Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ ЗОБРАЖЕНЬ ЗА ЇХНІМ ТЕКСТОВИМ ОПИСОМ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Головін Е.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Любченко В.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Головіну Еріку Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мобільного застосунку для створення зображень за їхнім текстовим описом

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 29 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, пакетний менеджер Swift Packages, бібліотека Foundation, бібліотека UIKit, мова програмування Swift, мова програмування SwiftUI, середовище розробки Xcode.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз проблеми створення зображень.
2. Аналіз існуючих технічних засобів для генерації зображень.
3. Створення методології розробки мобільного застосунку з можливістю генерації зображень за текстовим описом.
4. Розробка та тестування мобільного застосунку для генерації зображень.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми створення зображень за текстовим описом, постановка задачі, сучасні методи створення зображень, алгоритмічна модель створення зображень, методологія розробки мобільного застосунку для створення зображень, результати роботи застосунку.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-20.04.23	
3	Аналіз літератури з досліджуваної проблеми	21.04.23-25.04.23	
4	Аналіз технічних засобів	26.04.23-28.04.23	
5	Створення методології розробки	29.04.23-01.05.23	
6	Програмна реалізація	02.05.23-07.05.23	
7	Оформлення пояснювальної записки	08.05.23-31.05.23	
8	Перевірка на плагіат	01.06.23	
9	Рецензування	02.06.23	
10	Підготовка презентації та доповіді	03.06.23-06.06.23	
11	Занесення роботи в електронний архів	07.06.23	
12	Попередній захист кваліфікаційної роботи	07.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Любченко В.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 78 с., 2 табл., 47 рис., 2 дод., 39 джерел.

TEXT2IMG, ГЕНЕРАЦІЯ ЗОБРАЖЕНЬ, STABLE DIFFUSION, DALL-E, MIDJOURNEY, МОБІЛЬНИЙ ЗАСТОСУНОК, IOS.

Об’єктом роботи є Stable Diffusion алгоритм, за допомогою якого відбувається генерація зображень за описом.

Метою роботи є розробка iOS застосунку із застосуванням стабільної дифузійної нейронної мережі, спрямованого на генерацію зображень за їх текстовим описом. Вхідні дані включають в себе різноманітні варіації текстових описів зображень.

Проведено дослідження методів генерації зображень на основі нейронних мереж. Досліджено можливості нейронних моделей SDNN, особливості роботи моделі DALL-E та моделі Midjourney. Проведено дослідження методів створення мобільних застосунків на основі нейронної мережі SDNN.

У результаті роботи отримано iOS застосунок для створення зображень за їхнім текстовим описом.

TEXT2IMG, GENERATION OF IMAGES, STABLE DIFFUSION, DALL-E, MIDJOURNEY, MOBILE APPLICATION, IOS.

The object of the work is the Stable Diffusion algorithm, which is used to generate images according to the description.

The purpose of the work is to develop an iOS application using a stable diffusion neural network aimed at generating images based on their text description. The input data includes various variations of the textual descriptions of the images.

A study of image generation methods based on neural networks was conducted. The capabilities of SDNN neural models, the features of the DALL-E model and the Midjourney model have been studied. A study of methods for creating mobile applications based on the SDNN neural network was conducted.

As a result of the work, an iOS application for creating images based on their text description was obtained.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 6
Вступ 7
1 Аналіз проблеми створення зображень 8
1.1 Проблематика створення зображень за текстовим описом 9
1.2 Аналіз існуючих методів створення зображень 13
1.3 Нейронні мережі для роботи з зображеннями 14
1.4 Постановка задачі 17
2 Використання нейронних мереж для генерації зображень в iOS застосунках 19
2.1 Алгоритмічна модель для створення зображень за їх текстовим описом 19
2.1.1 Загальні принципи роботи алгоритму Stable Diffusion 19
2.1.2 Кодування текстового запиту 21
2.1.3 Створення інформації про зображення 23
2.1.4 Декодування зображення 25
2.2 Оптимізація гіперпараметрів для кращої продуктивності 28
2.3 Методологія розробки мобільного застосунку з використанням технології Stable Diffusion 31
3 Проєктування програмного продукту 34
3.1 Етапи програмної реалізації iOS застосунку 34
3.1.1 Модель DALL-E API 34
3.1.2 Модель Midjourney API 36
3.1.3 Впроваження API у застосунок 39
3.2 Розробка сценарію взаємодії користувача із застосунком 41
3.3 Тестування розробленого застосунку та аналіз результатів 46
3.4 Перспективи подальшої роботи 54
Висновки 56
Перелік джерел посилання 57
Додаток А Приклади коду застосунку 61
Додаток Б Знімки екранів застосунку 64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

text2img – text-to-image (текст в зображення)

2D – two-dimensional graphics (двовимірна графіка)

3D – three-dimensional graphics (тривимірна графіка)

CNN – Convolutional Neural Network (згорточна нейронна мережа)

RNN – Recurrent Neural Network (рекурентна нейронна мережа)

GAN – Generative Adversarial Network (генеративна змагальна мережа)

SDNN – Stable Diffusion Neural Network (стабільно-дифузійна нейронна мережа)

BERT – Bidirectional Encoder Representations from Transformers (представлення двонаправленого кодера від трансформаторів)

IIC – Image Information Creator (творець інформації про зображення)

API – Application Programming Interface (програмний інтерфейс застосунка)

VQ-VAE – Vector Quantized Variational AutoEncoder (векторний квантований варіаційний автокодер)

LLM – Large Language Model (велика мовна модель)

REST – Representational State Transfer (передача представницького стану)

HTTP – Hypertext Transfer Protocol (протокол передачі гіпертексту)

UI – User Interface (інтерфейс користувача)

img2img – image-to-image (зображення в зображення)

ВСТУП

З кожним днем технології штучного інтелекту все глибше проникають в усі сфери нашого життя. Зокрема з'являється велика кількість сервісів, здатних суттєво вплинути на роботу та бізнес. Сьогодні за допомогою штучного інтелекту можна запускати рекламні кампанії, писати тексти і навіть музику. Особливу увагу надають питанням генерування зображень, зокрема із тексту.

Нейромережа – це штучний інтелект, який працює за принципом людського мозку. Нейрони отримують, обробляють та віддають інформацію, а зв'язки її передають. Головна відмінність нейронів комп'ютерної мережі в тому, що їх треба постійно навчати. Людина ж розширює нейромережу, коли сама вчиться чомусь новому.

Останніми роками комп'ютерні нейромережі набули великого розвитку. Здебільшого їх використовують для завдань, де потрібно обробити текст, відео, аудіо чи іншу інформацію. Особливої популярності набули нейромережі, що здатні швидко генерувати зображення з текстового запиту, поєднувати графічні об'єкти чи відтворювати відсутні елементи.

Перетворення тексту на зображення – це процес створення реалістичного зображення, яке відповідає заданому текстовому опису та вимагає обробки нечіткої та неповної інформації в описах природною мовою.

Нейромережі та сервіси на їхній базі – технологія, яка змінює світ вже зараз. Останнім часом техгіганти активно інвестують в їх розвиток, а суспільство дискутує про те, які спеціальності штучний інтелект замінить швидше.

Мобільні застосунки для створення зображень за їхнім тектовим описом створюють унікальні зображення, які раніше не існували. Їх можна використовувати для фото-ілюстрацій статей, у копірайтингу та в рекламі. Вони не замінюють роботу дизайнерів та художників, але допомагають оптимізувати рутинні процеси.

1 АНАЛІЗ ПРОБЛЕМИ СТВОРЕННЯ ЗОБРАЖЕНЬ

Генерація зображень з тексту (text2img) – це завдання створення реалістичних зображень із текстових описів. Text2img є складною проблемою, яка потребує поєднання передових методів машинного навчання та глибокого розуміння природної мови та візуального сприйняття.

Однією з перших нейронних мереж, що спеціалізується на створенні зображень є ImageNet, розроблена у 2009 році [1]. Вона не покривала вирішення задачі створення унікальних артів, а скоріше виконувала завдання пошуку зображення у базі даних за назвою.

Широкого використання text2img набула після 2020 року. Була створена велика кількість різноманітних моделей, які дозволяють виконувати завдання генерації зображень у 2D та 3D форматах.

Таким чином, технологія text2img може широко використовуватись у рекламному бізнесі. Вона дозволить значно скоротити витрати на покупку фотографій та послуги дизайнерів. Приклад створення рекламного зображення фітнес-застосунку показано на рисунку 1.1.

Рисунок 1.1 – Результат створення зображення за запитом «реклама фітнес-застосунку»

1.1 Проблематика створення зображень за текстовим описом

Незважаючи на значний прогрес у сфері генерації зображень за останні роки, все ще є кілька проблем, які необхідно вирішити.

Однією з основних проблем є неоднозначність описів. Текстові описи можуть бути дуже суб'єктивними та тлумачитися різними способами. Наприклад, фразу «жінка в червоній сукні» можна інтерпретувати по-різному, і від використаної інтерпретації буде залежати кінцевий образ (рис. 1.2). Ця неоднозначність може ускладнити моделям text2img створення точних і послідовних зображень.

Рисунок 1.2 – Результат створення зображення за запитом «жінка в червоній сукні»

Ще одна проблема – складність природних сцен. Реальні зображення часто містять багато різних об’єктів, текстур та умов освітлення, і моделям text2img може бути важко зафіксувати всі ці деталі. Наприклад, створення зображення парку з багатьма різними видами дерев, трави та іншої рослинності може бути складним завданням для моделі text2img (рис. 1.3).

Рисунок 1.3 – Результат створення зображення за запитом «парк з великою кількістю дерев та квітами»

Крім того, моделі text2img також можуть мати проблеми зі створенням зображень із дрібними деталями, такими як текстури та дрібні об’єкти (рис. 1.4). Це пояснюється тим, що ці деталі часто важко представити за допомогою простих векторних зображень [2], які зазвичай використовуються як вхідні дані для моделей text2img.

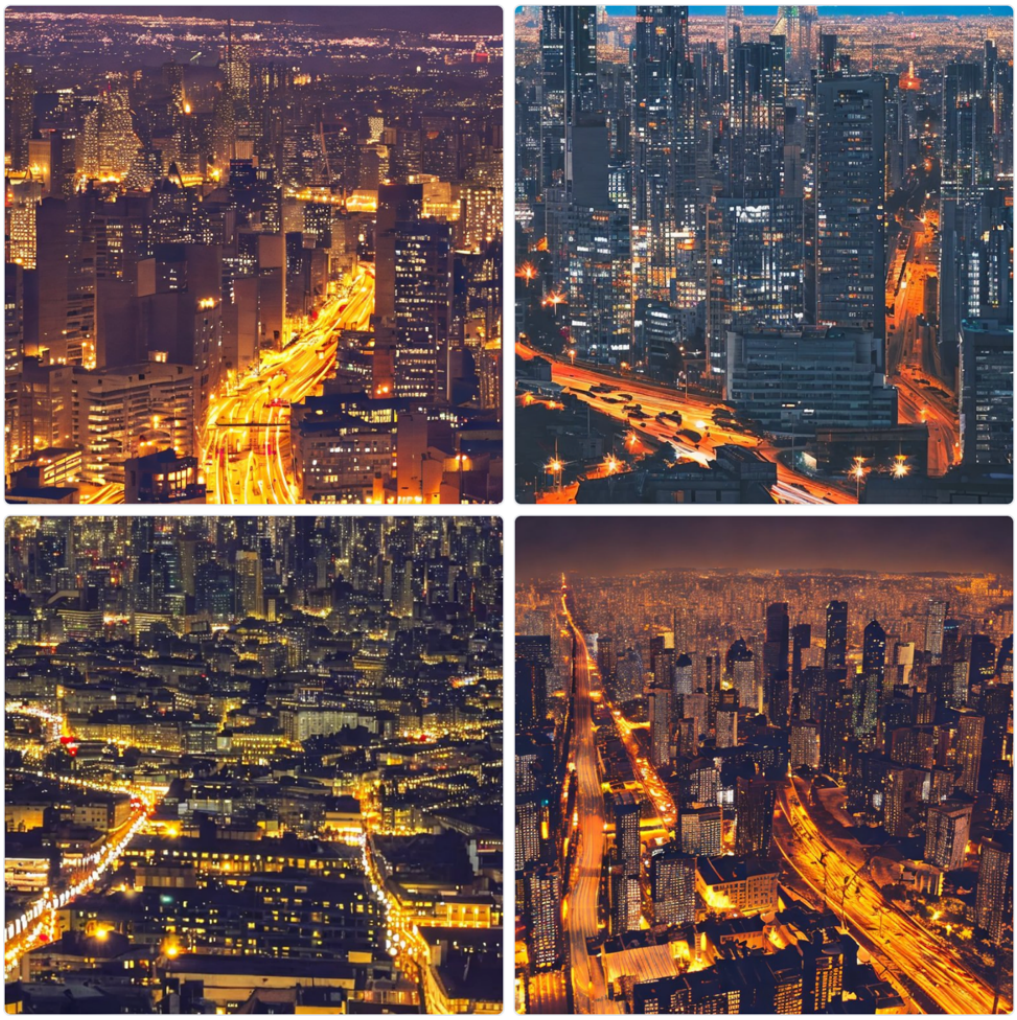

Рисунок 1.4 – Результат створення зображення за запитом «вид на місто з великою кількістю маленьких ліхтарів»

Нарешті, найбільшою проблемою постає створення зображень людей [3]. Цю проблему можна поділити на декілька категорій: створення обличь та людські кінцівки.

Обличчя людини має внутрішню та повну семантичну логіку, яку не можна знайти на цеглі будівлі або інших елементах, які мають майже завжди подібну логіку відтворення. Приклад вдалої та невдалої генерації зображень обличь відомих людей показано на рисунку 1.5.

