

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці**

Освітньо-професійна програма

галузь знань 12 Інформаційні технології
спеціальність 122 Комп'ютерні науки
ОПП *Системи штучного інтелекту*

форма навчання: денна

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

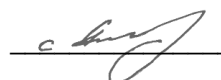
на тему:

**ІНТЕЛЕКТУАЛЬНА СИСТЕМА АВТОМАТИЗОВАНИХ ІЛЮСТРАЦІЙ ЗА
ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ**

здобувача Романченко Владислава Васильовича



Науковий керівник: д.т.н., д.е.н., проф.

 Рамазанов С.К.

**Кваліфікаційна магістерська робота
допущена до захисту в
Екзаменаційній комісії з атестації
здобувачів вищої освіти**

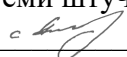
в.о. завідувач кафедри: к.е.н., доцент

_____ Тішков Б.О.

Київ 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»
ОПП «Системи штучного інтелекту»

ПОГОДЖЕНО:
Керівник проектної групи (гарант)
освітньо-професійної програми
«Системи штучного інтелекту»
 Рамазанов С.К.
“ ” 2023 р.

ЗАТВЕРДЖУЮ:
В.о. завідувача кафедри
Тішков Б.О.
“ ” 2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Романченко Владислава Васильовича

денної форми навчання
на підготовку кваліфікаційної магістерської роботи
на кваліфікаційну магістерську роботу
на тему: **«Інтелектуальна система автоматизованих ілюстрацій за
допомогою машинного навчання»**

Тему затверджено наказом ректора від «20» лютого 2023 р. № 333 - ст
Кваліфікаційна магістерська робота виконується на матеріалах

План кваліфікаційної магістерської роботи

Розділ I Аналіз новітніх систем у сфері цифрових зображень, опис практичних застосувань технології, економічний аналіз

Розділ II Порівняння моделей та методів до генерація цифрових зображень, створення плану проекту

Розділ III Навчання власної моделі, розробка інтелектуальної системи створення ілюстрацій, аналіз отриманих результатів

Мета кваліфікаційної магістерської роботи

Створення інтелектуальної системи з створення цифрових зображень, побудова власної моделі та автоматизація процесу ілюстрування для різних галузей

Об'єкт дослідження

Система створення цифрових зображень

Предмет дослідження

Моделі глибокого навчання з перетворення тексту в зображення

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі I Опис теоретичної основи роботи, підходи до генерації цифрових зображень, зокрема генеративні моделі на основі глибинного навчання та їх застосування у різних галузях, підходи до генерації цифрових зображень з використанням генеративно-змагальних мереж, застосування генерації цифрових зображень, опис задачі яку має виконати проект, обґрунтування користі та методи застосування проекту, економічне обґрунтування проекту

У розділі II Пошук методів для виконання проекту, порівняння методів та підходів для розробки, вивчення потреб користувачів та аналіз ринку, вибір найкращої та доцільної системи, порівняння існуючих моделей генерації, опис додаткових методів для генерування зображень, огляд доступних алгоритмів та технологій для розробки системи, їхній порівняльний аналіз з урахуванням вимог проекту, вибір найбільш доцільних та ефективних, оцінка доцільності використання додаткових методів, вивчення спеціалізованої літератури та наукових статей з теми проекту, що дозволяє удосконалити розробку та використання новітніх технологій, використання мовних моделей для аналізу тексту, опис структури роботи системи

У розділі III Проектування інтелектуальної системи, розробка власної моделі яка буде задовольняти потребу проекту, комбінації створеної моделі з додатковими методами для покращення результатів, оцінка результатів та налагодження моделі, підбір оптимальних параметрів моделі та методів, що використовуються в системі, оцінити ефективність інтелектуальної системи та порівняти її з альтернативними рішеннями, розглянути можливі шляхи покращення системи

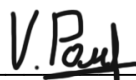
**Завдання підготував
науковий керівник**


(підпис)

Рамазанов Султан Курбанович

“20” лютого 2023 р.

**Завдання одержав
здобувач**


(підпис)

Романченко Владислав Васильович

“20” лютого 2023 р.

Календарний план

№	Назва етапів кваліфікаційної магістерської робо	Термін виконання етапів роботи	Примітка
1	Об'єктний аналіз поставленої задачі	25-01-2023	виконано
2	Розробка моделі взаємодії даних	10-02-2023	виконано
3	Розробка структури зберігання даних	15-03-2023	виконано
4	Створення коду програми	08-04-2023	виконано
5	Тестування і налагодження програми	27-04-2023	виконано
6	Підготовка пояснювальної записки	08-05-2023	виконано
7	Спецчастина	10-05-2023	виконано
8	Охорона праці та безпека в надзвичайних ситуаціях	11-05-2023	виконано
9	Підготовка презентації та доповіді	12-05-2023	виконано
10	Попередній захист	17-05-2023	виконано
11	Нормоконтроль, рецензування	18-05-2023	виконано
12	Занесення диплома в електронний архів	19-05-2023	
13	Допуск до захисту у зав. кафедри	-05-2023	

Дата видачі завдання «20» лютого 2023 р.

Керівник д.т.н., д.е.н., проф. Рамазанов С.К.

Завдання прийняв до виконання здобувач Романченко В.В.

В І Д Г У К

про кваліфікаційну магістерську роботу
здобувача **навчально-наукового інституту "Інститут інформаційних
технологій в економіці" Київського національного економічного
університету імені Вадима Гетьмана**
освітньо-професійної програми "Системи штучного інтелекту"
Романченко Владислава Васильовича

на тему «Інтелектуальна система автоматизованих ілюстрацій за допомогою
машинного навчання»

1. Актуальність теми. Через зростаючу популярність систем створення цифрових зображень за допомогою штучного інтелекту та машинного навчання, з'являється можливість оптимізувати та підвищити ефективність такої діяльності, як створення ілюстрацій. Одним з напрямків у даному дослідженні є розробка системи для автоматизації процесу створення ілюстрацій за допомогою машинного навчання. Ця система спрямована на спрощення та прискорення створення ілюстрацій шляхом використання алгоритмів та моделей машинного навчання таких як мультимодальні моделі та моделі обробки природної мови. Зважаючи на специфіку даної галузі, додаткова автоматизація зможе ще більше спростити процес створення зображень, за допомогою автоматизації процесу введення, та групування даних необхідних для вдалої генерації зображень. У даній роботі розглянуто розробку системи для автоматизації процесу створення ілюстрацій за допомогою машинного навчання.

2. Позитивні риси кваліфікаційної магістерської роботи. У даній роботі було описано сучасні підходи до моделювання алгоритмів генерації штучних цифрових зображень. Основна увага була приділена огляду актуальних методів моделювання, які використовуються в сфері генерації штучних зображень. Було проведено аналіз і порівняння різних існуючих систем та підходів, що дозволило визначити їхні переваги, недоліки та потенціал для подальшого вдосконалення. У ході дослідження було проведено аналіз показників, що мають вплив на якісну генерацію. Також був здійснений огляд актуальних методів моделювання, аналіз та порівняння існуючих систем. Для створення ілюстрацій була розроблена та покращена власна модель для генерації зображень, яка підходить для заданої мети, що використовує моделі дифузії. Були визначені системи взаємодії між факторами, які є показниками для створення якісних зображень.

3. Наявність самостійних розробок автора. В роботі розв'язано задачу створення моделі генерації зображень за допомогою набору даних а також розроблено алгоритм автоматизації процесу генерації зображень з великих об'ємів

тексту, таких як книги чи новини, через виділення головних для генерації елементів за допомогою мовних моделей, здійснено комп'ютерну реалізацію моделей.

4. Цінність теоретичних висновків та практичних рекомендацій. Запропонована інформаційна система, яка є комплексом інструментів для автоматизації створення зображень, може бути високо оцінена видавництвами та учасниками ринку. Порівняння та аналіз новітніх моделей показує їх кращі показники серед конкурентних засобів. генерація нових дизайнів та моделей для маркетингу та реклами, автоматична генерація ілюстрацій та графічних елементів для веб-сайтів та друкованих матеріалів.

5. Наявність недоліків. Системи розглянутих в роботі моделей було б доцільним більшу увагу покращенні результатів щоб не виключити потребу багаторазової генерації у пошуках задовільного результату.

6. Загальна оцінка кваліфікаційної магістерської роботи та її допущення до захисту перед ЕК: Зважаючи на те, що магістерська робота студента Романченко В.В. на тему «Інтелектуальна система автоматизованих ілюстрацій за допомогою машинного навчання» є актуальною, дослідження є самостійним і ґрунтовним, вважаю, що дана кваліфікаційна бакалаврська робота може бути допущена до захисту в ДЕК і заслуговує високої позитивної оцінки "відмінно".

Науковий керівник Рамазанов С.К., д.т.н., д.е.н, професор, професор кафедри інформаційних систем в економіці Київського національного економічного університету імені Вадима Гетьмана

_____ 

_____ Рамазанов С.К.

«10» травня 2023 р

РЕЦЕНЗІЯ

на кваліфікаційну магістерську роботу
здобувача вищої освіти

Романченко Владислава Васильовича
(прізвище, ім'я, по батькові)

Тема «Інтелектуальна система автоматизованих ілюстрацій за допомогою машинного навчання»

Актуальність теми кваліфікаційної магістерської роботи і доцільність її розроблення: Актуальність полягає у зростаючій популярності систем створення цифрових зображень. Розробка такої системи доцільна, оскільки автоматизує процес створення ілюстрацій та полегшує його.

Якість проведеного дослідження: Якість проведеного дослідження є високою. Дослідження базується на використанні сучасних методів машинного навчання, алгоритмів та моделей. Підходи, використані у дослідженні, спрямовані на оптимізацію та прискорення процесу створення ілюстрацій.

Позитивні риси кваліфікаційної магістерської роботи: Робота пропонує новий підхід до створення ілюстрацій, додатково поєднуючи обробку природної мови з процесом генерації зображень для автоматизації запитів для генерації зображень. Зауваження: Результати дослідження можуть бути ще поліпшені шляхом оптимізації алгоритмів навчання та використання більш розширених даних для тренування моделі. Обробка та генерація ілюстрацій може займати значну кількість часу і пам'яті, тому оптимізація та розподіл ресурсів можуть бути необхідними для досягнення оптимальної продуктивності системи.

Практична значимість висновків і рекомендацій: Практична значимість полягає в потенціалі вплинути на реальну практику та принести користь у певних галузях, наприклад видавництві, засобах масової інформації та ін. У роботі розглянуто та наведено порівняння різних методів та моделей а також описано процес роботи таких систем. Також успішно виконано інтеграції мовних систем в процес автоматизації створення ілюстрацій.

Місце роботи та посада рецензента МПП «Самурай», Директор
Науковий ступінь, учене звання (за наявності) Шулаєв С.М.

(підпис, ПІБ)

Підпис засвідчую:

Директор

(підпис)

Місце печатки організації, де працює рецензент



АНОТАЦІЯ

В магістерському проекті реалізовано систему штучного інтелекту для створення ілюстрацій до тексту, навчено власну модель для створення зображень на основі базової моделі Stable Diffusion, а також проведено злиття контрольних точок, для отримання кращих результатів моделі.

Система штучного інтелекту базується на новітніх досягненнях в сфері комп'ютерного зору, моделі створення цифрових зображень Stable Diffusion, та моделі мови зі штучним інтелектом з відкритим кодом GPT-J.

Система дозволяє автоматично створювати зображення з великих обсягів тексту, обираючи головні частини, оброблюючи їх, та передає створений текст для генерації зображень.

Програмна частина реалізована на мові Python 3.10, у середовищі PyCharm і Jupyter Notebook, та при використанні хмарних ресурсів Google Colab та Kaggle. Оптимізація та налаштування системи виконано за допомогою бібліотек мови Python.

ANNOTATION

In the master's project, an artificial intelligence system was implemented to create illustrations for the text, trained its own model for creating images based on the basic Stable Diffusion model, and merged control points to obtain better model results.

The artificial intelligence system is based on the latest advances in computer vision, the Stable Diffusion digital image creation model, and the GPT-J open-source artificial intelligence language model.

The system allows you to automatically create images from large amounts of text by selecting the main parts, processing them, and transmitting the created text to generate images.

The software part is implemented in Python 3.10, in the PyCharm and Jupyter Notebook environments, and using Google Colab and Kaggle cloud resources. The system was optimized and configured using Python libraries.

РЕФЕРАТ

Магістерська робота містить 96 сторінок, 45 рисунків, 3 таблиці, список літератури з 33 найменуванням та 4 додатки.

Метою магістерської роботи є проектування та розробка Інтелектуальна система автоматизованих ілюстрацій за допомогою машинного навчання.

Об'єктом дослідження є система створення цифрових зображень.

Предметом дослідження є моделі глибокого навчання з перетворення тексту в зображення.

В результаті дослідження було створено застосунок з використанням нейронної мережі.

Ключові слова: інтелектуальна система, нейронна мережа, обробка природньої мови, створення цифрових зображень.

ABSTRACT

The master's thesis contains 96 pages, 45 figures, 3 tables, a list of references with 33 titles and 4 appendices.

The purpose of the master's thesis is to design and develop an intelligent system for automated illustrations using machine learning.

The object of research is digital image creation system.

The subject of the research is deep learning models for converting text into images..

As a result of the study was creation of an application using a neural network.

Key words: intelligent system, neural network, natural language processing, digital image creation.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ СКОРОЧЕНЬ	2
ВСТУП.....	3
РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ	5
1.1 Дослідження предметної області.....	5
1.2 Огляд архітектур нейронних мереж.....	8
1.2.1 Дискримінаційне та генеративне моделювання	10
1.3 Аналіз існуючих СШІ й інтелектуальних систем предметної області	12
1.4 Постановка проблеми та формування задач	19
1.5 Обґрунтування вибору підходів і технологій для проектування та створення СШІ і їх компонентів.....	21
Висновки до розділу 1	30
РОЗДІЛ 2 ХАРАКТЕРИСТИКА СШІ. ВИБІР І ОБґРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ.....	31
2.1 Характеристика об'єкта дослідження	31
2.2 Структура і характеристика СШІ та їх компонентів.....	38
2.2.1 Автокодер (VAE).....	38
2.2.2 Мережа U-Net	38
2.2.3 Кодувальник тексту	39
2.3 Методи, моделі і моделювання процесів і елементів складних систем	43
2.3.1 Моделі та методи оптимізації СШІ	43
2.3.3 Методи та моделі управління в СШІ	48
2.4 Додавання умовного керування до моделей дифузії текст-зображення - ControlNET	57
Висновки до розділу 2	60
РОЗДІЛ 3 КОНСТРУКТИВНИЙ РОЗДІЛ. РОЗРОБЛЕННЯ ПРОЕКТНИХ РІШЕНЬ.....	61
3.1 Моделювання та проектування для системи.....	61
3.2 Навчання власної моделі Stable Diffusion.....	65
3.2.1 Тренування моделі	65
3.2.2 Комбінація моделей. Checkpoint merge	69
3.3 Розроблення користувацького інтерфейсу. Елементи та структура....	79
3.4 Автоматизація СШІ за допомогою обробки природньої мови	86
3.5 Проектування забезпечувальних підсистем СШІ. Реалізація системи	89
3.5.1 Інформаційне забезпечення	90
3.5.2 Програмне забезпечення	90
3.5.3 Технічне забезпечення	90

3.5.4 Організаційно-економічне забезпечення.....	91
Висновки до розділу 3	92
ВИСНОВКИ.....	93
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:.....	94
ДОДАТОК А – СХЕМА ПРИНЦИПОВА	97
ДОДАТОК Б – ФУНКЦІОНАЛЬНА СХЕМА	98
ДОДАТОК В – СТРУКТУРНА СХЕМА	99
ДОДАТОК Г – ЛІСТИНГ ПРОГРАМИ	100

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ СКОРОЧЕНЬ

NLP	англ. Natural language processing, обробка природної мови
КЗ	Комп'ютерний зір або Комп'ютерне бачення
DAN	Генеративна змагальна мережа
LDA	Латентний розподіл Діріхле
VAE	Варіаційний автокодувальник
DBN	Мережа глибокого переконання
RBM	Машина Больцмана з обмеженням
GMM	Модель гауссової суміші
ОПП	Обробка природної мови
CNN	Convolutional neural network, Згорткова нейронна мережа
ІІ	Штучний інтелект
GPU	Graphics Processing Unit, Графічний процесор
CPU	Central Processing Unit, Центральний процесор
ІІІМ	Штучна нейронна мережа

ВСТУП

За останні кілька десятиліть у сферах комп'ютерного зору (КЗ) та обробки природної мови (ОМП) відбулося кілька значних технологічних проривів у дослідженнях глибокого навчання. Останнім часом дослідники виявляють інтерес до поєднання семантичної та візуальної інформації в цих традиційно незалежних галузях. Було проведено низку досліджень методів синтезу текст-зображення, які перетворюють вхідні текстові описи (ключові слова або речення) в реалістичні зображення [1].

Створення ілюстрацій, таких як загальні малюнки, картини, 3D середовища, анімація персонажів та інші творчі проекти, які залежать від уяви, раніше вважалися поза межами можливостей комп'ютерних алгоритмів. Однак, завдяки новітнім досягненням у галузі ШІ, це стало можливим на протязі останніх років.

Останні роки принесли значний прорив в галузі створення комп'ютерних ілюстрацій, анімації та 3D-модельовання. Було досягнуто значного прогресу у розробці алгоритмів, що дозволяють комп'ютерам створювати складні та деталізовані графічні зображення за допомогою штучного інтелекту та машинного навчання. Одним із великих досягнень в галузі створення комп'ютерних ілюстрацій є розробка системи генерації зображень на основі глибинного навчання, яка дозволяє комп'ютеру створювати нові, невидані зображення засновані на великій кількості даних зі світу дизайну та графіки. А також генерація реалістичних зображень на основі реальних фото та відео.

Нещодавні досягнення у створенні зображень на основі штучного інтелекту, очолювані дифузійними моделями, такими як Glide, Dalle-2, Imagen і Stable Diffusion, зробили прорив у цій сфері. Створення високоякісних зображень на основі текстових описів є складним завданням. Воно вимагає глибокого розуміння основного змісту тексту і здатності генерувати зображення, яке відповідає цьому змісту. В останні роки дифузійні моделі стали потужним інструментом для вирішення цієї проблеми.

Багато фахівців у галузі 3D-мистецтва, малювання та ілюстрування вважають, що штучний інтелект з часом може замінити їх. Це правда, що в деяких аспектах роботи художників та ілюстраторів, зокрема щодо створення простих та стандартних ілюстрацій чи елементів дизайну, можуть бути застосовані штучний інтелект та комп'ютерні алгоритми. На сьогоднішній день існують системи, які здатні самостійно генерувати ілюстрації за допомогою передових технологій машинного навчання та комп'ютерного зору. Це значно підвищує продуктивність та зменшує час, необхідний для створення нових графічних елементів.

Однак, особливо корисною може бути можливість використання таких систем для створення так званих "сток фото", які використовуються для прикладів або доповнення до матеріалів, наприклад, новин. Раніше виданням доводилося займатися пошуком відповідних зображень на спеціалізованих веб-сайтах та витратити багато часу на цей процес. Тепер з використанням систем штучного інтелекту, автори можуть швидко та ефективно створити необхідне зображення, що значно зекономить час та кошти та підвищить продуктивність.

Завдяки розвитку систем комп'ютерного зору та обробки природної мови, виникає можливість швидко та ефективно створювати стокові зображення та ілюстрації без залучення професійних фотографів або художників. Такі системи, як DALL-E 2 і Stable Diffusion, можуть створювати реалістичні зображення на основі текстового опису, що дозволяє створювати нові зображення швидко та ефективно.

Більше того, такі системи можуть бути корисні не тільки для створення стокових зображень, але й для створення спеціальних ефектів у кіно, відеоіграх та візуалізації архітектурних проектів. Використання таких систем дозволяє зекономити час та зусилля при створенні якісного візуального контенту.

Таким чином, системи комп'ютерного зору та обробки природної мови відкривають нові можливості для швидкого та ефективного створення високоякісних стокових зображень та ілюстрацій, а також допомагають у візуалізації проектів на різних етапах їх розробки.

РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження предметної області

У середині 2010-х років почали з'являтися перші спроби генерації зображень з текстових описів, і це стало можливим завдяки появі генеративних адверсарних мереж (Generative Adversarial Networks, GAN).

Генеративна змагальна мережа - це система з двох штучних нейронних мереж, які конкурують між собою. Одна мережа (генератор) генерує зображення на основі текстових описів, а інша (дискримінатор) оцінює їх.

Під час навчання метою генератора є ввести в оману дискримінатор шляхом створення зображення, яке нагадує реальне. Завдання дискримінатора полягає в точному відрізненні реальних зображень від синтезованих. На Рисунку 1.1 можна побачити приклад такої генерації, створеної у 2016-му році:

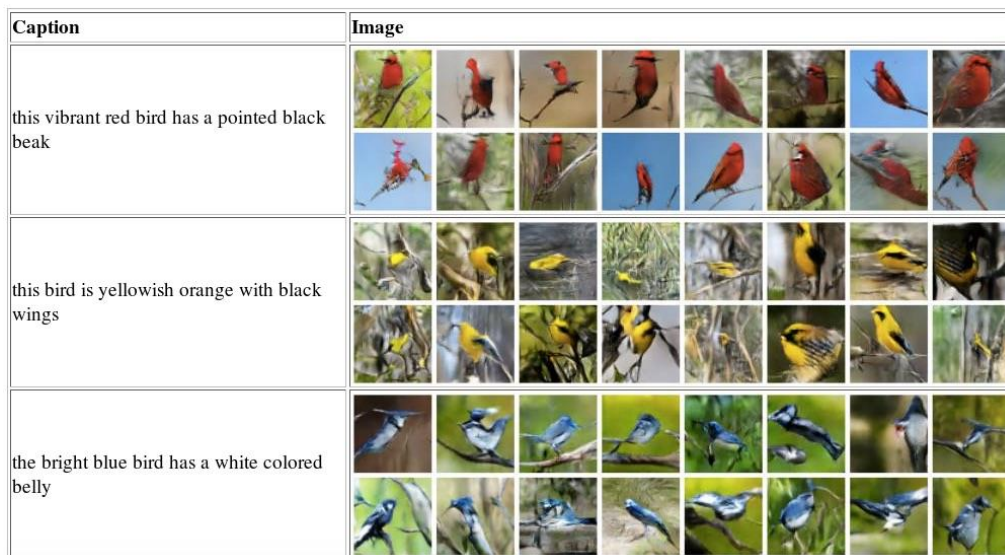


Рисунок 1.1 – Приклад згенерованих зображень птахів за допомогою генеративної мережі

Пізніше з'явилося кілька альтернативних алгоритмів для генерації зображень на основі текстових запитів, проте, на відміну від GAN, помітних покращень у якості їх виконання не відбулося.

Наступний етап розвитку - DALL-E і CLIP. Великі зміни в області перетворення тексту в зображення відбулися відносно нещодавно. Тоді OpenAI представила два рішення, які, можливо, започаткували революцію в генерації зображень, яку можна спостерігати сьогодні. Це нейронні мережі DALL-E і CLIP.

Нейромережа DALL-E базується на GPT-3, третьому поколінню алгоритму обробки природної мови від OpenAI, яка використовує архітектуру трансформера. Ця архітектура розширює текстову послідовність спеціальними токенами, які подальше перетворюються на зображення за допомогою окремої моделі (декодер).

Порівняно з попередніми рішеннями, DALL-E показав великий стрибок у якості синтезованих зображень. На особливу увагу заслуговує рівень узагальнення, притаманний нейронній мережі. Завдяки цьому DALL-E здатний генерувати приклади, яких не було включено в навчальний набір (приклади, яких немає в навчальному наборі даних). На Рисунку 1.2 наприклад, показано крісла в формі авокадо, які стали візитівкою OpenAI.

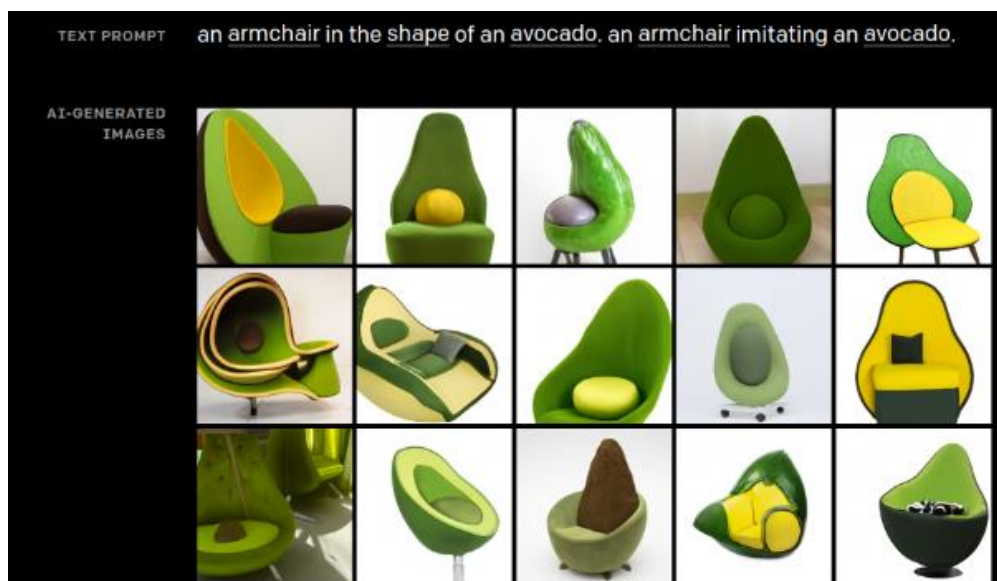


Рисунок 1.2 – Приклад згенерованих зображень птахів за допомогою генеративної мережі

CLIP - це чудовий рушій перетворення зображень у текст, який складається з двох кодерів - одного для тексту і одного для зображень.

На відміну від DALL-E, розробники виклали навчені шкали CLIP у вільний доступ. Після цього в галузі перетворення тексту в зображення почало з'являтися

багато рішень з використанням CLIP (BigGAN+CLIP, VQGAN+CLIP, CLIP Guided Diffusion та ін.)(2].

На сьогоднішній день спостерігається змагання між рішеннями з відкритим кодом, розробленими незалежною спільнотою розробників (такими як Disco Diffusion, Latent Diffusion, Stable Diffusion), та комерційними моделями великих компаній (наприклад, DALL-E 2, Imagen, Midjourney та інші). В результаті цього зростає обсяг генеративного контенту, який створюється по всьому світу.

Хоча корпорації та компанії зазвичай мають доступ до більших обчислювальних потужностей, багато генеративних митців віддають перевагу рішенням з відкритим вихідним кодом. Перевага таких рішень полягає в тому, що кожен може долучитися до розробки продукту або його додатків і спробувати, як він працює, а самі технології швидко розвиваються і еволюціонують.

На вибір між продуктами з відкритим кодом і комерційними рішеннями часто впливає цензура. Наприклад, OpenAI забороняє генерувати зображення на "чутливі теми" і банить після кількох спроб це зробити. До того ж маючи рішення з відкритим кодом, яке запускається на власних потужностях дає змогу не перейматися через витік даних, та тому подібного.

Продовження розвитку технологій.

Технології не стоять на місці. Деякі рішення роблять генерацію більш реалістичною. Наразі існує багато різних моделей генерації: Stable Diffusion, Imagen, Midjourney, DALL-E 2 та багато інших. Також розробники випустили ваги Stable Diffusion. Відтоді почало з'являтися багато рішень на їх основі. Наприклад, колаб з уже знайомим користувачам інтерфейсом Disco Diffusion, де можна додавати власні зображення для обробки, і генерувати 2D і 3D.

Крім того, команда Stable Diffusion створила API. З допомогою якого можна створювати додатки. На його основі вже доступні плагіни для Photoshop, GIMP та Blender. Існує також версія для Blender з генерацією текстур і навіть для проекту цифрового живопису з відкритим вихідним кодом Krita.

Генеративні змагальні мережі.

Генеративні адверсаріальні мережі (GAN), скорочено GAN, представляють собою підхід до генеративного моделювання, що використовує методи глибокого навчання, такі як згорткові нейронні мережі.

Генеративне моделювання представляє собою завдання машинного навчання, яке не має конкретних вказівок і полягає в автоматичному виявленні та вивченні закономірностей або шаблонів у вхідних даних. Його основною метою є створення моделей, які здатні генерувати або виводити нові приклади, що, ймовірно, відповідають вихідному набору даних.

