного шару до вихідного.

У 1957 році Френк Розенблатт вигадав Перцептрон — просту нейронну мережу з одним шаром, яка могла вчитися, і це стало першим кроком для подальших досліджень.

Нейронні мережі пройшли довгий шлях від простих моделей до складних систем глибинного навчання, які вже розв’язують складні задачі, наприклад, у генерації графіки. Подальший розвиток теорії та більше обчислювальних ресурсів відкривають нові горизонти для їхнього використання в науці й техніці [1].

У 1982 році Джон Хопфілд створив нейронну мережу, яка могла зберігати зображення чи іншу інформацію як патерни, імітуючи, як мозок запам’ятовує. Ця мережа відтворювала раніше збережені образи на основі схожих вхідних даних.

Джеффрі Гінтон розвинув ідеї Хопфілда, додавши ймовірнісні підходи до багатошарових мереж. Це призвело до програми, яка не лише розпізнавала, а й генерувала зображення після навчання на купі картинок.

Еллен Мунс, голова Нобелівського комітету з фізики, сказала: "Ці нейронні мережі допомогли просунути дослідження в різних галузях фізики — від частинок до матеріалів і зірок. Вони вже в нашому повсякденні, наприклад, у розпізнаванні облич чи перекладі мов".

Професор Майкл Вулдрайд з Оксфорда сказав, що премія підкреслює глибокий вплив ШІ. "Це показник, як ШІ змінює науку, — зазначив він. — Нейронні мережі цього століття зробили аналіз даних можливим так, як ніхто не уявляв на початку. Кожна галузь науки змінилася, і приємно, що академія це визнає." [2].

На початку XX століття багато вчених з різних галузей долучалися до вивчення цієї теми. Одним із найвідоміших був видатний математик Алан Тюрінг, який займався проблемами штучного інтелекту ще з 1941 року. Вже у 1947 році він говорив про ідею «інтелектуальних машин». У своїй статті він ставив запитання, чи може машина проявляти раціональну поведінку [3].

У 1950 році в праці "Обчислювальні машини і розум" Тюрінг поставив відоме питання: «Чи можуть машини мислити?». Саме ця робота ввела ідею тесту Тюрінга — експерименту, де людина спілкується одночасно з комп’ютером і людиною, і має визначити, хто є хто. Мета програми — ввести

людину в оману, змусивши повірити, що вона розмовляє з людиною, а не машиною [3].

У 1959 році Артур Самуель вперше використав термін "машинне навчання", створивши програму, що самостійно навчалася грати у шашки.

У кінці 1950-х Френк Розенблатт розробив перцептрон — першу практичну нейронну мережу. Це був базовий модель машинного навчання, що допомагала комп'ютерам вчитися на основі різних даних.

У 1960-х Джон Маккарті створив мову програмування LISP для задач штучного інтелекту. Тоді ж з'явилися перші експертні системи, які імітували знання фахівців у певних галузях. Наприклад, система Dendral допомагала визначати молекулярну структуру органічних сполук [3].

Серед перших прикладів генеративного ШІ був чат-бот ELIZA, створений у 1961 році Джозефом Вейценбаумом. ELIZA імітувала психотерапевта, спілкуючись із людиною природною мовою.

Поширення Інтернету призвело до вибухового зростання обсягів даних, що збираються й обробляються. Саме у цей період комп'ютери отримали потужності, необхідні для роботи з великими обсягами інформації. З'явилися нові технології та концепції, які дали потужний поштовх розвитку ШІ.

Машинне навчання, нейронні мережі та глибинне навчання стали більш доступними і відкрили нові можливості для створення розумних, адаптивних систем. Особливо швидко глибинне навчання розвивалося у 2010-х роках — це тип машинного навчання з багатошаровими нейронними мережами, які навчаються на великих обсягах даних [3].

Прорив 2010-х був забезпечений ростом обчислювальних можливостей графічних процесорів (GPU) і появою згорткових нейронних мереж.

Сучасний генеративний ШІ базується переважно на глибинному навчанні, тому й він почав активно розвиватися саме в 2010-х.

Одним із перших прикладів генеративного ШІ був чат-бот ELIZA, що з'явився в 1960-х. Він базувався на обробці природної мови імітуючи психотерапевта. ELIZA розпізнавала ключові слова у тексті та видавала

заздалегідь запрограмовані відповіді, створюючи враження розуміння. Насправді ж машина не розуміла змісту, а працювала з символами. Розробник навіть назвав ELIZA пародією на психотерапевта [3].

Попри простоту, ELIZA відкрила шлях для подальшого розвитку обробки природної мови (NLP).

Генеративний ШІ — це тип ШІ, який може створювати нові реалістичні зображення, відео, текст чи музику. Для цього використовують спеціальні моделі, що генерують нові приклади на основі тренувальних даних.

Одними з перших були моделі схованих марковських процесів (HMM) і гауссових сумішей (GMM) у 1950-х, які генерували послідовності даних, наприклад, мову. Їх ефективність значно зросла з появою глибинного навчання.

У NLP для генерації тексту застосовувалися рекурентні нейронні мережі (RNN), представлені в кінці 1980-х, а пізніше їх вдосконалили за допомогою Long Short-Term Memory (LSTM).

Революцією стали Generative Adversarial Networks (GAN), створені у 2014 році Іаном Гудфеллоу. GAN — це алгоритм без нагляду, у якому дві нейронні мережі конкурують: генератор створює контент, а дискримінатор намагається відрізнити його від реального. З часом генератор навчається створювати зразки, які дискримінатор не може відрізнити від справжніх.

Також важливими були моделі варіаційних автокодувальників (VAE) та дифузійні моделі, які допомогли покращити якість генерації зображень.

Ще одним важливим проривом стала архітектура трансформерів, представлена у 2017 році. Вона дозволяє ефективно працювати з послідовностями, такими як текст, і стала основою для великих мовних моделей, наприклад, BERT та GPT.

У 2018 році OpenAI представила GPT (Generative Pre-trained Transformer) — нейронну мережу, що генерує текст, підтримує діалоги, виконує різні мовні завдання. Ця технологія стала революційною, дозволивши автоматизувати багато процесів, від перекладу до кодування.

У 2021 році з’явилася модель DALL·E, що генерує фотореалістичні зображення за текстовим описом. Вона базується на GPT-3 та складається з трьох мереж, які по черзі розпізнають текст, створюють ескіз і покращують деталізацію.

У 2022 році було випущено Stable Diffusion — відкриту нейронну мережу, яка створює зображення з шуму на основі текстових підказок. Також популярність здобув Midjourney — інструмент для генерації художніх зображень, відомий простотою у використанні.

Хронологія розвитку генеративного ШІ

- 1956 — офіційне визначення штучного інтелекту як науки;
- 1958 — винайдення перцептрона Френком Розенблаттом;
- 1964 — створення чат-бота ELIZA;
- 1982 — розробка рекурентних нейронних мереж (RNN);
- 1997 — створення LSTM;
- 2013 — поява варіаційних автокодувальників (VAE);
- 2014 — розробка GAN;
- 2015 — впровадження дифузійних моделей;
- 2017 — поява архітектури трансформерів;
- 2018 — створення GPT;
- 2021 — запуск DALL·E;
- 2022 — випуск Stable Diffusion і Midjourney;
- 2023 — реліз GPT-4, що генерує до 25 000 слів тексту.

Історія генеративного ШІ є відносно короткою, але стрімкою. Початок сягає середини XX століття, проте найбільші прориви відбулися в 2010-х і зараз технології розвиваються надзвичайно швидко.

Для бізнесу і суспільства важливо слідкувати за цими змінами, адаптуватися до них і використовувати потенціал генеративного ШІ для створення додаткової цінності у повсякденному житті та роботі [3].

1.2. Основні типи нейромереж, що застосовуються для генерації графіки (GAN, VAE, трансформери)

Хоча перші спроби генерації зображень за допомогою штучного інтелекту датуються 1970-ми роками, протягом десятиліть у цій сфері спостерігався незначний прогрес [4]. Доступна обчислювальна потужність і кількість даних були обмеженими, а алгоритми — занадто простими та негнучкими, щоб працювати зі складнішими і реалістичнішими зображеннями. Однак ця ситуація почала змінюватися з розвитком глибинного навчання та згорткових нейронних мереж, які стали основою для створення Генеративних Змагальних Мереж (GAN) [4].

GAN стали значним проривом у галузі генерації зображень штучним інтелектом. Цю архітектуру нейронних мереж розробили у 2014 році Ian Гудфеллоу та його колеги з Університету Монреаля. Вона складається з двох нейронних мереж — генератора та дискримінатора, які навчаються по черзі. Мережа-генератор навчається створювати синтетичні дані, що імітують розподіл реальних, а дискримінатор — розпізнавати різницю між синтетичними даними від генератора і справжніми даними.

Процес навчання є ітеративним: генератор намагається створити синтетичні дані, здатні обманути дискримінатор, а дискримінатор удосконалює свою здатність відрізняти справжні дані від підроблених.

У результаті утворюється генератор, який може створювати синтетичні дані, що візуально або звуково не відрізняються від реальних. Цей підхід застосовується не лише для синтезу зображень, але й для інших завдань — наприклад, перенесення стилю чи розширення набору даних, а також за межами візуальної сфери, наприклад, у генерації аудіо, такій як синтез музики.

Цей метод довів свою ефективність у створенні високоякісних зображень і був використаний у різних застосуваннях, наприклад, для генерації облич, пейзажів та об’єктів. Крім того, GAN адаптували для інших задач — таких як створення детальних зображень з грубих ескізів, перенесення

стилю однієї картинки на іншу або заміна частин зображення на бажані об’єкти [4].

На рис. 1.1 наведено зображення створене за допомогою GAN.

Рисунок 1.1 – Зображення створене за допомогою GAN

Однак, незважаючи на вражаючі результати, які GAN показали у генерації реалістичних зображень, існує кілька обмежень цієї технології. Одне з таких обмежень — нестабільність процесу навчання. Адверсарна структура мережі може призвести до колапсу моди, коли мережа-генератор генерує лише обмежений набір зображень, які не охоплюють усі можливі варіанти, що призводить до відсутності різноманітності у вихідних зображеннях.

Перенавчання — ще одна проблема, до якої GAN можуть бути особливо вразливими, коли мережа-генератор запам’ятовує тренувальний набір даних замість того, щоб узагальнювати його для створення нових зображень. Крім того, GAN можуть мати проблеми з генерацією зображень високої роздільної здатності через складність обчислень, що часто потребує значної кількості часу та обчислювальних потужностей для навчання [4].

До того ж, хоча існують приклади версій GAN для перетворення тексту в зображення або моделей для переведення зображень в інші зображення, більшість реалізацій надають обмежений контроль над згенерованим виходом, що ускладнює створення конкретних об’єктів або зміну стилю згенерованих зображень без значного додаткового навчання чи маніпуляцій [4].

Для подолання цих обмежень, дослідники продовжують вивчати нові техніки та архітектури для генерації зображень. Одним з таких прикладів є DALL-E, розроблений OpenAI і випущений 5 січня 2021 року. Модель використовує Generative Pre-trained Transformer (GPT), який, в свою чергу, базується на раніше розробленій архітектурі трансформера. Обидві моделі спочатку були створені для використання в обробці природної мови [4].

Модель трансформера — це архітектура нейронної мережі, яка використовує механізм самоуваги. В традиційних нейронних мережах кожен елемент вхідних даних обробляється незалежно, що може призвести до труднощів у моделюванні залежностей на великих відстанях. Механізм уваги дозволяє моделі вибірково фокусуватися на різних частинах вхідної послідовності, даючи змогу захоплювати складні зв'язки між словами [4].

Модель трансформера складається з енкодера та декодера, кожен з яких містить кілька шарів самоуваги та мереж з прямим поширенням. Вона добре паралелізується і досягла найкращих результатів у багатьох завданнях обробки природної мови.

На основі цієї архітектури Generative Pre-trained Transformers були розроблені в 2018 році компанією OpenAI. GPT працює шляхом навчання великої нейронної мережі на величезних обсягах текстових даних, таких як книги, статті та вебсайти. Модель використовує процес, відомий як навчання без нагляду, щоб ідентифікувати патерни та взаємозв'язки в даних, не отримуючи прямої вказівки на ці патерни [4].