Кінцівки також постають великою проблемою для нейронної мережі. Руки можуть розмножуватися випадковим чином, пальці зливаються, треті ноги з’являються, а існуючі кінцівки зникають без сліду. Проблему створення людських кінцівок показано на рисунку 1.6 та на рисунку 1.7.

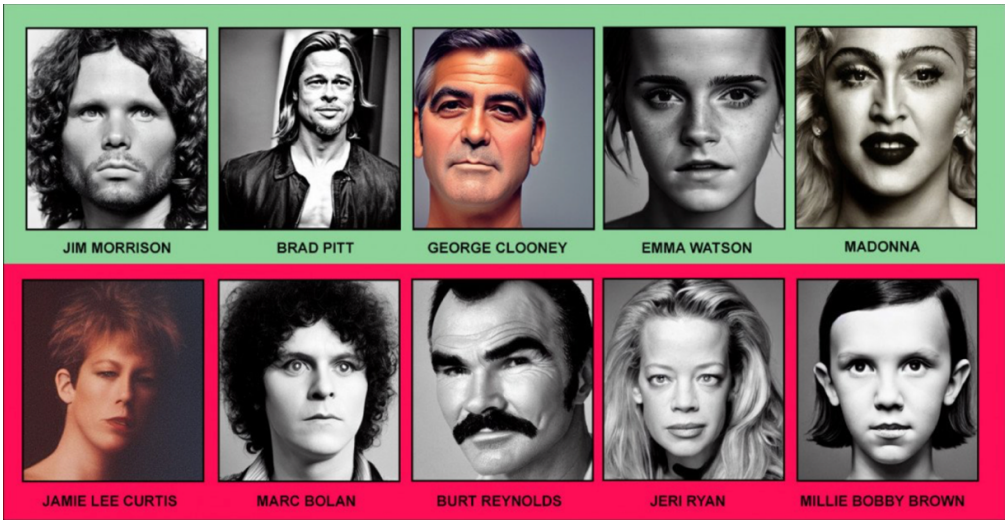

Рисунок 1.5 – Створені нейронною мережею обличчя відомих людей

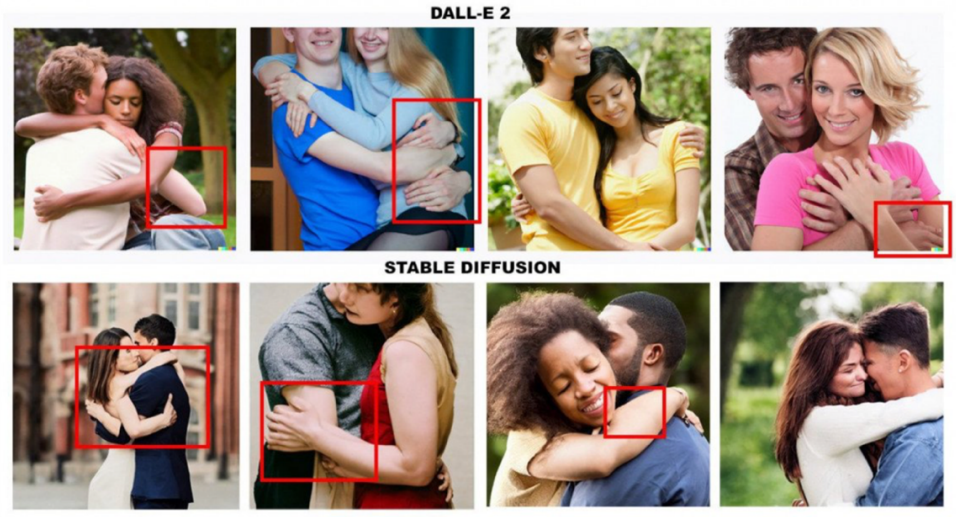

Рисунок 1.6 – Проблема створення зображень людських кінцівок

Рисунок 1.7 – Проблема створення зображень з декількома людьми та їх кінцівками

1.2 Аналіз існуючих методів створення зображень

Стосовно методології створення зображень було проведено ряд досліджень. Фаріба Йошефі, Женвен Даі, Карл Генрік Ек та Неіл Лавренсе запропонували нову архітектуру нейронної мережі – глибоку згорткову генеративно-змагальну мережу (Deep Convolutional Generative Adversarial Network) [4]. Основною ідеєю даної архітектури є використання шарів згортки, як у генеративній, так і в дискримінативній моделі. Дана модифікація дозволяє зробити навчання генеративно-змагальної мережі більш стійким. У роботі автори демонструють, що цей алгоритм добре показує себе у задачі генерації зображень, у тому числі людей.

Дідерік Кінгма та Макс Веллінг описують тип нейронної мережі – варіаційні автоенкодери [5]. Автоенкодери відрізняються тим, що кількість нейронів на вході у них збігається з кількістю на виході. Суть їх роботи полягає у стисненні вхідних даних та відображенні їх у прихований простір (latent-space), з подальшим відновленням, з метою одержати дані як найбільш близькі до вхідних. Недоліком цього підходу є недостатня якість одержаних результатів – зображення часто виходять розмитими.

В сервісі Illustration2Vec [6] представлено архітектуру нейронної мережі Deep Recurrent Attentive Writer (DRAW) [7]. Запропонована авторами статті нейронна мережа є різновидом варіаційного автоенкодера. Відмінною особливістю є поетапний процес генерації зображення, схожий на процес малювання в реальності.

Янгуа Джин, Джиакаі Занг, Мінджун Лі та інші використовують Deep Regret Analytic Generative Adversarial Networks [8]. Також вони розробили сервіс AnimeGAN [9]. Він дозволяє створювати картинки розмірністю 256×256, а також дає вибір із 12 параметрів, призначених для персоналізації зображення.

Сервіс Diffuse The Rest дозволяє малювати картину і за допомогою текстової підказки створює високоякісне реалістичне мистецтво [10].

Сервіс Latent Diffusion – це ще одна версія text2img [11]. Застосунок надає більше можливостей конфігурації, таких як розмір зображення, якість зображення та масштаб різноманітності. Для початку використання потрібно написати текстову підказку для генерації зображень.

1.3 Нейронні мережі для роботи з зображеннями

Нейронні мережі для обробки зображень – це тип алгоритму глибокого навчання, який особливо ефективний у розпізнаванні шаблонів на зображеннях. Ці мережі зазвичай складаються з кількох шарів штучних нейронів, які навчаються на великих наборах даних позначених зображень, щоб навчитися розпізнавати різні об’єкти, форми та особливості на зображеннях.

Існує кілька типів нейронних мереж, які зазвичай використовуються для роботи з зображеннями.

Згорткові нейронні мережі (CNN) – є одним із найпоширеніших типів нейронних мереж, які використовуються для обробки зображень [12]. Вони особливо ефективні в розпізнаванні просторових моделей на зображеннях, таких як краї, текстури та форми. Схема роботи згорткової нейронної мережі наведена на рисунку 1.8.

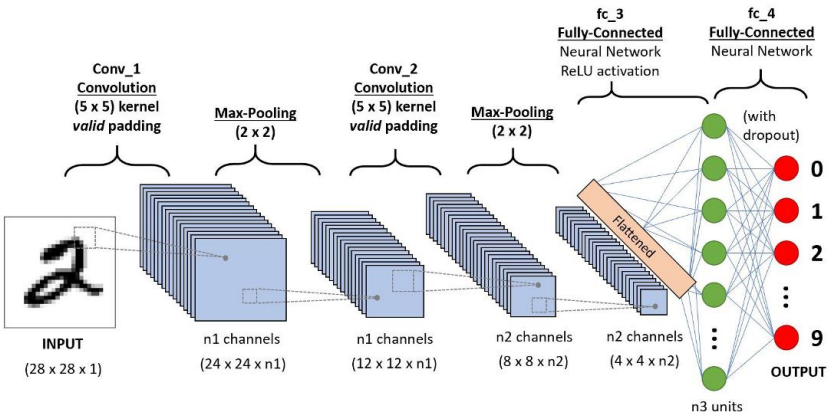

Рисунок 1.8 – Схема роботи CNN

CNN використовують згорткові шари для застосування набору навчених фільтрів до вхідного зображення, яке витягує функції із зображення, які потім пропускаються через серію повністю пов’язаних шарів, щоб зробити прогноз.

Рекурентні нейронні мережі (RNN) – це ще один тип нейронної мережі, який можна використовувати для обробки зображень [13]. Вони особливо ефективні при обробці послідовних даних, таких як дані часових рядів або відеокадри. RNN використовують повторюваний цикл зворотного зв’язку, щоб дозволити інформації перетікати від одного часового кроку до наступного, що робить їх добре придатними для обробки послідовностей зображень, що змінюються в часі. Схему роботи рекурентної нейронної мережі наведено на рисунку 1.9.

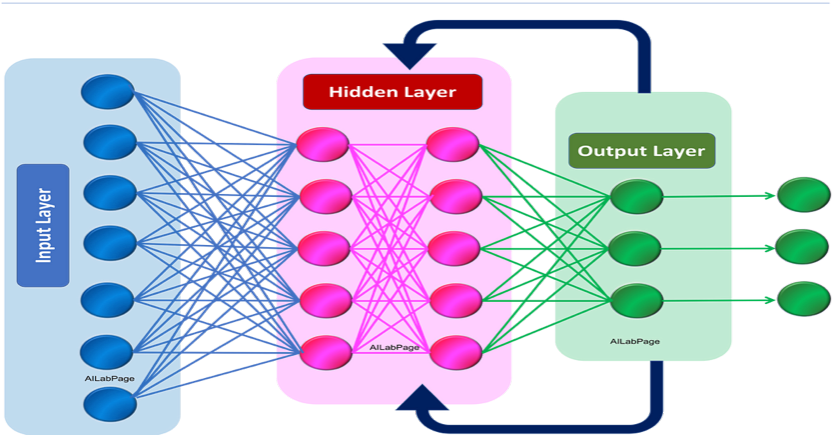

Рисунок 1.9 – Схема роботи RNN

Генеративні змагальні нейронні мережі (GAN) – це тип нейронної мережі, який можна використовувати для створення зображень і маніпулювання ними [14]. Вони складаються з двох нейронних мереж: мережі генератора, яка створює нові зображення, і мережі дискримінатора, яка намагається відрізнити справжні зображення від підроблених. Дві мережі навчаються разом у процесі, який спонукає генератор створювати все більш реалістичні зображення. Принцип роботи генеративної змагальної нейронної мережі наведено на схемі (рис. 1.10).

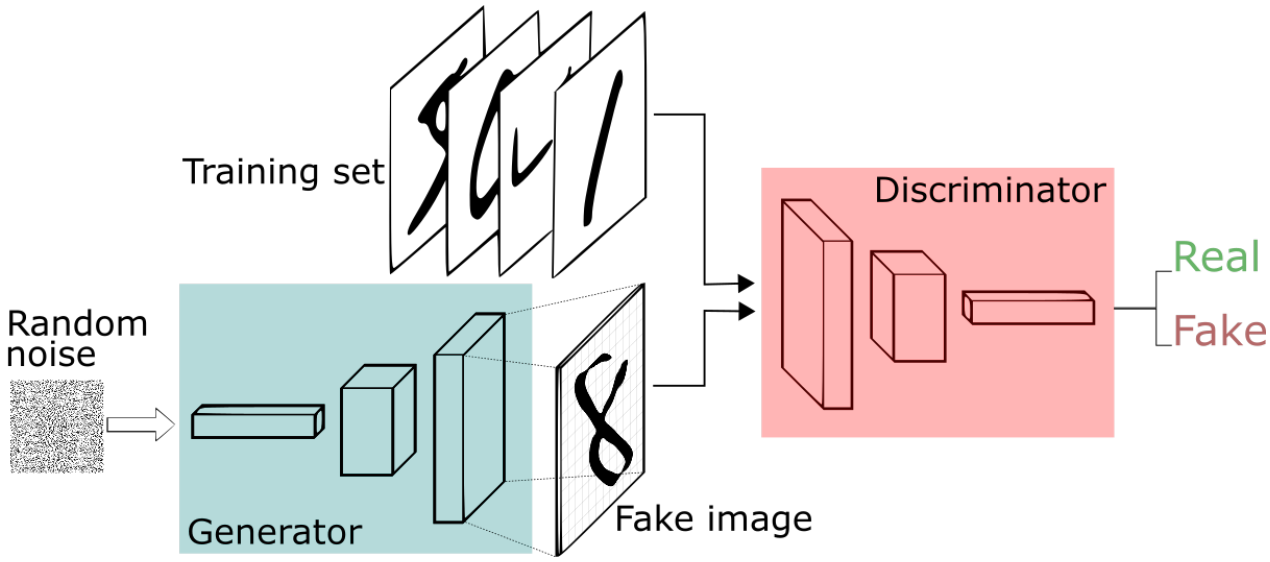

Рисунок 1.10 – Схема роботи GAN

Стабільні дифузійні нейронні мережі (SDNN) – це тип нейронних мереж, які розроблені для навчання з даних великої розмірності, таких як зображення чи аудіо [15]. SDNN базуються на принципах процесів дифузії, які є математичними моделями, які описують, як частинки поширюються з часом.

Ключова ідея SDNN полягає у використанні процесу дифузії для створення серії проміжних представлень вхідних даних, які охоплюють різні рівні абстракції. Кожне проміжне представлення отримується шляхом поширення попереднього представлення на набір вивчених операторів дифузії. Остаточне представлення потім отримується шляхом дифузії останнього проміжного представлення, доки воно не досягне стабільного стану.