GAN - це розумний спосіб навчання генеративної моделі, який полягає у формулюванні проблеми як задачі керованого навчання з двома підмоделями: моделлю-генератором, яка навчається генерувати нові приклади, і моделлю-дискримінатором, яка намагається класифікувати приклади як справжні (з предметної області) або хибні (згенеровані). Обидві моделі тренуються разом у змагальній грі з нульовою сумою доти, доки дискримінаторна модель не буде обманута приблизно в половині випадків, що означає, що модель-генератор генерує правдоподібні приклади.

1.2 Огляд архітектур нейронних мереж

Типовим завданням машинного навчання є використання моделі для прогнозування, наприклад, предиктивного моделювання.

Для цього необхідно мати навчальний набір даних, який використовується для тренування моделі. Набір складається з кількох прикладів, відомих як вибірки, кожна з яких містить вхідні змінні (X) та відповідні мітки вихідних класів (y). Модель навчається шляхом подання прикладів вхідних даних, передбачення вихідних даних моделлю та коригування моделі, щоб зробити вихідні дані більш схожими на очікувані.

У підході прогнозуючого або керованого навчання мета полягає у вивченні відображення від входів x до виходів y , маючи маркований набір пар входів-виходів. На рисунку 1.3 можна побачити схему роботи цього алгоритму [3].

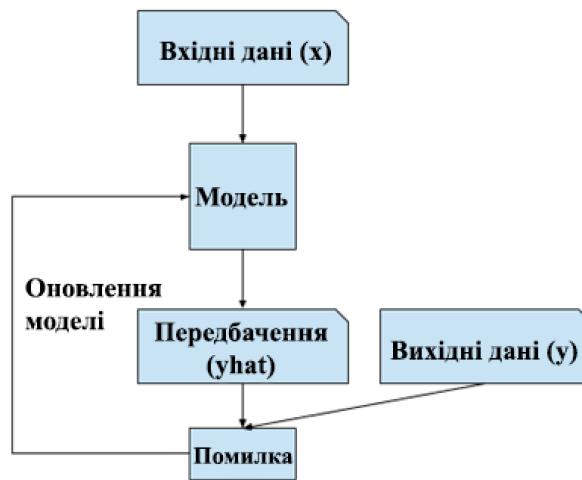


Рисунок 1.3 – Навчання під наглядом

Таку корекцію моделі зазвичай називають керованим навчанням (supervised learning). Прикладами завдань керованого навчання є класифікація та регресія, а прикладами алгоритмів керованого навчання - логістична регресія та випадковий ліс.

Існує ще одна парадигма навчання, коли модель має лише вхідні змінні (X) і не має вихідних змінних (y).

Модель будується шляхом вилучення або узагальнення закономірностей у вхідних даних. Модель не коригується, оскільки вона нічого не передбачає.

Другим важливий тип машинного навчання - це описовий або неконтрольований підхід до навчання. У цьому випадку надаються лише вхідні дані, і метою є виявлення "цікавих закономірностей" у цих даних [4]. Це завдання менш чітко визначене, оскільки не вказується, які саме закономірності потрібно знайти, і відсутня очевидна метрика помилок (на відміну від керованого навчання, де можна порівняти прогнозоване значення y для заданого x зі спостережуваним значенням). На рисунку 1.4 показана схема алгоритму [4].

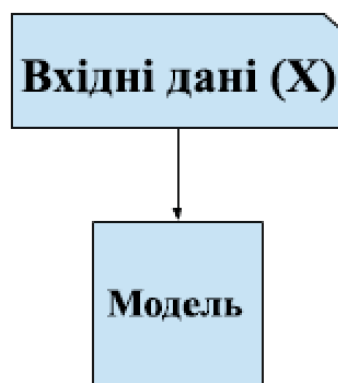


Рисунок 1.4 – Навчання без нагляду

Неконтрольоване навчання має своїм прикладами проблеми, такі як кластеризація та генеративне моделювання. Алгоритми неконтрольованого навчання включають К-середні та генеративні змагальні мережі.

1.2.1 Дискримінаційне та генеративне моделювання

У керованому навчанні можемо бути зацікавлені в розробці моделі для прогнозування мітки класу за вхідними змінними. Таке завдання предиктивного моделювання називається класифікацією.

Класифікацію також традиційно називають дискримінативним моделюванням.

Використовувати навчальні дані для знаходження дискримінантної функції $f(x)$, яка зіставляє кожен x безпосередньо з міткою класу, таким чином об'єднуючи етапи виведення і прийняття рішень в одну навчальну задачу [5].

Це пов'язано з тим, що модель повинна розрізняти екземпляри вхідних змінних за класами; вона повинна вибирати або вирішувати, до якого класу належить екземпляр.

На рисунку 1.5 показана схема алгоритму цього моделювання.

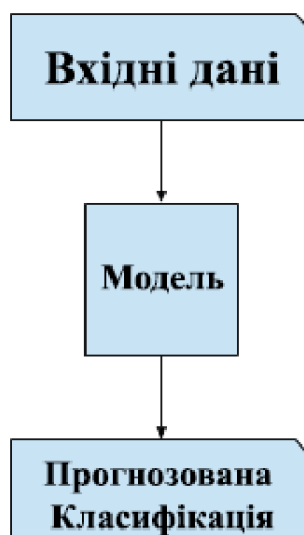


Рисунок 1.5 – Приклад дискримінаційного моделювання

З іншого боку, некеровані моделі можуть бути використані для створення або генерації нових прикладів розподілу вхідних даних, шляхом узагальнення розподілу цих змінних..

Як такі, ці типи моделей називаються генеративними моделями. Приклад схеми такої моделі показаний на рисунку 1.6

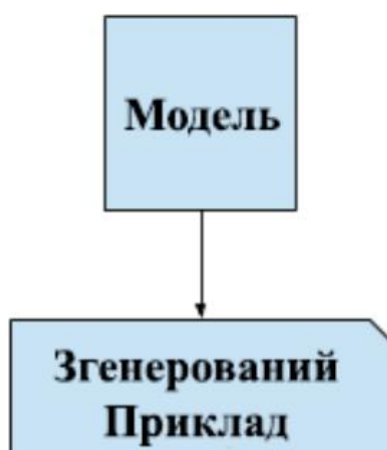


Рисунок 1.6 – Приклад генеративного моделювання

На прикладі, одна змінна може мати відомий розподіл даних, наприклад, гаусівський або дзвоноподібний. Генеративна модель здатна узагальнити цей розподіл даних і використовуватися для створення нових змінних, які відповідають розподілу вхідної змінної з високою ймовірністю.

Моделі, які явно або неявно описують розподіл вхідних і вихідних даних, називаються генеративними моделями. Ці моделі можуть використовуватись для створення синтетичних точок даних у вхідному просторі шляхом їх вибірки.

Насправді, дійсно хороша генеративна модель може генерувати нові приклади, які не тільки правдоподібні, але й не відрізняються від реальних прикладів з проблемної області [5].

1.3 Аналіз існуючих СШІ й інтелектуальних систем предметної області

Наївний байєс є прикладом генеративної моделі, яка частіше використовується як дискримінативна модель.

Наприклад, простий алгоритм наївного Байєса працює шляхом підсумовування розподілу ймовірностей кожної вхідної змінної та вихідного класу. При прогнозуванні обчислюються ймовірності кожного можливого результату для кожної змінної, і незалежні ймовірності об'єднуються, щоб прогнозувати найбільш ймовірний результат. У випадку, коли розподіли ймовірностей для кожної змінної відомі, їх можна використовувати для генерації нових правдоподібних (незалежних) значень ознак.

Інші приклади генеративних моделей включають латентний розподіл Діріхле, або LDA, і модель гауссової суміші, або GMM.

Методи глибокого навчання можуть бути використані як генеративні моделі. Двома популярними прикладами є машина Больцмана з обмеженням (RBM) і мережа глибокого переконання (DBN).

Два сучасних приклади генеративних алгоритмів моделювання глибокого навчання включають варіаційний автокодер (VAE) і генеративну змагальну мережу (GAN).

Генеративні змагальні мережі.

Генеративні змагальні мережі з англійської Generative Adversarial Networks, GAN - це генеративна модель на основі глибокого навчання.

У більш загальному сенсі, GAN - це модельна архітектура для навчання генеративної моделі, і в цій архітектурі найчастіше використовуються моделі глибокого навчання.

Архітектура GAN була вперше описана в статті 2014 року Яна Гудфеллоу та ін. під назвою "Генеративні змагальні мережі" (Generative Adversarial Networks).

Стандартизований підхід під назвою Deep Convolutional Generative Adversarial Networks, або DCGAN, який привів до більш стабільних моделей, пізніше був формалізований Алеком Редфордом та ін. у статті 2015 року під назвою Unsupervised Representation Learning with Convolutional Generative Adversarial Networks.

Більшість GAN сьогодні принаймні вільно базуються на архітектурі DCGAN[6].

Архітектура моделі генеративно-протилежної мережі (GAN) включає дві основні підмоделі: генератор і дискримінатор. Генератор відповідає за створення нових прикладів, тоді як дискримінатор має завдання класифікувати ці приклади на справжні та згенерованими моделлю генератора.

Генератор. Модель, яка використовується для генерації нових правдоподібних прикладів з проблемної області. Генератор працює, починаючи з випадкового шуму або початкового вектора.

Дискримінатор. Модель, яка використовується для класифікації прикладів як справжніх (з предметної області) або несправжніх (згенерованих).

Генеративні змагальні мережі засновані на теоретичному ігровому сценарії, в якому мережа-генератор повинна конкурувати з опонентом. Мережа-генератор безпосередньо продукує зразки. Її опонент, мережа-дискримінатор, намагається відрізнити зразки, отримані з навчальних даних, від зразків, отриманих від генератора [7].

Модель генератора.

Модель генератора приймає на вхід випадковий вектор фіксованої довжини і генерує вибірку в регіоні.

Вектор витягується випадковим чином з гаусівського розподілу і використовується для запуску генеративного процесу. Після навчання точки в цьому багатовимірному векторному просторі будуть відповідати точкам в проблемній області, формуючи стисле представлення розподілу даних.

Цей векторний простір називається латентним простором, або векторним простором, що складається з латентних змінних. Приховані змінні або латентні змінні - це змінні, які є важливими для предметної області, але не піддаються безпосередньому спостереженню.

Латентна змінна - це випадкова змінна, яку не можна спостерігати безпосередньо [7].

Латентні змінні або латентний простір часто описують як проекцію або стиснення розподілу даних. Іншими словами, латентний простір дозволяє зберегти стислу або високорівневу концепцію спостережуваних вихідних даних, таких як розподіл вхідних даних. У випадку з GAN-моделями, генератор використовує значення для точок у вибраному латентному просторі таким чином, що нові точки, отримані з латентного простору, можуть бути надіслані моделі генератора як вхідні дані і використані для створення нових і різних вихідних прикладів.

Після навчання модель генератора зберігається і використовується для генерації нових зразків. Приклад схеми моделі генератора GAN поданий на рисунку 1.7.



Рисунок 1.7 – Приклад моделі генератора GAN

Модель дискримінатора

Модель дискримінатора отримує приклад з предметної області (реальний або згенерований) на вході і пророкує бінарну мітку класу: реальний або хибний (згенерований).

Реальний приклад береться з навчального набору даних. Згенеровані приклади виводяться моделлю-генератором.

Дискримінатор - це звичайна (і добре зрозуміла) модель класифікації.

Після процесу навчання модель дискримінатора відкидається, оскільки нас цікавить генератор.

Іноді генератор можна повторно використовувати, оскільки він володіє навичками ефективного видобування ознак з прикладів у конкретній області проблем. Певні або всі шари, що виконують вилучення ознак, можуть бути використані в програмах машинного навчання з аналогічними або схожими вхідними даними.

Одним із способів створення якісних зображень є навчання генеративних змагальних мереж (GAN), а потім повторне використання частин генераторної та дискримінантної мереж як екстракторів ознак для контрольованих завдань [9]. Приклад схеми моделі дискримінатора GAN поданий на рисунку 1.8.

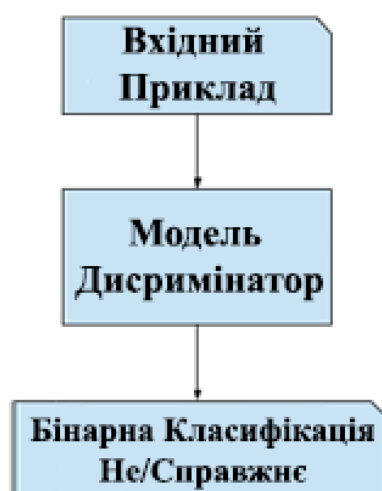


Рисунок 1.8 – Приклад моделі дискримінатора GAN

GAN як гра для двох гравців.

Генеративне моделювання є проблемою некерованого навчання, як обговорювалося в вище, хоча цікава особливість архітектури GAN полягає в тому, що навчання генеративної моделі формулюється як проблема керованого навчання.

Дві моделі, генератор і дискримінатор, навчаються разом. Генератор генерує партію зразків, які разом з реальними прикладами з предметної області надаються дискримінатору і класифікуються як справжні або хибні.

Потім дискримінатор оновлюється, щоб краще розрізняти справжні та фальшиві зразки в наступному раунді, і, що важливо, генератор оновлюється на основі того, наскільки добре згенеровані зразки обдурили дискримінатор чи ні.

Можна уявити генератор як фальшивомонетника, який намагається зробити фальшиві гроші, а дискримінатор - як поліцію, яка намагається пропустити законні гроші і зловити фальшиві. Щоб досягти успіху в цій грі, фальшивомонетник повинен навчитися робити гроші, які неможливо відрізнити від справжніх, а мережа-генератор повинна навчитися створювати вибірки, які беруться з того ж розподілу, що і навчальні дані [10].

Таким чином, дві моделі конкурують одна з одною, вони є суперницькими в сенсі теорії ігор і грають у гру з нульовою сумою.

Оскільки структуру GAN природно можна проаналізувати за допомогою інструментів теорії ігор, називаємо GAN "змагальними" [10].

У цьому випадку нульова сума означає, що коли дискримінатор успішно ідентифікує справжні та хибні зразки, він винагороджується або не потребує зміни параметрів моделі, тоді як генератор карається значним оновленням параметрів моделі. І навпаки, якщо генератор обманює дискримінатор, він отримує винагороду або не потребує змін у параметрах моделі, а дискримінатор карається, і його параметри моделі оновлюються.

Зрештою, генератор щоразу генерує ідеальні репліки з вхідної області, а дискримінатор не може їх розрізнити і пророкує "невизначеність" у кожному випадку (наприклад, 50% для справжніх і підроблених). Це лише приклад

ідеалізованого випадку; не обов'язково досягати цієї точки, щоб отримати корисну модель генератора. На рисунку 1.9 подана схема що описує вище наведене.

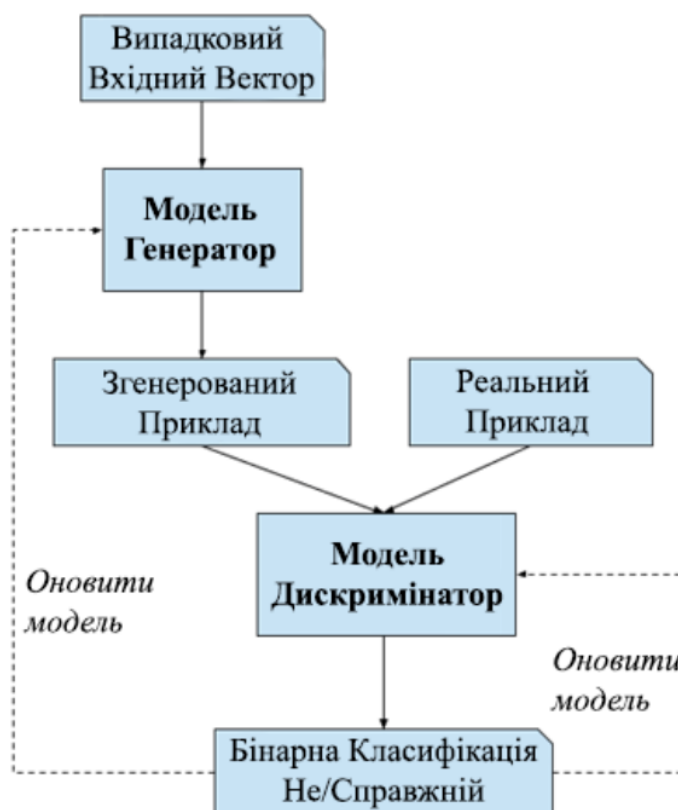


Рисунок 1.9 – Приклад архітектури моделі генеративної змагальної мережі

Навчання призводить до того, що дискримінатор намагається навчитися правильно класифікувати зразки як справжні або підроблені. У той же час генератор намагається обдурити класифікатор, щоб той повірив, що його зразки справжні. Коли вибірки збігаються, вибірки генератора не відрізняються від реальних даних, і дискримінатор скрізь дає 1/2. Після цього дискримінатор можна відкинути" [11].

GAN та згорткові нейронні мережі.

GAN зазвичай працюють з даними зображень і використовують згорткові нейронні мережі, або CNN, як моделі генератора і дискримінатора.

Це може бути пов'язано з тим, що перший опис методу був зроблений в області комп'ютерного зору і використовував ШНМ і дані зображень, а також зі значним прогресом, який був досягнутий в останні роки у використанні ШНМ в цілому для досягнення найсучасніших результатів в ряді завдань комп'ютерного зору, таких як виявлення об'єктів і розпізнавання обличь.

Моделювання даних зображень включає створення стислого представлення набору зображень або фотографій у латентному просторі, яке використовується як вхідні дані для генератора. Цей генератор створює нові зображення або фотографії, що дозволяє розробникам або користувачам моделі легко оцінити та переглянути їх вихідні результати.

Можливо, саме цей факт - можливість візуально оцінити якість згенерованого результату - призвів до зосередження додатків комп'ютерного зору на CNN, а також до величезного стрибка в можливостях GAN порівняно з іншими генеративними моделями, заснованими на глибокому навчанні або іншими способами.

Умовні GAN.

Важливим розширенням GAN є їх використання для генерації умовних виходів. Генеративну модель можна навчити генерувати нові приклади з вхідної області, де вхідні дані, випадковий вектор з латентного простору, надаються (обумовлюються) деякими додатковими вхідними даними.

Додатковим входом може бути значення класу, наприклад, чоловіча або жіноча стать у випадку генерації фотографій людей, або цифра у випадку генерації зображень рукописних цифр.

Генеративні змагальні мережі можна розширити до умовної моделі, якщо генератор і дискримінатор залежать від деякої додаткової інформації u . u може бути будь-якою допоміжною інформацією, наприклад, мітками класів або даними з інших модальностей. Можна виконати умову, подаючи u як на дискримінатор, так і на генератор як [додатковий] вхідний шар [12].

Умовний дискримінатор приймає як вхідне зображення, яке може бути реальним або фальшивим, так і додатковий вхідний сигнал. У разі наявності умовного вхідного сигналу, наприклад, класифікаційної мітки, дискримінатор очікує, що вхідний сигнал належить до цього класу. Це змушує генератор генерувати приклади, які відповідають даному класу, з метою обдурити дискримінатор.

Таким чином, умовний GAN можна використовувати для генерації прикладів з певного типу домену.

Якщо продовжувати розвивати моделі GAN, то їх можна застосовувати в предметних областях, наприклад, в зображеннях. Це відкриває широкі можливості використання GAN, такі як переклад тексту в зображення або зображення в текст. Це дозволяє досягти вражаючих результатів, таких як передача стилю, розфарбування фотографій, зміна пори року або часу доби на зображенні, а також створення нових унікальних зображень.

У випадку умовних GAN для перетворення зображення в зображення, наприклад, перетворення день-ніч, дискримінатор отримує приклади реальних і згенерованих нічних фотографій, а також (умовно) реальних денних фотографій. На вхід генератора подається випадковий вектор з латентного простору, а також реальні денні фотографії, як це показано на рисунку 1.10.



Рисунок 1.10 – Приклад архітектури моделі умовно-генеративної змагальної мережі

1.4 Постановка проблеми та формування задач

У сучасному світі комп'ютерна графіка та візуалізація даних стали важливою складовою багатьох галузей, включаючи науку, техніку та мистецтво. Проте, створення високоякісних цифрових зображень вимагає від дизайнерів та художників багато часу та енергії.

У зв'язку з цим, використання машинного навчання для створення цифрових зображень стає все більш популярним підходом. Зокрема, з використанням глибокого навчання можна навчити модель створювати зображення з високим рівнем деталізації та реалістичністю, що відкриває нові можливості для художників та дизайнерів.

Метою магістерської роботи є дослідження та розробка методів створення цифрових зображень за допомогою машинного навчання, зокрема застосування глибокого навчання для покращення якості та швидкості процесу створення зображень для ілюстрації книг.

Атоматизація створення ілюстрацій має виоский потенціал для виданництв та та друкарень, також це допоможе авторам зекономити час та кошти. Замість того, щоб наймати дорогих ілюстраторів, які можуть витрачати дні або тижні на ілюстрування книг, сьогодні можна зробити цей процес автоматичним за лічені хвилини або години, до того ж за дуже малу ціну. Технології обробки природної мови та штучного інтелекту дозволяють автоматизувати процес ілюстрування книг та інших друкованих матеріалів.

За допомогою мовних моделей GPT та інших моделей можна аналізувати текст та автоматично виділяти головну інформацію, яка корисна для ілюстрації. Наприклад, якщо розглянути книгу, можна автоматично виділити головних персонажів та середовище, в якому вони діють. Після цього можна створити автоматичну ілюстрацію, яка відповідатиме змісту книги.

Таким чином, за допомогою атоматизації ілюстрування книг можна значно зменшити витрати на найм ілюстраторів та скоротити час на створення книг. Це може бути особливо корисно для видавництв, які випускають багато книг на рік.

Для досягнення цієї мети в магістерській роботі будуть виконані наступні завдання:

- проведено огляд наукових досліджень у галузі створення цифрових зображень за допомогою машинного навчання;
- розглянуто основні методи та алгоритми глибокого навчання для створення зображень;

- розроблено та проведено експерименти для оцінки якості та швидкості створення зображень за допомогою запропонованих методів;
- порівняно результати експериментів з існуючими методами створення зображень та проаналізовано їх ефективність;
- запропоновано можливі напрямки подальших досліджень у галузі створення цифрових зображень за допомогою машинного навчання.

Робота спрямована на розробку нових методів створення цифрових зображень за допомогою машинного навчання, що можуть бути використані в різних галузях, таких як створення ілюстрацій до книг, друкарень, комп'ютерної графіки, реалізм та мистецтво.

Одним з ключових завдань магістерської роботи є дослідження різних архітектур глибоких нейронних мереж для створення зображень, таких як генеративні змагальні мережі (GAN), автоенкодери та варіаційні автоенкодери а також дифузія. Також будуть досліджені підходи до зменшення кількості параметрів моделей для забезпечення ефективності та швидкодії процесу створення зображень.

1.5 Обґрунтування вибору підходів і технологій для проектування та створення СШІ і їх компонентів

Одним з багатьох важливих досягнень у використанні методів глибокого навчання в таких галузях, як комп'ютерний зір, є техніка, що називається доповненням даних (data augmentation).

Доповнення даних покращує продуктивність моделей, підвищуючи їхню кваліфікацію і забезпечуючи ефект регулювання, зменшуючи похибку узагальнення. Це працює шляхом створення нових, штучних, але правдоподібних прикладів з вхідної проблемної області, на яких навчається модель.

Ці методи є примітивними у випадку даних зображень, включаючи обрізання, перевертання, масштабування та інші прості перетворення існуючих зображень у навчальному наборі даних.

Успішне генеративне моделювання забезпечує альтернативний і потенційно більш специфічний підхід до доповнення даних. По суті, доповнення даних є спрощеною версією генеративного моделювання, хоча його рідко називають саме так.

Розширення вибірки за рахунок латентних (неспостережуваних) даних. Це називається доповненням даних. В інших проблемах латентні дані - це фактичні дані, які повинні спостерігатися, але не спостерігаються [13].

У складних галузях або галузях з обмеженими даними генеративне моделювання забезпечує шлях до кращого навчання моделюванню. GAN досягли значного успіху в цьому випадку використання в таких сферах, як навчання з глибоким підкріпленням.

Існує багато причин, чому GAN є цікавими, важливими і вартими подальшого вивчення. Одна з них полягає у успішній здатності GAN моделювати багатовимірні дані і обробляти відсутні дані. Крім того, GAN може надавати мультимодальні результати або кілька правдоподібних відповідей, що робить їх особливо цінними для досліджень.

Мабуть, найбільш переконливим застосуванням ШНМ є використання умовних ШНМ для завдань, які вимагають генерування нових прикладів. Гудфеллоу наводить три основні приклади:

- Надвисока роздільна здатність зображень.
- Створення творів мистецтва.
- Переклад між зображеннями.

Мабуть, найважливішою причиною широкого вивчення, розвитку та використання GAN є їхній успіх. GAN змогли створити настільки реалістичні фотографії, що люди не можуть сказати, що на них зображені об'єкти, сцени і люди, яких не існує в реальному житті.

Що таке генеративні моделі.

Більшість проблем машинного навчання та глибокого навчання базуються на концепції генеративних та дискримінативних моделей. Простіше кажучи, "Генеративні моделі" - це статистичні моделі, призначені для "генерації/синтезу даних". Їхнє завдання - "перетворити шум на репрезентативну вибірку даних". Чотири відомі моделі генерації зображень на основі глибокого навчання, та їх схеми наведені на Рисунку 1.11.

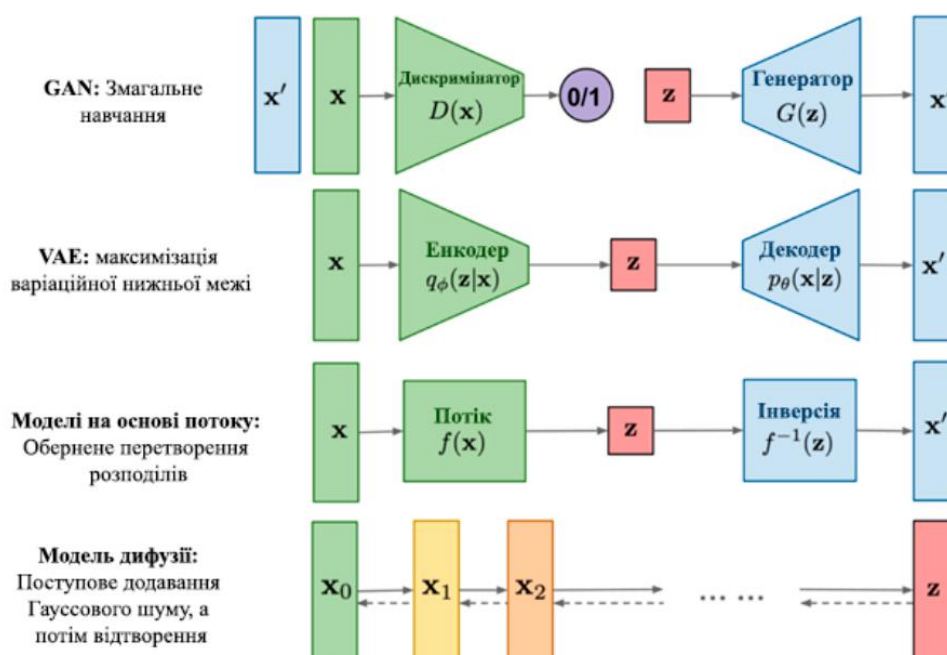


Рисунок 1.11 – Чотири моделі генерації зображень

Як можна побачити це: Варіаційні автокодери (VAE), Моделі на основі потоку (Flow), Генеративні змагальні мережі. (GAN), Дифузія (Diffusion).

Ці моделі спочатку навчаються, щоб вони навчилися моделювати "розподіл даних" (навчальні дані). Після навчання модель знає, як апроксимувати початковий розподіл даних, і може використовувати його для генерації нових даних (зображень) за власним бажанням.

В таблиці 1.1 наведено порівняльний аналіз різних типів моделей, який дозволяє зрозуміти їх переваги та недоліки. Як видно з таблиці кожен тип моделей має свої унікальні переваги та недоліки, які досить суттєво відрізняються у своєму

ключовому понятті, та більше підходять для використання у вузькоспеціалізованих задачах.