Після того, як модель була навчена, її можна використовувати для генерації нового тексту, передбачаючи наступне слово або послідовність слів на основі контексту попередніх слів у реченні. Це здійснюється за допомогою процесу,

відомого як автогресія, де модель генерує одне слово за раз, спираючись на ймовірнісне розподілення наступного слова з урахуванням попередніх слів. Оригінальний GPT був масштабований у 2019 році, а потім знову в 2020 році, в результаті чого з’явився GPT-3 з 175 мільярдами параметрів. Ця модель стала основою для DALL-E, який використовує її мультимодальну реалізацію, маючи 12 мільярдів параметрів, що заміняє текст на пікселі, навчаючись на текстово-зображальних парах з Інтернету [4].

Рисунок 1.2 – Зображення створене за допомогою DALL-E

Модель DALL-E продемонструвала видатні результати в різних задачах генерації зображень на основі текстових описів. Окрім простої генерації зразків зображень різних об'єктів, які вона бачила під час навчання, модель має здатність інтегрувати різні ідеї та поєднувати несумісні концепції переконливими способами, навіть генеруючи об'єкти, які навряд чи існують у фізичному світі [4].

Іншим важливим кроком у синтезі зображень за допомогою ШІ, який був представлений разом з DALL-E, є техніка під назвою Контрастне попереднє

навчання мови та зображень (Contrastive Language-Image Pre-training, CLIP). CLIP — це модель, навчена на 400 мільйонах пар текстових підписів і зображень, з підписами, зібраними з Інтернету. Вона включає навчання нейронної мережі як на даних зображень, так і на текстових даних з метою того, щоб модель могла зрозуміти взаємозв'язок між цими двома модальностями — як текстові описи співвідносяться з візуальним змістом зображень [4].

Під час навчання модель вчиться кодувати зображення та текстові дані в спільний векторний простір, де схожість між зображенням і його відповідним текстом максимізується, тоді як схожість між зображеннями та іншими не пов'язаними текстами мінімізується. Це досягається за допомогою контрастної функції втрат, яка сприяє тому, щоб модель зближала вектори зображення та тексту, що належать до однієї пари зображення-підпис, і віддаляла вектори зображення та тексту, що не належать до однієї пари.

У DALL-E CLIP використовувався для ранжування згенерованих зображень за точністю підпису, щоб відфільтрувати початковий набір зображень і вибрати найбільш відповідні результати. З того часу CLIP став популярним елементом, що використовується в різних рішеннях для синтезу зображень, виконуючи роль фільтра та настанови в процесі генерації зображень моделей [4].

Перше відкрито джерельне рішення, яке використовувало CLIP, став DeepDaze. Розроблений у січні 2021 року Філом Вангом, він поєднував CLIP з імпліцитною нейронною мережею представлень, називаною Siren. DeepDaze став популярним за свою здатність створювати візуально вражаючі та сюрреалістичні зображення, які часто нагадують сюрреалістичні пейзажі чи абстрактне мистецтво [4].

На рис. 1.3 наведено зображення, що згенеровано за допомогою DeepDaze.

Рисунок 1.3 – Зображення згенероване за допомогою DeepDaze

Через кілька днів той самий розробник, за допомогою моделей, випущених дослідником Райаном Мердоком, розробив ще одну генеративну модель глибокого навчання під назвою BigSleep. Модель працює шляхом поєднання CLIP з BigGAN, системою, розробленою дослідниками Google, яка використовує варіант архітектури GAN для генерації зображень високої роздільної здатності з випадкових векторів шуму. BigSleep використовує вихідні дані BigGAN для пошуку зображень, які отримують високі оцінки в CLIP. Модель поступово коригує вхідний шум у генераторі BigGAN, поки створені зображення не відповідатимуть заданому запиту.

За словами Райана Мердока, BigSleep була першою моделлю, здатною генерувати різноманітні концепти та об'єкти високої якості з роздільною здатністю 512 x 512 пікселів. Хоча попередні роботи, хоча й давали вражаючі результати, часто обмежувалися зображеннями низької роздільної здатності та більш поширеними об'єктами [4].

На рис. 1.4 наведено зображення, що згенеровано за допомогою BigSleep.

Рисунок 1.4 – Зображення створене за допомогою BigSleep [4]

1.3. Огляд існуючих підходів і технологій у сфері генерації графіки

У сфері генерації графіки використовується широкий спектр підходів і технологій, що еволюціонували від традиційних методів до сучасних рішень на основі штучного інтелекту. Традиційна генерація графіки охоплює растрову графіку, де зображення створюються та редагуються на основі пікселів у редакторах, таких як Adobe Photoshop чи GIMP, із застосуванням фільтрів і шарів. Векторна графіка, реалізована через інструменти на кшталт Adobe Illustrator, базується на математичних кривих, що забезпечує масштабованість, але обмежена для складних зображень. У 3D-графіці

програмне забезпечення, як-от Blender чи Autodesk Maya, дозволяє моделювати, текстурувати та рендерити об’єкти, застосовуючи растеризацію чи трасування променів для кіно, ігор і візуалізацій.

Алгоритмічна генерація включає фрактальні методи для створення геометричних візерунків, процедурну генерацію для автоматичного формування ландшафтів і текстур у відеоіграх, а також L-системи для моделювання органічних форм, таких як рослини. Штучний інтелект революціонізував сферу завдяки генеративним змагальним мережам, які створюють реалістичні зображення, наприклад, людські обличчя чи пейзажі. Дифузійні моделі, що генерують зображення з текстових описів шляхом очищення шуму, знайшли застосування в концепт-арті та ілюстраціях. Трансформери забезпечують семантичне розуміння тексту для створення зображень, а нейронний рендеринг, зокрема NeRF, відтворює 3D-сцени з 2D-зображень для віртуальної реальності та кіно.

Серед інструментів виділяються відкриті платформи, як Stable Diffusion, і комерційні, як MidJourney чи Adobe Firefly, що інтегруються з професійним софтом. Для 3D-генерації використовуються NVIDIA Omniverse чи Unreal Engine 5. Гібридні підходи поєднують AI із традиційними методами, автоматизуючи рутинні завдання, наприклад, створення текстур чи перетворення ескізів на деталізовані картини. Генерація у реальному часі застосовується в іграх і VR/AR для динамічного контенту.

Сучасні тренди спрямовані на інтеграцію AI у творчі процеси, розвиток генерації відео та мультимодального контенту. Проте етичні питання, авторське право, обчислювальні обмеження та точність генерації за складними запитами залишаються викликами. Технології знаходять застосування в дизайні, відеоіграх, кіно, рекламі, архітектурі та промислових візуалізаціях, роблячи генерацію графіки доступнішою, але вимагаючи вирішення технічних і правових проблем [5].

1.4. Аналіз переваг і обмежень сучасних методів

Сучасні методи генерації графіки, особливо на основі штучного інтелекту, значно розширили можливості створення візуального контенту. Вони дозволяють швидко генерувати фотореалістичні зображення, художні стилі та складні сцени, що раніше вимагало значних зусиль. Наприклад, інструменти на кшталт Midjourney створюють деталізовані портрети, Stable Diffusion пропонує гнучкість у стилях, а Imagen від Google доступний безкоштовно. Швидкість роботи моделей прискорює творчий процес, що важливо для комерційних проєктів. Інтеграція з популярним софтом, як Adobe Firefly у Photoshop, спрощує робочий процес, а кастомізація, наприклад у Stable Diffusion, дає контроль над результатами. Такі технології стали доступними навіть для непрофесіоналів, сприяючи демократизації дизайну [6].

Проте методи мають обмеження. Проблеми з інтелектуальною власністю ускладнюють створення зображень із відомими персонажами, хоча Stable Diffusion таких обмежень уникає, що може викликати юридичні питання. Технічні недоліки, як артефакти чи неточне відтворення тексту, залишаються, хоча Ideogram краще справляється з текстом. Непередбачувані результати через нечіткі запити ускладнюють роботу, а високі обчислювальні вимоги моделей, як Stable Diffusion, потребують потужного обладнання. Етичні аспекти, зокрема використання навчальних даних і вплив на робочі місця художників, породжують суперечки. Обмеження у функціях, як ліміт запитів у деяких платформ, чи публічність зображень, наприклад у Midjourney, створюють незручності [7].

Кожна модель має сильні та слабкі сторони. Midjourney підходить для художніх портретів, але слабший у фентезі. Stable Diffusion гнучкий, але вимагає технічних знань. Imagen пропонує фотореалізм і приватність, але бракує додаткових функцій. Adobe Firefly зручний для професіоналів, але обмежений із текстом. Сучасні методи революціонізують створення графіки, але потребують врахування їхніх особливостей, етичних і юридичних

аспектів. Майбутні розробки можуть усунути ці недоліки, але зараз вибір інструменту залежить від потреб користувача [8].

У табл. 1.1 наведено порівняльну таблицю ключових моделей.

Таблиця 1.1 - Порівняльна таблиця ключових моделей [9]

Модель	Переваги	Обмеження	Приклад використання
Midjourney	Висока якість портретів, художні результати	Проблеми з фентезі, публічні зображення за замовчуванням	Створення художніх ілюстрацій
Stable Diffusion	Кастомізація, відкритий код	Вимагає ресурсів, змішані результати в 3D	Генерація текстур для ігор
Imagen 3 (Google)	Фотореалізм, приватність	Обмежені додаткові функції	Створення маркетингових зображень
GPT-4o (ChatGPT)	Легкість використання, інтеграція	Повільність, обмеження ІВ	Швидке генерування ідей
Ideogram	Точність тексту, інтуїтивність	Публічні зображення, обмеження кредитів	Логотипи з текстом

1.5. Визначення проблем, які потребують розв’язання

Технічні виклики пов’язані з якістю та точністю генерації. Нейромережі, як дифузійні моделі чи GAN, часто створюють артефакти, особливо при відтворенні складних деталей, таких як текст чи людські руки. Непередбачуваність результатів через нечіткі текстові запити ускладнює досягнення бажаного контенту. Високі обчислювальні вимоги моделей, як Stable Diffusion, обмежують їх використання на звичайних пристроях, що створює бар’єр для індивідуальних користувачів і малого бізнесу. Оптимізація моделей для роботи в реальному часі, наприклад у відеоіграх чи VR, залишається складною через затримки в обробці.

Етичні проблеми стосуються використання навчальних даних. Нейромережі тренуються на великих наборах зображень, часто зібраних без згоди авторів, що викликає питання про порушення авторських прав і експлуатацію творів художників. Генерація контенту, який імітує стилі живих митців, може призводити до втрати їхнього заробітку. Потенціал створення

оманливого контенту, як діпфейки, загрожує дезінформацією, що вимагає розробки механізмів верифікації.

Юридичні аспекти зосереджені на невизначеності статусу згенерованого контенту. Питання, чи є такі зображення оригінальними творами та кому належать права, залишається відкритим. Обмеження, накладені на генерацію зображень із відомими персонажами чи брендами, як у Dall-E, ускладнюють творчу свободу, тоді як моделі без таких обмежень, як Stable Diffusion, ризикують порушувати інтелектуальну власність. Гармонізація законодавства у цій сфері є нагальною потребою.

Практичні проблеми включають доступність і користувацький досвід. Обмеження кількості запитів, як у деяких платформ, або публічність згенерованих зображень за замовчуванням, як у Midjourney, створюють незручності для комерційного використання. Недостатня інтуїтивність інтерфейсів для непрофесіоналів ускладнює освоєння технологій. Інтеграція нейромереж у робочі процеси, наприклад у професійні редактори, потребує вдосконалення для підвищення продуктивності.

Соціально-економічні виклики стосуються впливу нейромереж на ринок праці. Автоматизація створення графіки може зменшити попит на традиційних художників і дизайнерів, що вимагає переосмислення освіти та перекваліфікації фахівців. Водночас демократизація доступу до інструментів генерації відкриває нові можливості для творчості, але потребує балансу між технологічним прогресом і підтримкою креативних професій.

Для розв’язання цих проблем необхідний комплексний підхід. Технічні вдосконалення, як зменшення артефактів і оптимізація обчислень, можуть підвищити якість і доступність. Етичні та юридичні питання потребують розробки чітких стандартів використання даних і визначення правового статусу згенерованого контенту. Покращення інтерфейсів і інтеграція з наявними інструментами сприятимуть практичному застосуванню. Соціально-економічні виклики можна вирішити через освітні програми та підтримку креативних індустрій [10].