SDNN використовувалися для різноманітних програм, включаючи класифікацію зображень, аналіз аудіо та обробку природної мови. Вони також показали свою ефективність для трансферного навчання, яке передбачає використання попередньо навченої мережі для завдання, відмінного від того, для якого вона була навчена спочатку.

Схему роботи SDNN наведено на рисунку 1.11.

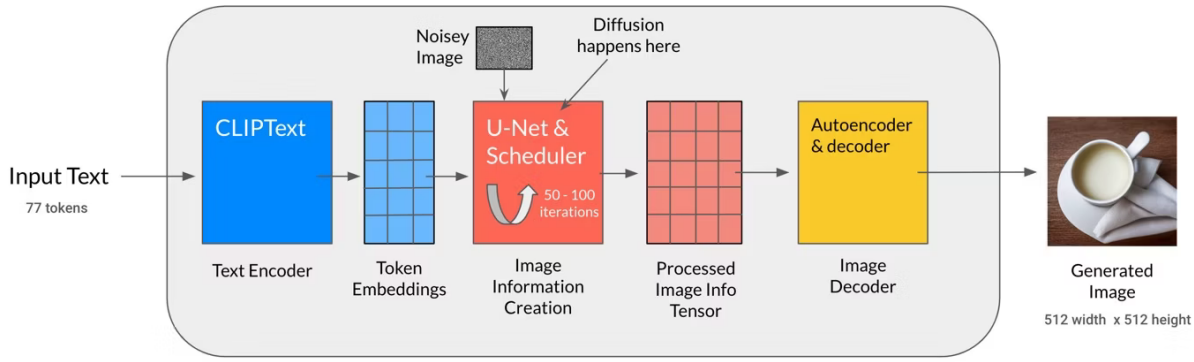

Рисунок 1.11 – Схема роботи SDNN

Нейронні мережі для обробки зображень мають численні застосування, включаючи розпізнавання об’єктів, виявлення обличь, сегментацію та генерацію зображень. Вони, також, використовувалися в різних галузях промисловості, таких як охорона здоров’я, автомобільна промисловість і роздрібна торгівля, для створення нових застосунків і вдосконалення існуючих.

1.4 Постановка задачі

Таким чином, створення зображень за їх текстовим описом, з використанням нейромережевих засобів є актуальним завданням у рекламному бізнесі.

Об’єктом роботи є Stable Diffusion алгоритм, за допомогою якого відбувається генерація зображень за описом.

Метою роботи є розробка iOS застосунку із застосуванням стабільної дифузійної нейронної мережі, спрямованого на генерацію зображень за їх текстовим описом. Вхідні дані включають в себе різноманітні варіації текстових описів зображень.

Для досягнення мети необхідно вирішити такі завдання:

– обрати нейронну мережу для генерації зображень;

– провести аналіз роботи алгоритму для генерації зображень;

– навести детальний опис етапів програмної реалізації iOS застосунку для генерації зображень;

– провести тестування розробленого застосунку та проаналізувати результати;

– виявити перспективи подальшої роботи;

– зробити висновки щодо виконаної роботи.

2 ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ В IOS ЗАСТОСУНКАХ

2.1 Алгоритмічна модель для створення зображень за їх текстовим описом

2.1.1 Загальні принципи роботи алгоритму Stable Diffusion

Модель Stable Diffusion підтримує можливість генерувати нові зображення з нуля за допомогою текстової підказки, що описує елементи, які слід включити або виключити з результату. Існуючі зображення можуть бути перемальовані моделлю, щоб включити нові елементи, описані текстовою підказкою за допомогою механізму дифузійного усунення шуму. Крім того, модель також дозволяє використовувати підказки для часткової зміни існуючих зображень шляхом замальовування, якщо використовується з відповідним інтерфейсом користувача, який підтримує такі функції.

Синтез зображень або генерація зображень досяг значного прогресу за останні роки [16]. Дифузійна модель заснована на нерівноважній статистичній фізиці, по суті, полягає в систематичному та повільному руйнуванні структури розподілу даних через ітераційний процес прямої дифузії. Потім починається процес зворотного розповсюдження, який відновлює структури в даних, створюючи дуже гнучку та зручну генеративну модель даних.

На рисунку 2.1 верхній рядок показує часові зрізи прямої траєкторії на 2-D даних швейцарського крену [17]. Розподіл даних (ліворуч) зазнає гауссової дифузії, яка поступово перетворює його на ідентичність-коваріацію (праворуч). Нижній рядок ілюструє часові зрізи від навченої зворотної траєкторії. Ідентифікаційно-коваріаційний гаусс проходить процес гауссової дифузії з вивченим середнім і коваріаційними функціями та поступово перетворюється назад у розподіл даних (ліворуч).

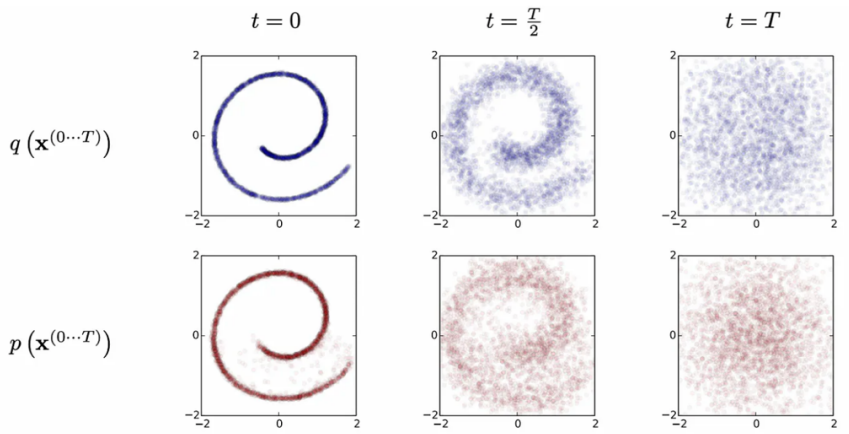

Рисунок 2.1 – Модель розподілу даних

Імовірнісні моделі дифузії є параметризованим ланцюгом Маркова (рис. 2.2), навченим за допомогою варіаційного висновку для отримання вибірок, що відповідають даним через кінцевий час. Переходи цих ланцюгів вивчаються для зворотного процесу дифузії, який є ланцюгом Маркова [17], який поступово додає шум до даних у протилежному напрямку вибірки доки сигнал не знищено.

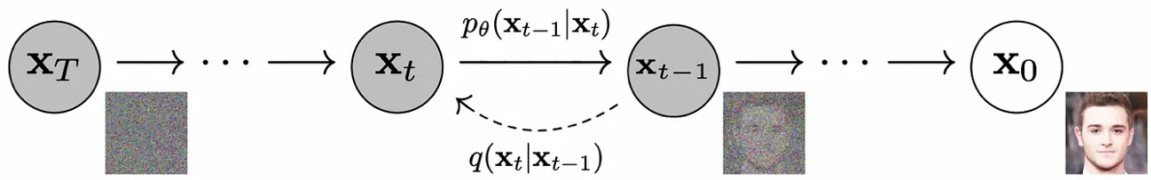

Рисунок 2.2 – Схема роботи ланцюга Маркова

Коли дифузія складається з невеликої кількості гауссівського шуму, достатньо встановити переходи ланцюга вибірки також на умовні гауссіани, що дозволяє особливо просту параметризацію нейронної мережі. Спрощена цільова функція:

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}[\| \epsilon - \epsilon_\theta(x_t, t) \|_2^2], \tag{2.1}$$

де $\epsilon_\theta(x_t, t)$; $t = 1, \dots, T$ – автокодер із шумом, який навчено передбачати знешумлений варіант вхідного сигналу x_t;

x_t – шумовий варіант вхідного сигналу x.

Навчання та оцінка дифузійних моделей вимагають величезних обчислювальних ресурсів, що обмежує доступність і зручність використання такого потужного класу моделей. Як і будь-яка модель правдоподібності, дифузійні моделі, навчені в просторі пікселів, можна грубо розділити на два етапи: перший – це етап перцептивного стиснення, який спрямований на видалення високочастотних деталей, але все ще вивчає невеликі семантичні варіації. А другий етап – етап генеративного моделювання, на якому вивчаються семантичні та концептуальні компоненти даних.

2.1.2 Кодування текстового запиту

Алгоритми кодування тексту в SDNN включають перетворення необроблених текстових даних у векторне представлення [18, 19], яке можна використовувати як вхідні дані для нейронної мережі. Мета кодування тексту – вловити семантичне значення тексту, зберігаючи його структуру та контекст.

Одним із часто використовуваних алгоритмів кодування тексту в SDNN є алгоритм BERT (Bidirectional Encoder Representations from Transformers). BERT – це попередньо навчений алгоритм на основі нейронної мережі, який вивчає векторне представлення слів на основі їх контексту. Це трансформаторна архітектура, яка використовує підхід самоконтролю для попереднього навчання на великих обсягах текстових даних без анотацій.

На високому рівні алгоритм BERT працює, приймаючи послідовність вхідних токенів (слів або підслів), а потім застосовуючи кілька рівнів нейронних мереж для створення контекстно-залежного представлення для

кожного токена. Контекстно-залежне представлення фіксує як значення, так і контекст вхідного токена.

Алгоритм BERT має дві основні цілі навчання: завдання моделювання замаскованої мови та завдання прогнозування наступного речення. Завдання моделювання замаскованої мови передбачає випадкове маскування деяких вхідних токенів, а потім прогнозування замаскованих токенів на основі контексту навколишніх токенів. Наступне завдання прогнозування речень включає в себе передбачення того, чи є два введені речення послідовними чи ні. Схема роботи алгоритму BERT представлено на рисунку 2.3.

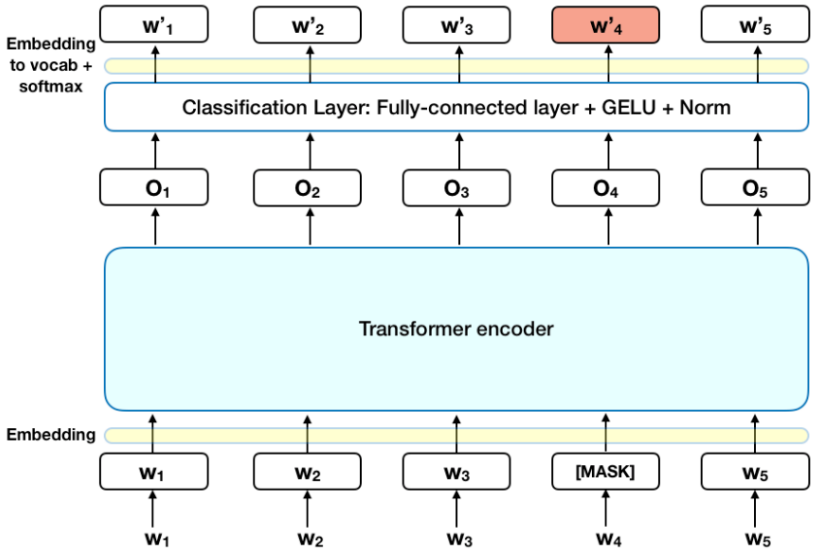

Рисунок 2.3 – Схема роботи алгоритму BERT

Алгоритм BERT працює наступним чином:

Крок 1. Токенізація: вхідний текст токенізується в послідовність підслів або слів.

Крок 2. Вбудовування: кожному токену призначається вектор вбудовування, який фіксує його значення та контекст. Вектори вбудовування вивчаються на етапі попереднього навчання.

Крок 3. Моделювання замаскованої мови: випадково вибрані лексеми маскуються, і алгоритм намагається передбачити замасковані лексеми на основі контексту навколишніх токенів.

Крок 4. Прогноз наступного речення: алгоритм передбачає, чи є два введені речення послідовними чи ні.

Крок 5. Точне налаштування: після попереднього навчання алгоритму BERT на великому масиві текстових даних його можна налаштувати для конкретного завдання, наприклад аналізу настроїв або класифікації тексту.

Алгоритм BERT використовує архітектуру на основі трансформатора, яка є багаторівневою архітектурою нейронної мережі, яка застосовує механізм самоконтролю для обробки вхідних маркерів. Механізм самоконтролю дозволяє моделі зважувати важливість різних вхідних токенів на основі їх контексту у вхідній послідовності.

2.1.3 Створення інформації про зображення

Створення інформації про зображення (Image information creator) є потужним інструментом для створення високоякісних зображень з високим ступенем різноманітності та складності. Схема роботи ІІС представлено на рисунку 2.4 [20].

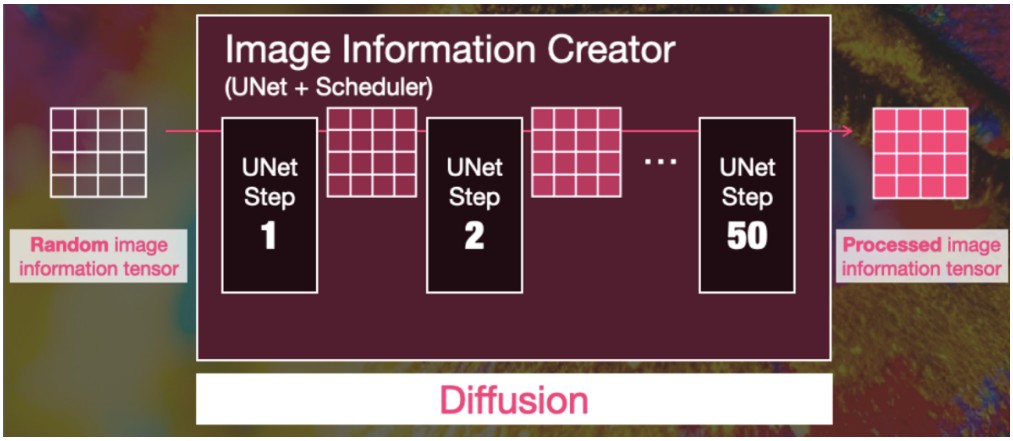

Рисунок 2.4 – Схема роботи ІІС

Алгоритм створення інформації про зображення у SDNN включає наступні кроки:

Крок 1. Ініціалізація. Генератор приймає вектор випадкового шуму як вхідний сигнал. Цей вектор шуму зазвичай складається з розподілу Гауса.

