Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ ГЕНЕРУВАННЯ ТЕСТОВИХ НАБОРІВ ДАНИХ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖ GAN
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Шемаєв Д.І.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

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

Допускається до захисту

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

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Шемаєву Денису Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення застосунку для генерування тестових наборів даних за допомогою нейромереж GAN

затверджена наказом університету від 15 травня 2023 року № 474 Ст

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

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі.

4. Перелік питань, що потрібно опрацювати в роботі

1. Генерація зображень за допомогою машинного навчання.
2. Переваги GAN нейромереж та методи їх застосування.
3.Функція втрат GAN.
4. Реалізація програмного застосунку на основі GAN нейромережі.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність генерації зображень, постановка задачі, тестові зображення.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Аналіз технічних засобів	21.04.23-30.04.23	
5	Розробка методу	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	07.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 56 с., 26 рис., 1 дод., 35 джерел.

НЕЙРОМЕРЕЖІ, GENERATIVE ADVERSARIAL NETWORKS, ОБРОБКА ЗОБРАЖЕНЬ, ГЕНЕРАЦІЯ ЗОБРАЖЕНЬ, АНАЛІЗ ДАНИХ.

Об’єктом роботи є генерування тестових наборів даних за допомогою нейромереж GAN.

Метою роботи є вивчення області нейромереж та розробка застосунку, що базуються на використанні функції втрат GAN, які дозволяють навчати систему генерації зображень.

Такі набори даних повинні бути створені з тестових даних і включати наступні критерії: містити достатню кількість даних для подальшої обробки, бути надійними для подальшої обробки в інших системах, генеруватися за допомогою шуму, включати специфічну структуру для більш ефективного навчання.

Деякі згенеровані зображення повинні бути показані користувачеві як приклад кінцевого результату. Такі дані необхідно зберігати в окремих файлах або передати іншим користувачам. У роботі буде використано нейронні мережі GAN для створення зображень.

NEURONETWORKS, GENERATIVE ADVERSARIAL NETWORKS, IMAGE PROCESSING, IMAGE GENERATION, DATA ANALYSIS.

The object of the work is the generation of test data sets using GAN neural networks.

The method of work is the study of the area of neural networks and the development of an application based on the use of GAN loss functions, which can be taught to the image generation system.

Such datasets should be created from test data and include the following criteria: contain a sufficient amount of data for further processing, be reliable for further processing in other systems, generated with noise, include a specific structure for more effective learning.

Some generated images should be shown to the user as the example of final result. Such data should be saved in separate files or transferred to other users. In work will be used GAN neural networks to generate images.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AGI – Artificial General Intelligence (загальний штучний інтелект)

GAN – Generative Adversarial Network (генеративна змагальна мережа)

ШІ – штучний інтелект

AI – Artificial Intelligence (штучний інтелект)

VAE – Variational Autoencoder (варіаційний автокодувальник)

CRM – Extended Relationship Management (розширене управління взаємовідносинами)

BI – Business Intelligence (бізнес-аналітика)

HRIS – Human Resource Management System (система управління людськими ресурсами)

IDE – Integrated Development Environment (інтегроване середовище розробки)

MNIST – Modified National Institute of Standards and Technology (модифікований національний інститут стандартів і технологій)

ZIP – формат архівації файлів та стиснення даних без втрат

ВСТУП

В сучасному світі ми дедалі частіше зустрічаємо нейронні мережі та також починаємо використовувати їх все частіше в повсякденному житті. Тому для навчання треба використовувати дані яких завжди не вистачає для збільшення точності навчених нейронних мереж.

Для створення таких зображень я буду використовувати GAN нейромережу яка і буде допомагати створювати такі дані.

GAN – це тип нейронної мережі, яка може генерувати нові дані з нуля. З невеликим введенням випадкового шуму може генерувати реалістичні зображення спалень, птахів тощо, як система була запрограмована.

GAN можна використовувати для створення нових даних у ситуаціях, коли дані обмежені, і це може виявитися дійсно корисним. Інколи створення даних може бути складним, дорогим і трудомістким. Але щоб нові дані були корисними. Зображення отримані на основі згенерованих даних, мають бути достатньо реалістичними, щоб їх можна було застосувати до реальних даних. Наприклад, якщо використовуєте підроблену мишу, щоб змусити кота полювати на мишей, потрібно переконатися, що підроблена миша дійсно виглядає як миша.

Іншими словами, GAN дозволяють створювати реалістичні дані, відкриваючи структуру даних. Це корисно, коли не можемо побачити цю структуру або не можемо витягнути її за допомогою інших методів.

Актуальність роботи полягає у тому, що дані дуже затребувані і із розвитком електронних застосунків вони стають ще більш необхідними для навчання різних нейромереж.

1 МАШИННЕ НАВЧАННЯ ЙОГО ПРИЗНАЧЕННЯ, ТИПИ ТА ВИКОРИСТАННЯ В ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

1.1 Що таке машинне навчання?

Підгалузь штучного інтелекту (ШІ) та інформатики під назвою машинне навчання фокусується на використанні даних і алгоритмів для моделювання того, як навчаються люди, поступово підвищуючи точність системи.

Тема обробки та зберігання даних значно просунулася за останні кілька десятиліть. Здатність зберігати й аналізувати величезні обсяги інформації у темпі, недосяжному для людського мозку, стала можливою завдяки зростанню потужності комп’ютерних систем і зменшенню їхнього розміру.

Ці технічні розробки створили новаторські товари на основі машинного навчання, які революціонізують наш спосіб життя. Механізм рекомендацій Netflix є одним із таких прикладів продукту. Завдяки алгоритмам машинного навчання Netflix може надати своїм глядачам індивідуальний вибір фільмів і серіалів відповідно до їхніх інтересів. Користувачі можуть зробити це, щоб скоротити пошук відповідного матеріалу та отримати більше задоволення від перегляду.

Безпілотні транспортні засоби є ще однією ілюстрацією інноваційного продукту, побудованого на основі машинного навчання. Ці автомобілі можуть самостійно працювати на дорозі завдяки розпізнаванню образів і обробці даних. Це не тільки робить водіння безпечнішим, але й знижує вартість транспортування та частоту аварій на дорогах.

Наука про дані, що швидко розвивається, включає машинне навчання як ключовий елемент. Алгоритми навчаються з використанням статистичних методів для створення класифікацій або прогнозів і для пошуку важливої інформації в проектах інтелектуального аналізу даних. Рішення, прийняті в результаті цих аналізів, в ідеалі впливають на ключові показники зростання

доходів і підприємств. Науковці даних будуть більш затребуваними, оскільки великі дані продовжуватимуть розвиватися та процвітати. Очікується, що вони допоможуть визначити найбільш відповідні бізнес-проблеми та інформацію, необхідну для їх вирішення. Алгоритми машинного навчання зазвичай створюються з використанням інфраструктур, які прискорюють розробку рішень, таких як TensorFlow і PyTorch.

Загалом, машинне навчання дозволяє розробляти нові товари, які можуть спростити наше життя, забезпечити безпеку та зберегти ресурси.

1.2 Типи машинного навчання

Класичне машинне навчання часто класифікують за тим, як алгоритм вчиться ставати точнішим у своїх прогнозах. Існує чотири основні підходи: навчання під контролем, навчання без контролю, навчання з напівконтролем і навчання з підкріпленням. Тип алгоритму, який оберуть науковці, залежить від того, який тип даних вони хочуть передбачити.

У цьому розділі будуть розглянуті ідеї генеративних моделей, розглянуті парадигми навчання з учителем та без, а також дискримінаційне моделювання проти генеративного.

1.2.1 Навчання з учителем проти навчання без учителя

Типова проблема машинного навчання включає використання моделі для прогнозування, наприклад прогнозує моделювання.