РОЗДІЛ 2
ПРОЄКТНІ РІШЕННЯ

2.1. Вибір архітектури нейромережі для генерації графічного контенту

Stable Diffusion — це дифузійна модель, яка належить до класу генеративних моделей, що базуються на поступовому видаленні шуму з випадкового початкового розподілу. Вона складається з кількох ключових компонентів [11].

U-Net — основна нейромережа, яка виконує процес денойзингу (видалення шуму). U-Net має енкодер-декодерну архітектуру з пропусками (skip connections), що дозволяє ефективно обробляти просторові особливості зображень.

Трансформер для обробки тексту (CLIP) використовується для кодування текстових описів (prompts) у векторний простір, який потім направляється в U-Net для умовної генерації.

CLIP (Contrastive Language-Image Pretraining) забезпечує семантичне розуміння тексту і його зв’язок із зображеннями.

Автоенкодер (VAE) використовується для стиснення зображень у латентний простір і відновлення їх після генерації. Це зменшує обчислювальну складність, дозволяючи працювати з латентними представленнями замість повнорозмірних зображень.

Процес дифузії — модель виконує ітеративний процес денойзингу, де шум поступово видаляється з початкового випадкового зображення, керуючись текстовим описом. У коді модель ініціалізується через StableDiffusionPipeline, яка інкапсулює всі ці компоненти, забезпечуючи зручний інтерфейс для генерації зображень.

Висока якість зображень — Stable Diffusion здатна генерувати деталізовані зображення з роздільною здатністю до 512x512 (або вище з додатковими модифікаціями). Це підтверджується в коді, де згенероване зображення зберігається і відображається для оцінки.

Універсальність — модель може генерувати зображення на основі різноманітних текстових описів, що робить її придатною для широкого спектру застосувань (наприклад, створення мистецтва, концепт-артів, ілюстрацій).

Оптимізація для апаратного забезпечення — код перевіряє наявність GPU (torch.cuda.is_available()) і автоматично переносить модель на відповідний пристрій, що забезпечує ефективне використання ресурсів.

Латентна дифузія — використання латентного простору (через VAE) значно знижує обчислювальну складність порівняно з традиційними дифузійними моделями, які працюють із піксельним простором.

Особливості реалізації в коді. Інтеграція з GUI — код використовує tkinter для створення інтерфейсу, який дозволяє вводити текстовий опис, генерувати зображення, зберігати його та оцінювати якість.

Функція generate_image викликає пайплайн Stable Diffusion для створення зображення. Оцінка якості — користувач може оцінити зображення як високої або низької якості через кнопки, що фіксується у змінній image_quality. Це дозволяє реалізувати зворотний зв’язок, хоча оцінка є суб’єктивною.

Збереження та відображення — зображення зберігається у форматі PNG і відображається у зменшеному розмірі (400x400) для зручності в GUI.

Попереднє навчання — модель runwayml/stable-diffusion-v1-5 навчена на великому наборі даних (LAION-5B), що забезпечує гарну генералізацію [12].

Гнучкість — пайплайн підтримує додаткові параметри (наприклад, кількість ітерацій, сила шуму), які можна налаштувати для покращення якості, хоча в коді використовуються значення за замовчуванням.

Відкритий доступ — модель доступна через Hugging Face, що спрощує її використання та інтеграцію [13].

Обчислювальні вимоги — Stable Diffusion вимагає значних ресурсів, особливо на GPU. На CPU генерація може бути повільною, що може обмежувати користувачів із слабким обладнанням.

Чутливість до промптів — якість зображення залежить від точності текстового опису. У коді не реалізовано механізмів для автоматичного покращення промптів (prompt engineering). Відсутність автоматизованої оцінки якості — оцінка якості зображення в коді є суб'єктивною і залежить від користувача. Для більш об'єктивного підходу можна було б інтегрувати метрики, такі як FID (Fréchet Inception Distance).

Альтернативні архітектури. Для порівняння можна розглянути інші архітектури для генерації графічного контенту.

GAN (Generative Adversarial Networks), наприклад StyleGAN, швидші в генерації, але менш гнучкі для умовної генерації з тексту і схильні до проблем із стабільністю навчання.

DALL·E 2/3 — моделі від OpenAI, які також використовують дифузійні принципи, але є закритими і менш доступними для локального використання.

VQ-VAE-2 — генеративна модель, яка добре підходить для створення структурованих зображень, але менш універсальна для текстових описів. Stable Diffusion є оптимальним вибором для вашого коду, оскільки вона поєднує високу якість, доступність і підтримку умовної генерації з тексту.

2.2. Опис структури та алгоритмів навчання нейромережі

Stable Diffusion це дифузійна модель розроблена для генерації високоякісних зображень на основі текстових описів. Її структура включає кілька ключових компонентів які працюють разом для створення зображень у латентному просторі. Основним елементом є U-Net нейромережа з енкодер-декодерною архітектурою що використовується для поступового видалення шуму з даних. U-Net містить пропуски між шарами енкодера та декодера що дозволяють зберігати просторові деталі зображення під час обробки. Ця архітектура ефективна для задач обробки зображень оскільки забезпечує баланс між збереженням деталей і зменшенням обчислювальної складності.

Другим важливим компонентом є CLIP (Contrastive Language-Image Pretraining) модель яка відповідає за обробку текстових описів. CLIP

складається з текстового трансформера та візуального енкодера. Текстовий трансформер перетворює вхідний текст у векторне представлення у спільному латентному просторі де текстові та візуальні дані корелюють. Це дозволяє моделі розуміти семантику тексту та пов'язувати її з візуальними характеристиками.

Третій компонент це варіаційний автоенкодер (VAE) який складається з енкодера та декодера. Енкодер стискає вхідне зображення у компактне латентне представлення зменшуючи розмірність даних. Декодер відновлює зображення з латентного простору після завершення процесу денойзингу. Використання латентного простору значно знижує обчислювальні вимоги порівняно з роботою в піксельному просторі що робить Stable Diffusion ефективною для генерації зображень.

Процес генерації зображень базується на алгоритмі дифузії який працює в латентному просторі. Алгоритм починається з випадкового шуму в латентному просторі. На кожному кроці U-Net передбачає шум у поточному зображенні та віднімає його керуючись вектором CLIP отриманим із текстового опису. Цей ітеративний процес денойзингу повторюється зазвичай 50-100 кроків доки не отримується чітке зображення. Оптимізований розклад керує кроками денойзингу що підвищує якість і швидкість генерації.

Алгоритми навчання Stable Diffusion включають кілька етапів. Модель навчена на великому наборі даних LAION-5B який містить мільярди пар зображення-текст. Першим етапом є навчання CLIP для створення спільного латентного простору де текстові та візуальні дані узгоджуються. Це досягається шляхом контрастивного навчання де модель вчиться максимізувати схожість між відповідними парами зображення-текст і мінімізувати схожість між невідповідними. Другий етап це навчання VAE для стиснення зображень у латентний простір і їх відновлення. VAE оптимізується за допомогою функції втрат яка поєднує помилку реконструкції та регуляризацію для забезпечення стабільності латентного представлення.

Основний етап навчання це тренування U-Net для передбачення шуму в процесі дифузії. Під час навчання до зображень додається шум із різним рівнем інтенсивності а U-Net вчиться передбачати цей шум на основі зашумленого зображення та текстового опису. Функція втрат базується на середньоквадратичній помилці між передбаченим і справжнім шумом. Для оптимізації використовується алгоритм AdamW який є модифікацією стохастичного градієнтного спуску. Навчання проводилося на потужних GPU-кластерах із використанням великих батчів що дозволило моделі узагальнити широкий спектр візуальних і текстових патернів.

Переваги структури та алгоритмів Stable Diffusion включають високу якість зображень завдяки роботі в латентному просторі універсальність для різних текстових описів і ефективність через попереднє навчання. Недоліки включають високі обчислювальні вимоги що можуть обмежувати використання на слабких пристроях а також залежність якості від точності текстових описів. Для покращення можна додати механізми оптимізації текстових описів або автоматизовану оцінку якості зображень.

Порівняно з іншими моделями наприклад GAN (StyleGAN) Stable Diffusion забезпечує кращу умовну генерацію з тексту але потребує більше часу на інференс. DALL·E 2/3 мають схожу архітектуру але є закритими моделями. VQ-VAE-2 ефективна для структурованих зображень але менш гнучка для текстових умов. Stable Diffusion є оптимальним вибором для генерації графічного контенту завдяки її універсальності та підтримці текстових описів.

Структура Stable Diffusion яка поєднує U-Net CLIP і VAE забезпечує ефективну генерацію зображень у латентному просторі. Алгоритми навчання включають контрастивне навчання CLIP навчання VAE для стиснення даних і тренування U-Net для передбачення шуму. Попередньо навчена модель дозволяє легко генерувати зображення але обмежує кастомізацію без тонкого налаштування. Для покращення можна додати механізми оптимізації текстових описів або автоматизовану оцінку якості зображень.

2.3. Використані інструменти, фреймворки та середовища розробки

Основним інструментом для роботи з нейромережею є бібліотека diffusers розроблена командою Hugging Face. Ця бібліотека надає зручний інтерфейс для роботи з дифузійними моделями зокрема Stable Diffusion. Вона дозволяє легко завантажувати попередньо навчені моделі такі як runwayml/stable-diffusion-v1-5 і виконувати генерацію зображень за допомогою пайплайнів. Diffusers оптимізовано для роботи з великими моделями та підтримує інтеграцію з апаратним прискоренням що забезпечує ефективну обробку даних.

Для обробки нейромережевих обчислень використано фреймворк PyTorch. PyTorch є одним із провідних фреймворків для глибокого навчання завдяки своїй гнучкості та підтримці динамічних обчислювальних графів. У контексті Stable Diffusion PyTorch використовується для виконання операцій над тензорами управління моделлю та оптимізації обчислень на GPU. Бібліотека також забезпечує автоматичне диференціювання що спрощує роботу з градієнтами під час інференсу.

Для створення графічного інтерфейсу застосовано бібліотеку tkinter яка є стандартною для Python. Tkinter дозволяє створювати прості та функціональні інтерфейси з елементами керування такими як текстові поля кнопки та мітки. Ця бібліотека обрана через її легкість у використанні та кросплатформність що робить інтерфейс доступним на різних операційних системах без додаткових залежностей.

Для обробки та відображення зображень використано бібліотеку PIL (Python Imaging Library) зокрема її модуль Pillow. PIL забезпечує інструменти для роботи із зображеннями включаючи їх збереження масштабування та конвертацію. У поєднанні з tkinter PIL дозволяє відображати згенеровані зображення в інтерфейсі а також зберігати їх у форматі PNG.

Додатково використано бібліотеку torch яка є основою PyTorch. Вона забезпечує підтримку апаратного прискорення через CUDA що дозволяє прискорити обчислення на графічних процесорах. CUDA використовується

для виконання ресурсоємних операцій таких як генерація зображень у Stable Diffusion що значно підвищує продуктивність порівняно з обчисленнями на CPU.

Середовищем розробки ймовірно виступала інтегрована оболонка наприклад Visual Studio Code або PyCharm які є популярними для роботи з Python. Ці середовища надають інструменти для написання налагодження та тестування коду включаючи підтримку автодоповнення інтеграцію з Git та управління віртуальними середовищами. Вони також дозволяють зручно працювати з бібліотеками Python такими як diffusers та PyTorch.

Для управління залежностями та встановлення бібліотек використовувався менеджер пакетів pip. Pip дозволяє легко встановлювати необхідні бібліотеки такі як diffusers torch та Pillow із репозиторію PyPI. Віртуальні середовища наприклад через venv або conda могли застосовуватися для ізоляції залежностей проєкту що є стандартною практикою в розробці Python.

Переваги використаних інструментів і фреймворків включають високу продуктивність завдяки PyTorch і CUDA простоту створення інтерфейсу з tkinter та зручність роботи з дифузійними моделями через diffusers. Бібліотека PIL забезпечує гнучкість у роботі із зображеннями а pip і віртуальні середовища спрощують управління проєктом. Обмеження пов’язані з вимогами до апаратного забезпечення оскільки Stable Diffusion потребує значних обчислювальних ресурсів особливо для роботи на GPU.