Крок 2. Дифузія. Генератор застосовує ряд кроків дифузії до вхідного шуму.

Кожен крок дифузії складається з лінійного перетворення, за яким слідує нелінійна функція активації, а потім крок нормалізації. Процес дифузії повторюється кілька разів, із кожним кроком збільшуючи рівень деталізації та складності створеного зображення. Процес дифузії починається з випадкового вхідного вектора шуму z розміром N, де N є розмірністю вхідного сигналу. Вектор шуму z перетворюється за допомогою вивченої матриці лінійного перетворення W розміром $N \times N$. Перетворений вектор позначається як h:

$$h = Wz. \tag{2.2}$$

Перетворений вектор h передається через нелінійну функцію активації f:

$$y = f(h). \tag{2.3}$$

Вихідні дані функції активації у потім нормалізуються за допомогою функції нормалізації g:

$$x = g(y). \tag{2.4}$$

Попередні кроки повторюються T разів для створення остаточного вихідного зображення. На кожному кроці t матриця лінійного перетворення W_t, функція активації f_t і функція нормалізації g_t використовуються для генерації вихідного вектора x_t. Після завершення процесу дифузії кінцевий

вихідний вектор x_T пропускається через кінцеву нелінійну функцію активації h:

$$x = h(x_T). \tag{2.5}$$

Метою алгоритму дифузії є створення високоякісних зображень, подібних до навчальних даних. Це досягається оптимізацією параметрів матриці лінійного перетворення W, функції активації f і функції нормалізації g за допомогою навчального набору даних. Оптимізація зазвичай виконується за допомогою зворотного поширення та стохастичного градієнтного спуску, де функція втрат визначається як від'ємна логарифмічна правдоподібність навчальних даних

Крок 3. Постобробка. Після завершення процесу розповсюдження генератор застосовує остаточну нелінійну функцію активації для створення остаточного вихідного зображення. Цей результат є тензором значень пікселів, який представляє згенероване зображення.

Крок 4. Вибірка. Щоб створити кілька зображень, генератор можна запустити кілька разів із різними випадковими введеннями шуму.

У процесі навчання оптимізуються параметри генератора за допомогою зворотного поширення та стохастичного градієнтного спуску. Мета полягає в тому, щоб максимізувати ймовірність генерації навчальних зображень, що зазвичай досягається мінімізацією негативної логарифмічної ймовірності навчальних даних.

2.1.4 Декодування зображення

Алгоритм декодування зображення в SDNN є процесом, зворотним алгоритму дифузії, який використовується для створення зображення. За наявності цільового зображення метою алгоритму декодування є відновлення

відповідного прихованого коду або вхідного вектора шуму, який використовувався для створення зображення (рис. 2.5) [20].

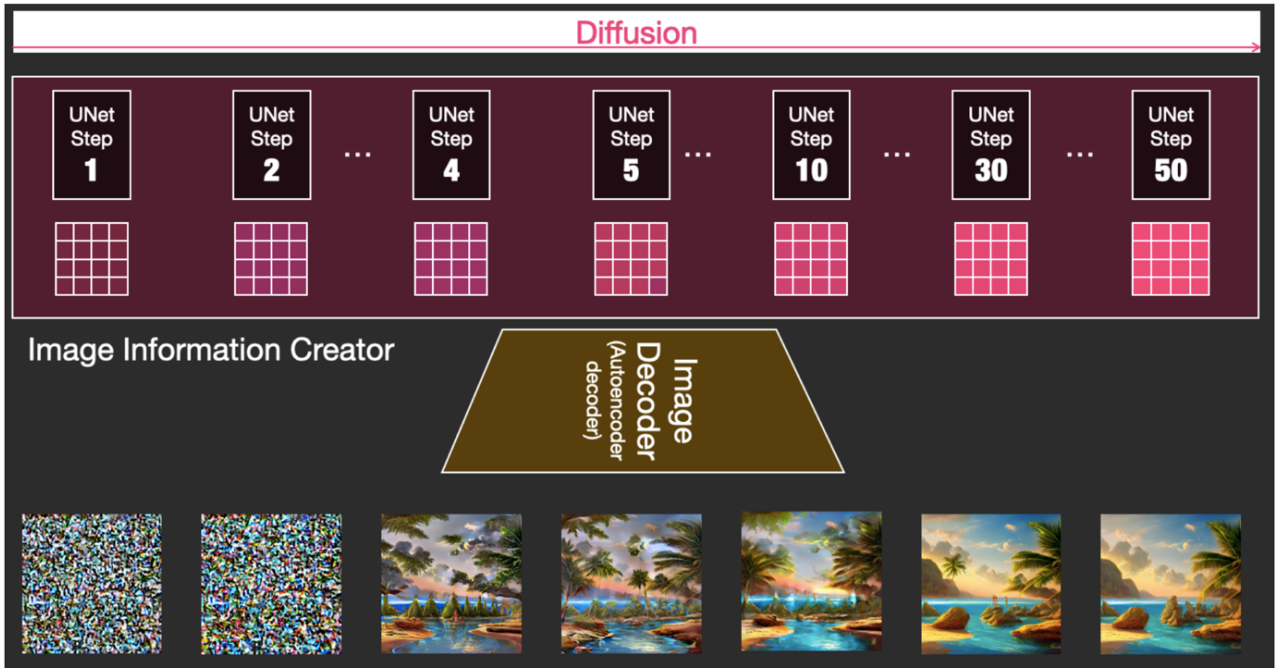

Рисунок 2.5 – Схема роботи декодера

Цього можна досягти за допомогою кодувальника на основі дифузії, який передбачає ряд математичних операцій, які застосовуються до цільового зображення [21].

Алгоритм декодування зображення в SDNN включає наступні кроки:

Крок 1. Процес декодування починається з цільового зображення x.

Крок 2. Цільове зображення x передається через нелінійну функцію активації f:

$$y = f(x). \tag{2.6}$$

Крок 3. Вихідні дані функції активації y потім нормалізуються за допомогою функції нормалізації g, яку можна вивчати:

$$z_0 = g(y). \tag{2.7}$$

Крок 4. Вхідний вектор шуму z_0 потім пропускається через процес дифузії, який включає серію T кроків. На кожному кроці t вхідний вектор шуму z_t перетворюється за допомогою вивченої матриці лінійного перетворення W_t розміром $N \times N$, де N є розмірністю вхідних даних. Перетворений вектор позначається як h_t:

$$h_t = W_t z_t. \tag{2.8}$$

Потім перетворений вектор пропускається через нелінійну функцію активації f_t:

$$y_t = f_t(h_t). \tag{2.9}$$

Потім вихідні дані функції активації y_t нормалізуються за допомогою функції нормалізації g_t, яку можна вивчати:

$$z_{t+1} = g_t(y_t). \tag{2.10}$$

Процес дифузії повторюється для T кроків для генерації остаточного вихідного вектора шуму z_T.

Крок 5. Кінцевий вихідний вектор шуму z_T потім перетворюється за допомогою вивченої матриці лінійного перетворення W_{T+1} розміром $N \times N$, так що:

$$h_{T+1} = W_{T+1}\, z_T. \tag{2.11}$$

Крок 6. Результат остаточного лінійного перетворення h_{T+1} потім передається через кінцеву нелінійну функцію активації g:

$$z = g(h_{T+1}). \quad (2.12)$$

Метою алгоритму декодування зображення в SDNN є відновлення прихованого коду або вхідного вектора шуму z, який використовувався для створення цільового зображення x. Це досягається оптимізацією параметрів матриць лінійного перетворення W_t і W_{t+1}, функцій активації f_t і f та функцій нормалізації g_t і g за допомогою навчального набору даних. Оптимізація зазвичай виконується за допомогою зворотного поширення та стохастичного градієнтного спуску, де функція втрат визначається як помилка реконструкції між цільовим зображенням x і згенерованим зображенням x_{hat}. Алгоритм декодування зображень у SDNN є потужним інструментом для реконструкції зображень і може використовуватися для широкого спектру застосувань, включаючи усунення шумів зображення, надвисоку роздільну здатність і малювання.

2.2 Оптимізація гіперпараметрів для кращої продуктивності

Деякий набір параметрів, які використовуються для контролю поведінки моделі та регулюються з метою отримання імпровізованої моделі з оптимальною продуктивністю називають гіперпараметрами.

Оптимізація гіперпараметрів є важливим кроком у покращенні продуктивності моделей SDNN для створення зображень у застосунках iOS. Розглянемо кілька потенційних підходів до оптимізації гіперпараметрів:

- пошук у сітці;
- випадковий пошук;
- байєсова оптимізація;
- автоматичне налаштування гіперпараметрів.

Пошук у сітці – це поширений підхід до налаштування гіперпараметрів, у якому визначається набір гіперпараметрів та їхні можливі значення, а модель навчається та оцінюється з кожною комбінацією гіперпараметрів. Це може бути трудомістким процесом, але він може допомогти визначити оптимальну комбінацію гіперпараметрів для певного завдання.

Випадковий пошук – це більш ефективний підхід до налаштування гіперпараметрів, у якому визначається набір гіперпараметрів та їхні можливі значення, а модель навчається та оцінюється за допомогою випадково вибраної комбінації гіперпараметрів. Це може бути швидше, ніж пошук у сітці, особливо якщо гіперпараметри мають великий діапазон можливих значень.

Байєсова оптимізація – це вдосконалений підхід до налаштування гіперпараметрів, у якому імовірнісна модель використовується для визначення найбільш перспективного набору гіперпараметрів, який слід спробувати наступним чином. Це може бути ефективнішим, ніж пошук по сітці та випадковий пошук, оскільки для пошуку оптимальних гіперпараметрів потрібно менше ітерацій.

Також доступно кілька автоматизованих інструментів і платформ, які можуть допомогти з налаштуванням гіперпараметрів, наприклад AutoML від Google і машинне навчання Azure від Microsoft. Ці інструменти використовують розширені алгоритми для автоматизації процесу налаштування гіперпараметрів, що може заощадити час і підвищити продуктивність моделі.

Розглянемо кожен з цих методів з математичної точки зору.

Нехай H – набір гіперпараметрів, де кожен гіперпараметр позначається h_i. Нехай V_i – набір можливих значень h_i, де i – індекс для гіперпараметра. Отже, маємо:

$$H = \{h_1, h_2, \ldots, h_n\}, \tag{2.13}$$

$$V_i = \{v_1, v_2, \ldots, v_m\}, \tag{2.14}$$

для кожного h_i.

Для методу пошуку по сітці ми можемо визначити сітку гіперпараметрів G, як декартів добуток множин V_i:

$$G = V_1 \times V_2 \times \ldots \times V_n . \quad (2.15)$$

Таким чином, G представляє всі можливі комбінації гіперпараметрів. Для кожної комбінації g у G можна навчити та оцінити модель із цими гіперпараметрами та записати показники ефективності.

Метою пошуку по сітці є визначення оптимальної комбінації гіперпараметрів, що призводить до найкращої продуктивності за метрикою оцінки. Для цього оцінюється продуктивність моделі для кожної комбінації гіперпараметрів і вибирається комбінація з найкращою продуктивністю.

Для виконання випадкового пошуку випадковим чином обирається комбінація гіперпараметрів із простору гіперпараметрів і навчається та оцінюється модель за допомогою цих гіперпараметрів. Цей процес повторюєтьмя протягом визначеної кількості ітерацій або доки не досягається задовільного рівня продуктивності.

Перевага випадкового пошуку перед пошуком по сітці полягає в тому, що він вимагає менше ітерацій для пошуку хорошого набору гіперпараметрів, оскільки він не обмежений сіткою можливих комбінацій. Крім того, це може бути більш ефективним з точки зору обчислень [22], оскільки можемо зосередитися на найбільш перспективних областях простору гіперпараметрів.

Однак випадковий пошук не гарантує, що досліджується весь простір гіперпараметрів, і він може пропустити певні комбінації гіперпараметрів, які важливі для оптимальної продуктивності. Тому він не завжди може знайти найкращий набір гіперпараметрів для даного завдання.

Для Байєсовської оптимізації визначаємо цільову функцію f, яка приймає набір гіперпараметрів h як вхідні дані та повертає показник продуктивності, такий як точність перевірки або втрати. На цьому етапі мета

полягає в тому, щоб знайти оптимальний набір гіперпараметрів, який максимізує f.

Байєсовська оптимізація відбувається ітеративно шляхом побудови імовірнісної моделі цільової функції f, яка передбачає виконання різних комбінацій гіперпараметрів на основі даних спостереження. Ця модель оновлюється після кожної ітерації, використовуючи результати оцінки моделі для кожної комбінації гіперпараметрів.

На кожній ітерації оцінюється модель з вибраною комбінацією гіперпараметрів і оновлюється ймовірнісна модель на основі нових даних. Цей процес повторюється, доки не досягається задовільний рів продуктивності або попередньо визначеної кількості ітерацій.

Байєсовська оптимізація має кілька переваг перед іншими методами налаштування гіперпараметрів, такими як пошук по сітці та випадковий пошук. Він більш ефективний, оскільки зосереджується на найбільш перспективних областях простору гіперпараметрів, і може забезпечити кращу продуктивність, оскільки враховує невизначеність у прогнозах моделі.