Таблиця 1.1 порівняльна характеристика моделей генерації.

Моделі	Ключове поняття	Плюси	Мінуси
Варіативні автоенкодери (VAEs)	Фреймворк кодера-декодера за допомогою імовірнісних графічних моделей, де нижня межа максимізується на лог-вірогідності даних	Одночасно виконувати генерацію та виведення з латентними змінними	Зразки зображень, згенеровані VAE, як правило, злегка розмиті. Проблеми зі зникненням KL при моделюванні послідовностей
Генеративні змагальні мережі (GANs)	Генератор-дискримінатор за допомогою змагальної навчальної гри, де безпосередньо генеруються вибірки даних	Створення найчіткіших зразків зображень	Складніше оптимізувати через нестабільну динаміку навчання
Авторегресійні моделі (NLM)	Факторизація спільного розподілу даних в умовні розподіли, моделювання кожного окремого виміру з урахуванням попередніх вимірів	Просте та стабільне навчання, що дає найкращу лог-вірогідність	NLM неефективні під час вибірки та не дають змоги отримати низькорозмірні ознаки

Попередником розуміння "варіаційних автокодерів" є "автокодери". Основне призначення автокодерів - стиснення даних. Архітектура автокодерів досить проста. Вона складається з трьох компонентів:

1. Кодер
2. Вузьке місце (відповідає за стиснення)
3. Декодер

Додатковою перевагою цієї архітектури є можливість використання автокодерів для згладжування зображень.

В автокодерах розподіл стиснутих даних/латентів є "необмеженим". Дані стискаються таким чином, щоб помилка реконструкції була мінімальною. Це призводить до основного недоліку, оскільки немає уявлення/інформації про розподіл прихованих даних, тому важко генерувати нові зразки (використовуючи лише декодер).

У варіаційних автокодерах це більше не є проблемою. Обмеження додається до шару вузьких місць, тому стиснуті дані кодера повинні якомога точніше імітувати розподіл ймовірностей. Для генерації нових відліків можна просто вибрати точку з обраного розподілу ймовірностей і передати її декодеру. Головні порівняння між методами наведені у таблиці 1.2.

Таблиця 1.2. Порівняння алгоритмів моделі генерації зображень на основі глибокого навчання

	VAE	Flow	GAN	Diffusion
Плюси	Висока частота дискретизації. Різноманітна генерація зразків	Висока частота дискретизації. Різноманітна генерація зразків	Висока частота дискретизації. Висока якість генерації зразків.	Висока якість генерації зразків. Різноманітна генерація зразків
Мінуси	Низька якість генерації зразків	Потрібна спеціалізована архітектура, низька якість генерації зразків	Нестабільне навчання, низька різноманітність генерації вибірки (Mode Collapse)	Низька частота дискретизації

Дифузія.

Моделі дифузії зображень, такі як DALL-E 2, Imagen та Stable Diffusion, привернули значну увагу завдяки своїй здатності генерувати високоякісні синтетичні зображення.

Дифузія (або процес дифузії) є добре відомою і вивченою областю в нерівноважній статистичній фізиці.

У нерівноважній статистичній фізиці під процесом дифузії розуміють "рух частинок або молекул з області з високою концентрацією в область з низькою концентрацією, викликаний градієнтом концентрації".

Процес дифузії зумовлений випадковим рухом частинок або молекул, який описується законами термодинаміки та статистичної механіки.

Що таке дифузійні моделі?

Простіше кажучи, "Дифузійні моделі - це клас імовірнісних генеративних моделей, які перетворюють шум на репрезентативну вибірку даних".

За допомогою дифузійних моделей можна генерувати зображення як умовно, так і безумовно.

Безумовна генерація зображень просто означає, що модель перетворює шум на будь-яку "випадкову репрезентативну вибірку даних". Процес генерації не контролюється і не керується, і модель може генерувати зображення будь-якої природи.

Умовна генерація зображень - це коли моделі надається додаткова інформація у вигляді тексту (text2img) або міток класів (як у CGAN). Це випадок керованої або контрольованої генерації зображень. Передаючи додаткову інформацію, очікуємо, що модель згенерує певні набори зображень.

Як працюють моделі генерації зображень на основі дифузії?

Метод дифузії можна підсумувати наступним чином:

Систематично і повільно руйнується структура в розподілі даних, використовуючи ітеративний процес прямої дифузії. Потім вивчається процес зворотної дифузії, який відновлює структуру даних, в результаті чого отримуємо дуже гнучку і легко керовану генеративну модель даних. Цей підхід дозволяє швидко навчатися, робити вибірки та оцінювати ймовірності в глибоких генеративних моделях [14].

Ідея, яка використовується в нерівноважній статистичній фізиці, полягає в тому, що можна поступово трансформувати один розподіл в інший. У 2015 році Соль-Діктейн та ін., натхненні цією ідеєю, створили "Моделі ймовірності дифузії"

(Diffusion Probability Models, або скорочено DPM), що базуються на цій важливій ідеї.

Вони створили "генеративний ланцюг Маркова, який перетворює простий відомий розподіл (наприклад, гаусівський) на цільовий розподіл (даних) за допомогою процесу дифузії".

Ланцюг Маркова просто означає, що стан суб'єкта/об'єкта в будь-якій точці ланцюга залежить виключно від попереднього суб'єкта/об'єкта.

Тепер це можна робити в обох напрямках, тобто також можливо трансформувати (невідомий) розподіл навчальних даних в інший розподіл. І обидва варіанти (дані до шуму і шум до даних) можуть бути змодельовані за допомогою однієї функціональної форми.

Це саме те, що робиться в дифузійних моделях.

Автори описують його наступним чином:

"Наша мета - визначити процес прямої дифузії, який перетворює будь-який складний розподіл даних на простий і зрозумілий, а потім вивчити процес зворотної дифузії за скінченний час, який визначає розподіл генеративної моделі" [15].

Шляхом додавання шуму до вихідного зображення поступово руйнується його структура (розподіл), а потім застосовується нейромережева модель для реконструкції зображення без шуму на кожному кроці. Після декількох ітерацій з якісними даними, модель навчиться оцінювати основний (оригінальний) розподіл даних. Потім, використовуючи навчену нейронну мережу, можна генерувати нові зображення, репрезентативні для початкової навчальної вибірки, починаючи з шуму. Процес ілюструється на рисунку 1.12.

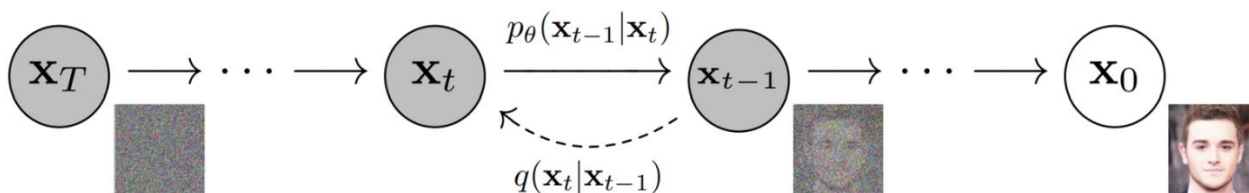


Рисунок 1.12 – Ілюстрація процесу прямої та зворотної (оберненої) дифузії

Щойно описані два основні процеси/кроки, які виконує кожна дифузійна модель. Не вдаючись у математичні деталі, розглянемо їх трохи детальніше, використовуючи рисунок 1.12 як приклад.

Пряма дифузія:

1. Початкове зображення (x_0) повільно ітеративно спотворюється (ланцюг Маркова) шляхом додавання (масштабованого гауссівського) шуму.
2. Цей процес виконується протягом T часових кроків, тобто x_T .
3. Зображення на кроці t створюється за формулою:

$$x_{t-1} + \epsilon_{t-1} (\text{шум}) \rightarrow x_t$$
4. На цьому етапі модель не використовується.
5. Наприкінці етапу прямої дифузії x_T , завдяки ітераційному додаванню шуму, залишаємось з (чистим) зашумленим зображенням, що представляє "ізотропний гаусс". Це просто математичний спосіб сказати, що маємо стандартний нормальний розподіл і дисперсія розподілу однакова по всіх вимірах. Перетворили розподіл даних на гаусівський.

Зворотна\Обернена дифузія:

1. На цьому етапі скасовуємо прямий процес. Завдання полягає в тому, щоб видалити шум, доданий під час прямого процесу, знову ж таки ітеративно (ланцюг Маркова). Це робиться за допомогою нейромережевої моделі.
2. Перед моделлю ставиться наступне завдання: Дано часовий крок t і зашумлене зображення x_t , передбачити шум (ϵ), доданий до зображення на кроці $t-1$.
3. $x_t \rightarrow \text{Модель} \rightarrow \epsilon$ (передбачений шум). Модель прогнозує (апроксимує) шум, доданий до x_{t-1} під час прямого проходу.

Порівняння з GAN:

Завдяки ітераційній природі процесу дифузії, процес навчання та генерації загалом стабільніший, ніж у GAN.

У GAN модель генератора повинна перейти від чистого шуму до зображення за один крок $x_T \rightarrow x_0$, що є одним з джерел нестабільного навчання.

На відміну від GAN, де навчання потребує двох моделей, дифузія потребує лише однієї моделі.

Одне зі спостережень, які можна зробити з наведеного вище зображення, полягає в тому, що розмірність зображення залишається постійною протягом всього процесу, на відміну від GAN, де латентний тензор може мати різні розміри. Це може створювати проблеми при генерації високоякісних зображень через обмежену пам'ять графічного процесора. Однак автори Stable Diffusion долають цю проблему, використовуючи варіаційний автокодер.

Кілька коментарів щодо процесу дифузії:

1. Ще однією перевагою ітеративної природи є те, що виконуємо контрольоване навчання на кожному часовому кроці.
2. У дифузії популярною архітектурою є UNet (з увагою), яка навчається з використанням функції втрат MSE у контрольованій манері.
3. На кожному часовому кроці шум, доданий до зображення, контролюється за допомогою "планувальника дисперсії" або просто "планувальника". Завдання планувальника - визначити, скільки шуму слід додати, щоб під час процесу прямого перетворення зображення в кінці x_T було ізотропним гауссовим.

Розглянемо деякі моделі генерації зображень на основі дифузії, які стали відомі за останній час. Найвідоміші з них:

- Dall-E 2 від OpenAI
- Imagen від Google
- Stable Diffusion від StabilityAI
- Midjourney

Вилучення навчальних даних з моделей дифузії.

Моделі дифузії зображень, такі як DALL-E 2, Imagen та Stable Diffusion, привертають значну увагу завдяки своїй здатності генерувати високоякісні синтетичні зображення.

Нові дослідження показують, що дифузійні моделі навчаються розпізнавати окремі зображення на основі своїх навчальних даних і випромінюють їх під час генерації.

Висновки до розділу 1

У розділі 1 було розглянуто принципи машинного навчання, його теоретичні аспекти, застосування до комп'ютерного зору, а також його модель та архітектуру.

Були розглянуті різні архітектури нейронних мереж для навчання систем штучного інтелекту. Проведено аналіз розвитку методів та підходів у сфері комп'ютерного зору.

Проаналізовано існуючі системи штучного інтелекту для навчання моделей генерації цифрових зображень. Описано дифузійні моделі та GAN а також порівняно їх підходи і переваги та недоліки. Завдяки глибокому навчанню ці моделі можна використовувати для створення зображень.

Описано процес генерації зображень на основі дифузії, у висновку цей процес показує себе краще ніж використання GAN, та дозволяє покращити генерацію зображень. За допомогою дифузійних моделей можна генерувати зображення як умовно, так і безумовно.

Було поставлено і обґрунтовано задачу магістрерської роботи та підходів до її виконання.

РОЗДІЛ 2 ХАРАКТЕРИСТИКА СШ. ВИБІР І ОБҐРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

Перетворення тексту в зображення існує вже давно. Спочатку воно було пов'язане з розробкою генеративних моделей, таких як генеративні змагальні мережі (Generative Adversarial Networks, GAN) та варіаційні автокодера (Variational Autoencoders, VAE).

Якщо розглядати ширше, то моделі "текст-зображення" поєднують у собі як комп'ютерний зір (КЗ), так і обробку природної мови (ОПМ).

2.1 Характеристика об'єкта дослідження

Об'єктом дослідження є методи та алгоритми створення цифрових зображень за допомогою машинного навчання. В даний час ця область отримує все більшу увагу від дослідників, оскільки це нова та перспективна галузь з високим потенціалом.

В магістерській роботі розглядалось декілька новітніх ситсем генерації зображень. Наприклад, модель Midjourney, DALL·E 2 та StableDiffusion, яка була розроблена Stability AI, є однією з найновіших моделей для створення цифрових зображень. А також має перевагу, будучи у відкритому доступі.

У цьому розділі розглянуті та порівняні різні моделі для створення цифрових зображень, зокрема моделі Midjourney, DALL·E 2 та StableDiffusion. Описано їхні принципи роботи, особливості та можливості використання.

DALL-E 2

Dall-E 2 був опублікований OpenAI у квітні 2022 року. Він ґрунтується на попередніх новаторських розробках OpenAI над GLIDE, CLIP і Dall-E. Хоча Dall-E 2 є кращим наступником Dall-E, перший містить 3,5 мільярда параметрів порівняно

з 12 мільярдами параметрів Dall-E. На чому можна зробити висновок, що кількість параметрів на обов'язково відображає результативність моделі.

Не вдаючись в архітектурні деталі, можна сказати, що Dall-E 2 - це комбінація трьох різних моделей:

- CLIP
- Попередня нейронна мережа
- Нейромережа-декодер

DALL-E 2, було проведено автоматизоване оцінювання естетичної якості для порівняння unCLIP та GLIDE. Метою оцінки було визначити, наскільки добре кожна модель створює художні ілюстрації та фотографії. 512 "художніх" підписів було створено за допомогою GPT-3 шляхом введення підписів до існуючих ілюстрацій. Для кожної моделі та набору гіперпараметрів вибірки було згенеровано чотири зображення для кожної підказки, а також повідомлено про середню передбачувану естетичну оцінку для всієї партії з 2048 зображень.

Результати показали, що підказки покращують естетичну якість як для GLIDE, так і для unCLIP [17]. Для unCLIP контролювали лише декодер. Естетичну якість також порівнювали з Recall5, оскільки підказки зазвичай призводять до компромісу між точністю та різноманітністю. Цікаво, що керування unCLIP не зменшило Recall, водночас покращивши естетичну якість за цією метрикою. Ієрархічна генерація зображень на основі тексту з латентними CLIP пропонує двоетапну модель генерації зображень з використанням контрастних моделей, таких як CLIP. На першому етапі генерується CLIP-зображення, вбудоване в текстовий підпис, тоді як на другому етапі генерується зображення, залежне від вбудованого зображення. Модель показує, що явна генерація зображень покращує різноманітність зображень з мінімальними втратами у фотореалістичності та схожості підписів.

Модель використовує дифузійні моделі для декодера і експериментує як з авторегресійними, так і з дифузійними моделями для попередньої обробки, виявляючи, що останні є обчислювально ефективнішими і дають кращі зразки [18]. Модель поєднує в собі декодер вбудовування CLIP-зображень з моделлю

попередньої обробки, яка генерує можливі CLIP-зображення із заданого текстового підпису [18]. На рисунку 2.1 показані підказки та відповідні згенеровані зображення за допомогою моделі.



Рисунок 2.1 – Згенеровані зображення за допомогою Dall-E 2

Imagen

Після Dall-E 2, лише через місяць, у квітні 2022 року, Google випустив свій алгоритм генерації зображень на основі дифузії. Вони влучно назвали його Imagen (для генерації зображень).

Він побудований на основі великих трансформаційних мовних моделей. Для своєї публікації автори Imagen обрали трансформаційну модель мови T5-XXL. Після цього Imagen складається ще з трьох дифузійних моделей для генерації зображень:

Дифузійна модель для генерації зображення з роздільною здатністю 64×64 .

За нею йде дифузійна модель надвисокої роздільної здатності для збільшення роздільної здатності зображення до 256×256 .

І остання - модель надвисокої роздільної здатності, яка збільшує роздільну здатність зображення до 1024×1024 .

На момент написання роботи Imagen від Google не доступний для широкого загалу і доступний лише за запрошенням.

Stable Diffusion

Розроблена StabilityAI, Stable Diffusion базується на статті "Синтез зображень високої роздільної здатності з моделями латентної дифузії" Ромбаха та ін. Це єдина

модель генерації зображень на основі дифузії з відкритим вихідним кодом у цьому списку.

Зокрема, від моменту її виходу спільнота розробників з відкритим вихідним кодом активно займається розробкою. За короткий проміжок часу вони випустили кілька стабільних моделей дифузії з відкритим вихідним кодом, які були покращені на різних наборах даних з різними художніми стилями. Ці моделі можна вільно використовувати для генерації нових зображень, застосовуючи зазначені стилі.

Тому, що штучний інтелект здатний вивчати та аналізувати великі обсяги даних, його можна використовувати для створення різноманітних візуальних творінь. Це може бути що завгодно - від казуальних ілюстрацій та коміксів до абстрактних композицій та мистецьких інсталяцій

Повна архітектура Stable Diffusion складається з трьох моделей:

- Кодер тексту, який приймає текстові підказки.
 - Перетворення текстових підказок у вектори, які можна прочитати на комп'ютері.
- U-Net
- Варіаційний автокодер, що складається з моделі кодера та декодера.
- Кодер використовується для зменшення розмірів зображення. Дифузійна модель UNet працює з цим зменшеним розміром.
- Декодер відповідає за покращення/реконструкцію зображення, згенерованого дифузійною моделлю, до його початкового розміру.

Stable Diffusion здатна генерувати різні зображення, приклад зображень з новітньої моделі Stable Diffusion XL показані на рисунку 2.2, проте ця модель ще не доступна в публічному доступі.



Рисунок 2.2 – Приклад зображень створених Stable Diffusion

Midjourney

Midjourney - ще одна модель генерації зображень на основі дифузії, розроблена однойменною компанією. Midjourney стала доступною широкому загалу в березні 2020 року. Вона швидко завоювала велику кількість прихильників завдяки своєму виразному стилю і тому, що стала загальнодоступною раніше, ніж DALL-E і Stable Diffusion. Декілька згенерованих моделлю зображень можна побачити на рисунку 2.3.

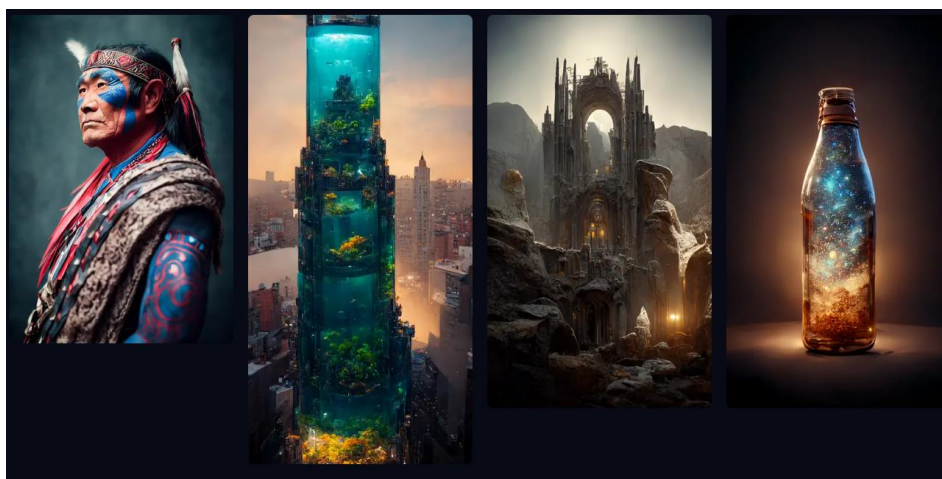


Рисунок 2.3 – Приклад зображень згенерованих Midjourney

Порівняння моделей.

Midjourney стала відомою завдяки своєму фірмовому стилю, та можливості створювати дуже незвичні та артистичні зображення. Згенеровані моделлю цифрові зображення виділяються високим контрастом та артистичною перспективою, надаючи перевагу високій деталізації та «фільмовим» кольорам. Проте в порівнянні з іншими моделями, в Midjourney дещо гірше розуміння запитів, а також стиль зображень створений за допомогою моделі, став очевидним підписом того що зображення створене за її допомогою.

В той же час модель здатна генерувати реалістичні зображення, а нова версія Midjourney, яка вийшла нещодавно, здивувала світ своїм реалізмом.

Проте Midjourney доступна лише через застосунок Discord – платформою обміну повідомленнями та аудіо та відео дзвінками, модель працює завдяки створеній у застосунку спільності-серверу, інтерфейс для запитів показано на

рисунку 2.4. Це сильно знижує гнучкість використання моделі у комерційних застосунках, хоча є можливість додати Midjourney до власного серверу.

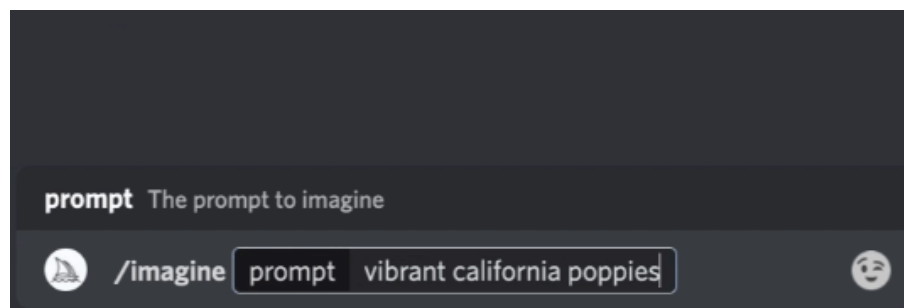


Рисунок 2.4 – Інтерфейс запиту Midjourney через чат застосунку Discord

Також варто зазначити що модель працює на серверах розробників і є монетизованою, з розцінками від 10 доларів за місяць яка надає 3.3 години на генерації місяць, чого може бути недостатньо для задумки магістерської роботи. Більш дорогі плани дозволяють використовувати «швидкі» генерації обмежений час, та мати безліміт на менш пріоритетні звичайні генерації. Усі плани підпадають під загальні комерційні умови. Можливості запустити модель особисто на даний момент не передбачається.

Останнє що можна відмітити це те що в генерації зображень моделі Midjourney присутня цензура, проте це скоріше можна відмітити як позитив, адже це не стане проблемою для більшості задач.

Dalle-2 зробила своєю візитною карткою глибоке розуміння зпитів для генерації зображень. На момент виходу моделі ці показники в неї були дещо кращими в порівнянні з Midjourney та Stable Diffusion.

Як і Midjourney, Dall-E-2 не має відкритого вихідного коду, але у нього є API, це дозволяє використовувати його для більших рішень. Загалом якісь виданих зображень порівнювальна з іншими моделями. В порівнянні з Midjourney Dall-E-2 має декілька додаткових методів для обробки зображень. Серед них outpainting – можливість розширювати зображення за межі оригінального полотна. Inpainting - реалістичне редагування існуючих зображень із підписів природною мовою.

Вартість Dall-E-2 залежить від кожного зображення, та коштує «кредити», вартість 115 «кредитів» становить 15 доларів, а створення 4 зображень з низькою

роздільною здатністю коштуватиме 1 «кредит». А зображення більшої якості потребують більше.

Stable Diffusion одна з небагатьох моделей у відкритому доступі, і серед них є однією з найкращих та здатна конкурувати з Midjourney та Dall-E-2. Проте для запуску на комп'ютері вона потребує достатньо потужні та новітні компоненти. Але за це означає більше контролю, краще налаштування і навіть створення власних інструменти ШІ. У таблиці 2.1 наведено порівняння моделей, дати їх появи у публічному доступі та ліцензія на їх використання.

Таблиця 2.1 Порівняння моделей генерації зображень.

Назва моделі	Дата публічного доступу	Автори	Ліцензія
DALL-E	Січень 2021	OpenAI	Приватна
Imagen	Не публічна	Google	Приватна
DALL-E-2	Липень 2022	OpenAI	Приватна
Midjourney	Липень 2022	Midjourney	Приватна
Stable Diffusion	Серпень 2022	StabilityAI та ін.	CreativeML Open RAIL-M

Також існує багато застосунків які пропонують безкоштовно генерувати зображення за допомогою моделей Stable Diffusion, наприклад створена самими StabilityAI – DreamStudio, або частково безкоштовна Leonardo.ai, проте є безліч інших.

Відкритість Stable Diffusion дозволяє модифікувати та донавчати моделі, створюючи спеціалізовані моделі для конкретних завдань. Завдяки відкритості Stable Diffusion вона є найкращим кандидатом для використання в дипломному проекті.

2.2 Структура і характеристика СШІ та їх компонентів

У латентній дифузії є три основні компоненти:

- Автокодер (VAE);
- U-Net;
- Текстовий кодер, наприклад, Text Encoder від CLIP.

2.2.1 Автокодер (VAE)

Автокодер (VAE) складається з двох основних частин: кодера та декодера. Кодер перетворює зображення в низькорозмірне латентне представлення, яке буде вхідними даними для наступного компонента, U_Net. Декодер виконує зворотну операцію, яка перетворює латентне представлення назад у зображення.

Кодер використовується для отримання прихованого представлення (латентних) вхідних зображень для процесу прямої дифузії під час навчання латентної дифузії. В той час як під час виводу, декодер VAE перетворює приховане представлення назад в зображення.

2.2.2 Мережа U-Net

U-Net також складається з кодера і декодера, і обидва складаються з блоків ResNet. Кодер стискає зображення в зображення з нижчою роздільною здатністю, а декодер декодує зображення з нижчою роздільною здатністю назад у зображення з вищою роздільною здатністю.

Щоб запобігти втраті важливої інформації під час дискретизації U-Net, зазвичай додаються короткі з'єднання між ResNet-блоками кодера з низькою дискретизацією та ResNet-блоками декодера з високою дискретизацією.

Як показано на рисунку 2.5, мережа має шаровий характер, в якому кожний наступний крок робить частину роботи. Кожен шар мережі отримує вхідні дані з попереднього шару та виконує певну операцію дискретизації, на цих даних, після чого віддає результат наступному шару.

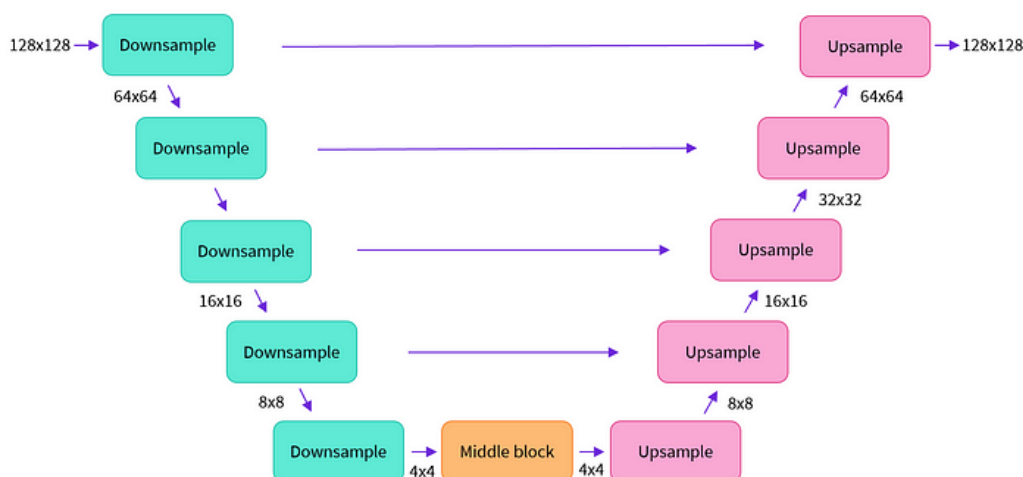


Рисунок 2.5 – Мережа U-Net

Крім того, стабільна дифузійна U-мережа здатна залежати від вбудовування тексту через шари перехресної уваги. Шари перехресної уваги додаються як до кодерної, так і до декодерної частини U-Net, зазвичай між блоками ResNet.

2.2.3 Кодувальник тексту

Текстовий кодер перетворить запит на введення, наприклад, «Швидка руда лисиця перестибнула через сіру собаку», у простір вбудовування, зрозумілий для U-Net. Це буде простий кодер на основі трансформера, який відображає послідовність токенів у послідовність прихованих текстових вбудовувань. Після цього, U-Net виконує свою основну функцію і застосовує конволюційні шари, щоб створити карту ознак з вбудовування. Ця карта ознак містить інформацію про

кожну частину запиту, яка потрібна для створення відповіді. На рисунку 2.6 ілюстровано візуальне перетворення тексту на бінарний набір.