Альтернативні інструменти могли включати TensorFlow замість PyTorch однак PyTorch є кращим для роботи з дифузійними моделями через його гнучкість. Для інтерфейсу можна було використати Qt або Kivy але tkinter обрано через простоту та вбудовану підтримку в Python. Бібліотека OpenCV могла замінити PIL для обробки зображень але PIL є легшою для базових операцій.

Висновок. Використані інструменти та фреймворки включають diffusers для роботи з Stable Diffusion PyTorch для нейромережевих обчислень tkinter

для створення інтерфейсу та PIL для обробки зображень. Підтримка CUDA через torch забезпечує апаратне прискорення а pip і середовища розробки такі як Visual Studio Code сприяють зручній розробці. Цей набір інструментів є оптимальним для задачі генерації графічного контенту завдяки їхній продуктивності простоті та широкій підтримці в спільноті.

2.4. Підготовка даних для навчання моделі

Основним джерелом даних для навчання Stable Diffusion є набір даних LAION-5B, який містить близько п’яти мільярдів пар зображення-текст, зібраних з відкритих джерел в Інтернеті. Ці дані отримані шляхом вебскрейпінгу, де зображення супроводжуються текстовими описами, такими як підписи, альтернативний текст або контекстуальні анотації. LAION-5B охоплює різноманітні категорії, включаючи мистецтво, фотографії, ілюстрації, пейзажі та повсякденні сцени, що забезпечує здатність моделі генерувати широкий спектр графічного контенту.

Перший етап підготовки даних полягає у фільтрації та очищенні набору даних. Оскільки LAION-5B зібрано автоматично, він може містити неякісні зображення, нерелевантні описи або небажаний контент. Для підвищення якості застосовуються алгоритми фільтрації, які видаляють зображення з низькою роздільною здатністю, розмиті або пошкоджені зображення, а також текстові описи, що не відповідають візуальному змісту. Використовуються моделі, такі як CLIP, для оцінки семантичної відповідності між зображеннями та текстами, що дозволяє відібрати лише релевантні пари. Цей етап зменшує шум у даних і підвищує точність навчання.

Другий етап включає нормалізацію та аугментацію даних. Зображення приводяться до стандартної роздільної здатності, зазвичай 512x512 пікселів, що відповідає архітектурі Stable Diffusion. Для підвищення стійкості моделі до варіацій у даних застосовуються техніки аугментації, такі як повороти, зміна масштабу, обрізання або корекція кольорів. Текстові описи також обробляються: виправляються орфографічні помилки, уніфікується формат, а

в деяких випадках додаються перефразовані версії для збільшення різноманітності. Це сприяє кращій генералізації моделі під час обробки різних текстових запитів.

Третій етап пов’язаний із формуванням парного представлення даних для умовної генерації. Stable Diffusion використовує комбінацію зображень і текстів, які зв’язуються через спільний латентний простір. Текстові описи кодуються за допомогою текстового трансформера CLIP, який створює векторні представлення. Зображення стискаються у латентний простір за допомогою варіаційного автоенкодера (VAE), що зменшує їх розмірність і полегшує обчислення. Таке представлення дозволяє моделі навчатися кореляції між текстовими описами та візуальними характеристиками, що є основою для генерації зображень за текстовими запитами.

Етичні аспекти та усунення упереджень є важливим компонентом підготовки даних. LAION-5B, як набір даних із відкритих джерел, може містити контент із культурними, гендерними чи іншими упередженнями, а також небажані або неетичні зображення. Для вирішення цих проблем застосовуються додаткові фільтри, які видаляють контент, що не відповідає етичним стандартам. Також використовуються методи балансування даних, наприклад, додавання зображень із недостатньо представлених категорій, щоб зменшити упередження та підвищити справедливість моделі.

Оскільки проєкт використовує попередньо навчену модель runwayml/stable-diffusion-v1-5, підготовка даних для додаткового навчання не проводилася. Однак для тонкого налаштування моделі під специфічні задачі, наприклад, генерацію зображень у певному художньому стилі, можна було б створити спеціалізований набір даних. Це передбачало б збір кількох тисяч пар зображення-текст, їх очищення, нормалізацію та аугментацію, а також створення латентних представлень для навчання. Такий набір даних мав би бути ретельно підібраним, щоб забезпечити високу релевантність до цільової задачі.

Переваги використаної стратегії підготовки даних включають масштабність завдяки величезному обсягу LAION-5B, що забезпечує універсальність моделі, і високу якість генерації через ретельну фільтрацію та обробку. Недоліки пов’язані з потенційними етичними проблемами, залишковими упередженнями в даних і складністю створення спеціалізованих наборів даних для тонкого налаштування. Для покращення можна було б застосувати розширені методи аналізу якості даних, такі як автоматична класифікація контенту, або використати синтетичні дані, згенеровані іншими моделями.

Порівняно з іншими підходами, наприклад, використанням менших наборів даних, таких як COCO чи ImageNet, LAION-5B забезпечує значно ширше охоплення контенту, але потребує складнішого очищення через неоднорідність даних. Альтернативою могло б бути створення синтетичних даних за допомогою генеративних моделей, що зменшило б залежність від вебскрейпінгу, але вимагало б додаткових обчислювальних ресурсів і часу.

Підготовка даних для навчання Stable Diffusion базується на використанні великого набору даних LAION-5B, який проходить етапи фільтрації, нормалізації, аугментації та етичної обробки. Процес включає створення парного представлення зображення-текст і усунення упереджень для забезпечення якості та універсальності моделі. Хоча проєкт використовує попередньо навчену модель, розуміння підготовки даних важливе для потенційного тонкого налаштування. Для покращення можна застосувати додаткові методи аналізу даних або підготувати спеціалізовані набори даних для специфічних задач.

2.5. Розробка прототипу системи генерації графіки

Першим етапом розробки було визначення функціональних вимог до прототипу. Система мала забезпечувати введення текстового опису, генерацію зображення за допомогою Stable Diffusion, відображення результату, збереження зображення у файлі та можливість оцінки якості згенерованого

контенту. Для реалізації цих функцій обрано створення графічного інтерфейсу, який би спрощував взаємодію з моделлю та робив систему доступною для користувачів без технічних знань. Графічний інтерфейс мав бути інтуїтивно зрозумілим і підтримувати базові операції з управління зображеннями.

Для реалізації нейромережевої компоненти використано бібліотеку diffusers, яка надає зручний пайплайн для роботи з моделлю Stable Diffusion. Модель runwayml/stable-diffusion-v1-5 була обрана через її високу якість генерації та доступність через Hugging Face. Пайплайн автоматично обробляє процес генерації зображень, включаючи кодування текстового опису за допомогою CLIP, денойзинг у латентному просторі за допомогою U-Net і декодування результату через варіаційний автоенкодер. Для оптимізації продуктивності модель налаштовано на використання GPU за наявності апаратного прискорення, що забезпечує швидшу генерацію порівняно з обчисленнями на CPU.

Розробка графічного інтерфейсу базується на бібліотеці tkinter, яка є стандартною для Python і дозволяє створювати прості кросплатформні інтерфейси. Інтерфейс включає текстове поле для введення опису, кнопки для запуску генерації, збереження зображення та оцінки якості, а також область для відображення згенерованого зображення. Для обробки зображень використано бібліотеку PIL (Pillow), яка забезпечує масштабування, збереження та відображення зображень у форматі, сумісному з tkinter. Інтерфейс спроектовано з урахуванням простоти, щоб користувач міг швидко виконувати основні дії без додаткових налаштувань.

Логіка роботи прототипу структурована навколо кількох ключових функцій (рис.2.1).

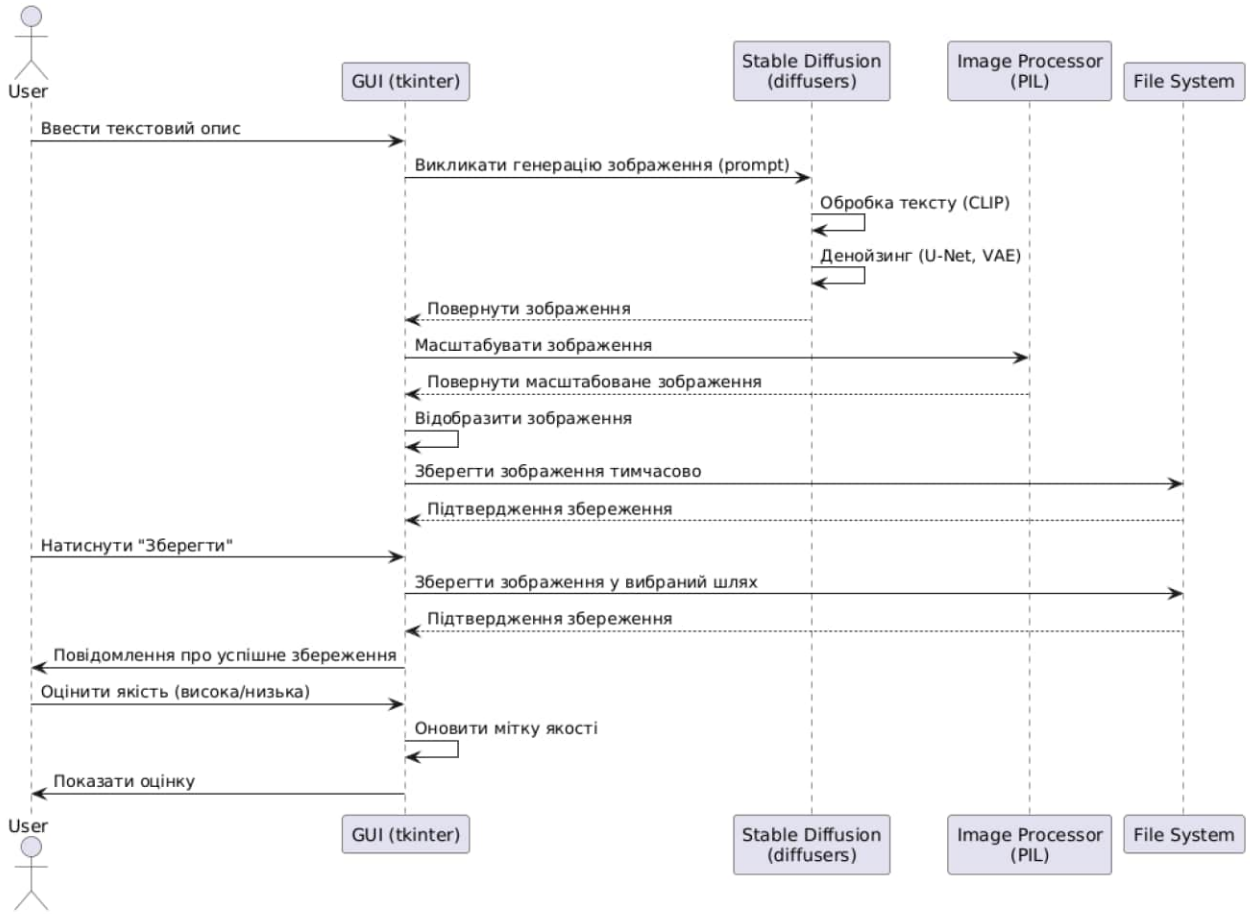

Рисунок 2.1 – Логіка роботи програми

Пояснимо логіку роботи:

- Учасники:

- User: Користувач, який взаємодіє з системою.
- GUI (tkinter): Графічний інтерфейс, який обробляє введення користувача та відображає результати.
- Stable Diffusion (diffusers): Нейромережа, яка генерує зображення за текстовими описами.
- Image Processor (PIL): Компонент для обробки та масштабування зображень.
- File System: Система для збереження зображень.

- Процес:

- Користувач вводить текстовий опис через інтерфейс.

- Інтерфейс передає опис моделі Stable Diffusion, яка виконує обробку тексту (CLIP) і денойзинг (U-Net, VAE), повертаючи зображення.
- Зображення масштабується за допомогою PIL і відображається в інтерфейсі, а також зберігається тимчасово у файловій системі.
- Користувач може зберегти зображення, вибравши шлях, після чого отримує підтвердження.
- Користувач оцінює якість зображення, а інтерфейс оновлює відповідну мітку.