Окрім того, налаштовуючи гіперпараметри для моделей SDNN у програмах iOS, важливо враховувати такі фактори, як використання пам’яті [23], час обробки та розмір набору даних зображення. Оптимізуючи гіперпараметри для цих факторів, можна досягти кращої продуктивності та генерувати зображення вищої якості за допомогою SDNN.

2.3 Методологія розробки мобільного застосунку з використанням технології Stable Diffusion

Мобільні програми все частіше використовують дані в реальному часі, щоб створити пряму взаємодію з користувачем. Одним із способів досягти цього є використання API. Розробка мобільного застосунку з використанням Stable Diffusion API може складатися з таких кроків:

Крок 1. Визначення варіанту використання. Потрібно визначитися з метою використання Stable Diffusion API. На цьому етапі зазвичай розглядаються такі варіанти, як 2D графіка, 3D графіка, відео.

Крок 2. Вибір платформи мобільної розробки. Після того, як буде визначено варіант використання, наступним кроком буде вибір платформи мобільної розробки, сумісної з Stable Diffusion API. На сьогоднішній день доступно кілька платформ для розробки мобільних пристроїв, у тому числі нативні платформи, такі як iOS і Android, і кросплатформні рішення, такі як React Native або Xamarin.

Крок 3. Інтеграція Stable Diffusion API. Після вибору платформи наступним кроком є інтеграція Stable Diffusion API у мобільну програму. Зазвичай це передбачає створення з'єднання з API та обробку обміну даними між мобільною програмою та API.

Крок 4. Реалізація оновлень даних у режимі реального часу. Якщо сценарій використання передбачає оновлення даних у режимі реального часу, наступним кроком є впровадження необхідної функціональності [24].

Крок 5. Перевірка та оптимізація. Після впровадження необхідної функціональності мобільну програму слід ретельно перевірити, що вона стабільна та працює добре. Продуктивність програми слід оптимізувати, щоб гарантувати, що вона може обробляти оновлення даних у реальному часі та функціонал працює без затримок або збоїв. Це може включати тестування продуктивності, тестування навантаження та стрес-тестування.

Крок 6. Розгортання та обслуговування. Після тестування мобільний застосунок можна розгорнути у відповідних магазинах застосунків і рекламувати користувачам. Важливо продовжувати підтримувати та оновлювати мобільну програму з часом, щоб переконатися, що вона залишається сумісною з Stable Diffusion API і продовжує задовольняти потреби користувачів. Можна випускати регулярні оновлення для виправлення помилок, додавання нових функцій і покращення загальної взаємодії з користувачем.

На рисунку 2.6 мобільний застосунок представлено об’єктом «Mobile App». Об’єкт «Канал даних» представляє певний канал даних, на який мобільна програма може підписатися та отримувати оновлення.

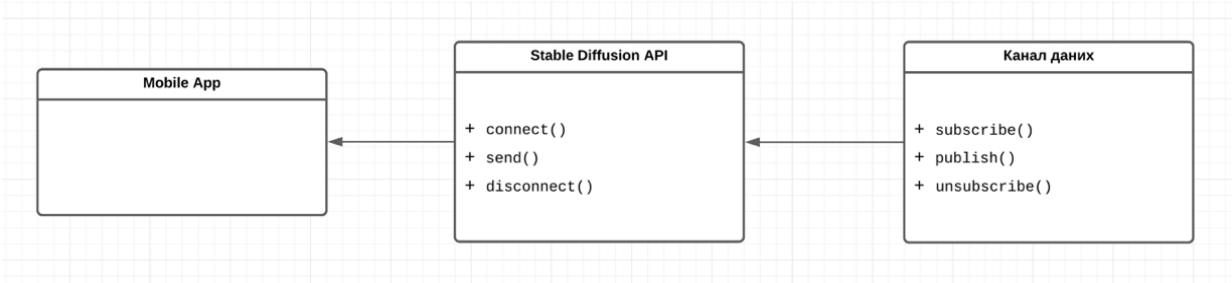

Рисунок 2.6 – Схема роботи декодера

Об’єкт «Mobile App» може з’єднуватися з об’єктом «Stable Diffusion API» за допомогою методу «connect()», який встановлює зв’язок між ними. Потім мобільна програма може використовувати метод «subscribe()» об’єкта «Канал даних», щоб підписатися на певні канали даних і отримувати оновлення в реальному часі, коли вони відбуваються. Об’єкт «Stable Diffusion API» може використовувати метод «publish()», щоб надсилати оновлення до каналів даних, на які є підписка, а об’єкт «Mobile App» може використовувати метод «unsubscribe()», щоб скасувати підписку на певний канал даних.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Етапи програмної реалізації iOS застосунку

3.1.1 Модель DALL-E API

DALL-E – це передова модель штучного інтелекту, розроблена OpenAI, яка поєднує розуміння мови та можливості створення зображень. Вона розшифровується як «розподілені, наближені та локально лінійні вбудовування», які є методами, які використовуються під час навчання моделі. Головна сторінка DALL-E представлена на рисунку 3.1 [25].

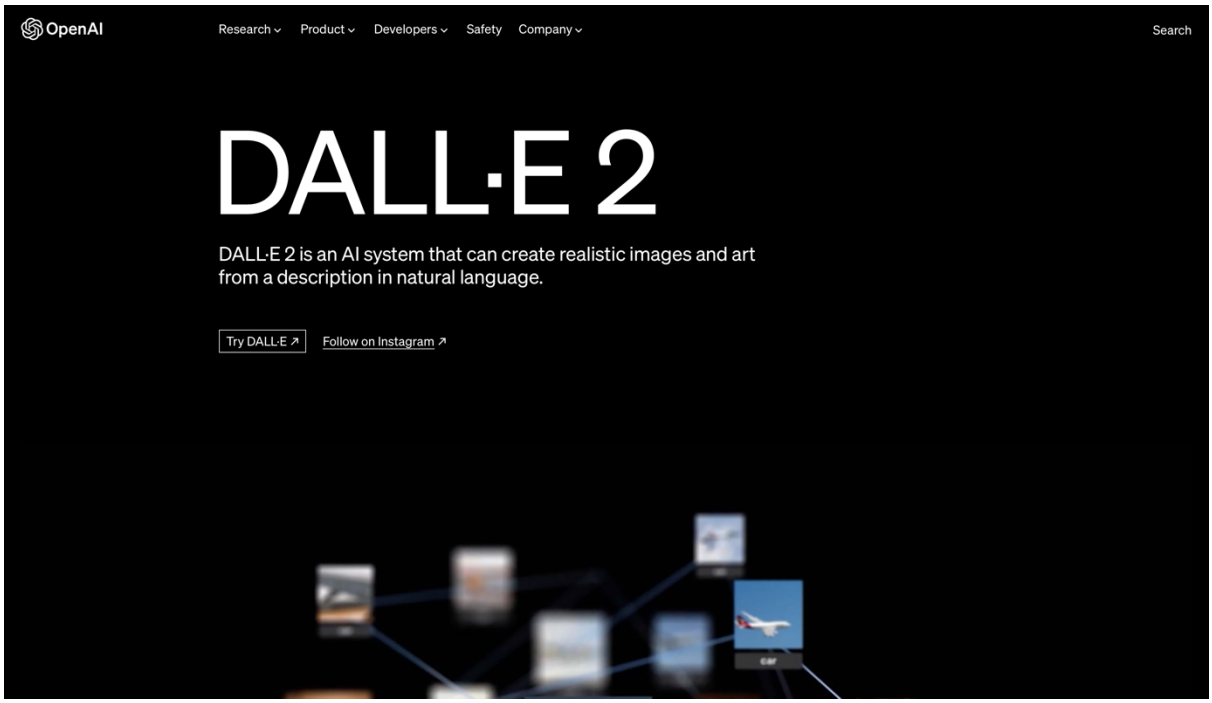

Рисунок 3.1 – Головна сторінка DALL-E

На відміну від традиційних моделей генерації зображень, які спираються на існуючі зображення, DALL-E має унікальну здатність створювати оригінальні зображення з текстових описів, яких вона ніколи раніше не бачила. DALL-E може створювати дуже деталізовані та зв’язні зображення на основі конкретних текстових підказок.

Навчання DALL-E включало великий набір даних пар тексту та зображення з мільйонами прикладів. Модель була навчена з використанням архітектури VQ-VAE-2, яка поєднує в собі ідеї автокодерів і векторного квантування.

У процесі навчання DALL-E навчилась пов’язувати текстові описи з відповідними зображеннями, відображаючи інформацію в просторі нижчого виміру. Це дозволяє моделі генерувати нові зображення шляхом декодування прихованого представлення назад у простір зображення.

Одним із чудових аспектів DALL-E є його здатність генерувати зображення для незвичайних або сюрреалістичних концепцій, які можуть не існувати в реальному світі. Вона може створювати образи, які поєднують різні предмети або мають фантастичні характеристики. Це робить DALL-E потужним інструментом для художніх і творчих застосунків.

Випуск DALL-E у 2021 році привернув значну увагу та продемонстрував потенціал моделей штучного інтелекту у створенні надзвичайно реалістичних і креативних зображень. Вона демонструє досягнення в дослідженнях штучного інтелекту та потенціал ШІ для допомоги в різних сферах, включаючи мистецтво, дизайн і створення візуального контенту, в тому числі для рекламних цілей.

Важливо зазначити, що хоча DALL-E може створювати вражаючі зображення, вона не має розуміння світу чи контексту, що стоїть за зображеннями, які вона створює. DALL-E зосереджена насамперед на створенні візуально цілісних зображень на основі текстових підказок і не розуміє семантичного значення чи ширшого контексту описів.

OpenAI продовжує розширювати межі можливостей штучного інтелекту за допомогою таких моделей, як DALL-E, сприяючи інноваціям і досліджуючи потенційні можливості застосування штучного інтелекту в різних сферах. Тому у 2022 році OpenAI представили бета модель DALL-E API для розробників (рис. 3.2) [26]. Однією з великих переваг є те, що ця модель є

безкоштовною. Завдяки цьому кожен розробник може вільно впроваджувати DALL-E у власні застосунки.

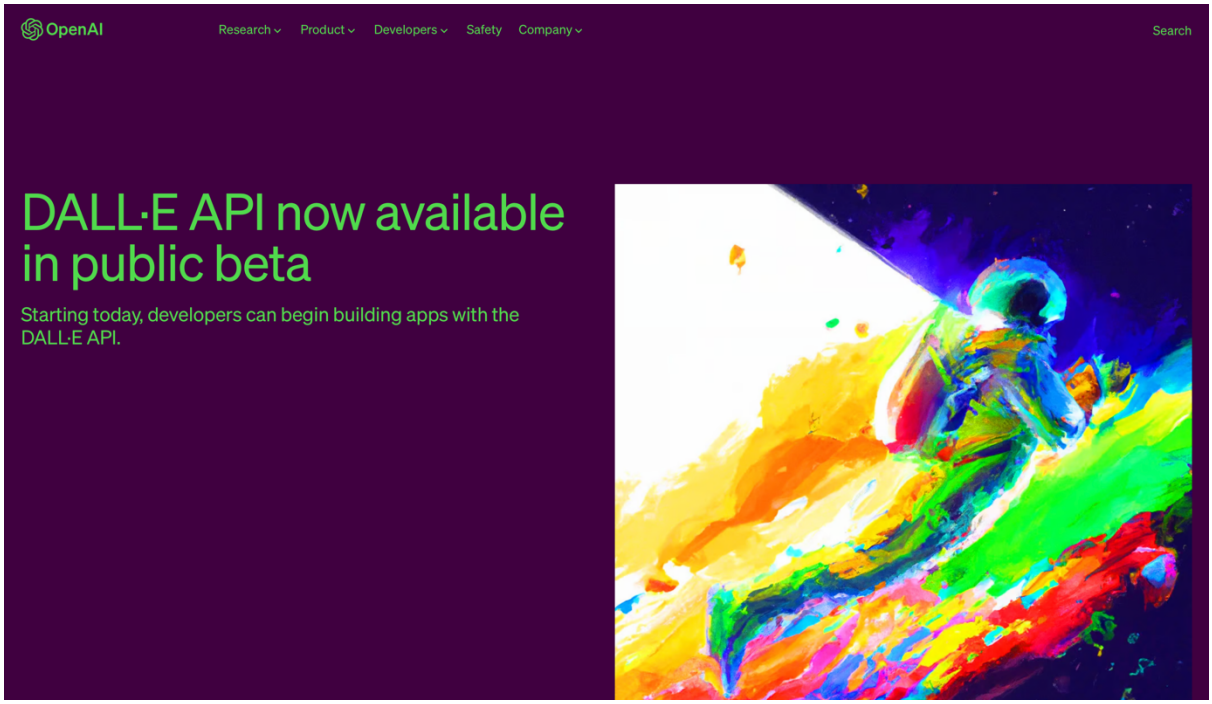

Рисунок 3.2 – Бета модель DALL-E API для розробників

3.1.2 Модель Midjourney API

Midjourney – це модель генеративного штучного інтелекту, створена та розміщена незалежною дослідницькою лабораторією Midjourney, Inc. у Сан-Франциско. Midjourney генерує зображення з описів природною мовою, які називаються «підказками», подібно до DALL-E.

Компанія Midjourney була заснована в 2021 році командою художників, технологів і дизайнерів. Місія компанії полягає в тому, щоб «розширити уявні можливості людей». Наразі Midjourney перебуває у відкритому бета-тестуванні та доступний лише за запрошенням (рис. 3.3).

Щоб використовувати Midjourney, користувачі надають підказку, що є коротким текстовим описом зображення. Потім Midjourney генерує кілька

зображень, які відповідають підказці. Після цього користувачі можуть вибрати зображення, яке їм найбільше подобається, і завантажити його.