Для цього потрібен навчальний набір даних, який використовується для навчання моделі, що складається з декількох прикладів, званих вибірками, кожна з яких містить змінні вхідні і вихідні мітки класу. Модель навчається на прикладах вхідних даних, роблячи прогнози вихідних даних та

подальшого коригування моделі, щоб зробити вихідні дані більш схожими на очікувані.

Таке моделювання зазвичай називають контрольованою формою навчання чи навчанням із учителем.

1.2.2 Контрольоване навчання

Контрольоване навчання – це тип методу машинного навчання, у якому надаємо зразки позначених даних системі машинного навчання, щоб навчити її, і на цій основі вона прогнозує результат.

Система створює модель, використовуючи дані з мітками, щоб зрозуміти набори даних і дізнатися про кожну з них. Після завершення навчання й обробки перевіряємо модель, надаючи зразки даних, щоб перевірити, чи прогнозує вона точний результат чи ні.

Метою навчання під наглядом є зіставлення вхідних даних із вихідними даними. Контрольоване навчання базується на нагляді, і це те саме, що коли студент навчається під наглядом вчителя. Прикладом навчання під наглядом є фільтрація спаму.

Контрольоване навчання можна далі згрупувати в дві категорії алгоритмів:

- класифікація;
- регресія.

1.2.3 Навчання без контролю

Навчання без нагляду – це метод навчання, при якому машина навчається без будь-якого нагляду.

Навчання надається машині з набором даних, які не були позначені, класифіковані чи категоризовані, і алгоритм повинен діяти з цими даними без будь-якого контролю. Метою неконтрольованого навчання є реструктуризація вхідних даних у нові функції або групу об’єктів зі схожими шаблонами.

У неконтрольованому навчанні не маємо заздалегідь визначеного результату. Машина намагається знайти корисну інформацію з величезної кількості даних. Далі його можна класифікувати на дві категорії алгоритмів:

- кластеризація;
- асоціація.

1.2.4 Навчання з підкріпленням

Навчання з підкріпленням – це метод навчання на основі зворотного зв’язку, за якого навчальний агент отримує винагороду за кожну правильну дію та отримує штраф за кожну неправильну дію. Агент автоматично навчається за допомогою цих відгуків і покращує свою продуктивність. У навчанні з підкріпленням агент взаємодіє з середовищем і досліджує його. Мета агента – отримати якомога більше бонусних балів, і, отже, він покращує свою продуктивність.

Робот-собака, яка автоматично навчається рухам рук, є прикладом навчання з підкріпленням.

1.3 Використання машинного навчання

Сьогодні машинне навчання використовується в широкому діапазоні програм. Мабуть, одним із найвідоміших прикладів машинного навчання в дії є механізм рекомендацій, який підтримує стрічку новин Facebook.

Facebook використовує машинне навчання, щоб персоналізувати спосіб доставки каналу кожного учасника. Якщо учасник часто зупиняється, щоб прочитати дописи певної групи, система рекомендацій почне показувати більше активності цієї групи раніше в стрічці.

За лаштунками механізм намагається посилити відомі шаблони поведінки учасників в Інтернеті. Якщо учасник змінить шаблони та не зможе прочитати дописи з цієї групи протягом наступних тижнів, стрічка новин буде відповідним чином скоригована.

Крім механізмів рекомендацій, машинне навчання можна використовувати в наступному:

– управління взаємовідносинами з клієнтами. Програмне забезпечення CRM може використовувати моделі машинного навчання для аналізу електронної пошти та спонукати членів відділу продажів першими відповідати на найважливіші повідомлення. Досконаліші системи можуть навіть рекомендувати потенційно ефективні відповіді;

– бізнес-аналітика. Постачальники BI та аналітики використовують машинне навчання у своєму програмному забезпеченні для виявлення потенційно важливих точок даних, шаблонів точок даних і аномалій.

– інформаційні системи кадрового забезпечення. Системи HRIS можуть використовувати моделі машинного навчання для фільтрації програм і визначення найкращих кандидатів на відкриту вакансію;

– безпілотні автомобілі. Алгоритми машинного навчання можуть навіть дозволити напівавтономному автомобілю розпізнати частково видимий об’єкт і попередити водія;

– віртуальні помічники. Розумні помічники зазвичай поєднують контрольовані та неконтрольовані моделі машинного навчання для інтерпретації природного мовлення та надання контексту.

1.3.1 Життєвий цикл машинного навчання

Машинне навчання наділило комп'ютерні системи здатністю автоматично навчатися без явного програмування. Але як працює система машинного навчання? Отже, це можна описати за допомогою життєвого циклу машинного навчання. Життєвий цикл машинного навчання – це циклічний процес створення ефективного проекту машинного навчання. Основною метою життєвого циклу є пошук рішення проблеми або проекту.

Життєвий цикл машинного навчання включає сім основних кроків, які наведено нижче:

- збір даних;
- підготовка даних;
- конфлікт даних;
- аналіз даних;
- тренування моделі;
- випробування моделі;
- розгортання.

Розуміння проблеми та усвідомлення її мети є двома найважливішими складовими всієї процедури. Оскільки позитивний результат залежить від глибшого розуміння проблеми, повинне бути повне її розуміння, перед початком життєвого циклу.

Щоб вирішити проблему, створюється система машинного навчання під назвою «модель» як частина всього процесу життєвого циклу. Ця модель

побудована шляхом «навчання». Однак, щоб навчити модель, потрібні дані, тому життєвий цикл починається зі збору даних.

1.3.2 Збір даних

Початковим етапом машинного навчання є збір даних. Метою цього кроку є виявлення та збір усіх проблем, пов’язаних із даними.

Оскільки дані можуть бути зібрані з різних джерел, включаючи файли, бази даних, Інтернет і мобільні пристрої. Для цього спочатку повинні визначити різні джерела даних. Це одна з найважливіших фаз життєвого циклу. Ефективність результату залежатиме від кількості та калібру зібраних даних. Прогноз буде точнішим, чим більше даних буде.

Цей крок включає такі завдання:

- визначте різні джерела даних;
- зберіть дані;
- інтегруйте дані, отримані з різних джерел.

Виконуючи вищезазначене завдання, отримаємо узгоджений набір даних, який також називають набором даних. Він буде використовуватися на подальших етапах.

1.3.3 Підготовка даних

На цьому єтапі повинні підготувати дані для майбутнього використання після їх збору. Підготовка даних – це процес організації та підготовки наших даних для використання в навчанні машинному навчанню.

На цьому етапі спочатку групуємо всі дані, а потім упорядковуємо їх випадковим чином.

Цей крок можна далі розділити на два процеси:

– дослідження даних. Це допомагає зрозуміти тип даних, з якими повинні працювати. Для цього маємо розуміти якість, формати та властивості даних. Більш точне розуміння даних призводить до успішних результатів. В цьому виявляємо кореляції, широкі закономірності та викиди;

– попередня обробка даних. Наступним кроком є попередня обробка даних для їх аналізу.

1.3.4 Суперечка даних

Очищення та перетворення непридатних необроблених даних у придатний для використання формат відоме як суперечка даних. Це процес підготовки даних для аналізу на наступному етапі шляхом правильного їх форматування, вибору змінної для використання та очищення даних. Це один із найважливіших етапів усієї процедури. Щоб вирішити проблеми з якістю, необхідне очищення даних.

Інформація, яка була зібрана, не завжди може бути для корисною; деякі з них можуть навіть не бути. У реальних програмах зібрані дані можуть мати різні проблеми, зокрема:

– відсутні значення;

– дубльовані дані;

– недійсні дані;

– шум.

Тому використовуємо різні методи фільтрації для очищення даних.

Виявлення та усунення вищезазначених проблем є обов’язковим, оскільки це може негативно вплинути на якість результату.