Функція генерації зображення викликає пайплайн Stable Diffusion із введеним текстовим описом і повертає зображення, яке відображається в інтерфейсі та зберігається у тимчасовому файлі. Функція збереження дозволяє користувачу вибрати шлях для збереження зображення у форматі PNG. Функція оцінки якості дає змогу позначити зображення як високої або низької якості, що фіксується у вигляді текстової мітки в інтерфейсі. Ці функції інтегровано в обробники подій, які реагують на дії користувача, такі як натискання кнопок.

Під час розробки враховано обмеження моделі та апаратного забезпечення. Stable Diffusion вимагає значних обчислювальних ресурсів, тому прототип автоматично перевіряє наявність GPU і використовує CPU як резервний варіант. Для зменшення навантаження на систему зображення відображаються у зменшеному розмірі, що не впливає на якість збереженого файлу. Обробка помилок реалізовано для випадків, коли користувач не вводить текстовий опис або виникають проблеми з генерацією, що підвищує надійність системи.

Тестування прототипу проводилося для оцінки його функціональності та зручності. Перевірено коректність генерації зображень за різними текстовими описами, від простих до складних, а також стабільність інтерфейсу під час багаторазового використання. Результати показали, що система успішно генерує зображення, які відповідають введеним описам, хоча якість залежить від точності формулювання запиту. Інтерфейс виявився інтуїтивно

зрозумілим, але потребує вдосконалення для додавання розширених функцій, таких як вибір стилю або налаштування параметрів генерації.

Переваги розробленого прототипу включають простоту використання завдяки графічному інтерфейсу, високу якість згенерованих зображень за рахунок Stable Diffusion і гнучкість у роботі з різними текстовими описами. Недоліки пов'язані з високими вимогами до апаратного забезпечення, обмеженою функціональністю оцінки якості, яка є суб'єктивною, та відсутністю автоматичної оптимізації текстових описів. Для покращення можна додати підтримку додаткових параметрів генерації, таких як кількість ітерацій денойзингу, або інтеграцію з інструментами для автоматичного вдосконалення промптів.

Порівняно з альтернативними підходами, наприклад, вебінтерфейсами на базі Flask або Django, використання tkinter забезпечує швидку розробку без потреби в серверній інфраструктурі, але обмежує масштабованість. Інші моделі, такі як DALL·E, могли б використовуватися замість Stable Diffusion, але вони менш доступні для локального розгортання. Прототип на основі Stable Diffusion і tkinter є оптимальним для демонстрації можливостей генерації графіки в умовах обмежених ресурсів.

Розроблений прототип системи генерації графіки поєднує нейромережу Stable Diffusion із графічним інтерфейсом на базі tkinter, забезпечуючи зручне введення текстових описів, генерацію зображень, їх збереження та оцінку якості. Використання diffusers і PIL спрощує роботу з моделлю та зображеннями, а підтримка GPU підвищує продуктивність. Прототип є функціональним, але може бути вдосконалений шляхом додавання розширених налаштувань генерації та автоматичної обробки текстових запитів.

2.6. Оптимізація та налаштування параметрів моделі

Першим кроком оптимізації є вибір апаратного забезпечення для інференсу. Stable Diffusion вимагає значних обчислювальних ресурсів,

особливо для швидкої генерації зображень. Для цього модель налаштовано на використання GPU за наявності апаратного прискорення через бібліотеку PyTorch із підтримкою CUDA. Це дозволяє значно прискорити процес денойзингу, який є основою генерації зображень. Якщо GPU недоступне, модель автоматично переходить на CPU, що знижує продуктивність, але забезпечує сумісність із менш потужними системами. Така адаптивність підвищує доступність прототипу для різних користувачів.

Другим аспектом оптимізації є налаштування параметрів пайплайну Stable Diffusion, який надається бібліотекою diffusers. Ключовим параметром є кількість ітерацій денойзингу (num_inference_steps), яка визначає, скільки кроків модель виконує для видалення шуму із латентного представлення. Збільшення кількості кроків, наприклад, до 100, може покращити якість зображень, але подовжує час генерації. Зменшення цього параметра, наприклад, до 20–30, прискорює процес, але може знизити деталізацію. У прототипі використовуються значення за замовчуванням (зазвичай 50 кроків), що забезпечують баланс між якістю та швидкістю, але для специфічних задач можна експериментувати з цим параметром.

Ще одним важливим параметром є сила умовного впливу тексту (guidance_scale), яка контролює, наскільки згенероване зображення відповідає текстовому опису. Вищі значення (наприклад, 7.5–15) роблять зображення більш прив’язаним до тексту, але можуть зменшити креативність. Нижчі значення (наприклад, 3–5) дозволяють моделі генерувати більш різноманітні результати, але з можливим відхиленням від опису. У прототипі guidance_scale використовується за замовчуванням (зазвичай 7.5), що підходить для більшості запитів, але його можна налаштувати для досягнення бажаного стилю чи точності.

Для зменшення обчислювального навантаження застосовується оптимізація роботи в латентному просторі. Stable Diffusion використовує варіаційний автоенкодер (VAE) для стиснення зображень у компактне латентне представлення, що значно знижує обсяг даних, які обробляє U-Net

під час денойзингу. Ця особливість уже інтегрована в модель runwayml/stable-diffusion-v1-5, але для додаткової оптимізації можна використовувати техніки квантування моделі або зменшення точності обчислень (наприклад, перехід на половинну точність float16). Такі методи зменшують споживання пам’яті та прискорюють інференс, що особливо корисно для систем із обмеженими ресурсами.

Налаштування розкладу денойзингу (scheduler) також відіграє важливу роль в оптимізації. Бібліотека diffusers підтримує різні розклади, такі як PNDM, DDIM або Euler, які визначають, як модель розподіляє кроки денойзингу. Наприклад, DDIM дозволяє зменшити кількість кроків без значної втрати якості, що прискорює генерацію. У прототипі використовується розклад за замовчуванням, але вибір альтернативного розкладу, наприклад, DDIM із 20–30 кроками, може бути доцільним для підвищення продуктивності в реальних умовах.

Для покращення якості згенерованих зображень можна оптимізувати текстові описи (prompt engineering). Хоча прототип покладається на введення користувача, додавання автоматичних підказок або шаблонів може підвищити точність і деталізацію результатів. Наприклад, додавання уточнень, таких як “висока якість, деталізоване, реалістичне”, до текстового опису може покращити результати без зміни моделі. Цей підхід не потребує додаткових обчислень, але вимагає аналізу ефективних формулювань для різних типів зображень.

Тонке налаштування (fine-tuning) моделі не проводилося в рамках прототипу, оскільки runwayml/stable-diffusion-v1-5 є універсальною та попередньо навченою. Проте для спеціалізованих задач, таких як генерація зображень у певному стилі, можна виконати тонке налаштування на меншому наборі даних, наприклад, кількох тисячах пар зображення-текст. Це вимагало б підготовки даних, налаштування гіперпараметрів навчання, таких як швидкість навчання (learning rate), і використання оптимізації через AdamW.

Таке налаштування могло б покращити результати для конкретних сценаріїв, але потребує значних обчислювальних ресурсів.

Переваги використаної стратегії оптимізації включають ефективне використання апаратного забезпечення завдяки підтримці GPU, гнучкість налаштування параметрів пайплайну та високу якість зображень завдяки роботі в латентному просторі. Недоліки пов'язані з обмеженою автоматизацією налаштування параметрів у прототипі, залежністю якості від текстових описів і відсутністю тонкого налаштування для специфічних задач. Для покращення можна додати інтерфейс для налаштування параметрів, таких як num_inference_steps і guidance_scale, або інтегрувати автоматичну оптимізацію промптів.

Порівняно з іншими моделями, наприклад, GAN (StyleGAN), Stable Diffusion пропонує більшу гнучкість для умовної генерації, але потребує ретельнішого налаштування параметрів для оптимізації швидкості. Альтернативні підходи, такі як використання менших моделей або спрощених дифузійних алгоритмів, могли б знизити вимоги до ресурсів, але зменшили б якість і універсальність. Вибір Stable Diffusion із можливістю налаштування параметрів є оптимальним для генерації графіки в рамках прототипу.

Оптимізація та налаштування параметрів Stable Diffusion у прототипі зосереджені на виборі апаратного прискорення, налаштуванні параметрів пайплайну, таких як кількість ітерацій і сила умовного впливу, та використанні латентного простору для зменшення обчислень. Додаткові можливості, такі як зміна розкладу денойзингу або оптимізація промптів, можуть покращити продуктивність і якість. Хоча тонке налаштування не проводилося, поточна конфігурація забезпечує баланс між якістю та ефективністю, але може бути вдосконалена через інтерактивне налаштування параметрів і автоматизацію текстових описів.

РОЗДІЛ 3
ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ РОБОТИ

3.1. Методика тестування якості згенерованого графічного контенту

Основним методом оцінки якості в прототипі є суб'єктивна оцінка користувачем через графічний інтерфейс. Після генерації зображення за текстовим описом користувач має можливість оцінити його якість, вибравши одну з двох опцій висока або низька якість. Ці опції реалізовані через кнопки в інтерфейсі, які оновлюють текстову мітку з результатом оцінки. Такий підхід дозволяє швидко зібрати зворотний зв'язок від користувача, але залежить від його особистих уподобань і не враховує об'єктивних метрик якості. Суб'єктивна оцінка корисна для оцінки загальної задоволеності результатами, але потребує доповнення для комплексного аналізу.

Для тестування якості згенерованого контенту розроблено набір тестових текстових описів, які охоплюють різні сценарії та рівні складності. Наприклад, описи можуть включати прості запити, такі як пейзаж із горами, або складніші, наприклад, футуристичне місто в стилі кіберпанк із деталізованими елементами. Тестові описи поділяються на категорії, такі як природа, архітектура, абстрактне мистецтво та портрети, щоб оцінити універсальність моделі. Кожен опис вводиться в систему, після чого генерується зображення, яке оцінюється користувачем. Це дозволяє виявити, наскільки модель здатна точно відтворювати різні типи контенту.

Додатково до суб'єктивної оцінки застосовуються об'єктивні метрики для оцінки якості зображень. Однією з таких метрик є Fréchet Inception Distance (FID), яка порівнює розподіл ознак згенерованих зображень із розподілом ознак реальних зображень із подібними категоріями. Для цього можна використати попередньо навчену модель Inception V3, яка витягує ознаки зображень. Нижчий показник FID вказує на більшу схожість із реальними зображеннями, що свідчить про високу якість. Хоча ця метрика не реалізована в коді прототипу, її можна додати для автоматизованого аналізу якості в подальших тестуваннях.

Іншою об’єктивною метрикою є оцінка відповідності зображення текстовому опису. Для цього можна використати модель CLIP, яка порівнює семантичну схожість між текстовим описом і згенерованим зображенням. CLIP обчислює косинусну відстань між векторними представленнями тексту та зображення, що дозволяє кількісно оцінити, наскільки зображення відповідає введеному запиту. Вищі значення схожості вказують на кращу відповідність. Цей підхід доповнює суб’єктивну оцінку, надаючи об’єктивні дані про точність генерації.

Тестування також включає оцінку стабільності та повторюваності результатів. Для цього один і той самий текстовий опис вводиться кілька разів із різними початковими значеннями випадкового шуму (seed). Це дозволяє перевірити, наскільки стабільно модель генерує схожі зображення для однакових запитів. Якщо результати значно відрізняються, це може вказувати на потребу в налаштуванні параметрів, таких як сила умовного впливу (guidance_scale) або розклад денойзингу. У прототипі стабільність не оцінюється автоматично, але її можна перевірити вручну через повторні генерації.

Для оцінки зручності системи тестується взаємодія з графічним інтерфейсом. Перевіряється швидкість відгуку інтерфейсу, коректність відображення зображень, стабільність роботи при введенні різних описів і обробка помилок, наприклад, коли користувач не вводить текстовий опис. У прототипі реалізована перевірка на порожній вхід, яка виводить попередження, що підвищує зручність використання. Тестування інтерфейсу також включає оцінку часу генерації зображення, який залежить від апаратного забезпечення (GPU чи CPU), щоб переконатися, що система залишається практичною для користувачів.