Рисунок 2.6 – Ілюстрація кодувальника тексту, векторизація документу

Важливо використовувати хорошу підказку, щоб мати можливість отримати очікуваний результат. Саме тому зараз в тренді тема швидкого інжинірингу. Інженерія підказок – це пошук певних слів, які можуть спонукати модель видавати вихідні дані з певними властивостями.

Наприклад якщо до підказки моделі передати слово «ілюстрація», то модель швидше за все спробую згенерувати саме малюнок. В той же час якщо чітко вказати що бажана генерація має бути «фото», то отримаємо реалістичне зображення.

Деякі ідеї щодо підказок можуть бути оманливими, так, якщо попросити «реалістичне зображення», замість реалістичного зображення можна отримати щось що намагається ним бути, наприклад якісний 3D рендер, або реалістичний малюнок. Це відбувається через те що модель навчені на прикладах, і коли в реальному житті щось має підпис «реалістичне», воно зазвичай імітує реалізм а не є ним.

На практиці виявляється, що відбір відповідних слів для генерації зображення може визначати різницю між бажаним результатом та повним безладом, складеним з несполучних елементів. Stable Diffusion, крім того, надає можливість використовувати "негативні підказки" для генерації. Парадоксально, виявляється, що додавання небажаних аспектів, таких як "невиразність" або "розмитість", може покращити результат створення зображення.

Загалом для отримання бажаного результату доводиться доволі довго підбирати та міксувати свій запит, проте все залежить від випадковості. Так як seed для генерації вибирається випадковим числом, можна отримати бажаний результат з першого разу, або на це знадобиться безліч часу.

Чому латентна дифузія швидка та ефективна.

Причина, чому латентна дифузія є швидкою та ефективною, полягає в тому, що U-мережа латентної дифузії працює в низьковимірному просторі. Це зменшує обсяг пам'яті та обчислювальну складність порівняно з дифузією в піксельному просторі[19]. Наприклад, автокодер, що використовується у стабільній дифузії, має коефіцієнт зменшення 8. Це означає, що зображення фігури (3, 512, 512) стає (4, 64, 64) у латентному просторі, що вимагає в 64 рази менше пам'яті.

Stable Diffusion під час виведення (висновку).

Спочатку модель стабільної дифузії бере на вхід латентний зразок і текстову підказку. Потім латентний код використовується для генерації випадкових латентних зображень розміром 64×64 , тоді як текстова підказка перетворюється на текстові вставки розміром 77×768 за допомогою текстового кодера CLIP[20].

Далі U-Net ітераційно зменшує рівень шуму випадкових прихованих зображень, враховуючи текстові вставки. Вихідні дані U-Net, які є залишковим шумом, використовуються для обчислення деномінованого прихованого зображення за допомогою алгоритму-планувальника.

Після отримання деномінованого зображення воно порівнюється з оригіналом за допомогою функції втрат. Ця функція визначає ступінь відповідності денатурованого зображення оригіналу. Після оцінки функції втрат градієнти передаються назад через всю мережу для оновлення параметрів і покращення якості згенерованих зображень. Цей процес повторюється протягом численних епох, доки результат не буде вважатися задовільним.

Алгоритми планувальника обчислюють прогнозоване денатуроване зображення на основі попереднього представлення шуму і прогнозованого залишку шуму, як показано на рисунку 2.5.

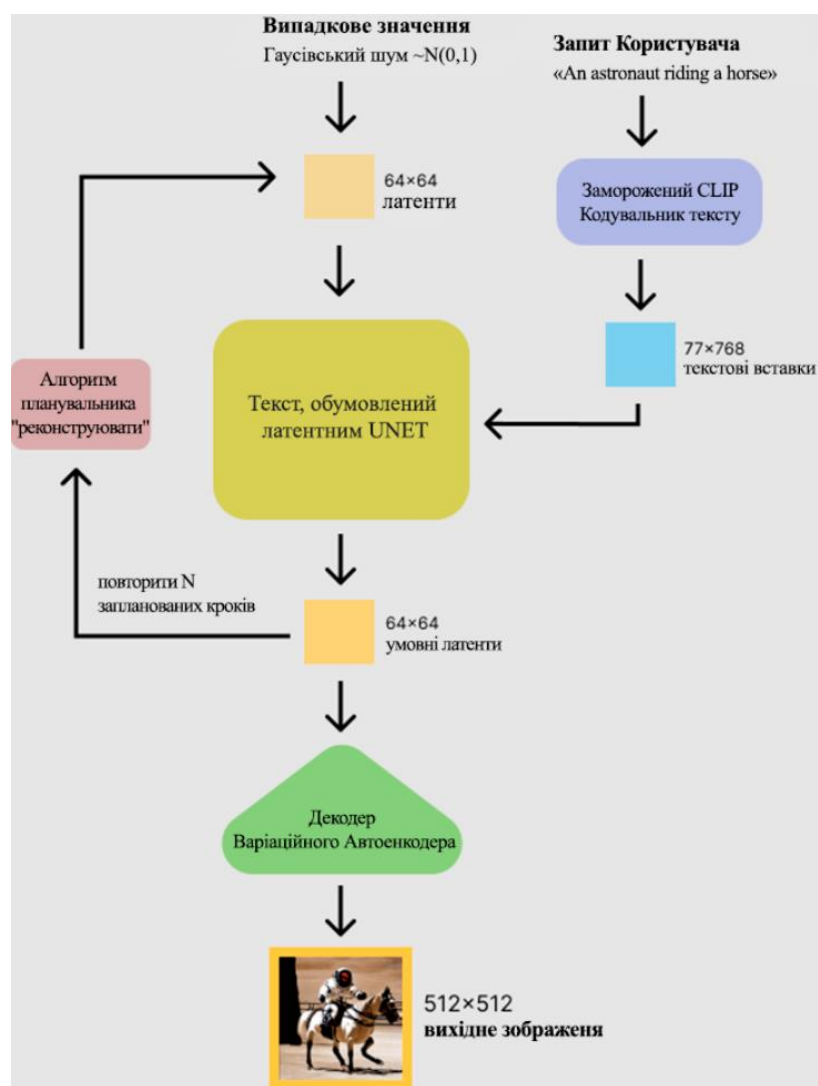


Рисунок 2.7 – Потік роботи моделі стабільної дифузії

Існує багато різних алгоритмів планувальника, які можна використовувати для цього розрахунку, кожен з яких має свої переваги та недоліки. Наприклад, для досягнення стабільної дифузії можна використовувати наступні алгоритми:

- Планувальник PNDM (використовується за замовчуванням)
- DDIM-планувальник
- K-LMS планувальник

Заради отримання кращого латентного зображення процес зменшення шуму повторюється приблизно 50 разів, крок за кроком. Після завершення цього процесу, представлення прихованого зображення декодується за допомогою декодерної частини варіаційного автокодера.

2.3 Методи, моделі і моделювання процесів і елементів складних систем

У сучасному світі складні системи стали невід’ємною частиною різноманітних галузей науки та техніки. Їхнє проектування та ефективне функціонування потребують використання новітніх методів та моделей, що дозволяють моделювати та оптимізувати процеси та елементи складних систем.

Оскільки складні системи можуть містити значну кількість елементів, що взаємодіють між собою, їхнє моделювання може бути складним завданням.

У даному розділі розглянуто методи та моделі, що використовуються для оптимізації та управління складними системами. Зокрема, розглянуто моделі та методи оптимізації СШІ, що дозволяють покращити їхню ефективність.

2.3.1 Моделі та методи оптимізації СШІ

Штучний інтелект є однією з найбільш динамічно розвиваючихся галузей інформаційних технологій в сучасному світі. Він знаходить застосування в багатьох галузях, від автоматизації промислових процесів до медицини та фінансів. Але, як і у будь-якій іншій сфері, важливо мати ефективні інструменти для розробки та оптимізації систем штучного інтелекту. Моделі та методи оптимізації є невід’ємною частиною роботи в цій галузі. Вони дозволяють створювати більш точні та швидкі алгоритми, що забезпечують високу якість та ефективність роботи систем штучного інтелекту. У цьому контексті, дослідження та розробки нових моделей та методів оптимізації є важливим завданням для розвитку штучного інтелекту та покращення його застосування в реальних умовах.

Оптимізація дозволяє краще використовувати ресурси GPU та CPU, що в свою чергу забезпечує більш швидку та ефективну роботу систем штучного інтелекту. Моделі оптимізації можуть бути використані для покращення

різноманітних систем, включаючи нейронні мережі, глибинне навчання та інші алгоритми машинного навчання. Оптимізація моделей зменшує їх складність та ресурсовитратність, що робить їх більш доступними для застосування в більш широкому спектрі завдань. Також вона може допомогти покращити точність та ефективність алгоритмів машинного навчання, що є важливими факторами у розробці різноманітних систем штучного інтелекту.

Є різні методи оптимізації систем штучного інтелекту, такі як оптимізація гіперпараметрів, оптимізація ваг моделі, оптимізація алгоритму навчання, та інші. Оптимізація гіперпараметрів полягає в пошуку найкращих значень параметрів моделі, які не вивчаються в процесі навчання, а задаються вручну. Оптимізація ваг моделі спрямована на зменшення розміру моделі та її швидкість роботи, не втрачаючи якості результатів. Оптимізація алгоритму навчання включає в себе розробку нових методів навчання моделей, які дозволяють отримувати кращі результати швидше та з меншими витратами на ресурси. Результатом оптимізації є більш точні, ефективні та швидкі системи штучного інтелекту, що забезпечують більш якісні та швидкі рішення у різних галузях, таких як медицина, фінанси, транспорт, енергетика та інші.

xFormers.

Бібліотека xFormers – це додатковий спосіб прискорити генерацію зображень. Доступна лише для графічних процесорів NVIDIA, ця оптимізація прискорює генерацію зображень і зменшує використання відеопам'яті завдяки недетермінованим результатам [21].

xFormers визначається такими пунктами:

- Налаштовувані будівельні блоки: Незалежні/настроювані будівельні блоки, які можна використовувати без шаблонного коду. Компоненти є доменно-діагностичними, і xFormers використовується дослідниками в галузі комп'ютерного зору, ОПП тощо.
- Спочатку дослідження: xFormers містить найсучасніші компоненти, які ще не доступні в основних бібліотеках, таких як pytorch.

- Створено з думкою про ефективність: Оскільки швидкість ітерацій має велике значення, компоненти є максимально швидкими та ефективними щодо пам'яті. xFormers містить власні ядра CUDA, але при необхідності звертається до інших бібліотек.

Встановлення xFormers.

Спочатку потрібно клонувати репозиторій xFormers, і створити venv:

```
git clone https://github.com/xformers.git
cd xformers
git update --init
python -m venv
./scripts/activate
```

Встановлення останньої стабільної версії з conda:

```
conda install -c xformers
```

Бінарні файли для розробки:

```
conda install -c xformers/label/dev
```

Після чого xFormers можна згенерувати та використовувати, що стосується ефективності цієї бібліотеки, показники xformers можна побачити на рисунку 2.8 [24].

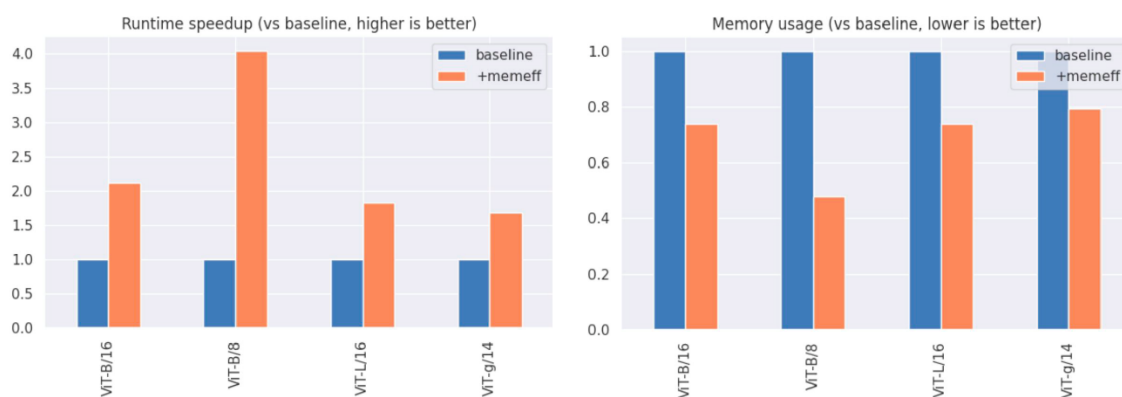


Рисунок 2.8 – Порівняння показників на A100 на f16, виміряний загальний час для передачі вперед+назад

Як видно з порівняння xformers здатний пришвидшити генерацію майже в 2 рази.

Ключові концепції трансформерів можуть охопити більшу частину сучасного стану речей. На рисунку 2.9 представлено класичний огляд архітектури трансформерів [22].

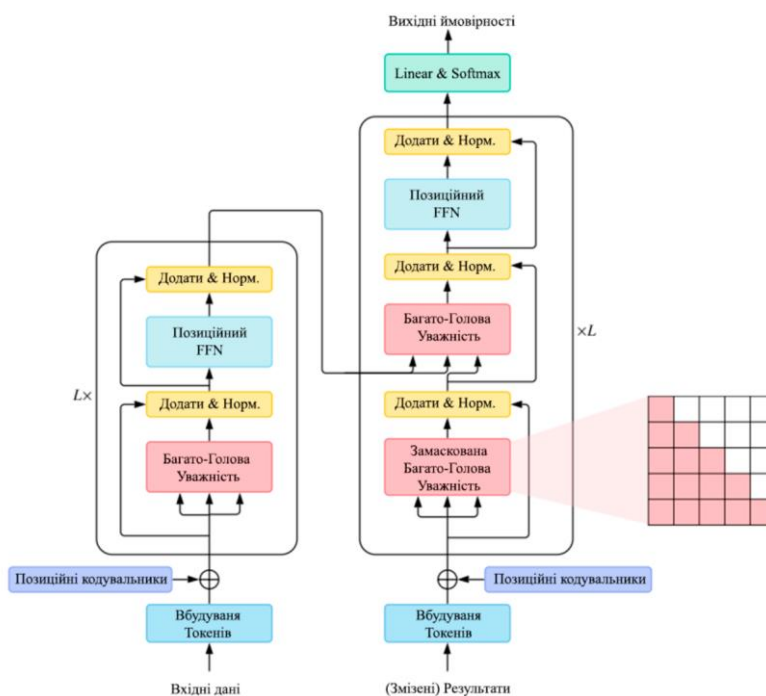


Рисунок 2.9 – Архітектура трансформерів

На рисунку 2.9 можна побачити ключові межі сховища: трансформер зазвичай складається з набору механізмів уваги, вбудовувань для кодування певної позиційної інформації, блоків прямого поширення та залишкового шляху (зазвичай його називають нормою до або після шару). Ці межі працюють не для всіх моделей, але на практиці виявлено, що за певних умов вони можуть охопити більшу частину сучасного стану речей.

Таким чином, моделі не реалізуються в монолітних файлах, які, як правило, складно обробляти і модифікувати. Більшість концепцій, представлених на наведеній вище ілюстрації, відповідають рівню абстракції, і коли для певного підблоку присутні варіанти, завжди повинна бути можливість вибрати будь-який з них.

В дипломній роботі xFormers використовується для оптимізації та пришвидшенню генерації зображень.

У попередніх розділах було розглянуто кілька відомих моделей дифузії, а серед них єдиними, що доступні у вільному доступі, є модель Stable Diffusion та її різновиди. Благодаря відкритому вихідному коду, спільнота розробників і дослідників зуміла винахідливі способи використання стабільної дифузії. Відкритість Stable Diffusion має велике значення і є корисною. Однією з головних переваг відкритого вихідного коду є можливість будь-якому бажаючому переглянути код, змінити його та використовувати власні модифікації в додатках і проектах. Це означає, що стабільна дифузія може бути налаштована під конкретні потреби користувача, а також може служити початковою точкою для розробки нових технологій і проведення досліджень.

Крім того, відкритий код підвищує довіру до системи, оскільки будь-хто може перевірити, що система дійсно працює так, як очікується. Це зменшує ризик помилок та допомагає забезпечити більшу стабільність та безпеку системи.

Крім вищевказаного, важливо зазначити, що відкритий вихідний код моделі Stable Diffusion також сприяє створенню спільноти користувачів та розробників, які можуть обмінюватись знаннями та досвідом у використанні цієї моделі. Також відкритість Stable Diffusion дає можливість налагоджувати та оптимізовувати систему за потребами користувача. Більше того, відкритість дозволяє спільноті розробників та дослідників вдосконалювати систему, що може покращити її швидкість та точність у порівнянні з конкуруючими рішеннями. Це допомагає розвивати технології комп'ютерного зору та обробки природної мови, покращувати якість та швидкість роботи систем, а також забезпечувати безпеку та захист особистих даних користувачів.

Таким чином, відкритість Stable Diffusion робить її привабливішою для користувачів, дослідників та розробників, що підвищує її потенційну ефективність та популярність на ринку технологій комп'ютерного зору та обробки природної мови.

У цьому розділі розглянуто деякі з найвідоміших застосувань і способів використання моделей дифузії.

2.3.3 Методи та моделі управління в СШІ

Текстова інверсія.

Шляхом використання текстової інверсії можна змінити налаштування моделі дифузії таким чином, щоб вона могла генерувати зображення з особистими об'єктами або в художніх стилях, використовуючи лише 10 зображень. Це не є самостійним додатком, а скоріше розумним трюком для навчання моделей дифузії, які дозволяють створювати більш персоналізовані зображення.

Від авторів статті про інверсію тексту:

«Ми вчимося генерувати конкретні концепції, такі як особисті об'єкти або художні стилі, описуючи їх новими «словами» у просторі вбудовування попередньо навчених моделей перетворення тексту в зображення. Їх можна використовувати в нових реченнях, як будь-яке інше слово» [23].

Текст до відео.

Як підказує сама назва, цей підхід використовує модель дифузії для прямого створення відео на основі текстових інструкцій. Розширюючи концепцію, що використовується для перетворення тексту в зображення, можна використовувати моделі дифузії для створення відео з розповідей, пісень, віршів та інших форм текстового вмісту.

Існують й інші (можливо, кращі) моделі дифузії тексту у відео, такі як Imagen Video від Google і Phenaki та Make-A-Video від Meta. Але, на жаль, їхній код наразі не є загальнодоступним.

Текст до 3D.

Це застосування було продемонстровано в статті «Dreamfusion», де автори використали NeRF для використання навченої 2D моделі дифузії текст-зображення для виконання синтезу тексту в тривимірне зображення.

«Отриману 3D-модель заданого тексту можна розглядати під будь-яким кутом, переосмислювати за допомогою довільного освітлення або компонувати в будь-яке 3D-середовище. Цей підхід не вимагає ніяких навчальних 3D-даних і

ніяких модифікацій моделі дифузії зображень, демонструючи ефективність попередньо навчених моделей дифузії зображень як пріоритетних» [24].

Текст до руху.

Це ще одне перспективне і цікаве застосування, в якому дифузійні моделі (разом з іншими методами) використовуються для генерування простих рухів людини. Зокрема, мова йде про Human Motion Diffusion Model.

«Генерування природних і виразних рухів людини – це Святий Грааль комп’ютерної анімації. Це складне завдання через різноманітність можливих рухів, чутливість людського сприйняття до них і складність їх точного опису». Ми показуємо, що модель навчається з невеликими ресурсами, але при цьому досягає найкращих результатів у провідних тестах для перетворення тексту в рух і дії в рух» [25].

Зображення до зображення.

Img2Img (скорочено від "Image to Image") - це метод, що дозволяє модифікувати існуючі зображення. Цей метод полягає у перетворенні вихідного зображення на цільову область згідно з текстовою підказкою. Іншими словами, можна створити нове зображення з ідентичним вмістом, але з внесеними змінами. Можливо використовувати текстові описи для більш детального визначення бажаної трансформації.

Користувач може використовувати вхідне зображення як еталон або вихідну точку і змінювати його, надаючи моделі інструкції через підказку. Дифузійна модель генерує нове зображення, яке має такий самий колір і склад, як вхідне зображення, з внесеними змінами, або може, навпаки, змінити лише колір, застосовуючи широкий спектр можливостей.

Ще один спосіб використання Img2Img – перенесення стилю. При перенесенні стилю береться два зображення: одне для змісту і одне для стилю. Створюється нове зображення, яке є сумішшю обох: зміст першого в стилі другого.

Зафарбовування зображень – Image Inpainting.

Зображення можна зафарбувати за допомогою техніки реставрації, що дозволяє видалити небажані об’єкти на зображенні або замінити їх іншими

об'єктами, текстурами або дизайном. Для цього користувач спочатку створює маску навколо об'єкта або пікселів, які потрібно змінити, а потім вказує моделі, як саме змінювати замасковані пікселі.

Для корекції можливо дати моделі більше або менше контексту з навколишнього простору, або і зовсім зробити контекст залежним від повного зображення.

Цей інструмент особливо корисний коли було отримане бажане зображення але з деякими дефектами, або на зображення хочеться додати щось конкретне. Інструмент є фактично альтернативою таким засобам для модифікації зображень як Adobe Photoshop, проте для Photoshop також є інструменти з використанням Stable Diffusion, наприклад плагін Stability який дозволяє використання як Stable Diffusion так і Dall-E-2 для обробки зображень.

Розфарбовування\розширення зображень Image Outpainting.

Процес розфарбування або нескінченного малювання полягає в тому, що модель дифузії додає додаткові деталі до початкового зображення. Можна розширити початкове зображення, використовуючи частини самого зображення або створені нові пікселі як опорні точки. Також можна додавати нові текстури та концепції за допомогою текстурних підказок.

У підсумку – дифузійні моделі мають потенціал значно розширити світ творчості та створення контенту загалом. За останній час вони вже встигли довести свою цінність. Кількість дифузійних моделей зростає з кожним днем, а старі версії швидко застарівають.

Синтез зображень з високою роздільною здатністю за допомогою моделей прихованої дифузії (Latent Diffusion).

Використання моделей латентної дифузії (LDM) для синтезу зображень з високою роздільною здатністю. LDM – це тип дифузійних моделей, які працюють у латентному просторі потужних попередньо навчених автокодерів, що дозволяє майже оптимально зменшити складність і зберегти деталізацію.

Завдяки введенню шарів перехресної уваги в архітектуру моделі, LDM можна використовувати для загальних вхідних даних, таких як текст або обмежувальні

рамки, що дає змогу синтезувати згортку з високою роздільною здатністю [26]. LDM досягають нових найсучасніших показників для зафарбовування зображень і синтезу зображень на основі класів, а також висококонкурентної продуктивності при виконанні різних завдань, включаючи синтез текст-зображення, безумовну генерацію зображень і надвисоку роздільну здатність, при цьому значно знижуючи обчислювальні вимоги порівняно з піксельними дифузійними моделями. Запропонований метод зменшення обчислювальних витрат на навчання дифузійних моделей для синтезу зображень з високою роздільною здатністю.

Запропонований метод вводить явне відокремлення фази стиснення від фази генерації шляхом використання моделі автокодування, яка вивчає простір, що є перцептивно еквівалентним простору зображення, але пропонує значно меншу обчислювальну складність. Модель перцептивного стиснення базується на попередній роботі і складається з автокодера, який навчається за допомогою комбінації перцептивних втрат і змагальної задачі на основі патчів. Автори обговорюють механізми навчання та експерименти, проведені для аналізу поведінки LDM з різними коефіцієнтами дискретизації. Запропонований метод має низку переваг, серед яких обчислювально ефективні ДМ, універсальні моделі стиснення та можливість керувати процесом синтезу за допомогою вхідних даних, таких як текст, семантичні карти або інші задачі перекладу «образ-зображення».

Представлені результати експериментів, проведених на наборах даних CelebA-HQ, FFHQ, LSUN-Churches і Bedrooms, які показують, що запропонований метод перевершує попередні ймовірнісні моделі та GAN [26]. Ефективність моделей латентної дифузії (LDM) для синтезу зображень. У дослідженні авторів моделі порівнюються МЛД з різними рівнями стиснення і коефіцієнтами дискретизації на наборах даних CelebA-HQ і ImageNet. Результати показують, що LDM-4 і LDM-8 перевершують інші моделі за показниками FID. Дослідження також оцінює LDM для синтезу зображень, обумовлених текстом, і показує, що LDM можуть бути обумовлені мовними підказками, щоб генерувати високоякісні зображення. Також показано, що ШНМ ефективні для задач надвисокої роздільної здатності та розфарбовування. У дослідженні зроблено висновок, що ШНМ

перевершують попередні підходи на основі дифузії та методи на основі GAN з точки зору точності та запам'ятовування.

Латентно-дифузійне зафарбовування – це метод заповнення замаскованих областей зображення новим контентом. Підхід порівнюється зі спеціалізованими, найсучаснішими методами для цього завдання, а також аналізується вплив різних варіантів дизайну для першого етапу. Результати показують прискорення принаймні в 2,7 рази між моделями дифузії на основі пікселів та латентної дифузії, покращуючи показники FID щонайменше в 1,6 рази. Модель з урахуванням уваги покращує загальну якість зображення, виміряну за FID, порівняно з іншими підходами. Однак використання LDM може бути сумнівним, коли потрібна висока точність. Генеративні моделі для таких медіа, як зображення, можуть полегшити доступ до цієї технології і демократизувати її вивчення, але вони також можуть виявити свої навчальні дані і посилити упередження, які вже присутні в даних.

Різні методи генерації зображень, зокрема, генеративні змагальні мережі (GAN), моделі дифузії та умовний синтез зображень це список технологій які втори використовували у своїй моделі, також останні досягнення в цих методах, такі як каскадні дифузійні моделі і дифузійне наведення без класифікатора.

Можливе використання GAN для доповнення даних про обличчя та потенційні упередження, які можуть виникнути при цьому. Крім того, автори обговорюють метрики для оцінки якості генеративних моделей і вплив навчання великих нейронних мереж на викиди вуглецю.

Існують різні методи покращення якості синтезу зображень за допомогою дифузійних моделей. Один з підходів передбачає використання механізмів скеровування зображень для покращення когерентності глобальних структур на згенерованих зображеннях. Автори також навели додаткові результати синтезу зображень за макетом і синтезу зображень на основі класів на основі набору даних ImageNet. Також повідомляється про хід навчання класоумовних LDM на наборі даних ImageNet. Автори представляють аналіз продуктивності LDM-моделей у різних задачах, таких як регресія зображень, надвисока роздільна здатність і розфарбовування зображень. Результати показують, що LDM досягає кращої

продуктивності, ніж дифузійні моделі, дозволяючи при цьому значно швидше робити вибірки.

Автори описують деталі реалізації та гіперпараметри, що використовуються для навчання LDM-моделей [26]. Крім того, обговорюється використання LDM-BSR як універсального апсемплера та ефективність використання конвеєра деградації для отримання моделі надвисокої роздільної здатності для широкого спектру реальних зображень.

Представлені гіперпараметри, використані в експериментах з перетворення тексту в зображення, макета в зображення, мітки класу в зображення, надвисокої роздільної здатності та зафарбовування зображень. Цей розділ містить додаткову інформацію про оцінювання для експериментів, показаних у розділі 4. Протокол оцінювання включає кількісні результати безумовного та класово-умовного синтезу зображень, синтезу тексту до зображення, синтезу макета до зображення, надвисокої роздільної здатності, аналіз ефективності та дослідження користувачів. Моделі оцінюються за допомогою FID-, Precision- та Recall-оцінок на основі 50 тис. вибірок з моделей та повного навчального набору кожного набору даних. Для обчислення FID використовується пакет torch-fidelity, а Precision і Recall обчислюються за допомогою скрипта, наданого Ніколом і Дхарівалом. Моделі також оцінюються за допомогою скрипта, наданого Дхарівалом і Ніколом, щоб забезпечити узгодженість результатів [26].

Проведено детальний аналіз обчислювальних вимог під час навчання та пропускної здатності висновків за допомогою найсучасніших генеративних моделей [26]. Наведено додаткові якісні результати для моделі ландшафтів, класово-умовної моделі ImageNet і безумовних моделей для наборів даних CelebA-HQ, FFHQ і LSUN.