1.3.5 Аналіз даних

Тепер очищені та підготовлені дані передаються на етап аналізу. Цей крок передбачає:

- вибір аналітичних методик;
- побудова моделей;
- перегляньте результат.

Мета цього етапу – створити модель машинного навчання, яка досліджуватиме дані різними аналітичними методами, а потім оцінюватиме результати. Щоб розробити модель з використанням підготовлених даних, спочатку визначаємо тип проблем. Потім обираємо такі методи машинного навчання, як класифікація, регресія, кластерний аналіз, асоціація тощо, і оцінюємо модель.

У результаті на цьому етапі беремо дані та створюємо модель за допомогою методів машинного навчання.

1.3.6 Модель поїзда

Тепер модель має бути навчена, щоб покращити її для кращого вирішення проблеми на наступному етапі.

Щоб навчити модель за допомогою різних методів машинного навчання, використовуємо набори даних. Модель необхідно навчити, щоб вона могла осягнути численні закономірності, закони та характеристики.

1.3.7 Тестова модель

Тестуємо модель машинного навчання після того, як її навчили на певному наборі даних. На цьому етапі надаємо нашій моделі тестовий набір даних, щоб перевірити, чи він точний.

Відповідно до потреб проекту чи завдання тестування моделі визначає її відсоток точності.

1.3.8 Розгортання

Розгортання, останній етап життєвого циклу машинного навчання, включає впровадження моделі в практичну систему.

Впроваджуємо модель у фактичну систему, якщо вона забезпечує точний результат, який відповідає нашим вимогам швидко та за планом. Однак спочатку визначимо, чи проект використовує надані дані для покращення продуктивності, перш ніж розповсюджувати їх. Остаточний звіт проекту складається на етапі розгортання.

1.4 Генерація зображень за допомогою машинного навчання

Алгоритми та моделі штучного інтелекту використовуються для створення реалістичних або творчих фотографій з нуля або на основі вже наявних фотографій. Цей метод використовує такі методи глибокого навчання, як варіаційні автокодери (VAE) і генеративні суперницькі мережі (GAN), щоб створювати естетично привабливі та узгоджені зображення, подібні до вмісту та представлення вхідних даних.

Перейдемо к огляду процесу створення зображення на основі штучного інтелекту.

Збір і підготовка даних: щоб забезпечити модель штучного інтелекту навчальними даними, збирається та готується чималий набір даних фотографій. Залежно від бажаного результату, набір даних може містити різноманітні зображення, включаючи фотографії, картини чи малюнки. На рисунку 1.1 наведено приклади вхідних даних для навчання штучного інтелекту.

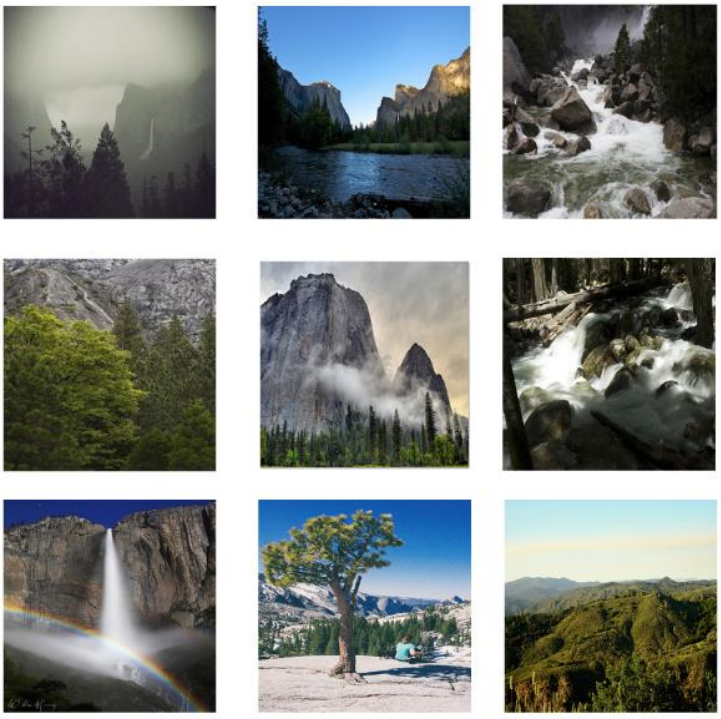

Рисунок 1.1 – Приклад вхідних даних для навчання штучного інтелекту

Навчання моделі: зібраний набір даних використовується для навчання моделі AI, якою зазвичай є GAN або VAE. Під час навчання модель отримує здатність розпізнавати статистичні закономірності та властивості вхідних фотографій. На відміну від VAE, які прагнуть навчитися компактному представленню даних, GAN мають мережу генератора та мережу дискримінатора, які конкурують одна з одною.

Прихований простір і кодування: щодо VAE, модель вчиться кодувати вхідні зображення в цю концепцію прихованого простору, який є представленням нижчого виміру. Без точних даних зображення цей

прихований простір є стиснутою версією вхідних фотографій, яка фіксує ключові характеристики.

Після навчання модель штучного інтелекту може створювати нові зображення шляхом вибірки з прихованого простору або шляхом введення випадкового шуму. За допомогою GAN мережа генератора створює нову картину з використанням попередньо вивчених моделей і характеристик. Зазвичай спочатку випадкові, створені візуальні елементи згодом покращуються протягом ітерацій. На рисунку 1.2 наведено приклад вихідних даних після навчання системи перетворення зображень з літа на зиму або навпаки.

Рисунок 1.2 – Приклад вихідних зображень після навчання

Передача стилю та умовна генерація: моделі AI також здатні передавати стиль, тобто створювати зображення, які поєднують стиль одного зображення з вмістом іншого. Крім того, умовне створення дозволяє користувачам змінювати певні характеристики або функції створених зображень, наприклад фон, об’єкти або кольори.

1.5 Приклади різних варіантів використання нейромереж для генерації зображень

Нейронні мережі, що генерують зображення, є потужною технологією, яка пропонує широкий спектр можливостей у сфері обробки зображень. Вони

використовуються для багатьох різних речей, включаючи створення обличчя, рестайлінг зображення, медичну діагностику, старіння обличчя, перетворення чорно-білого на кольорове, покращення зображення та переміщення людини на зображенні. Розглянемо кожну з цих особливостей докладніше.

Створюйте свіжі, реалістичні обличчя за допомогою нейронних мереж, яких навчили цьому робити. Вони досліджують атрибути та візуальні характеристики великої кількості даних про обличчя, перш ніж створювати оригінальні обличчя, яких не існує в реальному світі. Це може використовуватися для покращення анонімності суб'єктів у дослідженнях, у бізнесі відеоігор і кіноіндустрії. На рисунку 1.3 зображення показує повністю згенеровані обличчя, які досить важко відрізнити від реальних фотографій людей.

Рисунок 1.3 – Приклад згенерованих обличь за допомогою GAN

Зміна стилю зображення: змінюючи свій стиль відповідно до попередньо визначеного шаблону, нейронні мережі можна навчити модифікувати зображення. Наприклад, вони можуть надати малюнку креативного відчуття, наслідуючи стиль відомих художників, або перетворити знімок на стилізований ескіз. Це дозволяє створювати характерні, приголомшливі візуальні ефекти. На рисунку 1.4 зображено перетворення картини з одного стилю в інший.

Рисунок 1.4 – Приклад перетворення картини в іншому стилі

Медичні фотографії, особливо рентгенівські, можна аналізувати за допомогою нейронних мереж. Вони можуть виявити симптоми хвороби чи патології, допомагаючи клініцистам поставити точний діагноз. Цей метод забезпечує більш швидку та надійну інтерпретацію знімків, що спрощує роботу медичного персоналу та підвищує точність діагностики.

На рисунку 1.5 показано повністю згенеровані зображення рентгенівських знімків з різними захворюваннями.