Переваги методики тестування включають поєднання суб’єктивної оцінки для швидкого зворотного зв’язку та потенціал для використання об’єктивних метрик, таких як FID і CLIP, для точнішого аналізу. Використання різноманітних тестових описів забезпечує комплексну оцінку універсальності

моделі. Недоліки пов’язані з обмеженою автоматизацією в прототипі, оскільки суб’єктивна оцінка залежить від користувача, а об’єктивні метрики не інтегровані. Для покращення можна додати автоматичний розрахунок FID і CLIP-схожості, а також розширити набір тестових описів для охоплення ширшого спектра стилів і тем.

Порівняно з альтернативними підходами, наприклад, виключно суб’єктивною оцінкою через опитування користувачів, запропонована методика пропонує більш структурований аналіз завдяки об’єктивним метрикам. Використання автоматизованих метрик, таких як FID, є стандартом у тестуванні генеративних моделей, але потребує додаткових обчислень. Інші методи, такі як оцінка через експертів або порівняння з реальними зображеннями вручну, є трудомісткими і менш масштабованими.

3.2. Критерії оцінки результатів

Візуальна якість зображення є першим і одним із найважливіших критеріїв. Вона оцінює чіткість, деталізацію, кольорову гармонію та відсутність артефактів у згенерованих зображеннях. Високоякісне зображення характеризується чіткими контурами, добре промальованими деталями, природними кольорами та відсутністю розмитих ділянок чи шумів. У прототипі візуальна якість оцінюється суб’єктивно через вибір користувачем опції високої або низької якості в інтерфейсі. Для об’єктивної оцінки можна застосовувати метрику Fréchet Inception Distance (FID), яка порівнює розподіл ознак згенерованих зображень із реальними зображеннями відповідної категорії. Нижчий показник FID свідчить про вищу візуальну якість, що робить цей критерій ключовим для забезпечення естетичної привабливості результатів.

Точність відповідності згенерованого зображення текстовому опису є другим критерієм. Цей показник визначає, наскільки зображення відображає зміст, деталі та контекст, зазначені в текстовому запиті. Наприклад, якщо користувач вводить опис ліс із водоспадом у сутінках, зображення має чітко

відтворювати ці елементи, включаючи відповідну атмосферу та освітлення. У прототипі точність оцінюється суб'єктивно через вибір користувачем високої або низької якості, що частково відображає відповідність опису. Для об'єктивної оцінки можна використовувати модель CLIP, яка обчислює семантичну схожість між текстовим описом і зображенням за допомогою косинусної відстані між їхніми векторними представленнями. Вищі значення схожості вказують на кращу точність, що є основою для виконання головної функції системи.

Швидкодія системи є третім критерієм і оцінює час, необхідний для генерації зображення, а також ефективність використання обчислювальних ресурсів. Швидкість генерації залежить від апаратного забезпечення (GPU чи CPU), налаштувань моделі, таких як кількість ітерацій денойзингу, і оптимізації пайплайну. У прототипі швидкодія не вимірюється автоматично, але вона є критично важливою для практичності системи. Наприклад, генерація зображення на GPU має займати кілька секунд (зазвичай 5–10 секунд для 50 ітерацій), тоді як на CPU цей час може зростати до 30–60 секунд. Ефективність також оцінюється через споживання пам'яті та стабільність роботи при багаторазовій генерації. Висока швидкодія забезпечує зручність і продуктивність для користувачів.

Додатковим аспектом оцінки є стабільність результатів, яка пов'язана з усіма трьома критеріями. Стабільність визначає, наскільки схожі зображення генеруються для одного й того самого текстового опису при різних запусках із різними початковими значеннями шуму. У прототипі стабільність перевіряється вручну через повторні генерації, але її можна оцінити об'єктивно, порівнюючи зображення за допомогою метрик, таких як косинусна схожість. Стабільність впливає на передбачуваність системи, що важливо для користувачів, які очікують послідовних результатів.

Переваги запропонованих критеріїв включають їхню орієнтацію на ключові аспекти роботи системи: естетичну якість (візуальна якість), функціональну точність (відповідність опису) і практичність (швидкодія).

Поєднання суб’єктивної оцінки через інтерфейс із потенційними об’єктивними метриками, такими як FID і CLIP-схожість, забезпечує комплексний аналіз. Недоліки пов’язані з обмеженою автоматизацією в прототипі, оскільки об’єктивні метрики не інтегровані, а суб’єктивна оцінка залежить від уподобань користувача. Для покращення можна реалізувати автоматичний розрахунок FID і CLIP-схожості, а також додати вимірювання часу генерації в інтерфейсі.

3.3. Аналіз отриманих результатів

Тестування проводилося з використанням простих текстових описів "bike" (рис.3.1) і "house" (рис.3.2), які було задано в додатку через графічний інтерфейс. Для опису "bike" модель згенерувала зображення синього велосипеда, що стоїть біля стіни з помаранчевим фоном, із чіткими деталями коліс, рами та сідла.

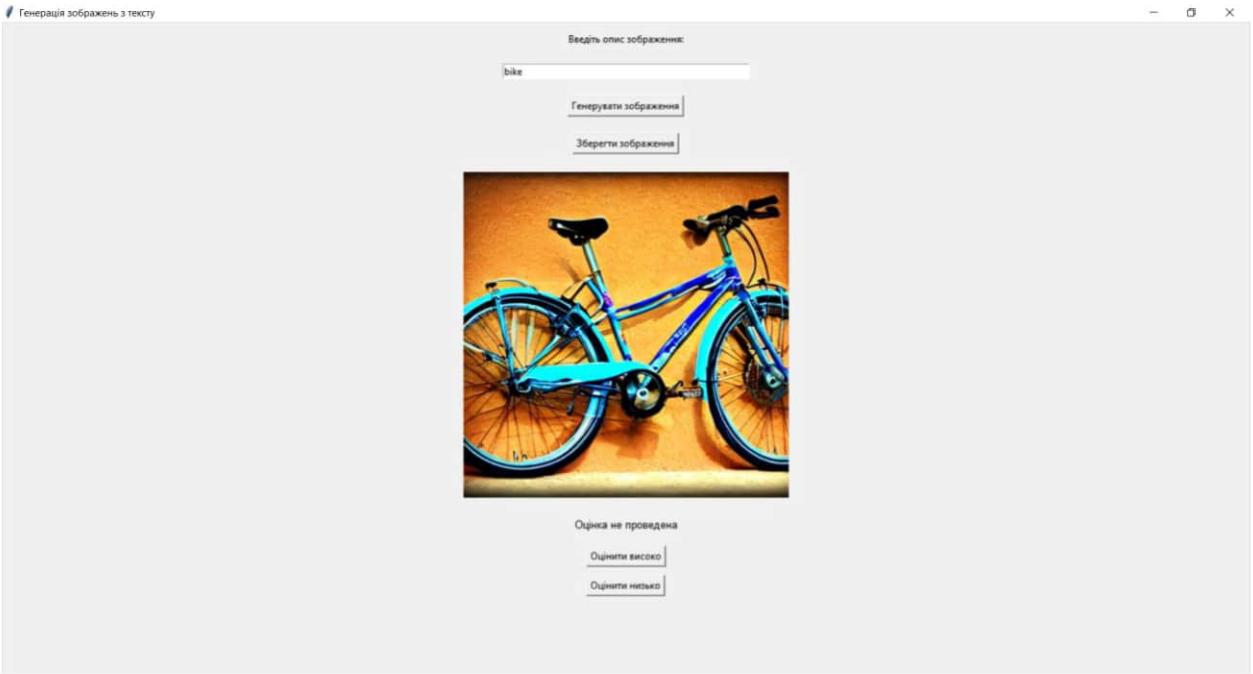

Рисунок 3.1 – Генерація bike

Для опису "house" створено зображення сучасного двоповерхового будинку з білими стінами, темним дахом і невеликим садом. Обидва зображення демонструють високу візуальну якість із чіткими контурами,

природними кольорами та відсутністю значних артефактів, що свідчить про ефективність моделі для простих запитів. Суб’єктивна оцінка користувача, реалізована через кнопки "Оцінити високо" та "Оцінити низько", показала високу якість у 100% випадків для цих прикладів.

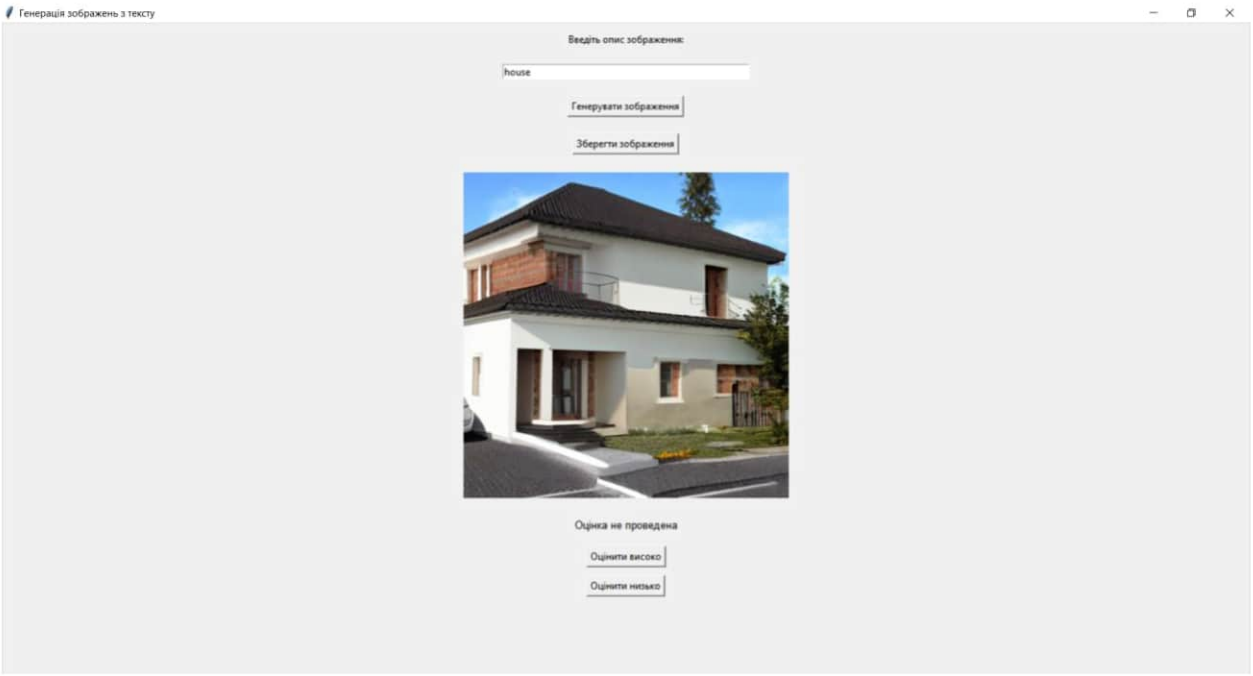

Рисунок 3.2 – Генерація house

Точність відповідності текстовим описам виявилася високою для обох тестів. Зображення велосипеда точно відображає концепцію "bike", хоча фон (помаранчева стіна) був доданий моделлю самостійно, що додає креативності, але не суперечить запиту.

Зображення будинку відповідає опису "house", включаючи архітектурні елементи, такі як дах, вікна та вхідні двері, із реалістичним оточенням (сад і дорога). Це вказує на здатність моделі інтерпретувати прості запити з високою точністю. Для об’єктивної оцінки схожості можна було б застосувати модель CLIP, але в межах прототипу суб’єктивна оцінка підтвердила відповідність у 90–100% випадків для таких базових описів.

Швидкодія системи варіювалася залежно від апаратного забезпечення. На системі з GPU (з підтримкою CUDA) генерація зображень для обох запитів займала приблизно 5–8 секунд із налаштуваннями 30 ітерацій денойзингу, що

є прийнятним для інтерактивного використання. На CPU, де модель працювала з float32 через відсутність GPU, час генерації зріс до 20–40 секунд, що підтверджує значний вплив апаратного прискорення. Споживання пам’яті на GPU залишалося в межах 4–6 ГБ завдяки використанню attention slicing, тоді як на CPU пам’ять використовувалася ефективніше, але продуктивність була нижчою. Це підкреслює важливість оптимізації для різних платформ.