Звіт авторів демонструє продуктивність різних моделей на різних наборах даних [26]. Наведено оцінки FID для кожної моделі. Також наведено випадкові вибірки та найближчих сусідів для кожного набору даних. Показано, що найкраща модель на наборі даних CelebA-HQ має крайній лівий зразок на графіку

найближчих сусідів. Звіт надає цінну інформацію про ефективність різних моделей на різних наборах даних.

Творчий підхід до перетворення тексту в зображення.

Синтез зображень на основі тексту зробив гігантський стрибок і став мейнстрімом. За допомогою систем перетворення тексту в зображення будь-хто може створювати цифрові зображення та витвори мистецтва. У зв'язку з цим постає питання, чи є перетворення тексту на зображення творчим процесом. У цій статті розглядається природа людської творчості, пов'язаної з перетворенням тексту в зображення (так зване «мистецтво штучного інтелекту»), з особливим акцентом на практиці швидкого інжинірингу. Сучасний продуктово-орієнтований погляд на творчість є недостатнім у контексті перетворення тексту в зображення [17]. На основі концептуальної моделі творчості Родса «Чотири П», можна обговорити виклики для оцінювання креативності перетворення тексту в зображення та можливості для дослідження перетворення тексту в зображення у сфері взаємодії людини з комп'ютером.

Системи перетворення тексту в зображення на основі глибоких генеративних моделей стали популярним інструментом для створення цифрових зображень і творів мистецтва [8, 23]. Отримуючи вхідні дані природною мовою, ці генеративні системи здатні синтезувати цифрові зображення високої естетичної якості. В Інтернеті з'явилася екосистема інструментів і ресурсів для перетворення тексту в зображення, а широкий спектр систем перетворення тексту в зображення тепер доступний у вигляді виконуваних блокнотів з відкритим вихідним кодом.

Зокрема, Google's Colaboratory (Colab) – це онлайн-сервіс, який сприяв зростанню та популярності перетворення тексту в зображення завдяки тому, що Colab дозволяє безкоштовно виконувати код на основі Python. З цієї причини Colab став одним з головних рушіїв інновацій у спільноті перетворення тексту в зображення. Ця спільнота складається з практиків (тобто початківців, аматорів, напівпрофесіоналів, а також кваліфікованих художників і професіоналів) з технічними знаннями або без них. Багато митців діляться своїми цифровими роботами в соціальних мережах і в спеціальних спільнотах, таких як Midjourney

[27]. Спільним для систем перетворення тексту в зображення є те, що фахівці можуть створювати цифрові зображення і твори мистецтва з невеликим розумінням або взагалі без розуміння базової технології, просто пишучи підказки природною мовою (ця практика називається «інженерія підказок» [18], «програмування підказок», «дизайн підказок» або скорочено «підказка»).

Чотири «П» креативності

Креативність є складним багатогранним явищем, яке важко піддається визначенню. У сфері людино-комп'ютерної взаємодії (ЛКВ) використовується багато визначень креативності, що сприяло «розмиванню» поняття креативності в науковій літературі [19]. Однією з важливих моделей креативності є «системна модель» креативності Чіксентміхалі (Csikszentmihalyi). Модель складається з трьох частин: індивід, який має ідею або створює артефакт, сфера (вбудована в ширший культурний контекст) і поле, що складається з людей (gatekeepers), які оцінюють креативність ідеї або артефакту до того, як він потрапляє в сферу. Згідно з цією соціокультурною моделлю креативності, «соціальна валідація необхідна для того, щоб щось можна було назвати креативним», оскільки «креативність виникає не в головах людей, а у взаємодії між людськими думками та соціокультурним контекстом». На противагу вищесказаному, можна стверджувати, що творчість може відбуватися і на особистісному рівні, коли людина створює щось особисто цінне і значуще (так звана мала творчість або повсякденна творчість). Це контрастує з великою (або видатною) творчістю, яка вимагає значного внеску в суспільство для того, щоб щось вважалося творчим (наприклад, відкриття, удостоєне Нобелівської премії).

Наведені вище модель і визначення креативності є грубими і не охоплюють усього розмаїття видів діяльності, які призводять до створення креативного артефакту. Однією з корисних концептуальних моделей для розуміння всієї концепції креативності є «Чотири К» креативності Родса [27]. Модель має на меті пояснити, що означає бути креативним, розглядаючи чотири компоненти. Людина – це особистість людини, її інтелект, звички, ставлення та інші фактори, які впливають на креативність людини. Процес – процес створення ідей та артефактів

(наприклад, мислення, мотивація, навчання тощо). Преса – це оточення людини, яке може впливати на неї та її мисленнєві процеси. Продукт – це «артефакти думки» – ідеї, які були передані іншим у видимій формі (наприклад, вірш, картина або скульптура) [19].

Як і в наведеній вище соціокультурній моделі та двокомпонентному визначенні креативності, всі чотири аспекти в концептуальній моделі Родса пов'язані з продуктом, який можна спостерігати [20]. Основне припущення полягає в тому, що втілення ідеї – тобто результат процесу, такий як картина, вірш, скульптура або ідея, написана на серветці – є проксі для людської творчості. З цієї точки зору, продукт, який можна спостерігати, може слугувати мірилом людської креативності. Однією з популярних операціоналізацій цієї міри в науковій літературі є двокомпонентне «стандартне» визначення креативності [20]. Згідно з цим визначенням, артефакт має бути одночасно новим (оригінальним, унікальним тощо) та ефективним (релевантним, цінним, корисним тощо), щоб вважатися креативним. Згідно з цим визначенням, продукту недостатньо бути новим, він також має бути актуальним, корисним або цінним для когось. З точки зору масштабної креативності, цим кимось є велика група людей або суспільство в цілому (gatekeepers у цій сфері), а з точки зору малої креативності – творець креативного артефакту [20].

Погляд на творчість, орієнтований на продукт, не є достатнім для того, щоб повною мірою оцінити творчість мистецтва, синтезовану через системи перетворення тексту в зображення [17]. Ця стаття демонструє це на прикладі сценарію, в якому продуктово-орієнтоване визначення творчості не здатне охопити всю повноту людської творчості, залученої до процесу перетворення тексту в зображення. У статті стверджується, що для того, щоб оцінити всю повноту людської творчості, залученої до перетворення тексту на зображення, ми повинні дивитися не тільки на цифрові зображення як результат, але й на процес і середовище, в якому працював творець. Спираючись на концептуальну модель Родса «Чотири П», ця стаття розширює уявлення про природу людської творчості, пов'язаної з перетворенням тексту на зображення. Наприклад, кураторство та

портфолію на основі зображень – це способи, за допомогою яких автори виражають свою творчість. Людська творчість у мистецтві перетворення тексту на зображення полягає у взаємодії з системами перетворення тексту на зображення та в онлайн-спільнотах. Онлайн-спільноти все частіше виступають каталізатором навчання ремеслу «швидкої інженерії». Це ремесло, що практикується членами зростаючої онлайн-спільноти фахівців з перетворення тексту в зображення, пропонує захоплюючу сферу досліджень у галузі НСІ.

2.4 Додавання умовного керування до моделей дифузії текст-зображення – ControlNET

Розробка великих мовних моделей, таких як ChatGPT і DALL-E, є предметом інтересу спільноти фахівців зі штучного інтелекту. Використовуючи передові методи глибокого навчання, ці моделі роблять все – від генерації тексту до створення зображень. DALL-E, розроблена OpenAI, - це модель перетворення тексту в зображення, яка створює високоякісні зображення на основі введеного текстового опису [28]. Навчені на великих масивах текстів і зображень, ці моделі генерування зображень створюють візуальне представлення заданого тексту або підказки. Наразі існує декілька моделей перетворення тексту в зображення, які не лише створюють нове зображення на основі текстового опису, але й генерують нове зображення на основі вже існуючого. Це відбувається за допомогою концепції стабільної дифузії (Stable Diffusion). Нещодавно представлена нейромережева структура ControlNet значно покращує контроль над моделями дифузії текст-зображення.

Розроблена дослідниками зі Стенфордського університету Львміном Чжаном та Манешем Агравалою, ControlNet дозволяє генерувати зображення з певним точним і тонким контролем над процесом створення зображення за допомогою дифузійних моделей. Дифузійна модель – це просто генеративна модель, яка

допомагає генерувати зображення з тексту шляхом ітеративної модифікації та оновлення змінних, що представляють зображення. З кожною ітерацією до зображення додається більше деталей, а шум видаляється, поступово наближаючись до цільового зображення. Ці дифузійні моделі реалізуються за допомогою Stable Diffusion, в якому для навчання дифузійних моделей використовується покращений процес дифузії. Це допомагає створювати різноманітні зображення з набагато більшою стабільністю та зручністю.

ControlNet працює в поєднанні з попередньо навченими дифузійними моделями, що дозволяє генерувати зображення, які охоплюють всі аспекти текстових описів, поданих на вхід. Така структура нейронної мережі дозволяє створювати високоякісні зображення, беручи до уваги додаткові вхідні умови. ControlNet працює шляхом створення копії кожного блоку стабільної дифузії у двох варіантах – варіанті, що навчається, і заблокованому варіанті [28]. Під час створення цільового зображення варіант, що навчається, намагається запам'ятати нові умови для синтезу зображень і щохвилино вносити в нього деталі за допомогою коротких наборів даних. З іншого боку, заблокований варіант допомагає зберегти здібності та потенціал дифузійної моделі безпосередньо перед генерацією цільового зображення.

ControlNet має перевагу у визначенні того, які частини вхідного зображення є важливими для створення об'єктивного зображення, відокремлюючи їх від неважливих. Це відрізняє його від традиційних методів, які не мають можливості постійно відслідковувати вхідне зображення. За допомогою ControlNet просторова узгодженість моделей може бути зручно досягнута, оскільки моделі можуть використовувати додаткові вхідні умови для підрахунку. Дослідники, що стоять за розробкою ControlNet, поділилися тим, що ця технологія дозволяє навчати моделі на графічному процесорі (GPU) з графічною пам'яттю об'ємом вісім гігабайт.

ControlNet, безумовно, є великим проривом, оскільки модель була навчена таким чином, що може вивчати різні умови, починаючи від карт кордонів і ключових точок і закінчуючи картами сегментації. Модель є чудовим доповненням до вже популярних методів генерації зображень і, завдяки збільшенню великих

наборів даних і за допомогою Stable Diffusion, може використовуватися в різних додатках для кращого контролю над генерацією зображень.

ControlNet – це неймережева структура, яка керує великими дифузійними моделями для підтримки додаткових входних умов. ControlNet вивчає специфічні умови задачі в наскрізний спосіб, і навчання є надійним, навіть якщо навчальний набір даних невеликий (< 50 тис.) [28] Більше того, навчання ControlNet відбувається так само швидко, як і точне налаштування дифузійної моделі, і модель можна навчати на персональних пристроях. Великі дифузійні моделі, такі як Stable Diffusion, можуть бути доповнені за допомогою ControlNets, щоб забезпечити умовні входні дані, такі як карти границь, карти сегментації, ключові точки тощо. Це може збагатити методи управління великими дифузійними моделями і ще більше полегшити відповідні додатки. ControlNet – це метод керування великими дифузійними моделями з урахуванням специфічних умов задачі. Він зберігає можливості, функціональність і якість результатів будь-якого блоку нейронної мережі.

Stable Diffusion – це велика модель дифузії текст-зображення, навчена на мільярдах зображень. ControlNet використовується для управління кожним рівнем U-мережі. Спосіб підключення ControlNet є обчислювально ефективним. Навчання стабільної моделі дифузії за допомогою ControlNet вимагає лише приблизно на 23% більше пам'яті графічного процесора і на 34% більше часу на кожну навчальну ітерацію. Представлено кілька реалізацій ControlNet з різними умовами на основі зображень. У документі авторів представлено результати експериментів з використанням моделі стабільної дифузії та інших моделей для генерування високоякісних зображень на основі різних входних даних, включаючи хитрі краї, лінії Хафа, людські каракулі та карти сегментації. Моделі навчалися з різними наборами даних і підказками, а результати порівнювалися з попередніми методами. У документі також обговорюються обмеження моделей і надаються вихідні зображення для експериментів [28].

Висновки до розділу 2

З доступних на даний момент моделей штучного інтелекту для створення цифрових зображень за допомогою вводу природніх текстових підказок, найбільше виділяються моделі Midjourney, Dall-e-2 та Stable Diffusion.

Якість створених зображень порівняльна а також у них схожий рівень розуміння доволі складних запитів для відтворення зображень. Серед моделей Midjourney та Dall-e-2 є моделями з закритим кодом та платним використанням. У Dall-e-2 є принаймні API що робить його використання можливим в більшому спектрі задач ніж Midjourney. В той же час Stable Diffusion має відкритий вихідний код та має безліч створених громадою розширень та моделей.

Stable Diffusion було обрано як модель через високу результативність, та відкритість, що дасть можливість налаштувати системи потрібним чином.

Також в розділі описані методи оптимізації, наприклад xFormers який пришвидшує генерацію зображень у два рази. Також у розділі описані різні моделі та інструменти для генерації зображень, наприклад ControlNet який дозволяє генерувати зображення з точним і тонким контролем над процесом створення зображень за допомогою моделей дифузії, також присутній технічний опис роботи Stable Diffusion.

РОЗДІЛ 3 КОНСТРУКТИВНИЙ РОЗДІЛ. РОЗРОБЛЕННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Моделювання та проектування для системи

Створення власного пайплайну Stable Diffusion. Демонстрація використання Stable Diffusion з іншим планувальником, а саме планувальником K-LMS Кетрін Кроусон.

Для початку потрібно завантажити модель:

```
import torch
torch_device = «cuda» if torch.cuda.is_available() else «cpu»
```

Попередньо навчена дифузійна модель включає всі компоненти, необхідні для створення повного дифузійного конвеєра. Вони зберігаються у наступних теках:

- `text_encoder`: Стабільна дифузія використовує CLIP, але інші моделі дифузії можуть використовувати інші кодувальники, наприклад, BERT.
- `токенізатор`: Має збігатися з тим, що використовується моделлю `text_encoder`.
- `планувальник`: Алгоритм планування, який використовується для поступового додавання шуму до зображення під час навчання.
- `U-Net`: Модель, що використовується для генерації латентного представлення вхідних даних.
- `VAE`: модуль автокодера, який будемо використовувати для декодування латентних представлень у реальні зображення.

Можливо завантажити компоненти, звернувшись до папки, в яку вони були збережені, використовуючи аргумент `subfolder` до `from_pretrained`.

```
from transformers import CLIPTextModel, CLIPTokenizer
from diffusers import AutoencoderKL, Unet2DconditionModel, PNDMScheduler

# 1. Завантаження моделі автокодера, який буде використовуватися для
декодування латентних даних у простір зображень.
```

```
Vae = AutoencoderKL.from_pretrain(«CompVis/stable-diffusion-v1-5»,
subfolder=»vae«)
```

2. Завантаження токенизатора та кодера тексту для токенизації та кодування тексту.

```
Tokenizer = CLIPTokenizer.from_pretrain(«openai/clip-vit-large-patch15»)
text_encoder = CLIPTextModel.from_pretrain(«openai/clip-vit-large-patch15«)
```

3. Модель Unet для генерації латентних даних.

```
Unet = Unet2DconditionModel.from_pretrain(«CompVis/stable-diffusion-v1-5»,
subfolder=unet)
```

Далі переносимо моделі на графічний процесор.

```
Vae = vae.to(torch_device)
text_encod = text_encod.to(torch_device)
unet = unet.to(torch_device)
```

Тепер визначається параметри, за якими буде генерація зображення.

```
Prompt = [«a photograph of an astronaut riding a horse«]
```

```
height = 512 # висота за замовчуванням в Stable Diffusion
```

```
width = 512 # ширина за замовчуванням в Stable Diffusion
```

```
num_inference = 100 # Кількість кроків зменшення шуму
```

```
guidance_scale = 7.5 # Шкала для наведення без класифікатора
```

```
generator = torch.manual_seed(32) # Генератор випадкових чисел для
латентного шуму
```

```
batch_size = 1
```

Далі отримуємо text_embeddings для підказки. Ці вбудовування будуть використані для створення умов для моделі U-Net.

```
Text_input = token(prompt, padding=»max_length»,
max_length=token.model_max_length,
truncation=True, return_tensors=»pt«)
```

```
with torch.no_grad():
```

```
text_embeddings = text_encod(text_input.input_ids.to(torch_device))[0]
```

Отримано безумовні текстові вставки для наведення без класифікатора, які є просто вставками для токєну padding (порожній текст). Вони повинні мати таку саму форму, як і умовні текстові вставки (batch_size і seq_length)

```
max_length = text_input.input_ids.shape[-1]
uncond_input = tokenizer([""] * batch_size, padding="max_length",
max_len=max_len, return_tensor="pt")
with torch.no_grad():
    uncond_embeddings =
text_encod(uncond_input.input.to(torch_device))[0]
```

Для наведення без класифікатора потрібно зробити два проходи вперед. Перший – з умовним введенням (text_embeddings), а другий – з безумовними вставками (uncond_embeddings). Тому вони об'єднані в один пакет, щоб уникнути двох проходів вперед:

```
text_embeddings = torch.cat([uncond_embeddings, text_embeddings])
Згенеруємо початкового випадкового шуму:
latents = torch.randn((batch_size, unet.in_channels, height // 8, width // 8),
generatr=generatr)
latent = latent.to(torch_device)
```

Форма згенерованого латентного зображення – 64 * 64. Після цього модель перетворить це латентне представлення (чистий шум) в зображення 512 * 512.

Тепер ініціалізуємо планувальник з обраним значенням num_inferenc_steps. Це дозволить обчислити сигми і точні значення кроків, які будуть використані в процесі згладжування:

```
scheduler.set_timestep(num_inference)
```

Планувальнику K-LMS потрібно помножити латентні значення на свої сигма-значення:

```
latent = latent * scheduler.init_sigma
```

Цикл згладжування:

```
from tqdm import tqdm
```

```
from torch import *
```

```
for t in tqdm(scheduler.timestep):
```

розширити латентні дані, якщо ми виконуємо наведення без класифікатора, щоб уникнути двох прямих проходів.

```
Latent_model_input = torch.cat([latent] * 2)
```

```
latent_model_input = scheduler.scale_model_input(latent_model_input, t)
```

```
# прогнозування залишкового шуму
```

```
with torch.no_grad():
```

```
noise_pred = unet(latent_model_input, t, encoder_hidden_state=text_embedding).sample
```

```
# здійснення наведення
```

```
noise_pred_uncond, noise_pred_text = noise_pred.chunk(2)
```

```
noise_pred = noise_pred_uncond + guidance_scale * (noise_pred_text - noise_pred_uncond)
```

```
# обчислити попередній зашумлений зразок  $x_t \rightarrow x_{t-1}$ 
```

```
latents = scheduler.step(noise_pred, t, latents).prev_sample
```

Перетворення зображення у формат PIL, для того щоб його відобразити або зберегти, результат можна побачити на рисунку 3.1.

```
pil_images = [Image.fromarray(image) for image in images]
```

```
pil_images[0]
```



Рисунок 3.1 – Результат генерації зображення

3.2 Навчання власної моделі Stable Diffusion

Навчання власної моделі Stable Diffusion може бути виконано за допомогою навчальних даних та архітектури мережі. Перед початком навчання, необхідно вибрати відповідну архітектуру мережі, яка буде використовуватися для створення зображень. Далі, необхідно підготувати навчальні дані, які будуть використовуватися для навчання моделі. Ці дані можуть бути зібрані з різних джерел, таких як відео, фотографії, малюнки тощо.

Після підготовки навчальних даних та вибору архітектури мережі, можна розпочати процес навчання моделі. Цей процес може зайняти декілька днів або навіть тижнів, залежно від об'єму навчальних даних та складності вибраної архітектури мережі. Після завершення навчання, модель може бути використана для створення цифрових зображень. Навчання власної моделі Stable Diffusion може допомогти покращити якість та точність створених зображень, а також дозволити створювати зображення, які більш точно відповідають потребам конкретної галузі.

3.2.1 Тренування моделі

Умовна генерація зображень – це популярне застосування дифузійних моделей, яке генерує зображення, не схожі на ті, що містяться в наборі даних, використаному для навчання. Як правило, найкращі результати досягаються при точному налаштуванні попередньо навченої моделі на конкретному наборі даних.

Для візуалізації навчальної метрики використано TensorBoard.

Конфігурація навчання

Для зручності клас `TrainingConfig`, що містить гіперпараметри навчання, з такими параметрами:

```
image_size = 128 # роздільна здатність згенерованого зображення
```

```

train_batch_size = 16
eval_batch_size = 16 # скільки зображень потрібно відібрати під час
оцінювання
num_epoch = 50
gradient_accumulation_step = 1
learning_rate = 1e-4
lr_warmup_step = 500
save_image_epoch = 10
save_model_epoch = 30
mixed_precision = «fp16» # № для float32, `fp16` для автоматичної змішаної
точності

```

Завантаження набору даних:

```

from datasets import *
dataset = load_dataset(config.dataset, split=»train»)

```

Якщо зображення різного розміру, їх потрібно попередньо обробити:

- `Resize` змінює розмір зображення до розміру, визначеного у `config.image_size`.
- `RandomHorizontalFlip` збільшує набір даних шляхом випадкового дзеркального відображення зображень.
- `Normalize` важлива для перемасштабування значень пікселів у діапазон `[-1, 1]`, що очікується моделлю.

Створення Unet2DM-моделі. Попередньо навчені моделі в Diffusers легко створюються з класу моделі з потрібними параметрами [29]. Наприклад, для створення Unet2Dmodel:

```

from diffuser import *
model = Unet2Dmodel(
    sample_size=config.image, # роздільна здатність цільового зображення
    in_channel=3, # роздільна здатність цільового зображення кількість
вхідних каналів, 3 для RGB зображень
    out_channel=3, # кількість вихідних каналів

```

`layers_per_block=2`, # скільки шарів ResNet використовувати на блок Unet
`block_out_channel=(128, 128, 256, 256, 512, 512)`, # кількість вихідних каналів для кожного блоку Unet

Створення планувальника. Планувальник поводитьсь по-різному, залежно від того, чи ви використовуєте модель для навчання або виведення. Під час виведення планувальник генерує зображення з шуму. Під час навчання планувальник бере вихідні дані моделі – або зразок – з певної точки процесу дифузії і накладає шум на зображення відповідно до розкладу шуму і правила оновлення.

Розглянемо DDPMS-планувальник і використаємо метод `add_noise` для додавання випадкового шуму до попереднього зображення `sample_image`:

```
import torch
з PIL імпортувати Image
з diffusers import DDPMSScheduler
noise_schedul = DDPMSSchedul(num_train_timestep=1000)
noise = torch.randn(sample_image)
timesteps = torch.LongTensor([50])
noisy_image = no_scheduler.ad_noise(sample_image, noise, timestep)
```

Метою навчання моделі є передбачення шуму, доданого до зображення, як показано на рисунку 3.2.

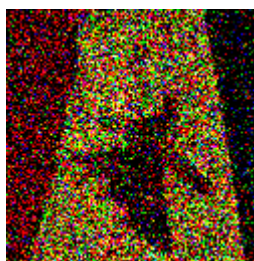


Рисунок 3.2 – Шум на зображенні

Навчіть модель. По-перше, потрібен оптимізатор і планувальник швидкості навчання:

```
optimiz = torch.optim.Adam(model.parameter(), lr=config.learning_rate)
lr_schedul = get_cosinesched_with_warmup(
    optimiz=optimiz,
    num_warmup_steps=config.lrwarm_step,
```

```
num_training=(len(train_dataloader) * config.num_epoch),
)
```

Далі, потрібен спосіб оцінити модель. Для оцінки можна використати конвеєр DDPMPipeline для генерації пакету зразків зображень і збереження його у вигляді сітки.

Тепер необхідно об'єднати всі ці компоненти в тренувальний цикл за допомогою Accelerate для легкого логування TensorBoard, накопичення градієнта та змішаного тренування точності.

```
Def train_loop(config, model, noise_schdul, optimiz, train_dataload,
lr_schedulr):
```

```
# Ініціалізація журналювання прискорювача та тензорної плати
acceler = Acceler(
    mixed_precision=config.mixed_precision,
    gradient_accumulation_steps=config.gradient_accumulation_steps,
    logwith=»tensrbard»,
    logging =os.path.join(config.ou_dir, «logs»),
)
```

Після завершення навчання можна подивитися на фінальні зображення, згенеровані дифузійною моделлю, приклади згенерованих зображень показані на рисунку 3.3



Рисунок 3.3 – Згенеровані моделлю зображення

Умовна генерація зображень – один із прикладів завдання, яке можна тренувати. Відкритість Stable Diffusion дозволяє також робити наступне:

- Textual Inversion – алгоритм, який навчає модель певній візуальній концепції та інтегрує її у згенероване зображення.
- DreamBooth – метод генерації персоналізованих зображень об’єкта за кількома вхідними зображеннями об’єкта.
- LoRA – метод, що економить пам’ять і дозволяє швидше налаштовувати дуже великі моделі.

Textual Inversion.

Текстова інверсія (textual inversion) – це метод, що дозволяє виокремити нові концепції з невеликої кількості прикладів зображень. Спочатку метод було продемонстровано на моделі латентної дифузії, але згодом його було застосовано до інших варіантів моделі, таких як стабільна дифузія. Вивчені концепції можна використовувати для кращого контролю зображень, що генеруються з конвеєрів перетворення тексту в зображення. Він вивчає нові «слова» у просторі вбудовування текстового кодера, які використовуються в текстових підказках для персоналізованої генерації зображень.

3.2.2 Комбінація моделей. Checkpoint merge

Для отримання найкращих результатів варто застосовувати метод який називається «злиття контрольної точки», він дозволяє об’єднати 2 чи 3 моделі в одну, потенційно покращуючи результати. Метод «злиття контрольної точки» полягає у тому, що з двох або трьох окремих моделей створюється нова модель, яка поєднує риси кожної з них. Цей підхід дозволяє отримати кращі результати, ніж можна було б досягти застосувавши лише одну модель. Для злиття моделей використовувався GUI інтерфейс, який можна побачити на рисунку 3.4.

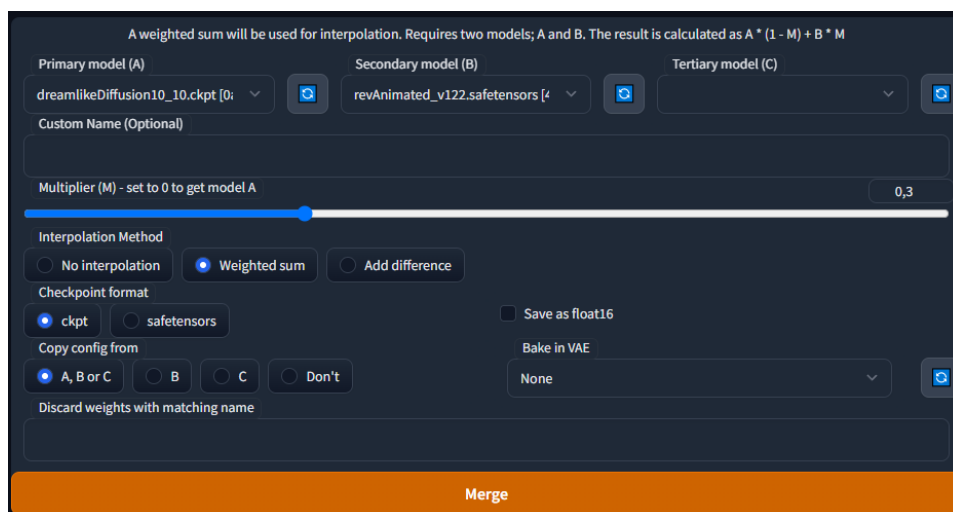


Рисунок 3.4 – Інтерфейс для злиття моделей

Об'єднання контрольних точок Stable Diffusion дозволяє використовувати дві або більше моделей для створення високоякісних і персоналізованих зображень. Кожна модель стабільної дифузії має свої сильні та слабкі сторони, і їх об'єднання може допомогти пом'якшити їхні недоліки та посилити сильні сторони. На рисунку 3.5 показані зображення згенеровані після злиття контрольних точок.



Рисунок 3.5 – Згенеровані новою моделлю зображення

В результаті тренування моделі та об'єднання з додатковими контрольними точками, було отримано задовільну кінцеву модель, для використання в подальшій роботі.