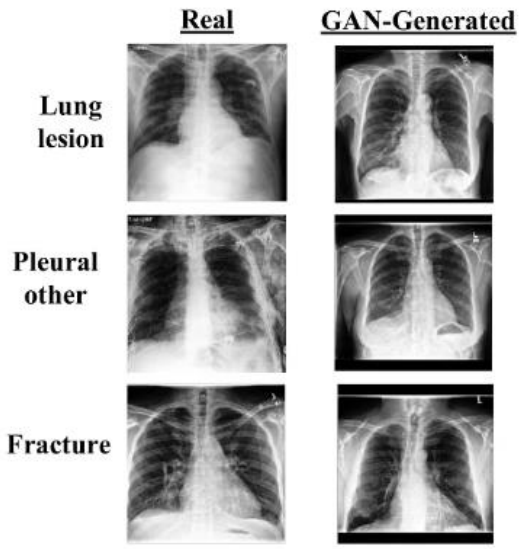

Рисунок 1.5 – Приклад генерації медичних знімків

Покращення зображення. Зменшуючи шум і відновлюючи деталі, нейронні мережі можуть покращити якість зображень. За допомогою своєї

навченої моделі вони можуть заповнити прогалини та відновити інформацію після аналізу зображення та виявлення шуму та втрати якості. Це може бути корисним під час відновлення пошкоджених або неякісних фотографій. На рисунку 1.6 показано зменшення шуму на фотографії та покращення її якості за допомогою GAN нейромереж.

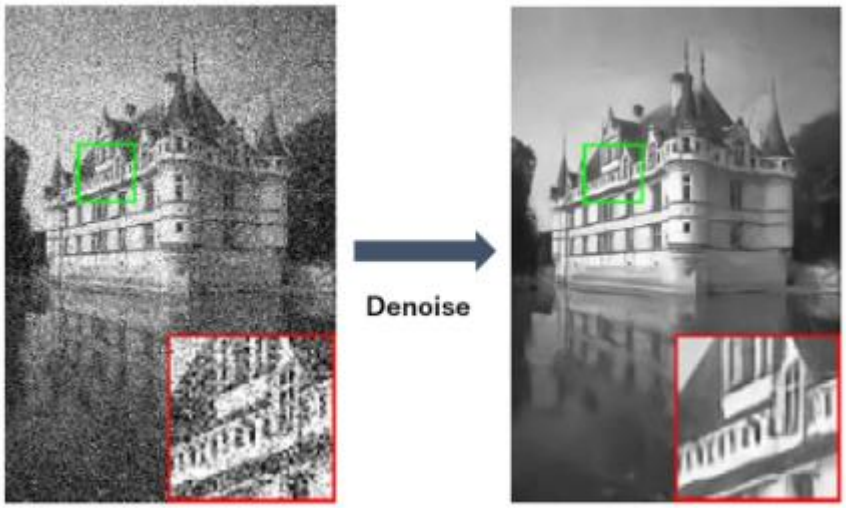

Рисунок 1.6 – Приклад зменшення шуму картинки

Зміна пози людини на фотографії. Нейронні мережі можна використовувати для зміни пози людини на фотографії. Вони можуть змінювати позицію або місце розташування людей, а також інших речей на фотографії. Це дозволяє легко змінювати композицію фотографії або переміщати певні елементи, щоб отримати бажане естетичне враження. На рисунку 1.7 показано зміну пози людини за допомогою обробки зображень нейромережою.

У сфері обробки зображень є кілька варіантів використання нейронних мереж для створення вищезгаданих типів зображень. Ці інновації збільшують нашу творчу свободу, підтримують медичні дослідження та пропонують зображення вищої якості, які допомагають у багатьох аспектах повсякденного життя.

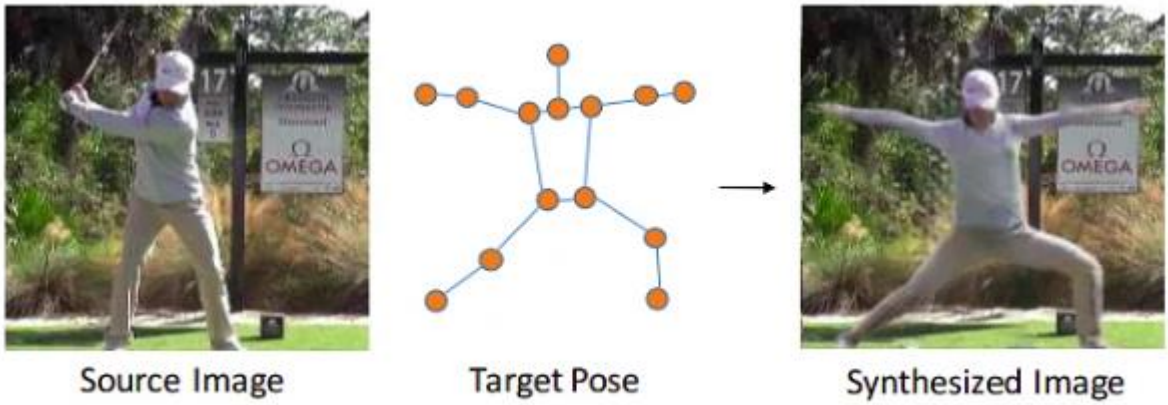

Рисунок 1.7 – Приклад зміни пози людини за допомогою GAN

1.6 Майбутнє машинного навчання

З огляду на те, що машинне навчання є сферою, що постійно розвивається, на яку впливають численні фактори, складно спрогнозувати її точне майбутнє. Однак машинне навчання, швидше за все, і надалі залишатиметься головною силою в багатьох галузях науки, техніки та суспільства, а також суттєвим внеском у технологічний прогрес. Створення інтелектуальних помічників, персоналізованої медичної допомоги та безпілотних автомобілів є одними з потенційних майбутніх застосувань машинного навчання. Такі важливі глобальні проблеми, як бідність і зміна клімату, можна вирішити за допомогою машинного навчання.

Ймовірно, машинне навчання розвиватиметься й покращуватиметься в майбутньому. Щоб зробити машинне навчання ще більш потужним і корисним, дослідники наполегливо працюватимуть над створенням нових алгоритмів і методів.

Створення загального штучного інтелекту (AGI) є однією з основних тем поточного дослідження в цій галузі. Розробка систем зі здатністю навчатися та виконувати різноманітні дії на рівні інтелекту, порівнянному з людським, називається AGI. Створення нових методів і алгоритмів, здатних

до абстрактного мислення, узагальнення та самонавчання, є необхідним для розвитку AGI.

Підвищення рівня автоматизації в ряді галузей, включаючи медицину, банківську справу, транспорт та багато інших, може бути результатом прогресу в машинному навчанні. Машинне навчання може стати основою для створення нових, передових технологій, які підвищать якість життя людей і сприятимуть розвитку суспільства в цілому.

Зрозуміло, що сфера продовжить розширюватися в майбутньому і може навіть досягти нових рівнів інтелектуальних досягнень, які на даний момент здаються недосяжними, враховуючи швидкі темпи технологічного прогресу та постійний інтерес дослідників до вивчення машинного навчання.

1.7 Постановка задачі

Таким чином, однією з критичних проблем у сфері обробки та створення зображень є розробка штучних даних. Це важливо, оскільки для ефективного навчання моделі машинного навчання потрібні великі, високоякісні та різноманітні дані. Для створення імітованих зображень за допомогою мереж глибокого навчання, зокрема мереж, що містять генеративні змагальні мережі (GAN), необхідно розробити алгоритм.

Мета створення техніки генерації зображень на основі мережі GAN полягає в тому, щоб навчити генератор створювати реалістичні зображення, ідентичні зображенням із вихідного набору даних. Для цієї процедури потрібно багато тренувальних фотографій.

Об’єктом роботи є генерування тестових наборів даних за допомогою нейромереж GAN.