Рисунок 3.3 – Головна сторінка Midjourney

Midjourney відрізняється здатністю створювати реалістичні та творчі зображення. Компанію також критикували за її потенціал використання для створення шкідливого або образливого контенту. Midjourney вжив заходів для вирішення цих проблем, зокрема запровадив політику модерації та надав користувачам інструменти для фільтрації шкідливого вмісту.

Midjourney працює на базі великої мовної моделі (LLM) під назвою BARD. BARD навчався на величезному наборі тексту та коду, і він може генерувати текст, перекладати мови, писати різні види творчого вмісту та інформативно відповідати на запитання.

Окрім того, Midjourney можна використовувати для створення зображень для різноманітних цілей, таких як реклама, дизайн та освіта.

Midjourney – потужний інструмент, який можна використовувати для створення різноманітних зображень. Місія компанії щодо розширення можливостей уяви людей є амбітною, але очевидно, що Midjourney має

потенціал справити значний вплив на те, як створюється та споживається мистецтво.

Midjourney не має офіційного API, але є кілька неофіційних API, створених сторонніми розробниками. Ці API дозволяють програмно взаємодіяти з Midjourney, що може бути корисним для таких завдань, як автоматизація створення зображень або інтеграція Midjourney з іншими програмами.

Одним із найпопулярніших неофіційних API для Midjourney є Imagine API (рис. 3.4). Imagine API [27] – це RESTful API, який дозволяє створювати, редагувати та видаляти зображення з Midjourney. Також можна використовувати Imagine API, щоб отримати інформацію про зображення, наприклад їх розмір, роздільну здатність і дату створення.

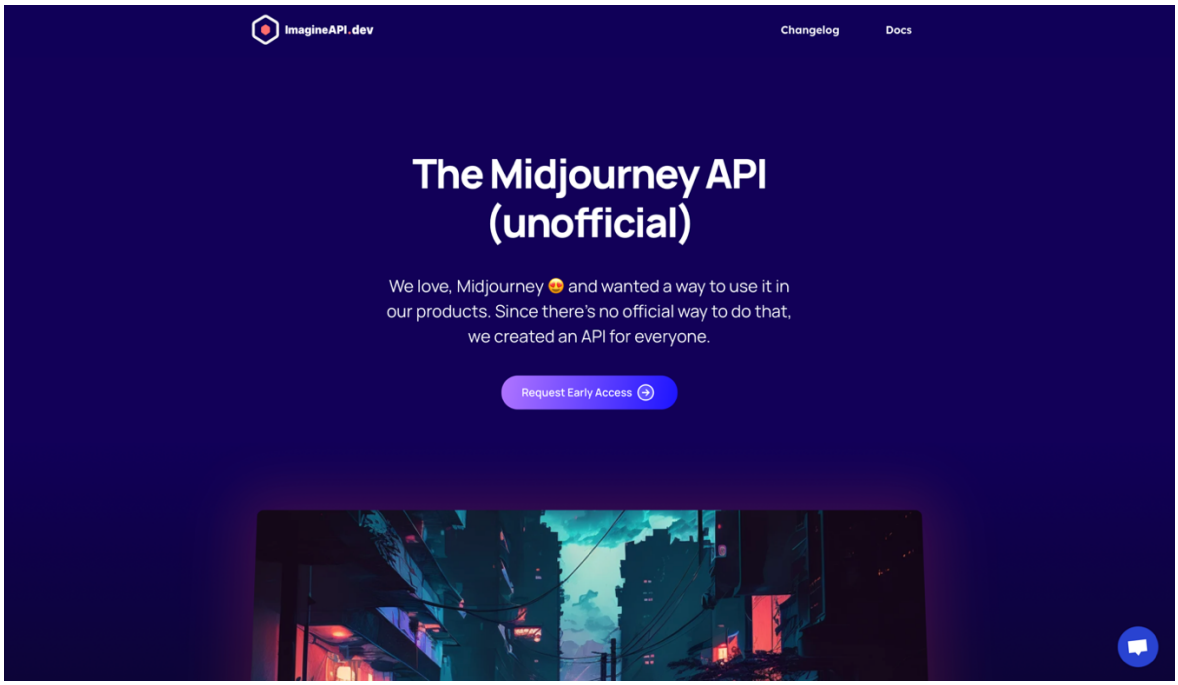

Рисунок 3.4 – Головна сторінка Imagine API

Щоб використовувати Imagine API, потрібно буде створити обліковий запис і отримати ключ API. Отримавши ключ API, можна використовувати його для надсилання запитів до API Imagine.

Іншим неофіційним API для Midjourney є API від The Next Leg (рис. 3.5) [28]. Це GraphQL API, який дозволяє взаємодіяти з Midjourney гнучкіше, ніж Imagine API. Можна використовувати Midjourney API від The Next Leg для створення, редагування та видалення зображень, а також для отримання інформації про зображення.

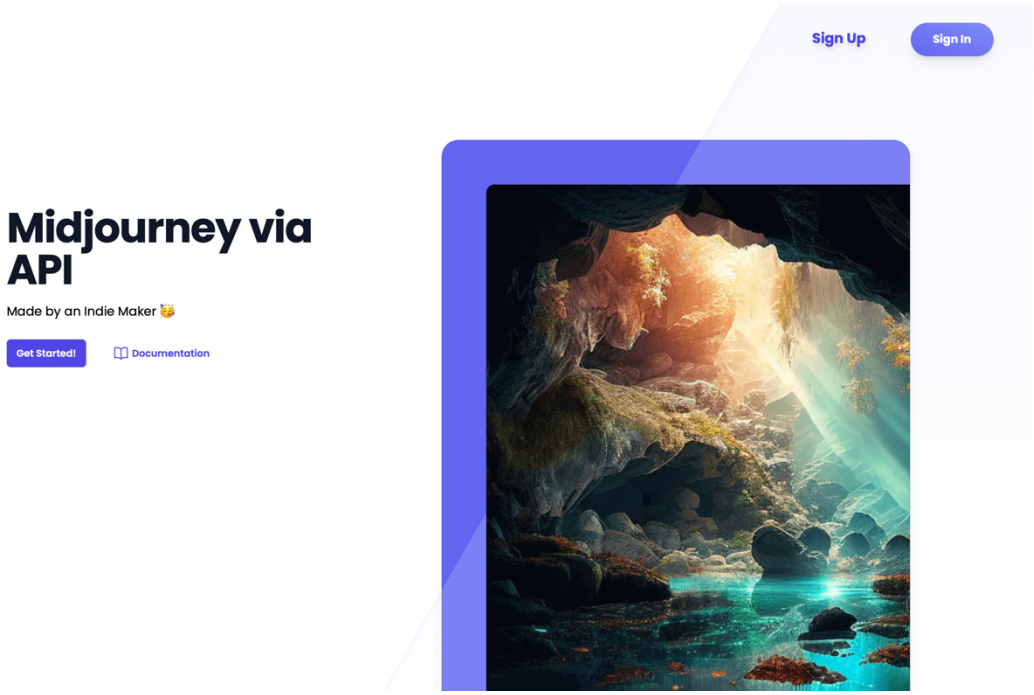

Рисунок 3.5 – Головна сторінка Midjourney API від The Next Leg

Щоб використовувати Midjourney API від The Next Leg, потрібно буде створити обліковий запис і отримати ключ API. Отримавши ключ API, можна використовувати його для надсилання запитів.

3.1.3 Впроваження API у застосунок

Для роботи з різноманітними API в iOS застосунках існує нативний інструмент – URLSession, але працювати з ним іноді буває складно. Для полегшення цього процесу існує бібліотека Alamofire.

Alamofire – це елегантний і зручний інтерфейс для мережевих запитів HTTP. Він створений на основі системи завантаження URL-адрес Apple, наданої фреймворком Foundation. Ядром системи є URLSession і підкласи URLSessionTask. Alamofire обгортає API у простий у використанні інтерфейс і надає різноманітні функції, необхідні для розробки сучасних програм за допомогою мережі HTTP.

Для впровадження Alamofire у застосунок існує спеціальний пакет «Alamofire Cocoa Pods». Для його використання потрібно щоб застосунок відповідав наступним вимогам:

- iOS 9.0+ / macOS 10.11+ / tvOS 9.0+ / watchOS 2.0+;
- Xcode 8.0+;
- Swift 3.0+;
- CocoaPods 1.1.0+.

Також для додавання «Alamofire Cocoa Pods» потрібно створити відповідний Podfile.

Основним методом роботи Alamofire є запит. Ця бібліотека підтримує такі HTTP запити, як GET, PUT, POST та DELETE.

Alamofire підтримує різні способи обробки відповіді. Очікується, що відповідь надасть клієнту запитуваний ресурс, повідомить клієнта, що запитану дію було виконано або повідомити клієнта, що під час обробки його запиту сталася помилка. Існують такі варіанти обробки відповіді:

- основна відповідь;
- відповідь JSON;
- відповідь даних;
- рядкова відповідь;
- декодована відповідь.

Базова відповідь не оцінює жодних даних відповіді, вона просто пересилає інформацію безпосередньо з URLSessionDelegate.

Обробник responseData використовує DataResponseSerializer для отримання та перевірки даних, які повертає сервер.

Обробник responseString використовує StringResponseSerializer для перетворення даних, повернутих сервером, у рядок із вказаним кодуванням.

Обробник responseDecodable використовує DecodableResponseSerializer для перетворення даних, повернутих сервером, у переданий тип Decodable за допомогою вказаного DataDecoder.

Для отримання запиту з DALL-E та Midjourney потрібно створити функції getImageFromDalle(prompt: String) та getImageFromMidjourney(prompt: String), які приймають строку з підказкою користувача типу String та робити запити на сервер.

Функції повертають результат з типом Data. Через це результат потрібно декодувати та перетворити в формат UIImage.

Приклади коду запитів наведено в додатку А.

3.2 Розробка сценарію взаємодії користувача із застосунком

Окрім розробки самого застосунку потрібно чітко продумати сценарії роботи користувача із застосунком. Цей процес також називають потоком користувача. Потік користувача, також відомий як шлях користувача або потік досвіду користувача, відноситься до послідовності кроків, які виконує користувач під час взаємодії з продуктом. Він окреслює шлях, який проходить користувач для виконання конкретного завдання або досягнення конкретної мети в системі. Потік користувачів є критично важливим аспектом розробки інтуїтивно зрозумілого та ефективного інтерфейсу.

Розробка ефективного потоку користувачів передбачає врахування цільових користувачів, їхні цілі та загальні цілі застосунку. Створюючи схему потоку користувачів, дизайнери та розробники можуть виявити потенційні

вузькі місця, оптимізувати взаємодію та створити більш оптимізовану та зручну для користувача роботу.

Детальний опис потоку користувача в застосунку зображено на рисунку 3.6.

Рисунок 3.6 – Опис потоку користувача в застосунку

В першу чергу користувач бачить стартовий екран (рис. 3.7). На ньому відображається процес підготовки даних до початку роботи.

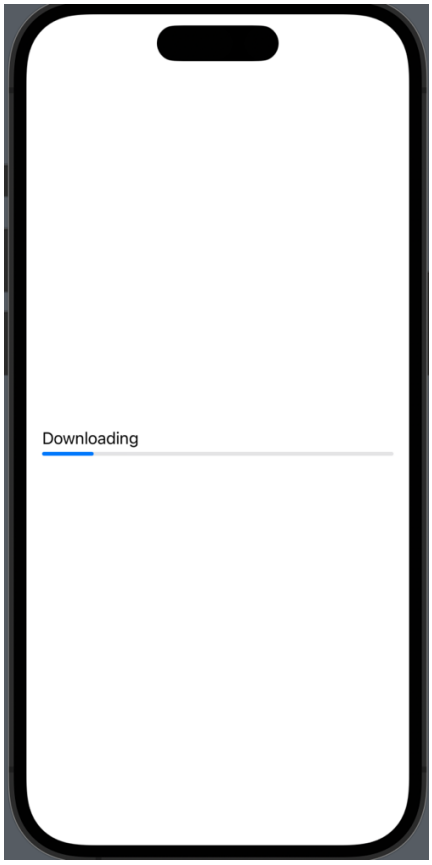

Рисунок 3.7 – Процес підготовки даних до початку роботи

Після підготовки застосунку до роботи користувачу надається доступ до текстового поля, в якому він має можливість ввести потрібний запит (рис. 3.8).