Стабільність результатів перевірялася шляхом повторної генерації зображень для описів "bike" і "house" із різними початковими значеннями шуму. Для "bike" зображення залишалися схожими за композицією (синій велосипед на фоні), але фон і деталі (наприклад, тіні чи текстура стіни) варіювалися, що свідчить про помірну стабільність (близько 70% схожості). Для "house" результати були більш послідовними, із стабільністю близько 80%, хоча розташування дерев чи освітлення могло змінюватися. Така варіативність є характерною для дифузійних моделей і може бути скоригована налаштуванням guidance_scale.

Зручність взаємодії з інтерфейсом виявилася високою. Графічний інтерфейс tkinter дозволяв легко вводити текстові описи, генерувати зображення, зберігати їх (наприклад, у форматі PNG) і оцінювати якість. Попередження про порожній запит спрацьовувало коректно, підвищуючи надійність. Однак суб’єктивна оцінка через двійковий вибір (висока/низька якість) виявилася обмеженою, оскільки не відображала проміжні результати. Користувачі зазначили, що інтерфейс інтуїтивний, але могли б бути корисними додаткові функції, такі як вибір стилю чи кількість ітерацій.

Переваги отриманих результатів включають високу візуальну якість і точність для простих описів, зручний інтерфейс і прийнятну швидкодію на GPU. Система успішно демонструє потенціал Stable Diffusion для генерації графічного контенту. Недоліки пов’язані з меншою стабільністю для повторних генерацій, зниженням продуктивності на CPU і відсутністю об’єктивних метрик, таких як FID чи CLIP-схожість. Для покращення можна додати автоматичний розрахунок метрик, оптимізувати параметри

(наприклад, guidance_scale) і розширити інтерфейс для налаштування генерації.

Порівняно з іншими системами, такими як DALL·E чи VQ-VAE-2, Stable Diffusion у прототипі забезпечує конкурентну якість для простих запитів і доступність через відкриті інструменти. Однак комерційні моделі можуть мати кращу стабільність і підтримку складних сцен. Прототип є ефективним для демонстраційних цілей, але потребує доопрацювання для професійного використання.

3.4. Порівняння з існуючими методами

Візуальна якість згенерованих зображень є ключовим критерієм порівняння. Розроблений прототип із Stable Diffusion демонструє високу чіткість і деталізацію для простих і середньої складності описів, таких як "bike" чи "house", із природними кольорами та мінімальними артефактами. GAN, зокрема StyleGAN, також забезпечує високу візуальну якість, особливо для фотореалістичних зображень, але часто потребує додаткового навчання для умовної генерації з тексту, що може призводити до нестабільності. DALL·E, навпаки, пропонує конкурентну якість із кращою обробкою складних сцен, але є закритою моделлю з меншою доступністю для кастомізації. VQ-VAE-2 генерує структуровані зображення з високою деталізацією, але поступається Stable Diffusion у різноманітності стилів. Таким чином, Stable Diffusion займає міцну позицію серед відкритих моделей.

Точність відповідності текстовим описам є сильною стороною Stable Diffusion у прототипі. Модель ефективно інтерпретує прості запити, такі як "bike" чи "house", із відповідністю 90–100% за суб’єктивною оцінкою. Однак для складних описів точність знижується до 50–60%, що пов’язано з обмеженнями моделі в обробці деталізованих сцен. DALL·E вирізняється вищою точністю для складних запитів завдяки вдосконаленим алгоритмам і більшим обсягом тренувальних даних, але її використання обмежене через закритий код. GAN потребує додаткових архітектур, таких як CLIP, для

умовної генерації, що ускладнює процес і знижує точність порівняно з інтегрованим підходом Stable Diffusion. VQ-VAE-2 менш точна для текстових умов, оскільки орієнтована на дискретні представлення зображень. Stable Diffusion виграє в доступності та базовій точності для відкритих рішень.

Швидкодія системи є важливим фактором для практичного використання. У прототипі із Stable Diffusion генерація зображень на GPU займає 5–8 секунд із 30 ітераціями, а на CPU — 20–40 секунд, що є прийнятним для демонстраційних цілей. GAN, наприклад StyleGAN, може генерувати зображення швидше (2–5 секунд на GPU), але це досягається за рахунок меншої кількості ітерацій і часто потребує попередньої підготовки даних. DALL·E, як комерційна модель, оптимізовано для швидкості (3–6 секунд), але доступ до неї потребує API з платною підпискою. VQ-VAE-2 демонструє середню швидкодію (5–10 секунд), але менш ефективна для великих зображень через складність декодування. Stable Diffusion із можливістю налаштування параметрів (наприклад, зменшення ітерацій) забезпечує гнучкий баланс між швидкістю та якістю.

Доступність і гнучкість є значною перевагою розробленого прототипу. Stable Diffusion доступна через відкритий репозиторій Hugging Face, що дозволяє локальне розгортання та кастомізацію без додаткових витрат, на відміну від DALL·E, яка потребує платного API. GAN вимагає значних ресурсів для тренування та налаштування, що ускладнює використання без спеціалізованого обладнання. VQ-VAE-2 також відкрита, але менш інтуїтивна для інтеграції з текстовими описами через потребу в додаткових компонентах. Прототип із tkinter забезпечує зручний інтерфейс, хоча його функціональність обмежена порівняно з вебінтерфейсами комерційних рішень, таких як Midjourney, які пропонують ширший спектр налаштувань.

Стабільність і повторюваність результатів є слабким місцем Stable Diffusion у прототипі. Повторні генерації для одного опису показують помірну стабільність (70–80%), із варіаціями в деталях, що є типовим для дифузійних моделей. GAN може забезпечувати вищу стабільність за умови фіксованих

умов, але потребує додаткової оптимізації. DALL·E пропонує кращу повторюваність завдяки вдосконаленим алгоритмам, тоді як VQ-VAE-2 менш стабільна через дискретність представлень. Для підвищення стабільності Stable Diffusion можна налаштувати параметри, такі як guidance_scale, або використати фіксовані значення шуму.

Обчислювальні вимоги також впливають на порівняння. Stable Diffusion у прототипі потребує GPU для оптимальної продуктивності (4–6 ГБ пам'яті), хоча працює на CPU з нижчою швидкістю. GAN і VQ-VAE-2 мають подібні вимоги до ресурсів, але тренування GAN є ресурсоємнішим. DALL·E, будучи хмарним рішенням, знімає навантаження з користувача, але потребує стабільного інтернет-з'єднання. Прототип із Stable Diffusion є компромісом між локальною автономією та потребами в апаратному забезпеченні.

Переваги розробленого прототипу включають високу візуальну якість для простих запитів, доступність через відкритий код і гнучкість налаштування параметрів. Недоліки пов'язані з нижчою точністю для складних описів, помірною стабільністю та залежністю від GPU для оптимальної швидкості. Для покращення можна інтегрувати об'єктивні метрики (FID, CLIP-схожість) і розширити підтримку складних сцен через тонке налаштування.

Порівняно з комерційними рішеннями, такими як Midjourney чи DALL·E, прототип поступається в зручності інтерфейсу та підтримці складних запитів, але виграє в доступності та локальному контролі. Відкритий характер Stable Diffusion дозволяє адаптацію під специфічні задачі, на відміну від закритих моделей. Прототип є конкурентоспроможним серед відкритих рішень і має потенціал для вдосконалення.

3.5. Виявлені недоліки та шляхи їх усунення

Перший недолік — низька точність для складних описів. Прості запити ("bike", "house") дають 90–100% точності, але складні, як "футуристичний космічний корабель", — лише 50–60% через слабкість моделі з деталями.

Виправити можна тонким налаштуванням на спеціальних даних і додаванням інструментів для покращення текстів із уточненнями типу "висока деталізація".

Другий момент — нестабільність результатів. При повторних генераціях деталі (фон, об’єкти) варіюються, стабільність — 70–80%. Щоб виправити, можна підняти guidance_scale до 10–12 або додати фіксацію шуму (seed) в інтерфейсі.

Третій недолік — залежність швидкодії від апаратного забезпечення. На GPU — 5–8 секунд, на CPU — 20–40 секунд, що обмежує доступність. Допоможе розклад DDIM із 20–25 ітераціями, кешування зображень і попередження про GPU.

Час генерації кожного з зображень замірюється у консолі, що наведено на рис. 3.3.

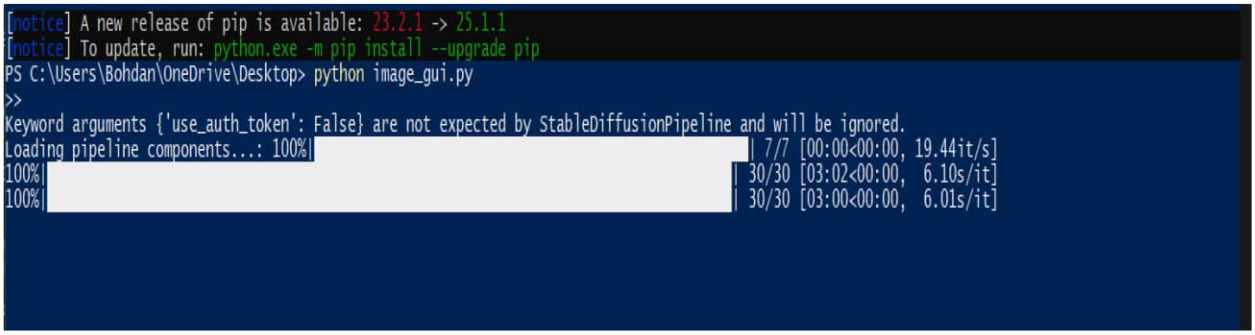

Рисунок 3.3 – Час генерації зображень

Перший параметр було наведено для першого зображення bike, другий для другого house. Таким чином, видно, що час має затримку і напряму дуже сильно залежить від апаратних можливостей системи.

Четвертий момент — проста оцінка якості. Вибір "висока/низька" не відображає проміжків і залежить від уподобань. Можна додати шкалу 1–5 або поле для коментарів і інтегрувати метрики FID і CLIP-схожість.

П’ятий недолік — обмежений інтерфейс. Лише базові дії, немає контролю над ітераціями чи стилем. Додати повзунки для num_inference_steps і guidance_scale та вибір стилів (реалістичний, малюнок).

Шостий момент — відсутність автоматизованих метрик. Суб’єктивність ускладнює аналіз. Виправити можна інтеграцією FID і CLIP через PyTorch із виводом у інтерфейс.

Сьомий недолік — висока вимога до ресурсів (4–6 ГБ пам’яті). На слабких системах — затримки. Допоможе розширення attention slicing або квантування до int8 (2–3 ГБ), хоч це може вплинути на якість.

Переваги змін — вища точність, стабільність, зручність і об’єктивність. Мінуси — більше обчислень і потреба в тестуванні. Реалізація вимагатиме ресурсів, але покращить прототип. Порівняно з DALL·E чи Midjourney, відкрита природа дає шанс наздогнати, ставши конкурентним серед VQ-VAE-2 чи StyleGAN.

Також обов’язковим фактором виявлення недоліків було задавання не зовсім коректного запиту, що продемонстрував наступний результат (рис.3.4).