Метою роботи є вивчення області нейромереж та розробка застосунку, що базуються на використанні функції втрат GAN, які дозволяють навчати систему генерації зображень.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів генерації зображень;

– розробити алгоритм генерації зображень на базі методу Кенні;

– реалізувати алгоритм знаходження ліній і кіл на базі методу функції втрат GAN;

– реалізувати комп’ютерний застосунок для генерації зображень за допомогою датасетів [1 – 12].

2 GAN НЕЙРОМЕРЕЖІ

2.1 Складові GAN нейромереж

Генеративні змагальні мережі, що породжують (англ. Generative Adversarial Nets, GAN) – алгоритм машинного навчання, що входить у сімейство породжуючих моделей і побудований на комбінації з двох нейронних мереж: генеративна модель G, яка будує наближення розподілу даних, і дискримінативна модель D, що оцінює ймовірність, що зразок прийшов з тренувальних даних, а не згенерованих моделлю G (рис. 2.1).

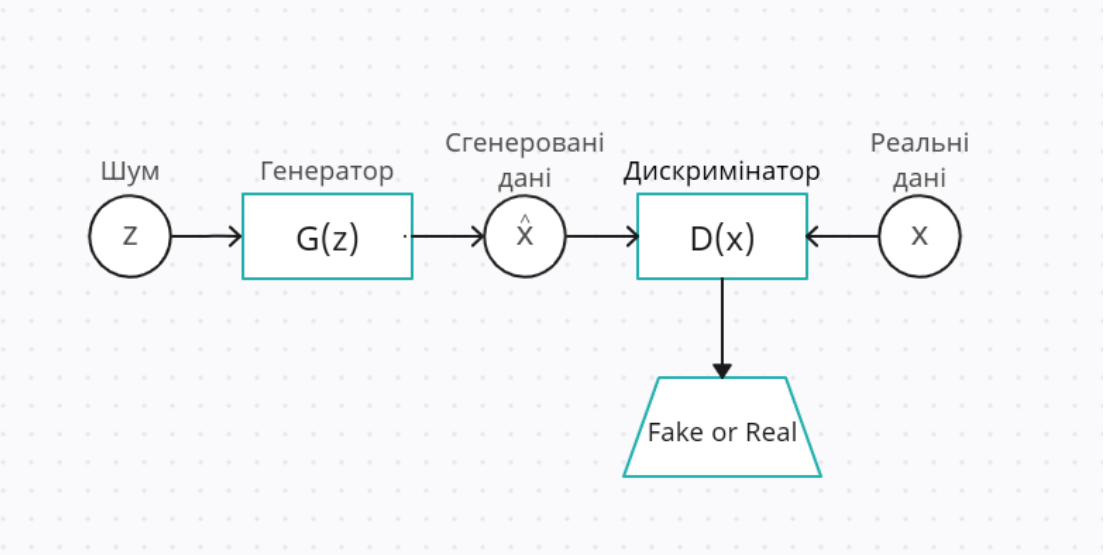

Рисунок 2.1 – Схема роботи алгоритму

Генеративне моделювання – це неконтрольоване навчальне завдання в машинному навчанні, яке передбачає автоматичне виявлення та вивчення закономірностей або закономірностей у вхідних даних таким чином, щоб модель могла використовуватися для генерування або виведення нових прикладів, які ймовірно могли бути взяті з вихідного набору даних.

Генеративні змагальні мережі можна розділити на два види. Один – традиційні GAN, а інший – GAN де навчання здійснюється офіційними дискримінаторами.

На рисунку 2.2 показано, що основний GAN сам по собі є контрольованим методом. Також CGAN (Conditional Generative Adversarial Network), AAGAN (Attention-to-Attention Generative Adversarial Network), GRAN (Graph Neural Network), і EvoGAN (Evolutionary Algorithm based Generative Adversarial Network) контролювані GAN. DCGAN (Deep Convolutional Generative Adversarial Network), LAPGAN (Laplacian Generative Adversarial Network), ACGAN (Auxiliary Classifier Generative Adversarial Network), InfoGAN (Information Maximizing Generative Adversarial Network) і CycleGAN (Cycle-Consistent Adversarial Networks) є прикладами неконтрольованих генеративних змагальних мереж. Серед комбінованих Generative Adversarial Networks, AAE (Adversarial Autoencoder) може бути контрольованим, неконтрольованим, а також напівконтрольованим. Як BiGAN (Bidirectional Generative Adversarial Networks), так і DVDGAN (Dual video discriminator Generative Adversarial Networks) можна контролювати і без нагляду. Інший комбінований GAN, SRGAN (Super Resolution Generative Adversarial Networks) є безконтрольним і частково під наглядом.

GAN – це розумний спосіб навчання генеративної моделі, створюючи проблему як проблему навчання під наглядом за допомогою двох підмоделей: моделі генератора, яку навчаємо генерувати нові приклади, і моделі дискримінатора, яка намагається класифікувати приклади як реальні (від домен) або підроблені (створені). Дві моделі тренуються разом у змагальній грі з нульовою сумою, доки модель дискримінатора не буде обманута приблизно в половині випадків, тобто модель генератора генерує правдоподібні приклади.

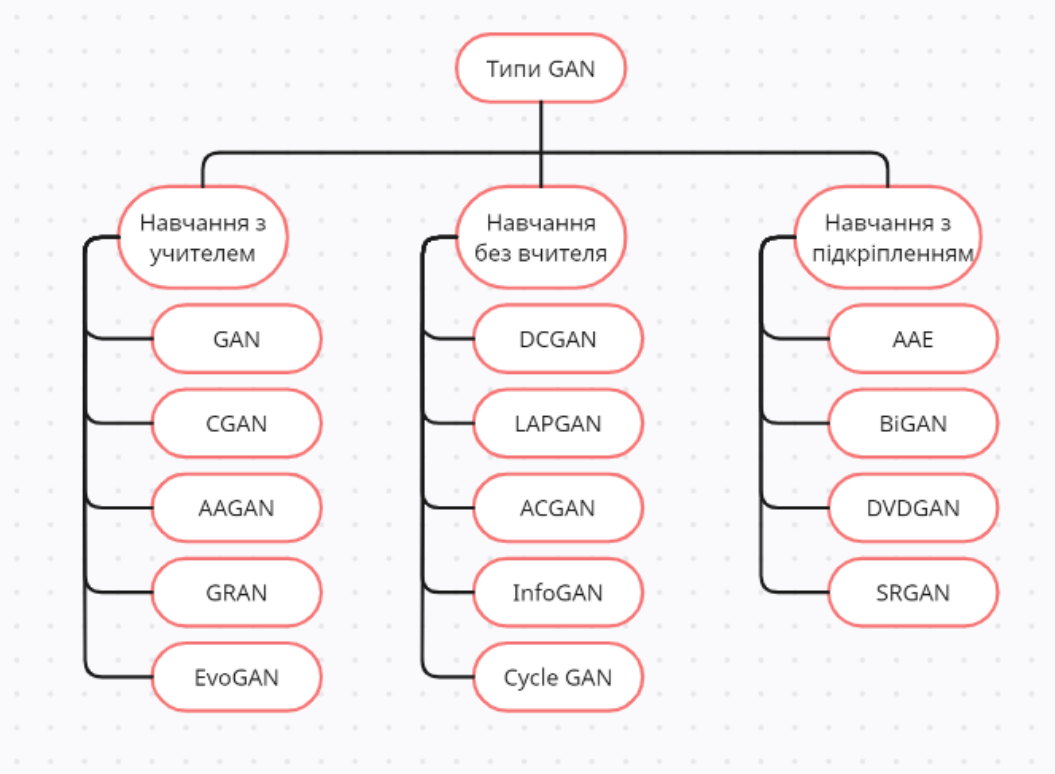

Рисунок 2.2 – Класифікація GAN

Хоча генеративна змагальна мережа – це стара ідея, що походить з теорії ігор, вони були введені в ком'юніті машинного навчання у 2014 році Єном Гудфеллоу та співавторами у статті «Генеративні змагальні мережі».