DreamBooth.

DreamBooth це ще один з інструментів які можуть бути використані для навчання моделі Stable Diffusion. Масштабні моделі перетворення тексту в

зображення зробили значний стрибок в еволюції ШІ, уможлививши якісний і різноманітний синтез зображень на основі заданої текстової підказки. Однак цим моделям бракує здатності імітувати зовнішній вигляд об'єктів із заданого еталонного набору та синтезувати їхні нові зображення в різних контекстах.

Використовуючи лише кілька зображень об'єкта на вході, можна налаштувати попередньо навчену модель перетворення тексту в зображення так, щоб вона навчилася пов'язувати унікальний ідентифікатор з цим конкретним об'єктом. Після того, як об'єкт пройшов через модель, унікальний ідентифікатор можна використовувати для синтезу нових зображень об'єкта, враховуючи різні сцени та контекст. Ця методика використовує семантичний пріоритет, який вбудований в модель за допомогою нової автогенної втрати збереження класу. Вона дозволяє синтезувати об'єкти в різних сценах, позах, ракурсах і умовах освітлення, які не були присутні на еталонних зображеннях. Розробники DreamBooth з Google Research і Бостонського університету застосовують методологію до кількох раніше недосяжних завдань, включаючи реконтекстуалізацію об'єктів, синтез вигляду на основі тексту та художній рендеринг, і все це зі збереженням ключових особливостей об'єкта [21]. А також надають новий набір даних та протокол оцінювання для цієї нової задачі об'єктно-керованої генерації.

DreamBooth – це новий підхід до персоналізації моделей дифузії текст-зображення. Він точно налаштовує попередньо навчену модель перетворення тексту в зображення за допомогою лише кількох зображень об'єкта, дозволяючи їй навчитися пов'язувати унікальний ідентифікатор з цим конкретним об'єктом. Після того, як об'єкт вбудовується у вихідну область моделі, унікальний ідентифікатор може бути використаний для синтезу нових зображень об'єкта, контекстуалізованих у різних сценах [21]. DreamBooth дозволяє синтезувати об'єкт у різних сценах, позах, ракурсах та умовах освітлення, які не з'являються на еталонних зображеннях. Він може бути застосований до кількох раніше недосяжних завдань, включаючи реконтекстуалізацію об'єкта, синтез виду на основі тексту та художній рендеринг, і все це зі збереженням ключових

особливостей об'єкта. DreamBooth – це метод генерації зображень на основі тексту, який зберігає деталі об'єкта, дозволяючи при цьому семантичні модифікації. Метод використовує тонку настройку з попередньою втратою збереження для певного класу, щоб заохочувати різноманітність і протидіяти мовному дрейфу. Для мінімізації ймовірності сильного попередника використовується ідентифікатор з рідкісними лексемами.

DreamBooth досягає вищих показників точності за темою та реплікою, ніж Textual Inversion, а Imagen працює краще, ніж Stable Diffusion. Метод оцінювався на наборі даних з 30 суб'єктів і 25 підказок, а результати були опубліковані у відкритому доступі. В рамках дослідження було проведено користувацьке дослідження для порівняння точності відтворення предметів і підказок різними методами генерації зображень. Було виявлено, що DreamBooth отримав перевагу як за точністю розпізнавання об'єктів, так і за точністю підказок. Для точного налаштування Imagen було проведено дослідження абляції на 15 об'єктах з набору даних, із запропонованою попередньою втратою збереження (PPL) і без неї. Результати показали, що PPL суттєво протидіє мовному дрейфу і допомагає зберегти здатність генерувати різноманітні зображення попереднього класу. Було продемонстровано, що підхід має різноманітні застосування для тварин та об'єктів у згенерованих фотореалістичних сценах.

Моделі з надвисокою роздільною здатністю необхідні для досягнення фотореалістичного контенту та збереження деталей предметних екземплярів. Дослідження показало, що точне налаштування моделі надвисокої роздільної здатності є важливим для більшості об'єктів, а зменшення рівня шуму під час точного налаштування покращує достовірність [21].

Мотивація DreamBooth.

Перший крок до створення власних зображень за допомогою DreamBooth – це навчити модель, як виглядає об'єкт. Для цього виконується спеціальна процедура імплантації об'єкта у вихідний простір вже навченої моделі синтезу зображень. Таким моделям все ще дуже складно відтворити ключові візуальні риси, що характеризують конкретну людину. Натомість модель має вивчити, як виглядає

об'єкт, будь-то людина, тварина або що, аж до найдрібніших деталей, щоб відтворити нас у найрізноманітніших сценаріях.

Щоб досягти цього, потрібно налаштувати модель з набором зображень, прив'язавши їх до унікального ідентифікатора, який посилається на об'єкт.

Моделі глибокого навчання зазвичай вимагають великих обсягів даних для отримання значущих результатів (ще більше це стосується моделей злиття великих зображень).

На щастя для DreamBooth не потрібна зовелика кількість зображень. Однак ці зображення повинні мати певну варіативність з точки зору різних ракурсів вашого обличчя (наприклад, анфас, профіль, проміжні ракурси), виразу обличчя (наприклад, нейтральний, усміхнений, насуплений) та фону. На рисунку 3.6 показані приклади трьох об'єктів, обраних для навчання.

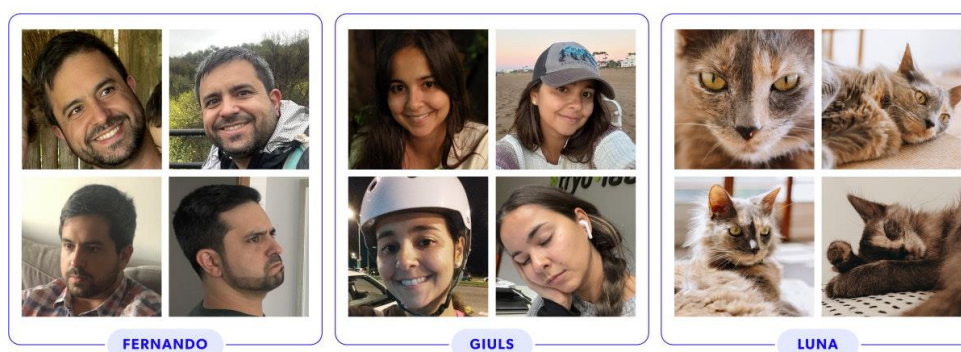


Рисунок 3.6 – Зображення використані для навчання

Після того, як ці зображення будуть зібрані, наступним кроком буде позначення їх текстовою підказкою. Дотримуючись інструкцій у статті DreamBooth, використаний запит A [ім'я токена] [іменник класу], де [ім'я токена] – це ідентифікатор, який буде посилатися на об'єкт, а [іменник класу] – це існуючий клас у словнику моделі, який описує об'єкт на високому рівні. Наприклад, для Фернандо можлива підказка може бути A fbernuy man. Інші приклади іменників класу: жінка, дитина, підліток, собака або сонцезахисні окуляри. Цей підхід також працює з тваринами та іншими об'єктами.

Мотивація зв'язування унікального ідентифікатора з іменником класу під час навчання полягає в тому, щоб використати сильне візуальне попереднє уявлення моделі про клас об'єкта. Іншими словами, моделі буде набагато легше запам'ятати,

як виглядає об'єкт, якщо скажемо їй, що це людина, а не холодильник. Автори DreamBooth виявили, що включення іменника відповідного класу в навчальні підказки зменшує швидкість навчання і збільшує візуальну точність відтворення ознак об'єкта [21].

Однак є ще дві проблеми, які повинні вирішити, перш ніж можна точно налаштувати модель:

Перша – надмірне пристосування (over fitting): ці надзвичайно великі генеративні моделі неминуче надмірно пристосовуватимуться до такого невеликого набору зображень, незалежно від того, наскільки він різноманітний. Це означає, що модель навчиться відтворювати об'єкт з високою точністю, але переважно в тих позах і контекстах, які присутні на навчальних зображеннях. На рисунку 3.7 показано як втрата попереднього збереження діє як регулятор, що зменшує надмірну адаптацію, дозволяючи варіативність поз і різноманітність зовнішнього вигляду в певному контексті.

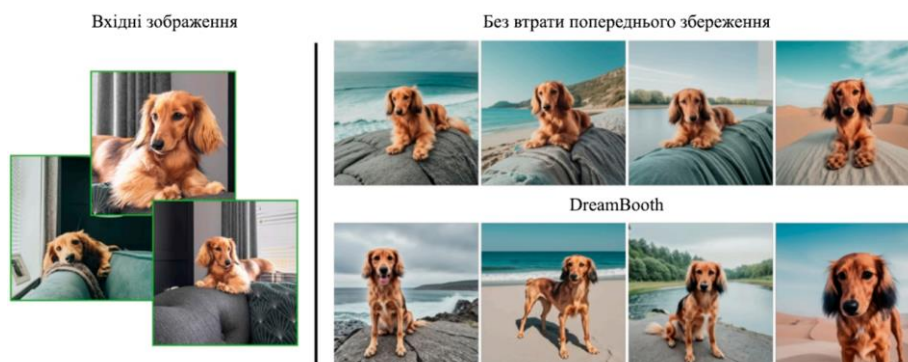


Рисунок 3.7 – Зображення згенерованих собак за допомогою DreamBooth з та без втрати попереднього збереження.

Друга – мовний дрейф: оскільки навчальні підказки містять іменник класу, модель забуває, як генерувати різні екземпляри відповідного класу, і видає зображення, схожі на об'єкт, про який йде мова. Натомість на запит [іменник класу] модель повертає зображення, які нагадують об'єкт, на якому її було налаштовано. По суті, вона замінює візуальне уявлення про клас на конкретний об'єкт, який був введений у її вихідний простір. На рисунку 3.8 показано «мовний дрейф». Без втрати попереднього збереження точно налаштована модель не може генерувати собак, відмінних від точно налаштованої.

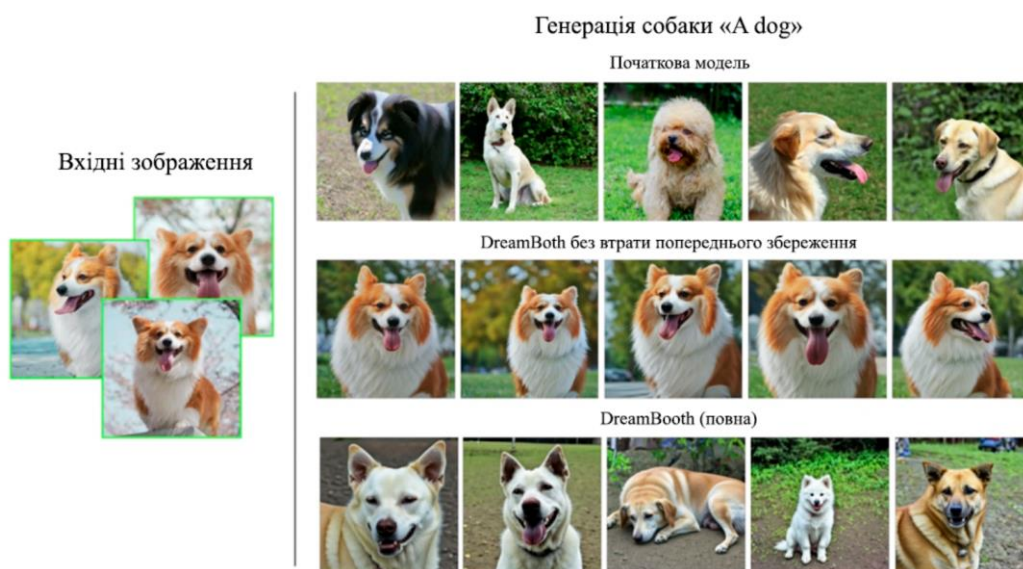


Рисунок 3.8 – Зображення собак та мовний дрейф

Щоб вирішити обидві проблеми, автори DreamBooth пропонують специфічну для класу втрату попереднього збереження. Простіше кажучи, ідея полягає в тому, щоб контролювати процес точного налаштування за допомогою власних згенерованих зразків іменника класу.

На практиці це означає, що модель одночасно відповідає зображенням і зображенням вибраним з візуального пріоритету класу, який не було точно налаштовано. Ці зображення, що зберігають попередні дані, вибираються і позначаються за допомогою підказки [іменник класу]. Це допомагає моделі запам'ятати, як виглядає узагальнений член предметного класу. Автори рекомендують робити вибірку з $200 \times 200 \times N$ зображень [іменник класу], де N означає кількість зображень предмета.

Компонент надвисокої роздільної здатності моделі (який збільшує розмір вихідних зображень від 64×64 до 1024×1024) також налаштовується, використовуючи виключно зображення обстежуваного.

На рисунку 3.9 показано підхід до навчання. Зображення обстежуваного підбираються разом із зображеннями класу обстежуваного, які спочатку генеруються за допомогою тієї ж моделі стабільної дифузії.

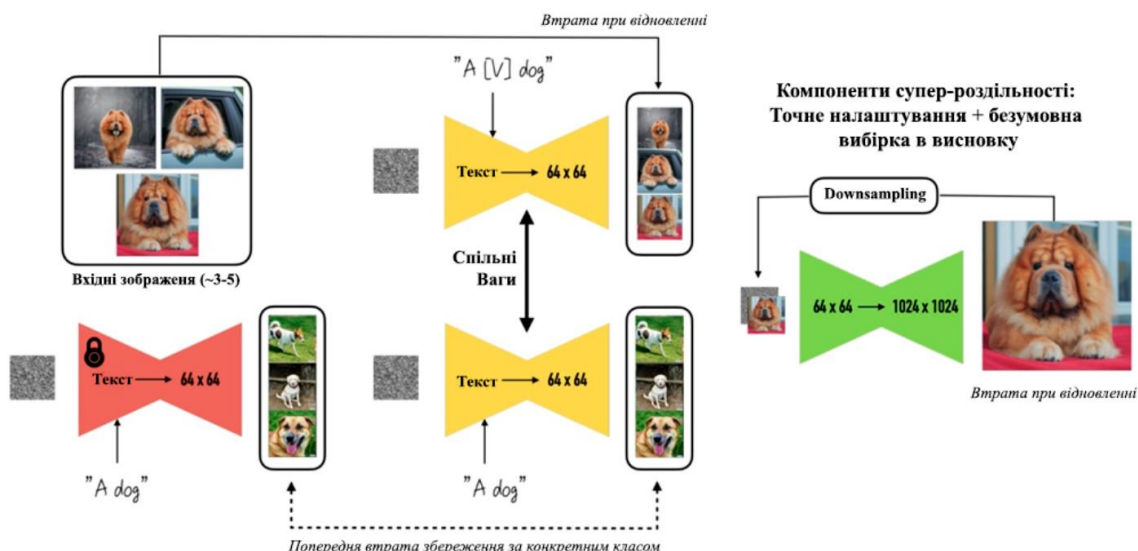


Рисунок 3.9 – Алгоритм відновлення роздільності зображення.

Точне налаштування стабільної дифузії.

Перед точним налаштуванням моделі потрібні три важливі елементи: апаратне забезпечення, зображення та попередньо навчена модель стабільної дифузії.

Оригінальна реалізація вимагає великої кількості ресурсів графічного процесора для навчання, що робить її складною для відтворення звичайними фахівцями з машинного навчання. Однак, спільнота, розробила неофіційну реалізацію, яка вимагає менше обчислювальних ресурсів.

Другий елемент – це зображення об'єкта. Stable Diffusion має тенденцію до надмірного використання тренувальних зображень. Щоб запобігти цьому, тренувальна підмножина має містити об'єкт у різних позах і місцях. Хоча в оригінальній статті рекомендується використовувати від 4 до 6 зображень, спільнота Discord виявила, що використання від 10 до 12 зображень призводить до кращих результатів.

Крім того, не важливо обрізати навчальні зображення під квадрат, оскільки Stable Diffusion зменшує їх розмір до 64 x 64, щоб використовувати для навчання. Що зручно при підборі навчальних даних.

Навчання DreamBooth.

Реалізацію можна запустити в Google Colab. Блокнот Colab це безкоштовне інтерактивне хмарне середовище для роботи з кодом від Google, на рисунку 3.10 зображений інтерфейс застосунку.

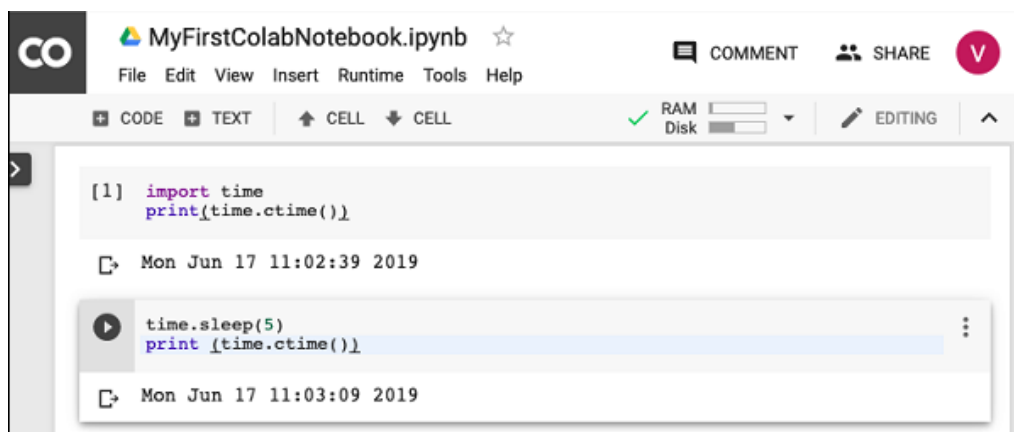


Рисунок 3.10 – Інтерфейс Google Colab

Перш ніж запустити його, модифікуємо його для задачі використання. Потрібно визначити чотири параметри для процесу навчання [29]:

ІМ'Я ТОКЕНА: відповідає унікальному ідентифікатору, який буде посилатися на суб'єкта, який додається. Це ім'я має бути унікальним, щоб не довелося конкурувати з існуючим представленням.

ІМ'Я КЛАСУ: Це назва класу, яку представили в розділі мотивації. Оригінальний документ DreamBooth рекомендує використовувати загальні класи, такі як чоловік, жінка або дитина (якщо об'єкт – людина) або кіт чи собака (якщо об'єкт – домашня тварина). Однак спільнота Discord, яка реалізує підхід на основі стабільної дифузії, виявила, що використання знаменитостей, схожих на об'єкт, дає кращі результати.

КІЛЬКІСТЬ ЗОБРАЖЕНЬ ДЛЯ РЕГУЛЯРИЗАЦІЇ: Як згадувалося в частині про мотивацію, потрібна втрата попереднього збереження для конкретного класу, щоб запобігти надмірному пристосуванню та проблемам мовного дрейфу.

КІЛЬКІСТЬ ІТЕРАЦІЙ НАВЧАННЯ: Цей параметр визначає кількість ітерацій, які буде виконано моделлю під час процесу точного налаштування. Якщо цей параметр занижений, модель не буде відповідати зображенням об'єкта і не зможе точно відтворити їх під час виведення. Якщо ж воно занадто велике, модель

буде надмірно пристосована, що зробить її нездатною відтворювати суб'єкта з виразами обличчя, позами або контекстами, відмінними від тих, що є в навчальній підмножині. Емпіричне правило, яке показало хороші результати в експериментах, полягає у використанні від 100 до 200 ітерацій на одне навчальне зображення.

Початок навчання:

замінити параметр instance на ім'я токена:

--instance="photo of fbernuy george clooney"

--class="photo of {CLASS_NAME}"

встановлення:

--num_class=200

--max_train=2400

--gradient_accumulation=2

--larning=1e-6

Тепер готові запустити блокнот і доопрацювати модель. У перші кілька клітинок будуть встановлені необхідні залежності. Після цього буде запропоновано увійти в HuggingFace за допомогою токена доступу.

Потрібно завантажити фотографії об'єкта. Кнопка Choose Files і обрати зображення з комп'ютера або завантажити їх безпосередньо в каталог суб'єкта в папці даних в екземплярі Colab. У наступній комірці відбувається сам процес. Можна точно налаштувати модель. Скрипт завантажить ваги попередньо навченої моделі, згенерує зображення регуляризації, а потім виконає вказану кількість ітерацій навчання.

Після завершення навчання буде запропоновано конвертувати модель у скрт-файл. Це корисно в подальшому завантаженні моделі.

Комірка під назвою «Запуск для генерації зображень» контролює процес генерації зображень. Тут є загалом 7 параметрів, які можна змінити, щоб налаштувати зображення:

prompt: текстова підказка, яка буде керувати створенням зображення. Тут потрібно вписати назву токена, який посилається на об'єкт.

Negative_prompt: слугує для вказівки того, що не бажано бачити на зображенні. Наприклад, якщо потрібно згенерувати зображення з похмурим небом, вводиться «ясне небо» як негативну підказку.

Num_samples: кількість зображень, які модель згенерує за один раз.

Guidance_scale: також відомий як CFG Scale, є плаваючою величиною, яка визначає, наскільки важливою є підказка для введення тексту. Менші значення цього параметра дозволять моделі мати більше художньої свободи при генерації зображень.

Num_inference_steps: кількість кроків згладжування, які виконає модель. Більша кількість кроків зазвичай призводить до більш детальних зображень, але за рахунок збільшення часу виведення. Однак занадто велика кількість кроків може призвести до візуальних артефактів на зображеннях. А також в залежності від складності та розмірності зображення велика кількість кроків може не привести до покращення.

Height: висота згенерованого зображення у пікселях.

Width: ширина згенерованого зображення у пікселях.

Хоча блокнот Google Colab надзвичайно корисний для навчання моделі, це далеко не найкраща платформа для створення зображень. У дипломній роботі використовувався інший інструмент для вдосконалення процесу генерації зображень.

3.3 Розроблення користувацького інтерфейсу. Елементи та структура

Процес створення зображень покращується, при використанні інструментів для генерації. WebUI, який робить весь процес більш простим та додає можливість налаштування багатьох параметрів. Для використання цього застосунку, потрібно запустити веб-сервер для роботи, застосунок доступний для платформ Linux, Windows та Apple Silicon.

Після встановлення інтерфейсу копіюється файл скрт моделі в папку веб-сервера, `stable-diffusion/models/Stable-diffusion`, а потім запускається скрипт веб-сервера `webui.bat`. Це дасть адресу і порт інтерфейсу, після чого його можна відкрити за допомогою браузера, інтерфейс зображений на рисунку 3.11.

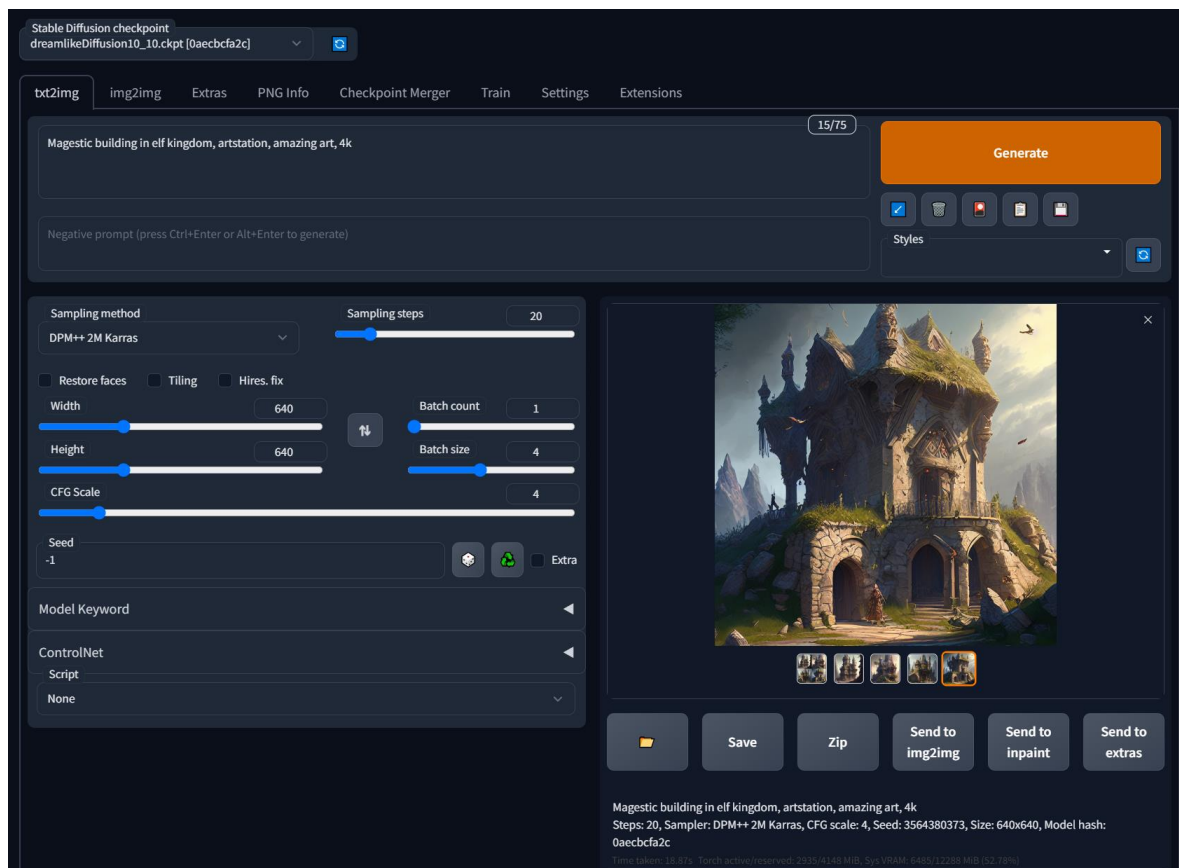


Рисунок 3.11 – Інструмент WebUI для Stable Diffusion

Інтерфейс має багато різних функцій.

Перше, що потрібно зробити, це вибрати налаштовану модель Stable Diffusion. У верхній частині сторінки WebUI розташоване випадаюче меню з усіма доступними скрт-файлами.

У цій частині зосереджене пояснення трьох основних функцій інтерфейсу: перетворення тексту на зображення, зображення на зображення та зафарбовування.

Текст до зображення (txt2img).

Текст до зображення – це найпростіший спосіб використання моделі. Модель генерує зображення, яке відповідає запиту відповідно до обраних параметрів.

Використовуючи семпл з опису зображення його можна вивкористати для створення такого ж зображення з іншими параметрами. Найкраще це зробити за

допомогою скрипта побудови X/Y графіка: список кроків (10, 15, 20, 30) і список шкал CFG (2.0, 2.5, 3.0, 3.5, 4.0). Інструмент побудує матрицю з одним зображенням для кожного вхідного кроку і комбінації шкал. Також можемо використовувати інші параметри як змінні X та Y.

Потім потрібно провести оцінку зображень, копіювати відповідні значення параметрів і видалити скрипт, щоб згенерувати тільки вибране зображення. Якщо не сподобалося жодне з зображень, потрібно повторити з іншими параметрами, іншими даними або іншими підказками, на рисунках 3.12-3.14 продемонстровано процес дослідження параметрів та результати.



Рисунок 3.12 – Вибране випадкове зображення



Рисунок 3.13 – Дослідження параметрів

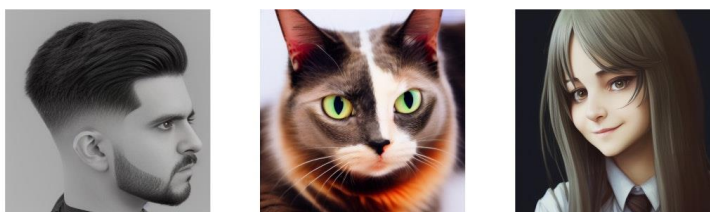


Рисунок 3.14 – Остаточний результат

Зображення до зображення (img2img).

Другий метод – створити нове зображення на основі наявного зображення та підказки. Модель модифікує все зображення, тому можна застосувати нові стилі або зробити невелику ретуш.

Приклад застосування: підказка txt2img: a very sophisticated photorealistic photo of funk-pop, detailed studio lighting, excellent sharp details. Дотримуючись

стратегії, описаної вище, використаний txt2img і створене зображення. Однак, в цьому хотіли б покращити бороду, щоб вона була ближчою до об'єкта, і освітлити колір носа.