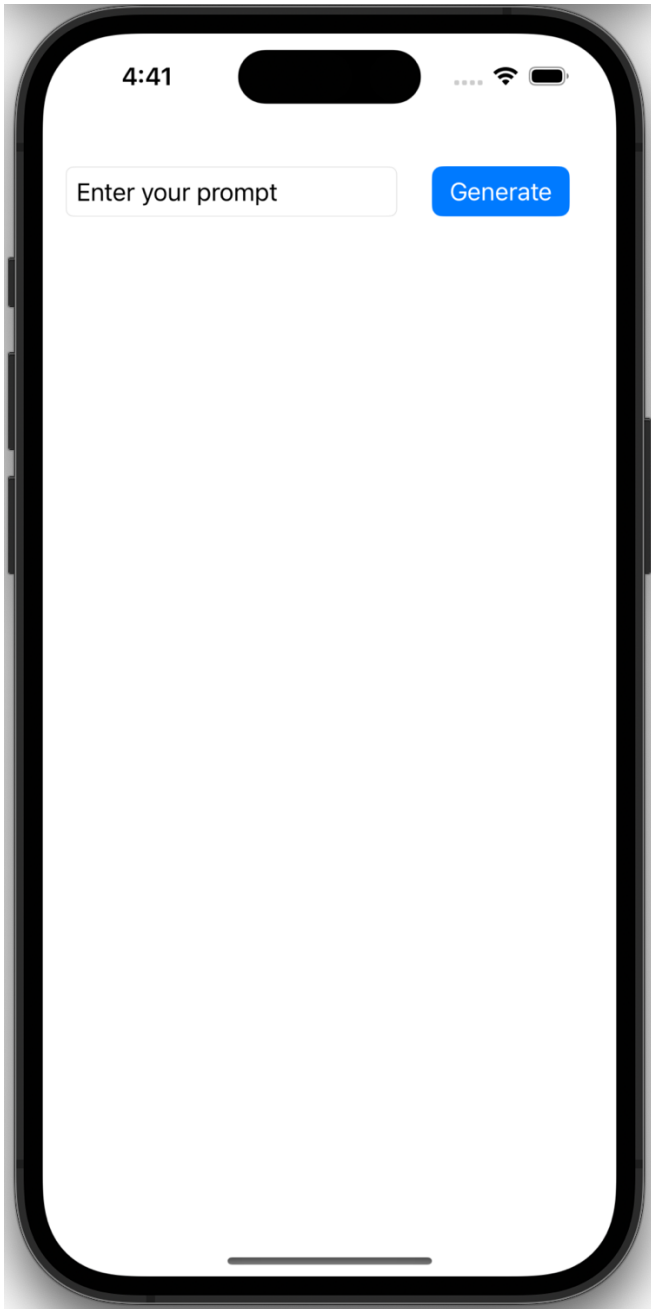

Рисунок 3.8 – Початковий екран з текстовим полем

Після натискання на кнопку «Generate» починається процес обробки запиту. На цьому етапі користувач повинен зачекати поки нейронні мережі згенерують відповідні до запиту зображення. Екран з показником процесу генерації зображено на рисунку 3.9.

Рисунок 3.9 – Процес генерації зображень

Після закінчення обробки запиту користувач може побачити результати згенерованих зображень на екрані (рис. 3.10).

Рисунок 3.10 – Результати згенерованих зображень

Далі можливі два варіанта розвитку подій. Якщо користувачу подобаються результати зображень за його запитом, то він може завантажити ці зображення собі на пристрій (рис. 3.11).

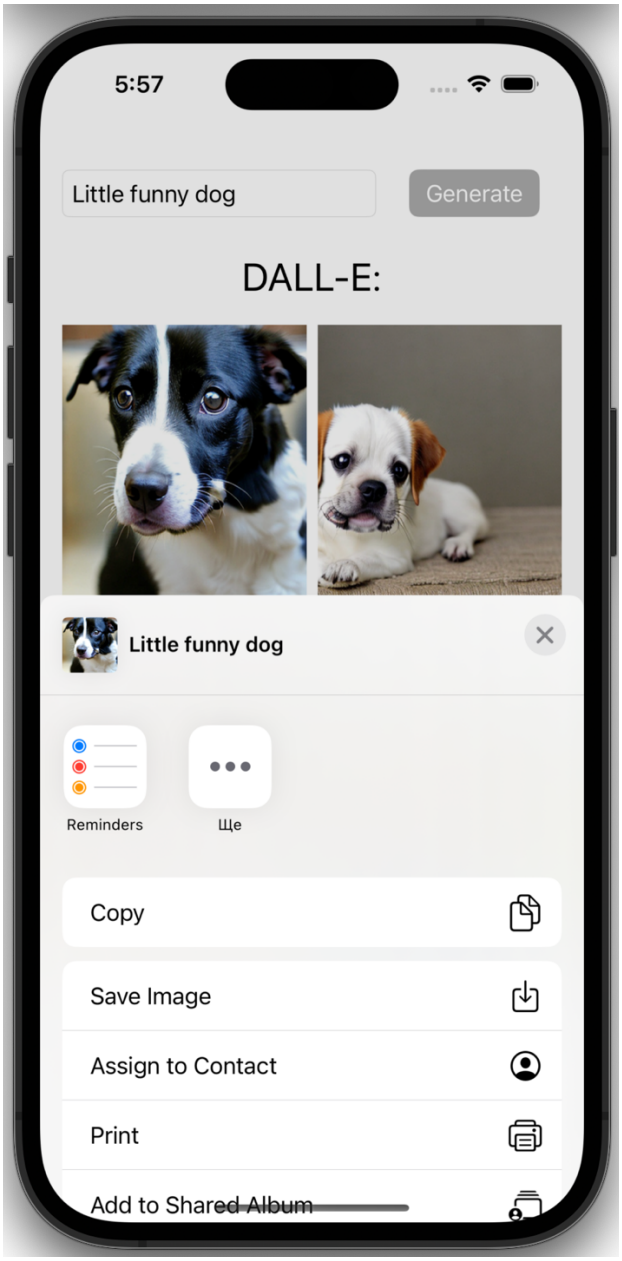

Рисунок 3.11 – Збереження на пристрій зображень

Якщо користувач не задоволений результатами, то він може змінити запит або продовжити з попереднім запитом та знову натиснути на кнопку «Generate». Після цього знову буде запущено процес генерації. Користувач може повторювати ці кроки, поки результати зображень його не задовільнять.

3.3 Тестування розробленого застосунку та аналіз результатів

Коли розробку застосунку завершено, розпочинається процес тестування.

Тестування застосунків iOS – це процес тестування, у якому застосунок iOS тестується на реальних пристроях Apple, щоб перевірити, чи працює він належним чином чи ні для конкретних дій користувача, таких як час встановлення, інтерфейс користувача, досвід користувача, зовнішній вигляд, поведінка, функціональність, час завантаження, продуктивність, список App Store, підтримка версій ОС тощо [29].

В даній роботі тестування було поділено на 2 етапи:

- тестування коду;
- тестування результатів роботи застосунку.

Для тестування коду зазвичай використовуються Unit тести (XCTests) та UI тести (XCUITests).

Фреймворк XCTest [30] використовується для написання модульних тестів для проєктів Xcode, які бездоганно інтегруються з робочим процесом тестування Xcode.

Тести стверджують, що під час виконання коду виконуються певні умови, і фіксують помилки тестування з необов’язковими повідомленнями, якщо ці умови не виконуються. Тести також можуть вимірювати продуктивність блоків коду, щоб перевірити регресію продуктивності, і можуть взаємодіяти з інтерфейсом користувача програми для перевірки потоків взаємодії користувача.

Після запуску Unit тестів було підтверджено, що функції застосунку працюють вірно, усі умови було виконано.

Результати проходження Unit тестів показано на рисунку 3.12.

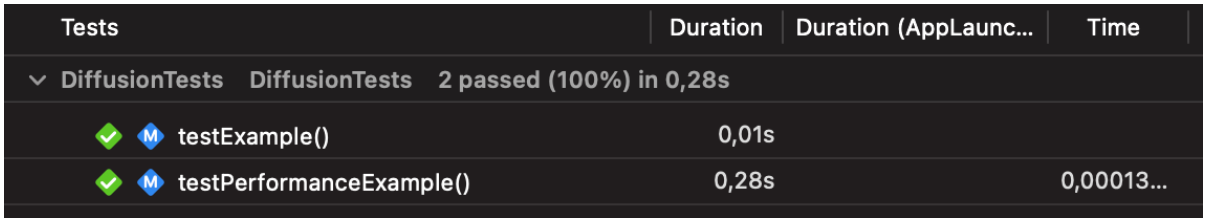

Рисунок 3.12 – Результат Unit тестів

UI тести стосуються тестування інтерфейсу користувача та взаємодії з елементами на екрані за допомогою тестів. Це допомогає переконатися та перевірити, що частина інтерфейсу користувача програми працює належним чином, а також виявити будь-які регресії через зміни в коді, пов'язаному з інтерфейсом.

Apple надає структуру під назвою XCUITest [31] для тестування інтерфейсу користувача. Віна спирається на дані, які технологія доступності використовує для взаємодії з екраном. Робота з цими даними схожа на написання модульних тестів. Створення тестових класів інтерфейсу відбувається за допомогою підкласу XCUITestCase.

Результати проходження UI тестів показано на рисунку 3.13.

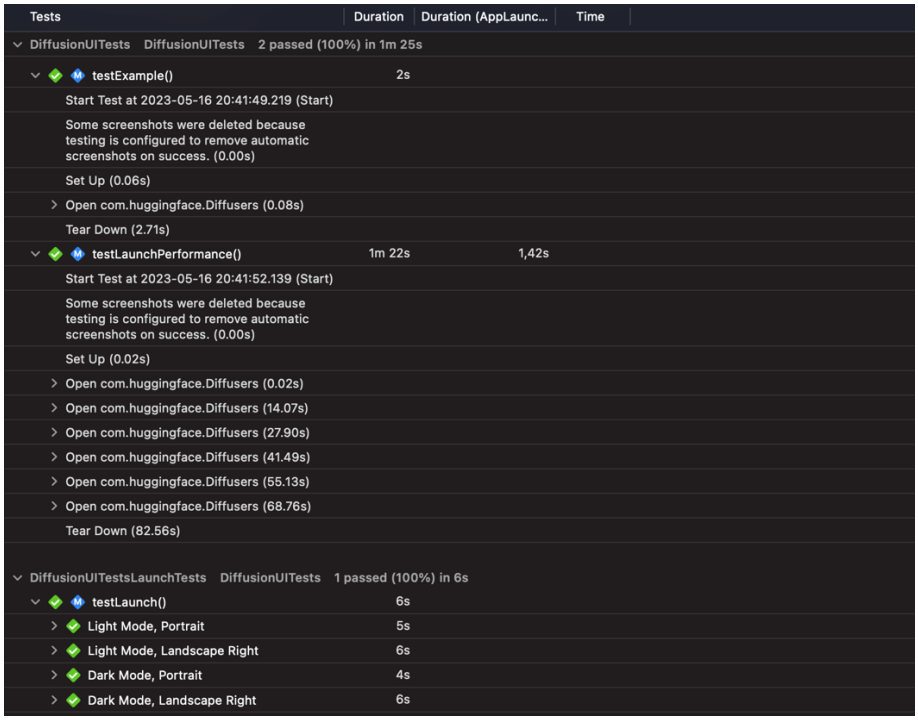

Рисунок 3.13 – Результат UI тестів

Наступний етап – перевірка роботи застосунка, а саме його функціональність. Для цього було створено декілька підказок запитів, що відрізняються за довжиною та кількістю уточнюючих деталей довжиною від 1 до 31 слів:

– «кішка»;

– «маленька кішка»;

– «маленький кіт з м’ячом»;

– «маленький кіт з червоною візерунковою кулькою»;

– «маленький кіт з червоною візерунковою кулькою на килимі»;

– «маленька кішка з червоною візерунковою кулькою на білому вовняному килимі»;

– «маленька кішка з червоною візерунковою кулькою на білому вовняному килимі у великій квартирі»;

– «маленький кіт сидить з червоною візерунковою кулькою на білому вовняному килимі у великій квартирі з рожевими стінами»;

– «маленький кіт сидить з червоною візерунковою кулькою на білому вовняному килимі у великій квартирі з рожевими стінами та великими вікнами»;

– «маленький кіт сидить з червоною візерунковою кулькою на білому вовняному килимі у великій квартирі з рожевими стінами та великими вікнами з фіолетовими шторами»;

– «маленький кудлатий кіт сидить з червоною візерунковою кулькою на білому вовняному килимі у великій квартирі з рожевими стінами та великими вікнами з фіолетовими шторами»;

– «маленький кудлатий чорний кіт сидить з червоною кулькою з візерунками на білому вовняному килимі у великій квартирі з рожевими стінами та великими вікнами з фіолетовими шторами»;

– «маленький кудлатий чорний кіт сидить з червоною кулькою з малюнком на білому вовняному килимі у великій квартирі з рожевими стінами та великими вікнами з фіолетовими шторами в Італії»;

– «маленький кудлатий чорний кіт сидить з червоною кулькою з малюнком на білому вовняному килимі у великій квартирі з рожевими стінами та великими вікнами з фіолетовими шторами в Італії у вересні».

Скріншоти застосунку з результатами запитів представлено в Додатку Б.

Далі було проведено аналіз результатів для кожної моделі. Основними показниками для аналізу було обрано кількість слів, час виконання запиту та якість зображення. Якість зображення є доволі суб'єктивним показником, тому для оцінки якості увага зверталась на якість промальовування об'єктів (чи є на них пошкодження) та те, наскільки зображення відповідає запиту.

На основі зібраних даних було сформовано таблиці 3.1 та 3.2.

Таблиця 3.1 – Результати роботи моделі DALL-E

Запит	**Кількість слів**	**Час виконання запиту**	**Якість зображення**
1	**2**	**3**	**4**
cat	1	11 с	Висока
little cat	2	12 с	Висока
little cat with ball	4	13 с	Висока
little cat with red patterned ball	6	12 с	Висока
little cat with red patterned ball on the carpet	9	13 с	Висока
little cat with red patterned ball on the white wool carpet	11	13 с	Висока
little cat with red patterned ball on the white wool carpet in the big flat	15	12 с	Середня

Продовження таблиці 3.1

1	2	3	4
little cat sitting with red patterned ball on the white wool carpet in the big flat	16	12 с	Середня
little cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls	19	11 с	Середня
little cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows	22	12 с	Середня
little cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains	25	14 с	Низька
little shaggy cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains	26	12 с	Низька
little shaggy black cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains	27	12 с	Низька

Продовження таблиці 3.1

1	2	3	4
little shaggy black cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains in Italy	29	11 с	Низька
little shaggy black cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains in Italy in September	31	13 с	Низька

Таблиця 3.2 – Результати роботи моделі Midjourney

Запит	**Кількість слів**	**Час виконання запиту**	**Якість зображення**
1	**2**	**3**	**4**
cat	1	40 с	Висока
little cat	2	41 с	Висока
little cat with ball	4	47 с	Висока
little cat with red patterned ball	6	41 с	Висока
little cat with red patterned ball on the carpet	9	48 с	Висока
little cat with red patterned ball on the white wool carpet	11	40 с	Висока
little cat with red patterned ball on the white wool carpet in the big flat	15	49 с	Висока

Продовження таблиці 3.2

1	2	3	4
little cat sitting with red patterned ball on the white wool carpet in the big flat	16	40 с	Висока
little cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls	19	39 с	Висока
little cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows	22	49 с	Висока
little cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains	25	44 с	Висока
little shaggy cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains	26	40 с	Висока
little shaggy black cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains	27	43 с	Висока

Продовження таблиці 3.2

1	2	3	4
little shaggy black cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains in Italy	29	47 с	Висока
little shaggy black cat sitting with red patterned ball on the white wool carpet in the big flat with pink walls and big windows with purple curtains in Italy in September	31	50 с	Середня

Також було побудовано графіки залежності часу виконання запиту (рис. 3.14) та якості зображення (рис. 3.15) від кількості слів в запиті.