Після навчання нейронні мережі досить добре розпізнають голоси, зображення та об'єкти в кожному кадрі відео, навіть коли ви відтворюєте відео. Скажімо, ви не можете купити дорогу картину відомого художника; чи можете ви створити/згенерувати штучного художника, який може малювати, як будь-який відомий художник, вивчаючи його/її минулі колекції?

Відповідь – так – за допомогою генеративних змагальних мереж (GAN) ви можете. Генеративні змагальні мережі (GAN) – це клас алгоритмів, які використовуються в неконтрольованому навчанні. Вам не потрібні мітки для вашого набору даних, щоб навчити GAN.

Візьмемо приклад художника, щоб зрозуміти GAN. Інтуїтивно зрозуміла ідея GAN полягає в тому, що існує глибока нейронна мережа експерта та глибока нейронна мережа учня. Потім змушуємо їх воювати один проти

одного, нескінченно намагаючись перевершити один одного. У процесі вони обоє стають сильнішими.

Експерт володіє інформацією про оригінал картини. Учень навчається за кілька проходів, де він генерує нейронну мережу та передає свої результати експерту для перевірки правильності. Робота експерта полягає в тому, щоб визначити справжність і підробку. Оскільки процес триває, експерт і учень навчаються по черзі. Отже, експерт і учень – вони залежать один від одного для ефективного навчання. Якщо один з них виходить з ладу, вся система виходить з ладу.

Архітектура GAN складається з наступних компонентів:

– навчальний набір даних: зразок реальних даних, яким навчаємо нашу мережу Discriminator розрізняти справжні та підроблені дані. Тип даних, які ви хочете створити, залежить від вашого навчального набору даних;

– випадковий шум: він діє як початкова точка (необроблений вхід) для нашої мережі генератора. Цей шум перетворюється на підроблені дані за допомогою генератора;

– генеруюча мережа: це нейронна мережа, яка приймає випадковий шум як вхідні дані та створює фальшиві дані як вихідні дані. Його мета – генерувати дані, які мережа Discriminator не може виявити як підроблені.

– мережа дискримінатора: це також нейронна мережа, яка приймає дані з навчального набору даних і підроблені дані як вхідні дані та класифікує їх як справжні або підроблені;

– ітераційне навчання: оскільки GAN складається з двох нейронних мереж, фаза навчання одночасно навчає генератор і дискримінатор.

2.2 Функція втрат GAN

Математична функція, відома як функція втрат GAN (Generative Adversarial Network), є важливою для процесу навчання та покращення здатності моделі GAN створювати високоякісні дані.

GAN має генератор і дискримінатор як дві основні частини. Генератор відповідає за створення свіжих зразків, які максимально схожі на фактичні дані. З іншого боку, дискримінатор відіграє роль класифікатора та намагається відрізнити зразки, створені генератором, від фактичних зразків із навчального набору даних.

Функція втрат GAN є мірою здатності дискримінатора розрізняти справжні та синтетичні зразки, а також здатності генератора «обдурити» дискримінатор. Мета полягає в тому, щоб досягти балансу між двома елементами таким чином, щоб генератор виробляв дані, які неможливо відрізнити від справжніх зразків, а дискримінатор не міг послідовно розрізняти реальні та створені зразки.

Ключовим кроком у створенні моделей GAN є вибір найкращої функції втрат. Залежно від конкретних вимог проблеми та характеристик моделі можна використовувати багато типів функцій втрат, наприклад бінарну перехресну ентропію або втрату Вассерштейна.

Загалом, взаємодія між генератором і дискримінатором під час навчання визначається функцією втрат GAN, що гарантує збалансований і ефективний метод отримання нових даних.

Для більш зрозумілого сприйняття буде наведена формула 2.1

$$V(D,G) = E_{x\sim Pdata(x)}[logD(x)] + E_{z\sim Pz(z)}[\log\Big(1 - D\big(G(z)\big)\Big)], \quad (2.1)$$

де *G* – генератор;

D – дискримінатор;

Pdata(x) – розподіл реальних даних;

P(z) – розподіл генератора;

x – вибірка з *Pdata(x)*;

z – вибірка з *P(z)*;

D(x) – мережа дискримінатора;

G(z) – генераторна мережа.

GAN сформульовано як мінімаксну гру, де Дискримінатор намагається мінімізувати свою винагороду *V(D, G)*, а Генератор намагається мінімізувати винагороду Дискримінатора або, іншими словами, максимізувати його втрати.

2.3 Проблеми тренування

Підготовка та нестабільність роботи генеративно-конкурентних мереж добре відомі. Одна проблема полягає в тому, що вибірка зображень генератора буде обмежена лише тими, які дозволять йому обдурити дискримінатор, якщо його швидкість навчання надто швидша, ніж у дискримінатора. Насправді генератор зводиться до створення єдиної універсальної картинки, щоб обдурити дискримінатор в результаті навчання. Ця проблема називається «Режим згортання».

Дискримінатор працює з відносно невеликим градієнтом, коли генератор фактично обманює, тому він не має достатньо даних, щоб знайти правильне рішення, яке покращить реалістичність зображення.

Зусилля дослідників для вирішення цих проблем були зосереджені на зміні структури функції втрат. Пряме коригування, запропоноване Xudong Мао замінити функцію втрат парою функцій на основі найменших квадратів. Застосовуючи незатухаючі градієнти, це забезпечує стабільність тренувального процесу, кращу якість зображення та зниження ризику колапсу.

Труднощі з отриманням фотографій високої роздільної здатності також були проблемою для дослідників, частково тому, що більш детальне зображення надає дискримінаторам більше інформації для ідентифікації фальшивих зображень. Сучасні генеративні змагальні мережі починають із зображень із низькою роздільною здатністю для навчання мережі та поступово додають додаткові шари, доки не буде досягнуто цільового розміру зображення.

Стабільність усього процесу, а також швидкість і якість кінцевого зображення значно підвищуються, коли шари з більш високою роздільною здатністю поступово додаються під час навчання генеративно-конкурентних мереж [13 – 21].

3 КОМП'ЮТЕРНА МОДЕЛЬ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був розроблений алгоритм для фільтрації зображень за допомогою високочастотних фільтрів. Для реалізації було обране середовище PyCharm і мови програмування Python для розробки програмного додатку з метою створення зображень, що може бути використано з різними перевагами. Мова програмування Python має власне інтегроване середовище розробки (IDE), PyCharm, яке було створено з метою роботи саме з мовою Python. PyCharm пропонує різноманітні функції, включаючи автозаповнення коду, налагодження, візуальне керування версіями, аналіз коду тощо. Ці функції PyCharm роблять роботу з проектами Python простою та ефективною, що робить його оптимальним варіантом для створення програмного забезпечення, зосередженого на створенні зображень.

Python є популярною та потужною мовою програмування, яка має широкий вибір бібліотек та фреймворків. Приклади передових інструментів для впровадження складних алгоритмів машинного навчання, обробки зображень і генеративного моделювання включають TensorFlow, PyTorch, Keras і OpenCV. Більшість із цих бібліотек мають обширну документацію та процвітаючу спільноту розробників, що допомагає зробити ресурси, пакети та підтримку більш доступними. Що дозволяє використати їх в реалізації програмного застосунку для покращення ефективності його роботи.

Ще одна перевага використання Python полягає в тому, що він не залежить від платформи, тобто програмне забезпечення, створене з його використанням, може працювати в інших операційних системах, таких як Windows, macOS або Linux, без серйозних налаштувань.