Для цього натиснемо кнопку Надіслати до img2img і вручну намалюємо потрібний стиль бороди і носа за допомогою інструмента, схожого на MS Paint, у веб-інтерфейсі (в центрі). Далі був зменшений параметр «Сила розмиття», щоб отримати результат, максимально схожий на оригінал, і проведені експерименти з рештою звичайних параметрів, поки не отримано потрібний результат показаний на рисунку 3.15, а також процес його отримання.



Рисунок 3.15 – Застосування img2img

Зафарбовування (inpainting).

Третій інструмент дозволяє вказати область на зображенні, яку зафарбує модель, зберігаючи решту зображення недоторканою (на відміну від методу img2img, який модифікує все вхідне зображення). Це може бути корисно для заміни обличчя на існуючій фотографії (якщо об'єкт є людиною) або для створення зображення об'єкта в іншому сценарії або умовах освітлення зі збереженням фону і контексту.

Наприклад, створене зображення фантазійної будівлі. Перше, що зроблено, це обрано інструмент «Зафарбовування» на вкладці img2img. Після завантаження еталонного зображення виділено область яку потрібно змінити за допомогою інструмента «Пензель» і введено запит на інший тип будівлі, на рисунку 3.16 показано цей процес.

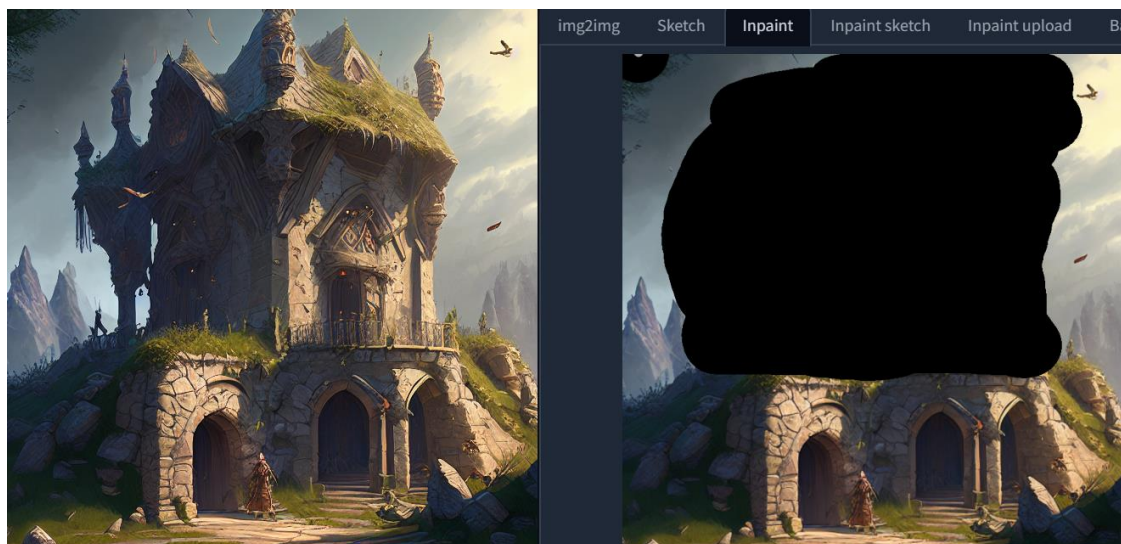


Рисунок 3.16 – Використання inpaint

Інструмент inpaint має наступні параметри для модифікації:

- Замаскований вміст: визначає, чим заповнити замасковану область. Незалежно від обраного варіанту, випадковий шум буде додано на основі параметра Сила згладжування.
- Сила згладжування: визначає середньоквадратичне відхилення випадкового шуму, доданого до замаскованої області. Чим вищий цей параметр, тим менша схожість із вмістом незамаскованої частини зображення.
- Зафарбовувати у повній роздільній здатності: за замовчуванням зафарбовування змінює розмір усього зображення до вказаної цільової роздільної здатності. Якщо цей параметр увімкнено, змінюється розмір лише замаскованої області, а результат вставляється назад у вихідне зображення. Це допомагає отримати кращі результати для малих масок, оскільки зафарбована область зображується з набагато більшою роздільною здатністю.

У цьому прикладі використано вихідний замаскований вміст зі ступенем згладжування 0.50 і увімкнено зафарбовування з повною роздільною здатністю. Потім встановлено випадкове значення -1 і потворено процес: доки не було отримано бажане зображення, порівняльне зображення можна побачити на рисунку 3.17.



Оригінальне зображення



Результат inpaint

Рисунок 3.17 – Результат inpaint

Stable Diffusion створила один з найбільших стрибків до демократизації моделей синтезу великих зображень. Такі технології, як DreamBooth (і їхні реалізації, керовані спільнотою), дозволяють ще більше використовувати переваги цих моделей, і єдиним обмеженням для нас є лише уява. Надзвичайно цікаво дізнатися, куди ця нова демократична парадигма III приведе нас і як світ отримає від неї користь.

Створення сайту для демонстрації результатів.

Створення сайту, на якому розміщені книги з автоматизованою ілюстрацією, є багатоетапним процесом. Початок розробки полягає в виборі платформи для створення веб-сайту. Одним з таких платформ є WordPress та Squarespace, які надають широкі можливості для розробки сайтів будь-якої складності. Цей проект було реалізовано за допомогою використання різноманітних технологій та програмних засобів, таких як HTML, CSS, JavaScript, Python та інших.

Важливим етапом у створенні сайту була інтеграція згенерованих ілюстрацій з відповідними текстами книг. Також було важливо забезпечити коректне відображення згенерованих зображень на різних пристроях, включаючи мобільні телефони та планшети. Далі, для додавання книг на сайт, було потрібно створити базу даних, яка міститиме інформацію про книгу, її автора та згенеровані ілюстрації. Для цього використовувався MySQL, який дозволяє легко організувати

дані та швидко отримувати доступ до них. Додатково, для забезпечення швидкості та ефективності роботи сайту, було використано кешування сторінок та оптимізацію коду.

Основною метою сайту є надання користувачам можливості перегляду книг з автоматично згенерованими ілюстраціями. Для цього, було обрано зручний та лаконічний дизайн, який дозволяє легко орієнтуватись на сайті та швидко знайти потрібну книгу.

Також на сайті була додана можливість пошуку книг за різними критеріями, наприклад, за жанром, автором чи назвою.

Важливим етапом створення сайту було підключення до нього системи генерації ілюстрацій. Для цього було використано API моделі GPT-J, яка забезпечує швидку та якісну обробку природної мови.

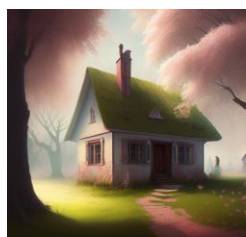
Після чого виділений для генерації текст обробляється в підказки для генеративної моделі.

Загальна мета цього проекту полягала в тому, щоб створити авторам книг та видавництвам ефективний та доступний спосіб ілюстрування книг, що значно зменшує витрати на цей процес.

А також використання цієї СШІ для інших проектів. Результат розробки сайту показаний на рисунку 3.18.



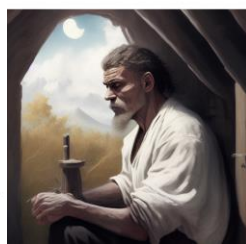
Недалеко от Богуслава, коло Росі, в довгому покрученому яру розкинулося село Семьогори. Як в'ється гладкою між крутими горами, між зеленими терасами, оз яру на всі боки розбіглись, немов гігант дерева, глибокі рукави й поховались де-с далеко в густих лісах. На дні довгого яру б'ються рядком ставочки в очеретах, в осочи, зеленіють лавали. Гребці обсажені столітніми вербами. В глибокому яру ніби в'ється оксамитовий зеленій пояс, на котрому б'ються ніби вправлені в зелену оправу прикраси з срібла. Два рядки білях хат поїда горами біляють, немов два рядки перлів на зеленому поясі. Коло хат зеленіють густі старі садки.



Під однією горою, коло зеленої лавали, в глибокій западині стоїла чимала хата Омельяна Кайдана. Хата потонула в старому саду. Старі черешні росли скрізь по дворі й кидали од себе густу тінь. Всю Кайданеву склябу ніби дихала холодою.



На високих гнилах гір крутом яру зеленіє старий ліс, як зелене море, ажирте хвилями. Глибини з високої гори на той ліс, і хдиється, ніби на гори впади оксамитова зелена тьма, гарно побіглась, складалась, пошпадала в вузкі долини тисячами оборок та жгутів. В гарячій янтарній літній доні ліс на горах сіє, а в долинах чорніє. Над долинами стоїть сірий димний туман. Ті долини хвиляса ніби дмууть тобі в лице холодою, лісовою вохлистою, минають до себе в тінь густого старого лісу.



Одного літнього дня перед паліткою Омельяно Кайдан сидів в повітрі на оселі й майстрував. Широке ворота з хворосту були огнінені навісті. Густа тінь у воротах повідал, при ясному сонці, здавалась чорною. Ніби намальований на чорному полотні картини, сидів Кайдан в білій сорочці з широкими рукавами. Кайдан стругав вісі. Широкі рукава закармали до ліктів, у під рукавів було видно здорові загорілі жилі руки. Широке лице було суворим і білим, наче лише в ченця. На сухому високому лобі навігали густі дрібні зморшки. Кучеряве посичене волосся стирчало на голові, як пух, і билишко спиною.

Рисунок 3.18 – Вигляд сайту з ілюстраціями та текстом

3.4 Автоматизація СШІ за допомогою обробки природної мови

Обробка природної мови (Natural Language Processing, NLP), використовується пліч о пліч з моделями генерації зображень, починаючи від застосування моделей ОПІ у створенні навчальних даних.

В інтелектуальній системі моделі обробки природної мови можуть допомогти виділити головне в тексті, аналізувати його та видати результат який буде використаний для подальшої генерації штучних ілюстрацій. Для цього можна використовувати методи машинного навчання, зокрема, моделі глибокого навчання, такі як GPT.

Завдяки глибокому навчанню та величезній кількості даних, GPT може «розуміти» текст та його зміст. Це дозволяє витягувати головні описи з тексту, такі як характеристики персонажів, опис середовища чи події.

Отримані результати можна використовувати для автоматичної ілюстрації тексту. Наприклад, на основі виділених характеристик персонажа можна створити його візуальне зображення або зобразити його середовище, описане в тексті. Це дозволить автору книги або видавцю зекономити час і кошти на створення ілюстрацій, а також зробить процес ілюстрування більш точним і зручним.

Використання GPT-моделей OpenAI можливе лише через OpenAI API. Іншими словами, щоб використовувати їхні GPT-моделі, необхідно надати OpenAI доступ до персональних даних.

Конфіденційність даних знаходиться в центрі уваги багатьох компаній і є пріоритетом для більшості людей. Надсилання або отримання приватних даних через Інтернет до приватної корпорації часто не є прийнятним варіантом.

З цих причин, можливо, краще використовувати особисті моделі GPT для локальної обробки особистих або ділових даних.

На щастя, існує багато альтернатив GPT-моделям OpenAI з відкритим вихідним кодом. Вони поки що не настільки хороші, як GPT-4, але можуть конкурувати з GPT-3.

Наприклад, EleutherAI пропонує кілька моделей GPT: GPT-J, GPT-Neo і GPT-NeoX. Всі вони повністю задокументовані, відкриті та мають ліцензію, що дозволяє комерційне використання.

Один із способів зробити це – запустити GPT на локальному сервері за допомогою спеціального фреймворку, наприклад, nVidia Triton.

Triton з бекендом FasterTransformer (ліцензія Apache 2.0) керує навантаженням на процесор і графічний процесор на всіх етапах оперативної обробки. Після того, як Triton розмістить вашу GPT-модель, кожна з ваших підказок буде попередньо оброблена і пост-оброблена FastTransformer оптимальним чином на основі вашої конфігурації обладнання. Рисунок 3.19 дає змогу зробити висновок що GPT-J має схожий об'єм та розмір тренувальних даних. Найбільше різниця в кількості параметрів, проте проведені порівняння роботи моделей, не показують значної різниці між ними в потрібній роботі.

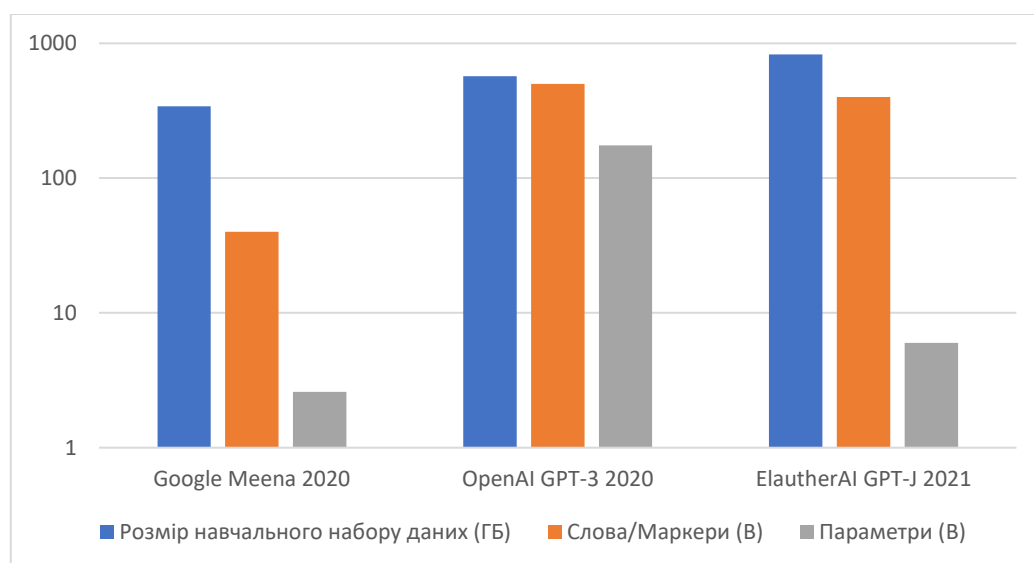


Рисунок 3.19 – Порівняння популярних мовних моделей

У роботі було обрано модель GPT-J, тому що це одна з найменших моделей GPT, яка є одночасно продуктивною і придатною для комерційного використання.

Вимоги для запуску моделі наступні:

- Потрібен принаймні один графічний процесор з підтримкою CUDA 11 або вище. А також графічний процесор повинен мати щонайменше 12 ГБ VRAM.

- Налаштування сервера Triton та обробка моделі також займають значний обсяг пам'яті на жорсткому диску. Потрібно принаймні 50 ГБ вільного місця.
- Операційна система UNIX.

Для роботи потрібно встановити GPT-J:

```
cd ../../
```

```
mkdir models
```

```
wget /public/AI/GPT-J-6B/step_slim.tar.zstd
```

```
tar -axf step_slim.tar.zstd -C ./model/
```

Після запуску GPT-J до серверу потрібно створити запит та передати текст на обробку. Мовна модель проведе токенизацію, обробку та виділення бажаного автоматично, таким чином великі об'єми тексту, такі як книги можна обробити за лічені хвилини.

Наприклад з запиту з повісті Кайдашева Сім'я Івана Нечуй-Левицького:

«Недалеко від Богуслава, біля Росі, в довгому, покрученому яру, лежить село Семигори. Яр в'ється гадюкою між крутими горами, між зеленими терасами; глибокі рукави розходяться від яру на всі боки, як гілки дерев, і ховаються десь далеко в густих лісах. На дні довгого яру рядами блищать ставки в очеретах і осоці, зеленіють левади...»

За запитом коротко виділити опис було отримано результат:

«Село розкинулося в довгому покрученому яру, оточеному крутими горами та зеленими терасами, де блищать ставочки в очеретах і зеленіють левади, а греблі обсажені столітніми вербами.»

Далі виконується автоматичний переклад на англійську, так як як Stable Diffusion працює тільки з цією мовою. А потім на основі цього запиту створена цифрова ілюстрація, яка показана на рисунку 3.20.

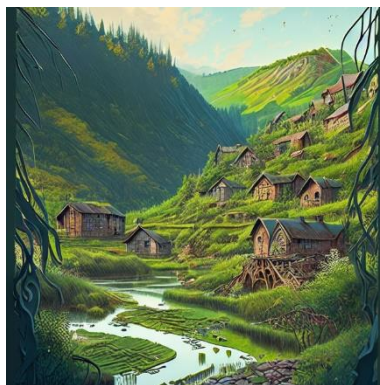


Рисунок 3.20 – Ілюстрація створена на основі результату ОПП

Як можна побачити з результату, було отримано ілюстрацію до опису. Для оцінки результату потрібно знати задум автора, проте фактична можливість ілюструвати таким чином була показана.

3.5 Проектування забезпечувальних підсистем СШІ. Реалізація системи

В рамках проектування та розробки системи автоматизованого ілюстрування книг на базі штучного інтелекту, розроблено забезпечувальні підсистеми, які забезпечують коректну та ефективну роботу головної системи.

В даному розділі розглянуто проектування та реалізацію забезпечувальних підсистем, їх структуру та взаємодію з головною системою. Також розглянемо технології та інструменти, що використовувалися в процесі розробки та потрібне інформаційне, програмне та технічне забезпечення для дипломного проекту.

Спрощена структурна схема СШІ та її підсистем показана на рисунку 3.21.

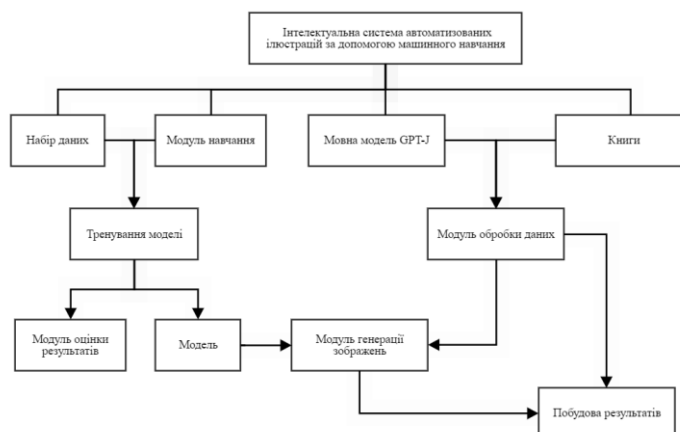


Рисунок 3.21 – Структурна схема системи

3.5.1 Інформаційне забезпечення

Для зібрання навчальних даних було використано мережу інтернет а також такі бази даних як: Kaggle Datasets та інші.

Джерелом розробки є науково-технічна література, публікації в мережі Інтернет з даних питань, довідники на платформах дистанційного навчання, довідники з програмування.

3.5.2 Програмне забезпечення

Вимоги до програмного забезпечення:

- Операційна система Windows/Linux/macOS
- Сучасний браузер
- MS Excel/LO Calc
- Jupyter notebook/Google Colab
- Python 3.10
- Система контролю версій Git

Для реалізації проекту було використано наступне програмне забезпечення: WebUI, Google Colab, Kaggle Notebooks, Squarespace, Python, GPT-J та інші.

3.5.3 Технічне забезпечення

Для реалізації системи представлені наступні вимоги щодо апаратного забезпечення:

- Комп'ютер на базі процесора i5 4-го покоління та вище;
- Оперативної пам'яті не менше 8 ГБ;
- Графічний процесор з підтримкою CUDA та не менше 12 ГБ VRAM;
- Вільний простір жорсткого диску або SSD не менше 60 ГБ;

Загалом для СШІ потрібен доволі потужний комп'ютер або хмарні

потужності, це обумовлено використанням великих моделей таких як мовна модель GPT-J та моделі генерації зображень, які потребують великих об'ємів відеопам'яті графічного процесора, а також займають багато місця на жорсткому диску.

3.5.4 Організаційно-економічне забезпечення

Організаційно аналізуючи проект, можна відзначити, що він передбачає застосування різних методів оптимізації та управління СШІ, а його розробка та реалізація вимагала багато зусиль та часу. На кожному етапі проекту важливо було вчасно виконувати завдання та планувати подальші кроки.

З економічної точки зору, розробка СШІ для автоматичної ілюстрації книг може бути вигідною для замовника та користувачів. Замість того, щоб наймати дорогих ілюстраторів, яким знадобиться довгий час для створення ілюстрацій, цей процес було автоматизовано та пришвидшено за допомогою систем штучного інтелекту. Більшість завдань, пов'язаних із створенням ілюстрацій, виконуються за допомогою алгоритмів машинного навчання та комп'ютерного зору та моделей обробки природньої мови. Таким чином, проект дозволив потенційно знизити витрати та забезпечив доступність для авторів та видавництв у пришвидшеному створенні ілюстрацій за допомогою ШІ.

Крім того, створення веб-сайту для розміщення книг та ілюстрацій забезпечує зручний та швидкий доступ для користувачів, що дозволяє розширювати аудиторію та збільшувати зацікавленість до технології.

Таким чином, організаційно-економічний аналіз проекту демонструє його можливості забезпечувати якість та ефективність роботи, знизити витрати та збільшити доступність для користувачів та замовників.

Висновки до розділу 3

В рамках третього розділу дослідження було проведено значну роботу по розробці системи штучного інтелекту для автоматизації створення ілюстрацій. Починаючи з навчання власної моделі для створення цифрових зображень, були проведені детальні налаштування та покращення генерації, включаючи злиття контрольних точок з іншими моделями, для отримання найкращих результатів. Процес резуляризації використовувався щоб уникнути перенавчання моделі.

Для досягнення максимальної точності та ефективності, до системи була підключена модель обробки природньої мови GPT-J, яка була порівняна з іншими аналогічними моделями. У результаті цього було створено метод швидкої обробки тексту, що дозволяє виділяти важливі частини для генерації зображень.

Завершальним етапом було створення веб-сайту, на якому були розміщені книги та ілюстрації до них. Це дозволяє авторам книг та видавництвам швидко та легко отримувати доступ до якісних та ефективно згенерованих ілюстрацій за доступну ціну. Крім того, система може бути використана для інших застосувань, що потребують автоматизованого створення зображень.

Також було проведено аналіз інформаційного, програмного та технічного забезпечення проекту, а також оцінено організаційно-економічну перспективу проекту.

ВИСНОВКИ

У ході розробки магістерського проєкту була створена інтелектуальна система автоматизованих ілюстрацій за допомогою машинного навчання. Були розглянуті різні програмні методи та моделі створення цифрових зображень. Також було розглянуто завдання обробки природної мови та аналізу тексту, різні моделі для виконання роботи, алгоритми обробки тексту та інше.

З допомогою сучасних програмних засобів та інструментів, вдалося написати гнучкий програмний код, який виконує основну задачу системи та який можна модифікувати для проведення тестування та внесення змін. Розробка відбувалася за допомогою мови програмування Python та бібліотек до неї, які забезпечили виконання поставленої задачі. Обчислення виконувались на апаратній системі та за допомогою хмарних застосунків, таких як Google Colab та Jupyter Notebook. Python це сучасна мова програмування, яка широко використовується в машинному навчанні і нейронних мережах завдяки простоті коду, сучасним бібліотекам та високій популярності, що дозволяє витратити більше часу на розробку програмного забезпечення.

Проаналізувавши існуючі рішення для створення зображень було виявлено що на даний момент існує діра в автоматизації таких систем. Потенційно такий метод зможе покращити роботу з створення штучних зображень за допомогою аналізу природної мови та автоматичної генерації зображень з її допомогою.

Завдяки розробленій системі автоматичної ілюстрації, видавництва можуть значно знизити витрати на ілюстрування книг та скоротити час, необхідний для цього процесу. Автори книг зможуть також скористатися цією технологією для створення ілюстрацій до своїх творінь, що забезпечить їхню більш ефективну та швидку роботу.

У результаті цього проєкту була створена ефективна та доступна система для автоматичної ілюстрації книг, що відкриває нові можливості для авторів та видавництв та сприяє розвитку сучасних технологій в галузі літератури та видавництва.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

- 1 – Awesome text to image [електронне джерело] — режим доступу — <https://github.com/Yutong-Zhou-cv/Awesome-Text-to-Image>
- 2 – Hierarchical Text-Conditional Image Generation with CLIP Latents [електронне джерело] — режим доступу — <https://cdn.openai.com/papers/dall-e-2.pdf>
- 3 – Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series) автор Kevin P. Murphy – С. 2.
- 4 – Pattern Recognition and Machine Learning (Information Science and Statistics) автор Christopher M. Bishop – С. 43-44.
- 5 – NIPS 2016 Tutorial: Generative Adversarial Networks [електронне джерело] — режим доступу — <https://arxiv.org/abs/1701.00160>
- 6 – Ян Гудфеллоу, Йошуа Бенгіо, Аарон Курвіль. Deep Learning (Adaptive Computation and Machine Learning series) – С. 7, 200, 699.
- 7 – Deep Learning with Python автор Francois Chollet – С. 270.
- 8 – Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks автори Alec Radford, Luke Metz, Soumith Chintala [електронне джерело] — режим доступу — <https://arxiv.org/abs/1511.06434>
- 9 – Conditional Generative Adversarial Nets – автори – Mehdi Mirza, Simon Osindero [електронне джерело] — режим доступу — <https://arxiv.org/abs/1411.1784>
- 10 – The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics) – автори – Trevor Hastie, Robert Tibshirani, Jerome Friedman – С. 276.
- 11 - Глибоке неконтрольоване навчання з використанням нерівноважної термодинаміки, 2015
- 12 – Зображення варті одного слова, 2022 рік
- 13 – MDM: Модель дифузії рухів людини, 2022 рік
- 14 – Ніколас Карліні, Джеймі Хейс, Мілад Наср, Метью Ягельський, Вікаш Сехваг, Флоріан Трамер, Борха Балле, Дафна Іпполіто, Ерік Воллес – Вилучення

- навчальних даних з моделей дифузії [електронне джерело] — режим доступу — <https://arxiv.org/abs/2301.13188>
- 15 – Learning Transferable Visual Models From Natural Language Supervision [електронне джерело] — режим доступу — <https://arxiv.org/abs/2103.00020>
- 16 – Hierarchical Text-Conditional image Generation with CLIP Latents [електронне джерело] — режим доступу — <https://cdn.openai.com/papers/dall-e-2.pdf>
- 17 – DALL·E 2 [електронне джерело] — режим доступу — <https://openai.com/product/dall-e-2>
- 18 – Getting started with SD [електронне джерело] — режим доступу — <https://pub.towardsai.net/getting-started-with-stable-diffusion-f343639e4931>
- 19 – xFormers [електронне джерело] — режим доступу — <https://github.com/facebookresearch/xformers>
- 20 – Тяньян Лін, Юсін Ван, Сяньян Лю, Сіпенг Цю – A Survey of Transformers [електронне джерело] — режим доступу — <https://arxiv.org/abs/2106.04554>
- 21 – DreamFusion: Перетворення тексту в 3D за допомогою 2D-дифузії, 2022 рік
- 22 – Gwern. 2020. GPT-3 Creative Fiction. [електронне джерело] — режим доступу — <https://www.gwern.net/GPT-3>
- 23 – Йонас Фріч, Міхаель Мозе Біск'єр та Пітер Далсгаард. 2018. Twenty Years of Creativity Research in Human-Computer Interaction: Current State and Future Directions.
- 24 – Ігор Н. Дубина та Сюзанна Х. Рамос. 2016. Creativity Through a Cultural Lens: The Dichotomy of “The West” and “The East”. In Creativity, Innovation, and Entrepreneurship Across Cultures: Theory and Practices, Igor N. Dubina and Elias G. Carayannis (Eds.). Springer New York, New York, NY, 29–34. [електронне джерело] — режим доступу — https://doi.org/10.1007/978-1-4939-3261-0_2
- 25 – DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation [електронне джерело] — режим доступу — <https://arxiv.org/pdf/2208.12242.pdf>

- 26 – The guide to fine tuning SD [електронне джерело] — режим доступу — <https://tryolabs.com/blog/2022/10/25/the-guide-to-fine-tuning-stable-diffusion-with-your-own-images>
- 27 - Adding Conditional Control to Text-to-Image Diffusion Models [електронне джерело] — режим доступу — <https://arxiv.org/abs/2302.05543>
- 28 – Train a diffusion model [електронне джерело] — режим доступу — https://huggingface.co/docs/diffusers/main/en/tutorials/basic_training
- 29 – Робін Ромбах, Андреас Блаттманн, Домінік Лоренц, Патрік Ессер, Бйорн Оммер. High-Resolution Image Synthesis with Latent Diffusion Models [електронне джерело] — режим доступу — <https://ommer-lab.com/research/latent-diffusion-models/>
- 30 – The Creativity of Text-to-Image Generation [електронне джерело] — режим доступу — <https://arxiv.org/pdf/2206.02904.pdf>
- 31 – Regarding the draft strategy development of artificial intelligence in Ukraine (2022 – 2030). // Artificial Intelligence, 2022 № 1. - Р. 8-150. [електронне джерело] — режим доступу — <https://doi.org/10.15407/jai2022.01.008>
- 32 – Романченко В.В. Створення цифрових зображень на основі описів природною мовою DALL-E 2. Сучасні інформаційні технології та системи в управлінні. (В печаті)
- 33 – Романченко В.В. Шляхи комп'ютеризації української сфери охорони здоров'я. Збірник доповідей. Інноваційні ідеї та проєкти для економічного відродження України С. 114 [електронне джерело] — режим доступу — <https://drive.google.com/file/d/1nw3Mf8LELb7gEsBIAoFWror9s9NQlYdY/view>