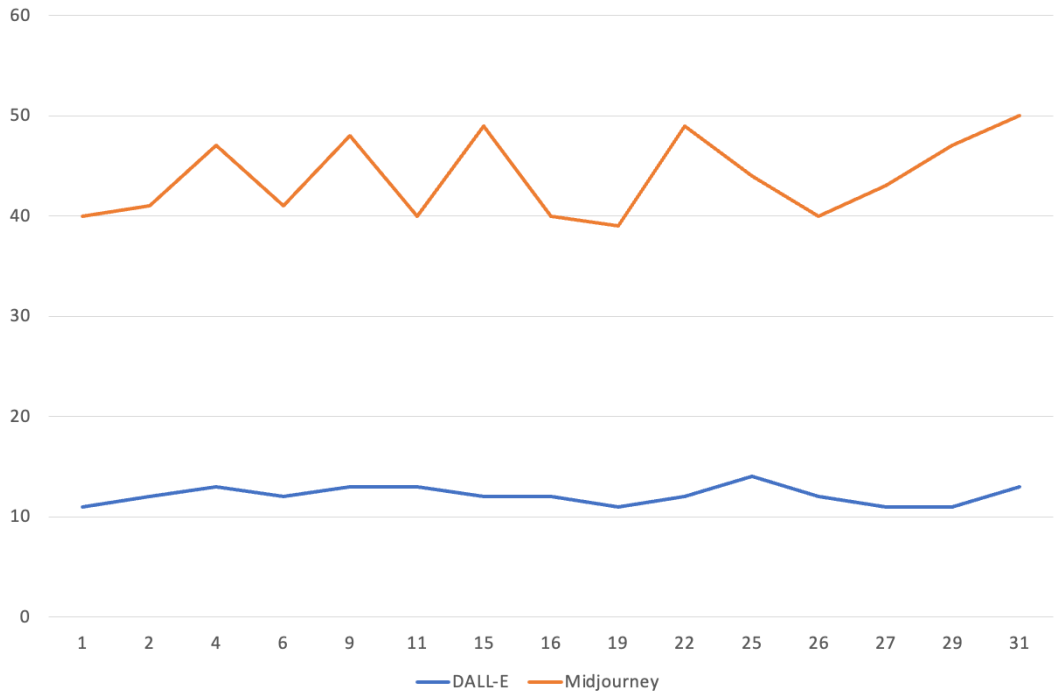

Рисунок 3.14 – Графік залежності часу виконання запиту від кількості слів в запиті DALL-E та Midjourney

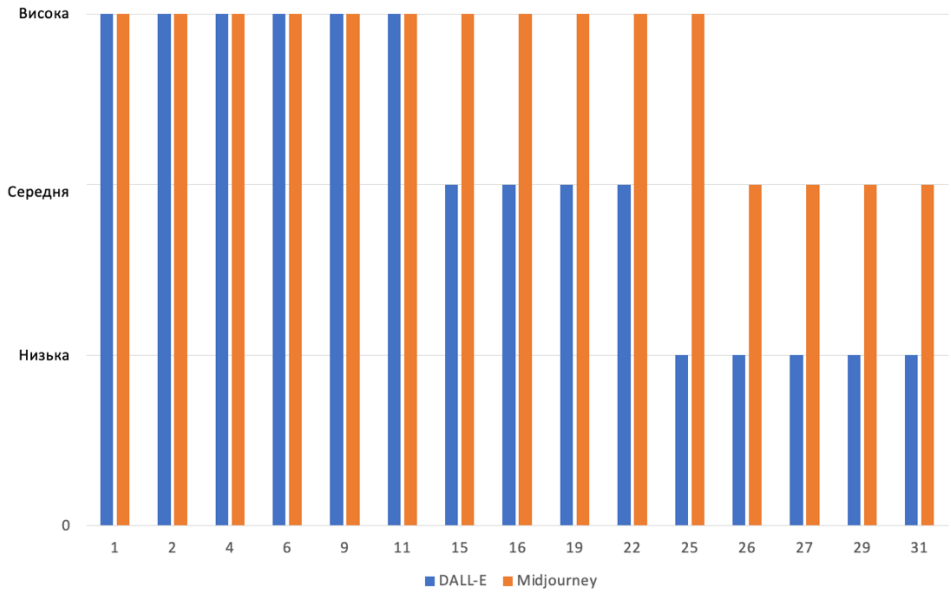

Рисунок 3.15 – Графік залежності якості зображення від кількості слів в запиті DALL-E та Midjourney

Як видно з результатів, наведених в таблицях та на графіках, модель DALL-E виконує запит дуже швидко та якість зображень є доволі високою, якщо запит не перевищує 11 слів. Модель Midjourney виконує запит набагато повільніше, але має значно вищу якість зображення на запитах великої довжини.

З цього можна зробити висновок, що якщо користувач потребує швидкої відповіді без надмірної деталізації, то йому краще скористатися моделлю DALL-E.

Якщо користувач потребує значної деталізації зображень, то йому краще скористуватися результатами моделі Midjourney.

3.4 Перспективи подальшої роботи

Реалізований iOS застосунок можна покращити, додавши новий функціонал:

– можливість збереження одного обраного зображення на пристрій;

- додатковий екран зі збільшеним зображенням;
- можливість надиктувати запит, а не вводити його.

Окрім цього можливо додати функціонал генерації нових зображень за вхідною фотографією (img2img).

Необхідно провести додатковий аналіз та розширити кількість моделей, які будуть працювати над задачею генерації зображень та провести оптимізацію для прискорення генерації хображень [32 – 39].

Взявши до уваги ці деталі, застосунок стане суттєво кориснішими, та його можна буде застосовувати в комерційних цілях.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений мобільний застосунк для створення зображень за їхнім текстовим описом.

Для досягнення мети роботи було вивчено проблематику створення зображень за текстовим описом, проведено аналіз існуючих методів створення зображень, розглянуто типи нейронних мереж для роботи з зображеннями.

Для реалізації проєкту було обрано стабільні дифузійні нейронні мережі (SDNN), проведено аналіз роботи алгоритму моделі Stable Diffusion для генерації зображень. В роботі наведено детальний опис етапів програмної реалізації iOS застосунку на основі моделей DALL-E API та Midjourney API. За результатами роботи було проведено тестування розробленого застосунку та аналіз результатів.

Застосунок створено з використанням сучасних технологій, а саме: середовища для розробки програмного забезпечення XCode, мови програмування Swift, бібліотеки Foundation, бібліотека Alamofire, бібліотеки UIKit та git, фреймворку XCTest, інструменту управління залежностями CocoaPods.

Виконана робота має практичне застосування – розроблено iOS застосунок для створення зображень за їхнім текстовим описом, наданого користувачем, що при певному розширені функціоналу (додати можливість збереження одного зображеня на пристрій; створити додатковий екран зі збільшеним зображенням; надати можливість надиктувати запит та інших) можна буде застосовувати його як для особистого, так і комерційного використання.

Такий мобільний застосунок може бути використано для створення унікальних і креативних дизайнів для веб-сайтів, застосунків, рекламних матеріалів та інших візуальних проєктів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ImageNet. URL: https://www.image-net.org (дата звернення 13.04.2023).

2. Lyashenko, V., Babker, A., & Lyubchenko, V. (2017). Wavelet Analysis of Cytological Preparations Image in Different Color Systems.

3. Tvoroshenko, I., & Koriakin, I. (2021). Analysis of methods for detecting and classifying the likeness of human features.

4. Yousefi, F., Dai, Z., Ek, C. H., & Lawrence, N. (2016). Unsupervised learning with imbalanced data via structure consolidation latent variable model. arXiv preprint arXiv:1607.00067.

5. Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

6. Illustration2Vec. URL: https://github.com/rezoo/illustration2vec (дата звернення 10.04.2023).

7. Gregor, K., Danihelka, I., Graves, A., Rezende, D., & Wierstra, D. (2015, June). Draw: A recurrent neural network for image generation. In International conference on machine learning(pp. 1462-1471). PMLR.

8. Jin, Y., Zhang, J., Li, M., Tian, Y., Zhu, H., & Fang, Z. (2017). Towards the automatic anime characters creation with generative adversarial networks. arXiv preprint arXiv:1708.05509.

9. AnimeGAN. URL: https://github.com/TachibanaYoshino/AnimeGAN (дата звернення 11.04.2023).

10. Diffuse The Rest. URL: https://huggingface.co/spaces/huggingface-projects/diffuse-the-rest (дата звернення 11.04.2023).

11. Latent Diffusion Models. URL: https://github.com/CompVis/latent-diffusion (дата звернення 11.04.2023).

12. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., & Vlasenko, N. (2023). Explanation of CNN image classifiers with hiding parts. In Explainable Deep Learning AI (pp. 125-146). Academic Press.

13. What are recurrent neural networks? URL: https://www.ibm.com/topics/recurrent-neural-networks (дата звернення 15.04.2023).

14. Generative adversarial networks. URL: https://developers.google.com/machine-learning/gan (дата звернення 15.04.2023).

15. Stable Diffusion Clearly Explained! URL: https://medium.com/@steinsfu/stable-diffusion-clearly-explained-ed008044e07e (дата звернення 24.04.2023).

16. Lyashenko, V., Babker, A., & Lyubchenko, V. (2017). The study of blood smear as the analysis of images of various objects.

17. Stable Diffusion: Theory and Applications. URL: https://medium.com/cj-express-tech-tildi/stable-diffusion-theory-and-application-a0f98881cb03 (дата звернення 21.04.2023).

18. Lyashenko, V., Lyubchenko, V., & Mohammad, A. (2015). Computing features of elementary projective orthogonal transformations of an image.

19. Putyatin, Y., Lyashenko, V., Ahmad, M. A., Lyubchenko, V., & Ahmad, N. A. (2015). A Theoretical Interpretation for the Study of Images Processing.

20. The Illustrated Stable Diffusion. URL: https://jalammar.github.io/illustrated-stable-diffusion/ (дата звернення 13.04.2023).

21. Lyashenko, V., Mohammad, A., Lyubchenko, V., & Kobylin, O. (2015). Methodology of Correlation Analysis in Solution of a Problem of Normalization of Projective Image Transformations.

22. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

23. Tvoroshenko, I., & Gorokhovatskyi, V. (2022). The Application of Hybrid Intelligence Systems for Dynamic Data Analysis.

24. Дубницький, В. Ю., Кобилін, А. М., & Кобилін, О. А. (2021). Виконання на мобільних пристроях арифметичних операцій з використанням аксіом класичного та нестандартного інтервального аналізу.

25. DALL·E 2. URL: https://openai.com/product/dall-e-2 (дата звернення 09.05.2023).

26. DALL·E API now available in public beta. URL: https://openai.com/blog/dall-e-api-now-available-in-public-beta (дата звернення 09.05.2023).

27. The Midjourney API (unofficial). URL: https://www.imagineapi.dev (дата звернення 13.05.2023).

28. Midjourney via API. URL: https://www.thenextleg.io/?gclid=CjwKCAjw04yjBhApEiwAJcvNoW9QKj4Pt0I80ARj18tv7tUQyCq0GWLzSMm6EpgRW-JXB4XEXje80hoClaMQAvD_BwE (дата звернення 13.05.2023).

29. Tvoroshenko, I. S., & Kuznetsov, M. (2021). About the role of testing in process of mobile application development.

30. XCTest. URL: https://developer.apple.com/documentation/xctest/ (дата звернення 01.05.2023).

31. User Interface Tests. URL: https://developer.apple.com/documentation/xctest/user_interface_tests (дата звернення 01.05.2023).

32. Mustafa, S. K., Kopot, M., Ahmad, M. A., Lyubchenko, V., & Lyashenko, V. (2020). Interesting applications of mobile robotic motion by using control algorithms.

33. Zeleniy, O., Rudenko, D., Lyubchenko, V., & Lyashenko, V. (2022). Image Processing as an Analysis Tool in Medical Research.

34. Velykodniy, S. S., Burlachenko, Z. V., & Zaitseva-Velykodna, S. S. (2019). Graphic databases reengineering in BRL-CAD open source computer-aided design environment. Modeling of the structural part.

35. Yin, G., Liu, B., Sheng, L., Yu, N., Wang, X., & Shao, J. (2019). Semantics disentangling for text-to-image generation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 2327-2336).

36. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, IEEE Access, 10, pp. 124738-124746.

37. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

38. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, Сучасні інформаційні системи, 6(3), С. 5-12.

39. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, Сучасні інформаційні системи, 7(1), С. 5-13.