Враховуючи ці переваги, використання PyCharm і мови програмування Python для створення програмного застосунку що буде генерувати зображення виглядає виправданим (рис. 3.1).

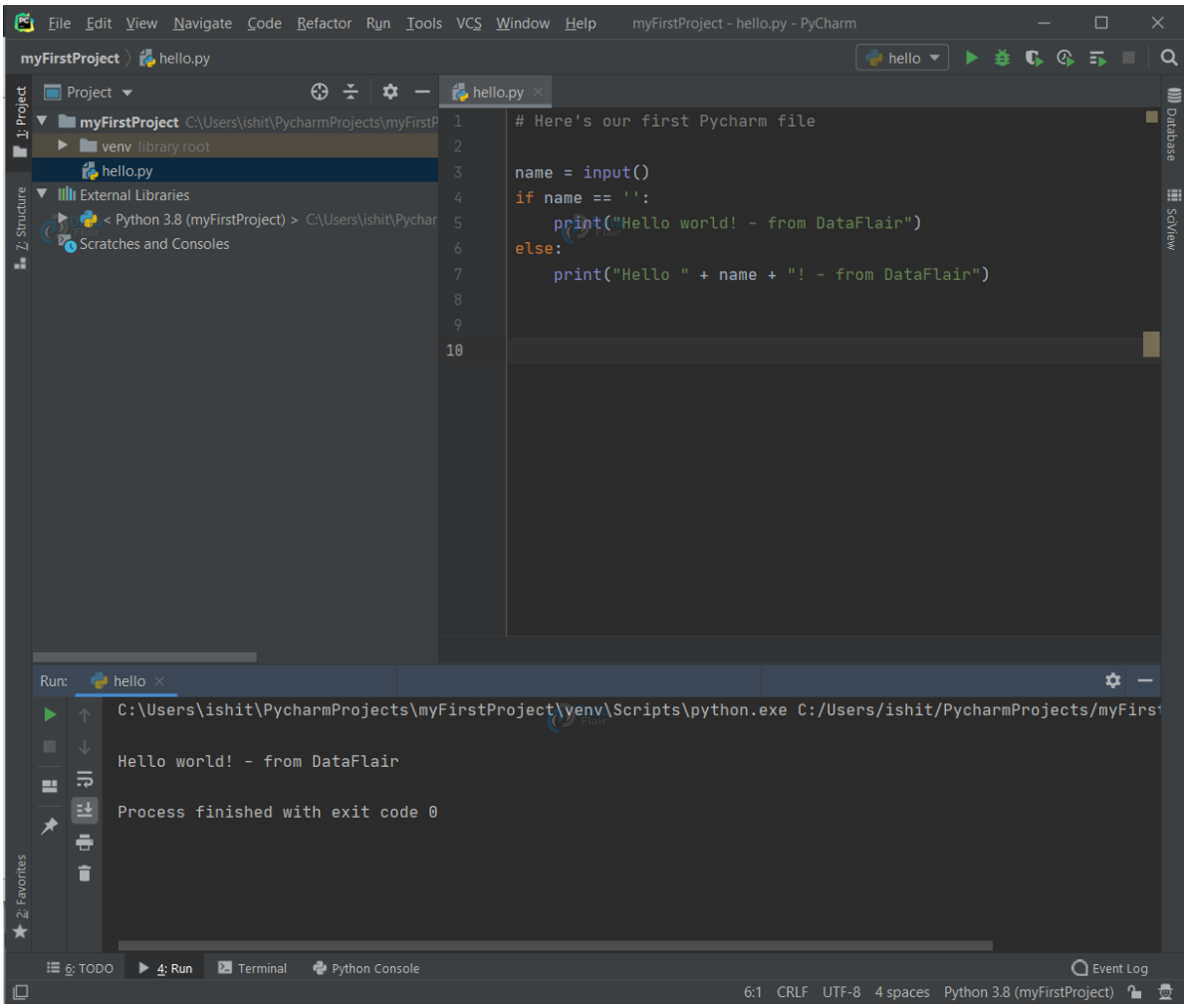

Рисунок 3.1 – Приклад інтерфейсу PyCharm

Для реалізації будемо використовувати PyCharm, язик програмування Python та такі бібліотеки як:

– numpy;
– keras;
– pycharm;
– math;
– tensorflow та інші.

Інтегроване середовище розробки (IDE) мови програмування Python, PyCharm, пропонує широкі можливості для створення програм для наукових досліджень зображень.

Зручний і простий інтерфейс PyCharm, який сприяє ефективності програміста, є однією з його основних переваг. На додаток до автоматичної підказки, перевірки граматики та допомоги в рефакторингу, він пропонує можливість створювати, змінювати та налагоджувати код Python.

Крім того, PyCharm пропонує інтеграцію з широким набором практичних інструментів, які спрощують створення програм для створення зображень. Він забезпечує інтеграцію з такими фреймворками глибокого навчання, як TensorFlow або PyTorch, наприклад, що полегшує створення та навчання моделей генерації зображень.

Крім того, PyCharm підтримує Git, систему контролю версій, яка спрощує підтримку та моніторинг версій програми. Це особливо корисно в наукових дослідженнях, оскільки важливо документувати зміни коду та працювати разом з іншими дослідниками.

Крім того, PyCharm пропонує налагодження коду, що полегшує аналіз і відстеження того, як виконуються програми. Це надзвичайно важливо для створення генеративних моделей, оскільки дозволяє виявляти та виправляти проблеми кодування.

Загалом PyCharm є ефективним інструментом дослідження зображень, який спрощує написання та налагодження коду, інтегрується з бібліотеками глибокого навчання та має систему контролю версій, яка допомагає у створенні програмного забезпечення для обробки зображень.

3.2 Бібліотеки використані в програмному застосунку

3.2.1 Numpy

Бібліотека Numpy має вирішальне значення для обробки та модифікації зображень у виробництва зображень. Можливості масиву Numpy дозволяють використовувати числові дані для створення, зміни та аналізу зображень.

Розробка масивів, які представляють піксель зображення, є одним із важливих елементів виробництва зображення. Ширину та висоту зображення можна зменшити за допомогою перших двох осей тривимірного масиву в Numpy, а значення кольорів (наприклад, червоний, зелений і синій) можуть міститися на третій осі.

У Numpy для створення зображень доступні численні методи, такі як додавання шуму, створення градієнтів, розміщення графічних форм і заповнення масиву випадковими значеннями. Потім цей масив можна використовувати як основу для створення шуму або абстрактної графіки.

Numpy також пропонує практичні інструменти для маніпулювання зображеннями. Зображення може змінюватися або трансформуватися в результаті дій різних команд, залежно від потреб дослідження.

При розширенні можливостей numpy для створення зображень часто використовуються інші спеціальні бібліотеки, такі як OpenCV або PIL (Python Image Library). Ці бібліотеки дуже допомагають вченим, які працюють із створенням зображень, оскільки вони надають функціональність для завантаження, зберігання та аналізу зображень, які доповнюють numpy.

З точки зору наукових досліджень створення зображень, numpy – це програма, яка пропонує практичні функції для синтезу, редагування й аналізу зображень. При використанні в поєднанні з іншими спеціалізованими бібліотеками він пропонує потужні можливості для вирішення проблем, пов’язаних з обробкою зображень.

3.2.2 Keras

Keras – це інтерфейс високого рівня для створення та навчання нейронних мереж. Побудова та навчання глибоких нейронних мереж зі здатністю створювати нові зображення на основі навчальних даних здійснюється за допомогою Keras у контексті досліджень створення зображень.

Простота використання та універсальність Keras є двома його видатними якостями. Завдяки з'єднанню кількох рівнів (таких як згортковий, субвідповідальний і вихідний рівні) він пропонує практичний інтерфейс для створення складних моделей нейронної мережі. Крім того, Keras пропонує варіанти налаштування параметрів моделі, включаючи кількість шарів, розмір фільтра, швидкість навчання тощо.