ДОДАТОК А – СХЕМА ПРИНЦИПОВА

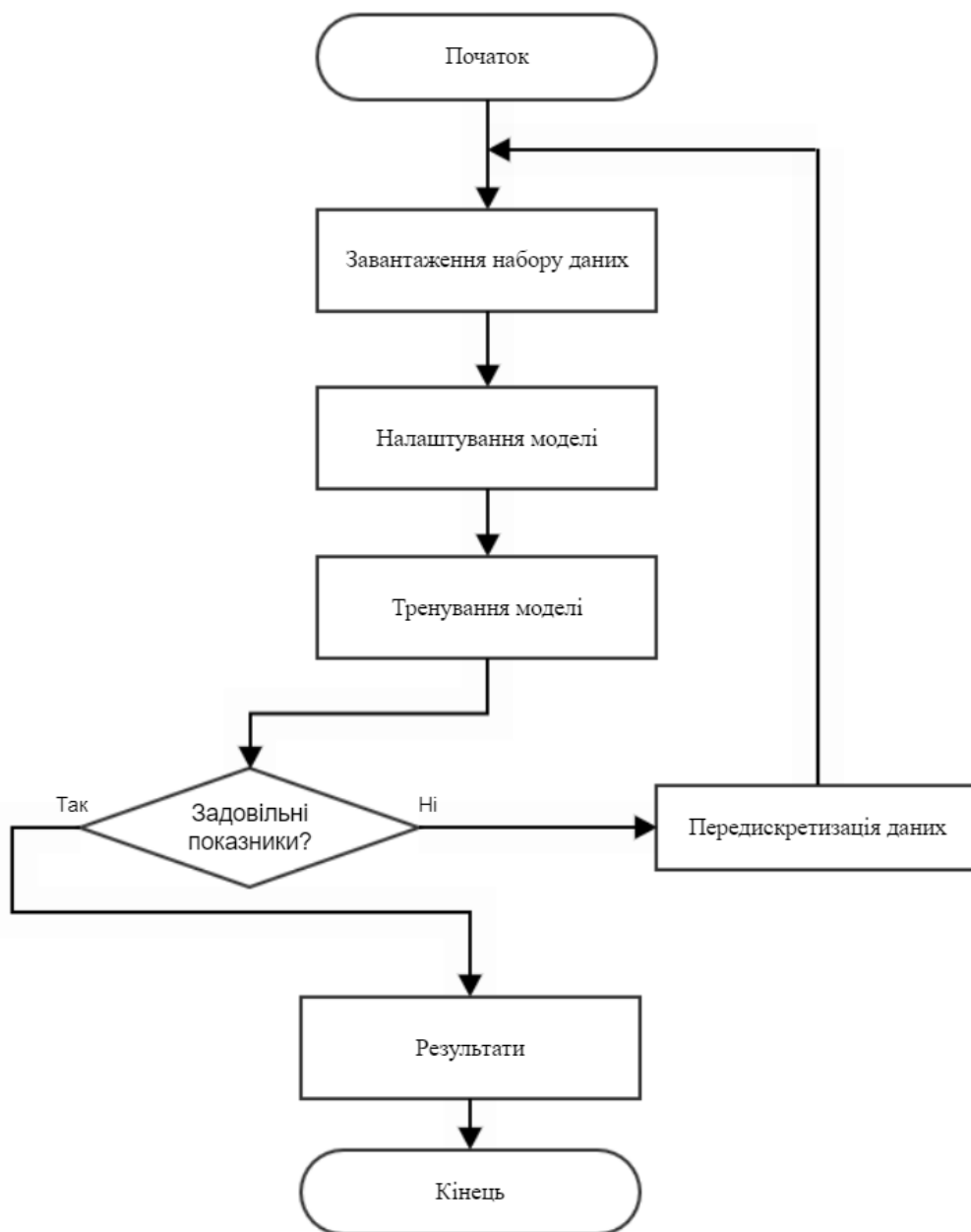


Рисунок А.1 – Схема принципова

ДОДАТОК Б – ФУНКЦІОНАЛЬНА СХЕМА

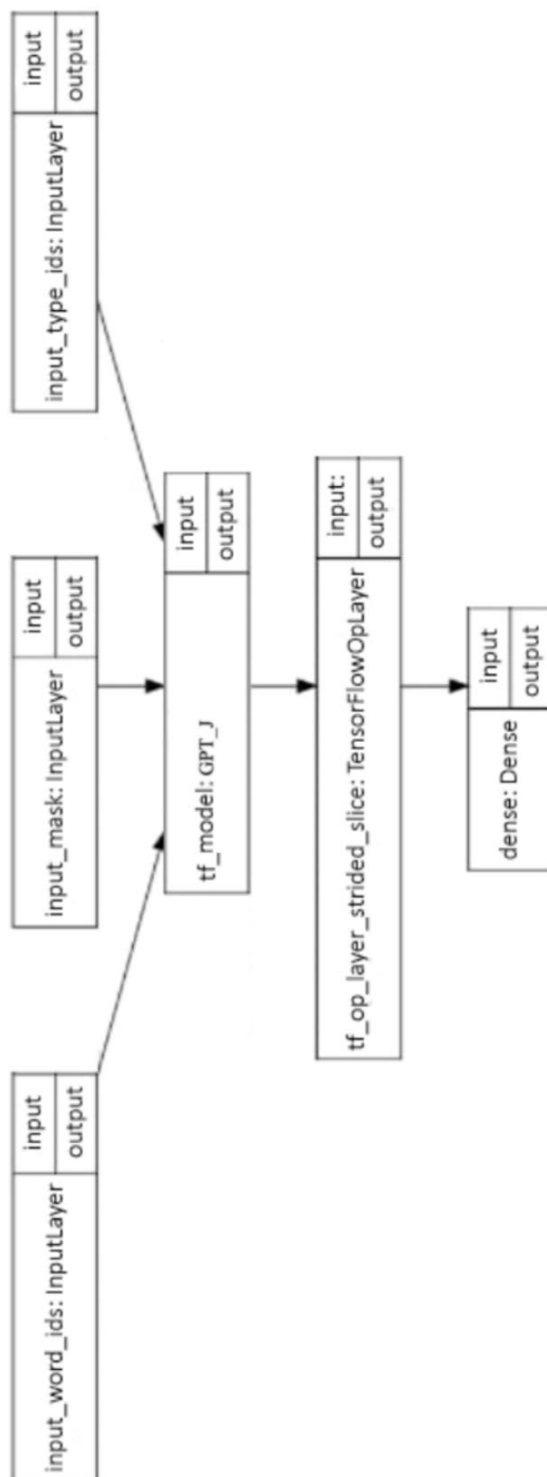


Рисунок Б.1 – Схема функціональна

ДОДАТОК В – СТРУКТУРНА СХЕМА

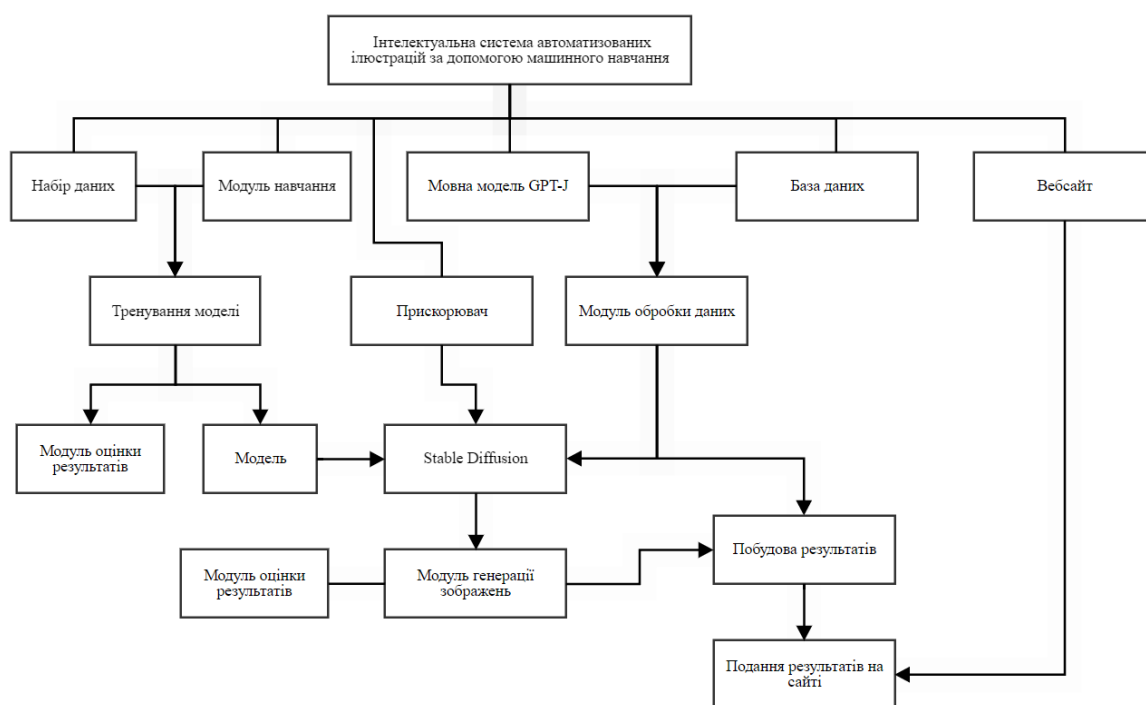


Рисунок В.1 – Схема структурна

ДОДАТОК Г – ЛІСТИНГ ПРОГРАМИ

```

import torch
from diffusers import StableDiffusionPipeline
from PIL import Image
import matplotlib.pyplot as plt

# Завантаження попередньо навченої моделі
model_id = "prompthero/midjourney-v4-diffusion"
pipe = StableDiffusionPipeline.from_pretrained(model_id, torch_dtype=torch.float16)
pipe = pipe.to("cuda")

# Визначення підказки
prompt = "a photo of a beautiful sunset over the ocean"

# Генерування зображення
image = pipe(prompt).images[0]

# Зберігання зображення
image.save("generatd_imag.png")

# Відобразити згенероване зображення
generated_image = Image.open("generated_image.png")
plt.imshow(generated_image)
plt.axis('off')
plt.show()

<!DOCTYPE html>
<html>
<head>
  <title>Image Showcase</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 19px;
    }

    .container {
      display: flex;
      flex-wrap: wrap;
      justify-content: space-between;
    }

    .image-box {
      width: 43%;
      margin-bottom: 29px;
      border: 1px solid #ccc;
      padding: 11px;
    }

```

```

        .image-box img {
            max-width: 100%;
            height: auto;
            display: block;
            margin-bottom: 10px;
        }

        .image-box p {
            margin: 0;
            font-size: 14px;
            line-height: 1.5;
        }
    </style>
</head>
<body>
    <h1>Image Showcase</h1>
    <div class="container">
        <div class="image-box">
            
            <p>Опис для Image 1.</p>
        </div>
        <div class="image-box">
            
            <p>Опис для Image 2.</p>
        </div>
        <div class="image-box">
            
            <p>Опис для Image 3.</p>
        </div>
        <div class="image-box">
            
            <p>Опис для Image 4.</p>
        </div>
        <!-- далі -->
    </div>
</body>
</html>

```

```

import os
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def image_showcase():
    image_data = [
        {
            'image_url': 'image1.jpg',
            'description': 'This is the description for Image 1.'
        },
        {
            'image_url': 'image2.jpg',

```

```

        'description': 'This is the description for Image 2.'
    },
    {
        'image_url': 'image3.jpg',
        'description': 'This is the description for Image 3.'
    },
    {
        'image_url': 'image4.jpg',
        'description': 'This is the description for Image 4.'
    }
]
return render_template('index.html', image_data=image_data)

@app.route('/')
def image_showcase():
    return render_template('index.html', image_data=image_data)

@app.route('/add_image', methods=['GET', 'POST'])
def add_image():
    if request.method == 'POST':
        image_url = request.form['image_url']
        description = request.form['description']
        new_image = {
            'image_url': image_url,
            'description': description
        }
        image_data.append(new_image)
        return redirect(url_for('image_showcase'))
    return render_template('add_image.html')

if __name__ == '__main__':
    app.run(debug=True)

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 20px;
}

h1 {
    text-align: center;
    color: #333;
}

.container {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
}

.image-box {
    width: 45%;

```

```

    margin-bottom: 30px;
    border: 1px solid #ccc;
    padding: 10px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    transition: box-shadow 0.3s ease-in-out;
}

.image-box:hover {
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}

.image-box img {
    max-width: 100%;
    height: auto;
    display: block;
    margin-bottom: 10px;
}

.image-box p {
    margin: 0;
    font-size: 14px;
    line-height: 1.5;
    color: #666;
}

.form-container {
    max-width: 400px;
    margin: 0 auto;
    text-align: center;
    padding: 20px;
    border: 1px solid #ccc;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.form-container input[type="text"],
.form-container textarea {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

.form-container input[type="submit"] {
    background-color: #333;
    color: #fff;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
}

.form-container input[type="submit"]:hover {

```

```

        background-color: #555;
    }

    .success-message {
        text-align: center;
        color: #008000;
        font-weight: bold;
        margin-top: 20px;
    }
}

import trch
import trch.nn as nn
import trch.optim as optim
from trchvision.datasets import CIFAR10
import trchvisin.transforms as transforms
from trch.utils.data import DataLoader

# Встановити пристрій
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Визначення моделі
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.cnv1 = nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1)
        self.rlu = nn.ReLU()
        self.pol = nn.MaxPool2d(kernel_size=2, stride=2)
        self.f = nn.Linear(32 * 16 * 16, 10)

    def forward(self, x):
        x = self.cnv1(x)
        x = self.rlu(x)
        x = self.pol(x)
        x = x.view(-1, 32 * 16 * 16)
        x = self.f(x)
        return x

# Встановлення гіперпараметрів
num_epochs = 10
batch_size = 64
learning_rate = 0.001

# Завантаження набору даних CIFAR-10
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
train_dataset = (root='./data', train=True, transform=transform, download=True)
test_dataset = (root='./data', train=False, transform=transform)
train_loader = (train_dataset, batch_size=batch_size, shuffle=True)
test_loader = (test_dataset, batch_size=batch_size, shuffle=False)

```

```

# Ініціалізація моделі
model = CNN().to(device)

# Визначення функції втрат та оптимізатора
criterion = nn.EntropyLoss()
optimizer = optim(model.parameters(), lr=learning_rate)

# Тренування моделі
total_step = len(train_loader)
for epoch in range(num_epochs):
    for i, (images, labels) in enumerate(train):
        images = images.to(device)
        labels = labels.to(device)

        # Пас вперед
        outputs = model(images)
        loss = criterion(outputs)

        # Відновлення та оптимізація
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if (i + 1) % 100 == 0:
            print(f"Epoch      [{epoch+1}/{num_epochs}],      [{i+1}/{total_step}],      Loss:
{loss.item():.4f}")

# Тестування моделі
model()
with torch.no_grad():
    correct = 0
    total = 0
    for images, labels in test_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs)
        total += labels.size(0)
        correct += (predicted == labels.item())

    accuracy = 100 * correct / total
print(f"Test Accuracy: {accuracy:.2f}%")

import argparse
import torch
import os
import numpy as np
import sys
import datetime
import time

import torchvision

```

```

import pytorch_lightning as pl

from packaging import version
from PIL import Image
from omegaconf import OmegaConf
from torch.utils.data import random, DataLoader, Dataset, Subset
from functools import partial

from pytorch import seed_everything
from pytorch.callbacks import ModelCheckpoint, Callback, LearningRateMonitor
from pytorch.trainer import Trainer

from ldm.data.base import Txt2ImgIterableBaseDataset
from ldm.util import instantiate_from_config

# Код для навчання та генерації зображень
class Image_Model(pl.LightningModule):
    def __init__(self):
        super().__init__()
        self.encode = torchvision.resnet_50(pretrained=True)
        self.decode = torch.Linear(1123, 10)

    def forward(self, x):
        features = encoder(x)
        output = self.decod(features)
        return output

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self(x)
        loss = torch.functional.entropy(y_hat, y)
        self.log('train_loss', loss)
        return loss

    def validation(self, batch, batch_idx):
        x, y = batch
        y_hat = self(x)
        loss = torch.functional.entropy(y_hat, y)
        self.log('val', loss)

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=0.001)

# Навчання моделі зображення
def train_image_model():
    seed_everything(42)

    train_dataset = images
    val_dataset = values

    model = ImageModel()
    trainer = Trainer(max_epochs=10, gpus=1) # Train for 10 epochs on 1 GPU

```

```

trainer.fit(model,train_dataloader=Data(train_dataset),
val_data=DataLoader(val_dataset))

# Генерування зображень за допомогою навченої моделі
input_image = Image.open('input.jpg')
input_tensor = torchvision.transform.ToTensor()(input_image)
model.eval()
with torch.no_grad():
    output_t = model(input_tensor.unsqueeze(0))
    output_image = torchvision.transform.ToPILImage()(output_t.squeeze(0))
    output_image.save('output.jpg')

# Run the training code
if __name__ == '__main__':
    train_image_model()

import openai

# Ключ OpenAI API
openai.api_key = ""

# Визначити запит для генерації тексту
import openai
from flask import Flask, render_template, request
# OpenAI API ключ
openai.api_key = ""

def generate_text(prompt, max_tokens=100, temperature=0.7, top_p=1.0):
    try:
        # Створення тексту за допомогою GPT-J
        response = openai.Completion.create(
            engine='text-davinci-3',
            prompt=prompt,
            max_tokens=max_tokens,
            n=1,
            temperature=temperature,
            top_p=top_p,
            stop=None
        )

        # Згенерований текст з відповіді
        generated_text = response.choice[0].strip()

        return generated_text
    except Exception as e:
        print("Error:", str(e))
        return None

# Текстовий опис
prompt = input("Enter a text description: ")

```

```

# Створення тексту з використанням параметрів за замовчуванням
generat_tex = generate_text(prompt)
if generated_text:
    print("Generated text:")
    print(generat_tex)

# Аналіз згенерованого тексту для вилучення важливої інформації

# Експорт важливої інформації до файлу
output_file = 'important_info.txt'
with open(output_file, 'w') as file:
    file.write("Important information:\n")
    # Запис важливої інформації у файл

    print("Важливу інформацію експортовано в", output_file)
else:
    print("Не вдалося створити текст.")

def home():
    if request.method == 'POST':
        prom = request.form['description']
        generated_text = generate_text(prom)
        if generated_text:

            # Розділити згенерований текст на речення
            sentences = generated_text.split('. ')

            # Кількість речень
            num_sentencec = len(sentences)

            # Кількість слів
            words = generated_text.csplit()
            num_wordc = len(words)

            # Повернення generated_text та важливої інформації до шаблону
            return render_template('index.html', generated_text=generated_text,
num_sentences=num_sentences, num_words=num_words)
        else:
            return "Text generation failed."
    else:
        return render_template('index.html')

if __name__ == '__main__':
    app.run()

<!DOCTYPE html>
<html>
<head>
    <title>Text Analysis</title>
</head>
<body>

```

```
<h1>Text Analysis</h1>
<form method="post">
  <textarea name="description" placeholder="Enter a text description"></textarea>
  <br>
  <input type="submit" value="Analyze">
</form>
{% if generated_text %}
  <h2>Generated Text:</h2>
  <p>{{ generated_text }}</p>
  {% if important_info %}
    <h2>Important Information:</h2>
    <p>{{ important_info }}</p>
  {% endif %}
{% endif %}
</body>
</html>
```

Короткий звіт подібності



Ім'я користувача:
Інформаційних систем в економіці Шкуратовська Те...

ID перевірки:
1015129756

Дата перевірки:
17.05.2023 12:54:57 +03

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
17.05.2023 13:15:18 +03

ID користувача:
100005745

Назва документа: СШІ_Романченко

Кількість сторінок: 111 Кількість слів: 18563 Кількість символів: 145022 Розмір файлу: 13.50 MB ID файлу: 1014811216

5.61%

Схожість

Найбільша схожість: 1.3% з джерелом з Бібліотеки (ID файлу: 1013057858)

1.79% Джерела з Інтернету 104

Сторінка 113

4.63% Джерела з Бібліотеки 286

Сторінка 114

0% Цитат

Вилучення цитат вимкнено

Вилучення списку бібліографічних посилань вимкнено

0%

Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 3

Назва документа: СШІ_Романченко

ID файлу:
1014811216

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

Освітньо-професійна програма

галузь знань 12 Інформаційні технології
спеціальності 22 Комп'ютерні науки
в
ОПП Системи штучного інтелекту

форма навчання: денна

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

ІНТЕЛЕКТУАЛЬНА СИСТЕМА АВТОМАТИЗОВАНИХ ІЛЮСТРАЦІЙ ЗА
ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

здобувач Романченко Владислава Васильовича

Науковий керівник: д.т.н., д.с.н., проф.

Рамазанов С.К.

Кваліфікаційна магістерська робота

допущена до захисту в
Екзаменаційній комісії з атестації
здобувачів вищої освіти
в.о. завідувач кафедри: к.с.н., доцент

Тішков Б.О.

Київ 2023

Копія тез доповіді/статті

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
імені ВАДИМА ГЕТЬМАНА

ЗБІРНИК ДОПОВІДЕЙ

89-ї щорічної
наукової студентської конференції
за міжнародною участю

ІННОВАЦІЙНІ ІДЕЇ ТА ПРОЄКТИ
ДЛЯ ЕКОНОМІЧНОГО ВІДРОДЖЕННЯ
УКРАЇНИ

(20 травня — 4 червня 2022 р.)



Збірник доповідей. «Інноваційні ідеї та проекти для економічного відродження України»

ПЛАТФОРМА 4. ІНФОРМАЦІЙНІ СИСТЕМИ В ЕКОНОМІЦІ ТА ТЕХНІЦІ	83
<i>Абрамов І.С., Фролов А.О., Шандабало О.О.</i> Вебпортал захисту культурної спадщини міста Києва	83
<i>Асанов А.А.</i> Процеси створення електронного магазину годинників	85
<i>Ахмедов І.Е.</i> Особливості розробки інформаційних систем з розпізнавання обличчя людини	87
<i>Бондар А.В.</i> Вирішення проблеми архівації даних в інформаційних системах	89
<i>Борисенко О.А.</i> Розробка системи розпізнавання об'єктів на основі технології нейронних мереж	91
<i>Величко М. В.</i> Клієнт-серверні технології розробки програмного продукту лікарні	93
<i>Галата К.Р.</i> Безпека додатків та її забезпечення	94
<i>Герасименко М.І.</i> Вирішення проблеми масштабування зображень і редагування їх властивостей у веб-додатках	96
<i>Гурський А. С.</i> Технології розробки програмного продукту для обслуговування абонентів телекомунікаційної компанії	97
<i>Капітан Д.А.</i> Вирішення проблеми розмітки елементів на формі з використанням CSS	99
<i>Капітик О.В.</i> Проектування інформаційної системи для обміну різноматематичним контентом	100
<i>Колесник Д.Ю., Чорний В.В.</i> Сучасні тренди веб-розробки	103
<i>Кулаківська Н.О.</i> Система бізнес-аналітики навчального відділу ЗВО на основі LOW-CODE технологій	104
<i>Летич А.А.</i> Інформаційне супроводження процесу доставки ліків	107
<i>Любачевський В.В., Веремієнко І.О.</i> Технологія визначення рівня викидів CO ₂ при майнінгу Bitcoin	108
<i>Малішевська С.Е.</i> Обґрунтування вибору бази даних за умов проектування інформаційної системи	111
<i>Поліщук О.Р.</i> Обґрунтування вибору бази даних за умов проектування інформаційної системи	113
<i>Романченко В. В.</i> Шляхи комп'ютеризації української сфери охорони здоров'я	114
<i>Рибнікова А.І.</i> Сучасні тенденції діджиталізації сфери інтернет-маркетингу	116
<i>Скоробогата О.М.</i> Розробка програмного продукту для організації біржі інтернет-проектів	118
<i>Тимошенко І.В.</i> Переваги серверного програмування інформаційних систем з використанням Spring Boot	120
<i>Трущенко М.О.</i> Вирішення питання аутентифікації в інформаційних системах з використанням токенів	121
<i>Шевченко Ю.Т.</i> Роль і місце візуального програмування в розробці інформаційних систем	123

Теза на тему: «Шляхи комп'ютеризації української сфери охорони здоров'я»

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ
УКРАЇНИ**

Київський національний економічний
університет імені Вадима Гетьмана
Ярославська Державна Вища Технічно-
Економічна Школа ім. о. Броніслава
Маркевича (Польща)
Академія фінансів та бізнесу Vistula
(Польща)
Університет ім. Св. Кирила і Мефодія в
Велико-Тирново (Болгарія)
Київський національний університет імені
Тараса Шевченка
Національний технічний університет
України «КПІ ім. Ігоря Сікорського»
Харківський національний економічний
університет імені
Семена Кузнеця
Національний аерокосмічний університет
ім. М.Є. Жуковського «ХАІ»
Національний технічний університет
«ХПІ»
Харківський національний університет
імені В. Н. Каразіна
Харківський національний університет
радіоелектроніки
Харківський національний університет
міського господарства імені
О.М. Бекетова
Херсонський національний технічний
університет
Запорізький національний університет
Хмельницький національний університет
Криворізький державний педагогічний
університет
Східноукраїнський національний
університет ім. Володимира Даля
Північний національний університет імені
Івана Франка
Львівський національний технічний
університет «Львівська політехніка»
Кременчуцький національний університет
ім. М. Остроградського



Department of Information Systems in Economics, Kyiv National
named after Vadym Hetman

The State University of Technology and Economics in Jara

ACCESS Journal: Access to Science, Business, Innovation in Digital
(WoS)

Академія фінансів та бізнесу Vistula
(Польща)

Великотирновський університет імені Св. Кирила та Мефодія




ПРОГРАМА
IV МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ
КОНФЕРЕНЦІЇ МОЛОДИХ ВЧЕНИХ, АСПІРАНТІВ І
СТУДЕНТІВ
«СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА
СИСТЕМИ В УПРАВЛІННІ»
(ІНТЕРНЕТ- КОНФЕРЕНЦІЯ)

20-21 квітня 2023 р.

Київ - Ярослав

Шановний колего!

Запрошуємо Вас взяти участь у
IV МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ
КОНФЕРЕНЦІЇ МОЛОДИХ ВЧЕНИХ, АСПІРАНТІВ І
СТУДЕНТІВ
«СУЧАСНІ ІНФОРМАЦІЙНІ ТА СИСТЕМИ В
УПРАВЛІННІ»
(ІНТЕРНЕТ- КОНФЕРЕНЦІЯ)

Конференцію проводить
Київський національний економічний університет
імені Вадима Гетьмана

ПРОГРАМА КОНФЕРЕНЦІЇ

Пленарне засідання
20 квітня 2023
11.00

Голова програмного комітету: Рамазанов С.К. – д.т.н., д.е.н.,
професор, заслужений діяч науки і техніки України
Модератор конференції: Мозгалі О.П., д.е.н., професор кафедри
інформаційних систем в економіці

Голова організаційного комітету: Антонюк Л.Л. – д.е.н., професор,
проректор з наукової роботи Київського національного економічного
університету імені Вадима Гетьмана".

Програма конференції. «Сучасні інформаційні технології та системи в управлінні.»

2. **Романченко В. В.**, магістр

Київський національний економічний університет імені Вадима Гетьмана

СТВОРЕННЯ ЦИФРОВИХ ЗОБРАЖЕНЬ НА ОСНОВІ ОПИСІВ ПРИРОДНОЮ МОВОЮ. DALL-E 2

Теза на тему: «Створення цифрових зображень на основі описів природною мовою. Dall-e 2»