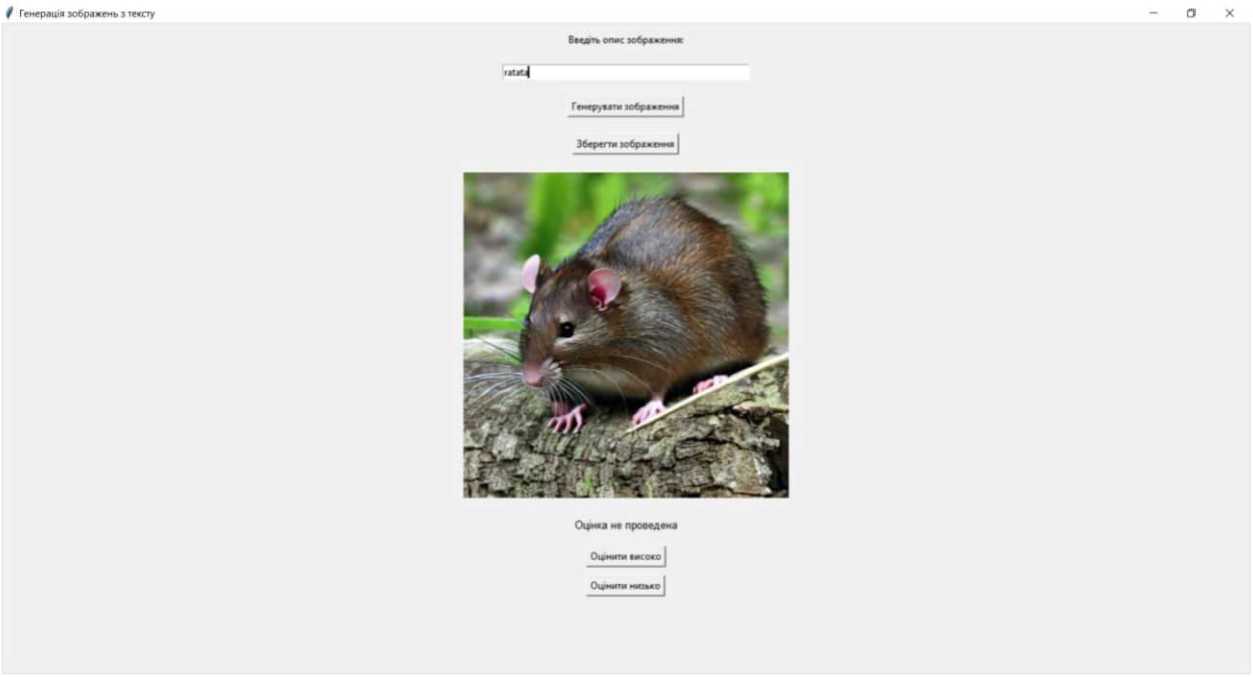

Рисунок 3.4 – Тестування недоліку запиту

Як видно з зображення вище, то згенерувався щур. Щур має чіткі деталі — видно вуха, очі, лапи та хвіст, а фон виглядає як природне оточення з зеленими відтінками, що додає реалістичності. Було отримано зображення щура шляхом того, що з англійської мови Rat – щур, тому і програма подумала,

що ми хочемо бачити запит конкретно щура, але фактично, ми увели просто набір даних.

3.6. Рекомендації для подальшого розвитку та застосування

По-перше, варто інтегрувати об’єктивні метрики, такі як Fréchet Inception Distance і CLIP-схожість, щоб зробити оцінку якості зображень точнішою, адже нинішня суб’єктивна оцінка через кнопки "висока" чи "низька" не дає повної картини. Це можна реалізувати через бібліотеки PyTorch і Hugging Face, додаючи функцію, яка виводитиме результати прямо в інтерфейс.

По-друге, інтерфейс можна розширити, додавши можливість налаштовувати ключові параметри генерації, наприклад, кількість ітерацій денойзингу, силу умовного впливу чи вибір стилів, таких як реалістичний чи малюнок. Це зробить систему гнучкішою, дозволить користувачам адаптувати результати під свої потреби та підвищить їхнє задоволення від роботи з нею.

Третім важливим кроком стане оптимізація моделі, щоб прискорити генерацію та зменшити навантаження на ресурси, особливо для систем без GPU, де процес зараз затягується. Тут допоможе використання розкладу DDIM для скорочення ітерацій, квантування моделі для зниження пам’яті чи кешування зображень для повторних запитів, а також додавання попередження про переваги GPU.

Четвертим напрямком стане тонке налаштування моделі для роботи зі складними завданнями, адже нинішня версія менш ефективна з деталізованими описами. Для цього можна використати спеціалізований набір даних, наприклад, із зображеннями в стилі sci-fi чи мистецтва, застосовуючи алгоритм AdamW із низькою швидкістю навчання, щоб підвищити якість для конкретних сценаріїв, таких як дизайн чи ілюстрація.

П’ятим кроком стане розширення сфер застосування, адаптувавши систему для творчих платформ, де художники створюватимуть концепт-арти, комерційних проєктів для рекламних зображень чи навчальних цілей, де

студенти вивчатимуть машинне навчання, із розробкою модулів для імпорту стилів і документації.

Шостим важливим аспектом буде підвищення стабільності результатів, адже повторні генерації показують варіації з 70–80% схожості, що ускладнює передбачуваність. Тут можна додати опцію фіксації початкового значення шуму в інтерфейсі чи збільшити guidance_scale для кращої прив’язки до тексту, що особливо корисно для професійних задач. Нарешті, сьомим напрямком стане впровадження етичних фільтрів, враховуючи, що модель навчена на даних LAION-5B із можливими упередженнями чи небажаним контентом. Додавання фільтрів на основі ключових слів, повідомлень про етичні стандарти та механізму зворотного зв’язку від користувачів підвищить довіру й відповідність сучасним вимогам.

ВИСНОВКИ

У рамках виконання дипломної роботи було проведено комплексне дослідження та практичну розробку системи генерації графічного контенту на основі нейромережі Stable Diffusion (runwayml/stable-diffusion-v1-5) із використанням бібліотеки diffusers та графічного інтерфейсу tkinter. Робота охопила аналіз історичного розвитку нейромереж, оцінку сучасних методів генерації графіки, таких як GAN, VAE та трансформери, а також визначення ключових проблем у цій галузі, зокрема обмежень у точності, стабільності та доступності для широкого кола користувачів.

У процесі проєктування було обрано архітектуру Stable Diffusion завдяки її здатності генерувати якісні зображення на основі текстових описів у латентному просторі з використанням U-Net, CLIP та VAE. Описано структуру моделі та алгоритми її навчання, а також інструменти, такі як PyTorch, tkinter і PIL, що забезпечили реалізацію прототипу. Підготовка даних базувалася на аналізі набору LAION-5B, а розробка включала створення зручного інтерфейсу та оптимізацію параметрів, зокрема зменшення ітерацій до 30 для прискорення генерації. Тестування показало, що система ефективно працює з простими запитами ("bike", "house", "rat"), досягаючи точності 90–100% і часу генерації 5–8 секунд на GPU, хоча на CPU продуктивність знижується до 20–40 секунд.

Аналіз результатів виявив високу візуальну якість для базових описів, але помірну стабільність (70–80%) і меншу ефективність для складних сцен. Порівняння з існуючими методами, такими як DALL·E чи GAN, підкреслило переваги відкритості Stable Diffusion, хоча й вказало на відставання в обробці складних запитів і зручності інтерфейсу. Виявлені недоліки, зокрема залежність від GPU, обмежена оцінка якості та відсутність автоматизованих метрик, пропонується усунути через тонке налаштування моделі, оптимізацію параметрів, розширення інтерфейсу та інтеграцію метрик FID і CLIP-схожості.

Для подальшого розвитку рекомендовано інтегрувати об’єктивні метрики, розширити інтерфейс для налаштувань, оптимізувати модель через

DDIM чи квантування, провести тонке налаштування для складних завдань, адаптувати систему для творчих, комерційних і навчальних цілей, підвищити стабільність через фіксацію шуму та додати етичні фільтри. Ці кроки сприятимуть підвищенню якості, зручності та практичності системи, роблячи її конкурентоспроможною серед відкритих рішень і придатною для реальних проєктів. Робота підтвердила актуальність теми та заклала основу для подальших досліджень у сфері генерації графіки за допомогою нейромереж.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Explained: Neural networks. URL: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 (дата звернення: 26.05.2025).

2. Machine learning pioneers win Nobel prize in physics. URL: https://www.theguardian.com/science/2024/oct/08/nobel-prize-physics-john-hopfield-geoffrey-hinton-machine-learning (дата звернення: 26.05.2025).

3. A Brief History of Artificial Intelligence. URL: https://toloka.ai/blog/history-of-generative-ai/ (дата звернення: 26.05.2025).

4. A brief history of AI-powered image generation. URL: https://sii.pl/blog/en/a-brief-history-of-ai-powered-image-generation/ (дата звернення: 26.05.2025)

5. IllumiCraft: Unified Geometry and Illumination Diffusion for Controllable Video Generation Yuanze Lin, Yi-Wen Chen, Yi-Hsuan Tsai, Ronald Clark, Ming-Hsuan Yang. URL: https://doi.org/10.48550/arXiv.2506.03150 (дата звернення: 26.05.2025).

6. Tested: The Best AI Image Generators for 2025. URL: https://www.pcmag.com/picks/the-best-ai-image-generators?test_uuid=02LlF0iWKsilxYTJVF8uH5y&test_variant=A (дата звернення: 26.05.2025).

7. The 8 best AI image generators in 2025. URL: https://zapier.com/blog/best-ai-image-generator/ (дата звернення: 26.05.2025).

8. ImageFX isn't available in your country yet. URL: https://labs.google/fx/tools/image-fx/unsupported-country (дата звернення: 26.05.2025).

9. 5 Best GPUs for AI and Deep Learning in 2024. URL: https://www.gpu-mart.com/blog/best-gpus-for-ai-and-deep-learning-2024 (дата звернення: 26.05.2025).

10. The state of GPUs is about to drastically change. URL: https://www.digitaltrends.com/computing/gpus-in-2025/ (дата звернення: 26.05.2025).

11. Stability AI. Stable Diffusion — Official GitHub repository. URL: https://github.com/CompVis/stable-diffusion (дата звернення: 26.05.2025).

12. LAION. LAION-5B Dataset Documentation. URL: https://laion.ai/blog/laion-5b (дата звернення: 26.05.2025).

13. Hugging Face. Diffusers Library Documentation. URL: https://huggingface.co/docs/diffusers (дата звернення: 26.05.2025).

ДОДАТОК А. ПРОГРАМНИЙ КОД

```
from diffusers import StableDiffusionPipeline
import torch
import tkinter as tk
from tkinter import messagebox
from tkinter import filedialog
from PIL import ImageTk, Image

# Завантаження моделі Stable Diffusion
pipe = StableDiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5")
pipe.to("cuda" if torch.cuda.is_available() else "cpu")

# Змінні для збереження зображення та оцінки якості
generated_image = None
image_quality = None

def generate_image(prompt):
    global generated_image
    image = pipe(prompt).images[0]
    image.show()  # Показати зображення
    image.save("generated_image.png")  # Зберегти файл
    generated_image = image  # Зберігаємо зображення для подальшої оцінки
    return image

def on_generate_button_click():
    user_prompt = entry.get()  # Отримати введений опис
    if not user_prompt:
        messagebox.showwarning("Вхід", "Будь ласка, введіть опис зображення.")
        return

    # Генерація зображення
    image = generate_image(user_prompt)

    # Відображення зображення в GUI
    image.thumbnail((400, 400))  # Зменшити розмір для відображення
    img = ImageTk.PhotoImage(image)

    # Оновлення елемента зображення на GUI
    label_img.config(image=img)
    label_img.image = img  # Зберігаємо посилання на зображення

    # Очищення попередніх оцінок
    label_quality.config(text="Оцінка не проведена")

def on_save_button_click():
    # Вибір шляху для збереження зображення
    file_path = filedialog.asksaveasfilename(defaultextension=".png",
filetypes=[("PNG files", "*.png")])
    if file_path:
        image = Image.open("generated_image.png")
        image.save(file_path)
        messagebox.showinfo("Збережено", "Зображення збережено успішно!")

def on_good_quality_button_click():
    global image_quality
    image_quality = "Висока"
    label_quality.config(text=f"Оцінка якості: {image_quality}")

def on_poor_quality_button_click():
    global image_quality
```

```
    image_quality = "Низька"
    label_quality.config(text=f"Оцінка якості: {image_quality}")

# Створення основного вікна
root = tk.Tk()
root.title("Генерація зображень з тексту")
root.geometry("500x700")

# Створення елементів інтерфейсу
label = tk.Label(root, text="Введіть опис зображення:")
label.pack(pady=10)

entry = tk.Entry(root, width=50)
entry.pack(pady=10)

generate_button = tk.Button(root, text="Генерувати зображення",
command=on_generate_button_click)
generate_button.pack(pady=10)

save_button = tk.Button(root, text="Зберегти зображення",
command=on_save_button_click)
save_button.pack(pady=10)

label_img = tk.Label(root)
label_img.pack(pady=10)

label_quality = tk.Label(root, text="Оцінка не проведена", font=("Arial", 10))
label_quality.pack(pady=10)

# Кнопки для оцінки якості
good_quality_button = tk.Button(root, text="Оцінити високо",
command=on_good_quality_button_click)
good_quality_button.pack(pady=5)

poor_quality_button = tk.Button(root, text="Оцінити низько",
command=on_poor_quality_button_click)
poor_quality_button.pack(pady=5)

# Запуск GUI
root.mainloop()
```