Найпопулярнішим використанням Keras у контексті виробництва зображень є побудова генеративних суперницьких мереж (GAN). Генератор і дискримінатор є двома основними частинами GAN. Дискримінатор прагне відокремити створені зображення від справжніх, тоді як генератор створює нові зображення, використовуючи випадковий шум як вхідний сигнал. Здатність генератора генерувати зображення посилюється цією конкурентною взаємодією між ним і дискримінатором.

Keras має вбудовані можливості для створення та вдосконалення GAN. Зображення можна створювати та обробляти за допомогою згорткових шарів, а функції втрат (такі як двійкова крос-ентропія або середньоквадратична помилка) можна використовувати для оцінки ефективності результатів.

Як відправну точку для створення зображень, Keras також дозволяє використовувати попередньо навчені моделі (наприклад, моделі VGG або ResNet). Щоб підвищити якість створюваних зображень, він дозволяє використовувати найсучасніші архітектури та зважені навчені параметри.

Усі ці фактори разом роблять Keras потужним інструментом для наукових досліджень зображень, що дозволяє вченим швидко створювати та

навчати складні глибокі нейронні мережі для створення свіжих високоякісних зображень.

3.2.3 Tensorflow

TensorFlow – це відкрите середовище розробки та навчання для моделей глибокого навчання. TensorFlow використовується для створення та навчання генеративних моделей, таких як автокодери або генеративні суперницькі мережі (GAN) у контексті наукових досліджень виробництва зображень.

Парадигма обчислення графів TensorFlow є однією з його основних функцій. Обчислювальна модель TensorFlow побудована на побудові обчислювального графа, у якому вузли замінюють операції, а ребра – дані, які проходять через ці процеси. Ця техніка забезпечує ефективні обчислення, що є вирішальним для створення зображень, особливо для великих наборів даних.

У TensorFlow доступні численні вбудовані функції для створення генеративних моделей. Для проектування та навчання моделей генерування зображень він пропонує широкий спектр шарів (таких як згортковий, субвідповідальний і вихідний), функцій активації та функцій втрати.

Функції розподіленого навчання TensorFlow є одними з його найефективніших інструментів. TensorFlow дозволяє розгортати обчислення на кількох пристроях (таких як графічні процесори або розподілені обчислювальні кластери), сприяючи швидшому навчанню моделей створення зображень і обробці величезних обсягів даних.

TensorFlow також пропонує інструменти для перегляду та відстеження навчання моделі. TensorBoard можна використовувати, наприклад, для моніторингу прогресу навчання, вивчення показників і перегляду створених зображень.

TensorFlow, ефективна платформа для дослідження зображень, пропонує інструменти для створення та вдосконалення генеративних моделей, розподіленого навчання та візуалізації результатів.

3.3 Розробка програмного застосунку

Програмний застосунок реалізує генеративно-протилежні мережі (GAN) для генерації зображень з ZIP архіву що буде розташований у папці з проєктом (рис. 3.2). Для цього використовує бібліотеку Keras з моделями і шарами, а також бібліотеку NumPy для обробки даних. Навчальний приклад включає в себе зображення маленьких дерев, але можна використовувати й інші зображення для тренування моделі.

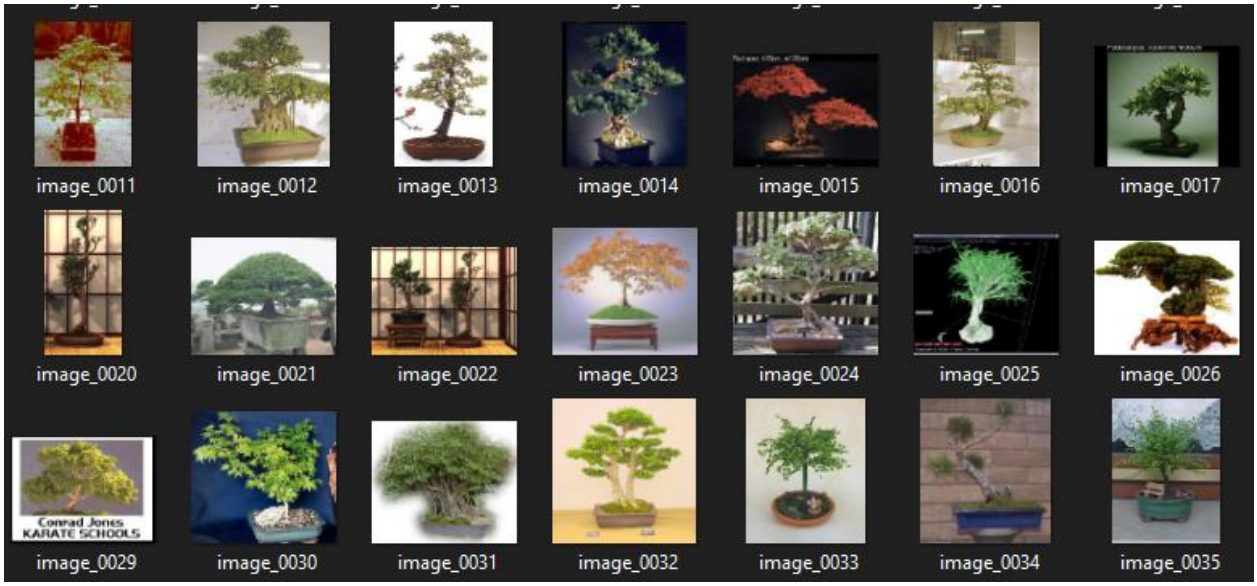

Рисунок 3.2 – Приклад вхідних даних з архіву

Програмний застосунок містить наступні основні компоненти:

– модуль розпаковки архіву та приведення зображень до єдиного формату: Розпаковує архів в вибрану папку де будуть зберігатися вхідні дані, при повторному відкритті програми буде зроблена перевірка на існування файлів в папці, якщо такі будуть повторна розпаковка архіву не почнеться для економії ресурсів. Далі програма зчитує зображення з папки та додає до масиву для подальшого використання. Форматує дані до єдиного розміру для подальшої передачі їх до дискримінатору.

Для початку розпаковки програмний застосунок визначає наявність архіву в форматі *.zip (де, * – будь-яка назва архіву) в папці з проєктом. Після цього розавковує файли в папку під назвою Unpacked де будуть зберігатися файли які будуть зчитані системою для подальшої обробки.

Наступним кроком зчитуємо всі файли які є картинками з папки Unpacked та додаємо їх до масиву ImagePaths. За допомогою бібліотеки Opencv зчитуємо з масиву картинки та приводимо їх до потрібного формату, у цьому випадку 64 на 64 пікселі. Зберігаємо в новий масив який потім переробимо в numpy масив, що є багатомірним. Приклад коду на мові Python наведено в додатку А з назвою «Розпаковка та приведення до єдиного формату»;

– модель генератора: Визначає архітектуру генератора, який приймає вхідні точки з латентного простору і генерує зображення. Генератор складається з повнозв’язаних шарів і функції активації.

Для реалізації моделі генератора було використано чотири приховані слої які створюють картинку починаючи з формату вісім на вісім і збільшуючи формат картинки в 2 рази на кожному слої до кінцевого формату в 64 на 64. Та в кінці при виводі зображення повинні визначити що картинка є різнокольоровою тож в пункті filter виводу з генератора зазначаємо три фільтри для кольорів червоний, зелений та синій які і складають будь-який

колір в певній пропорції кожного з трьох. Приклад коду на мові Python наведено в додатку А з назвою «Модель генератора»;

– модель дискримінатора: Визначає архітектуру дискримінатора, який приймає зображення як вхід і класифікує їх як реальні або фейкові. Дискримінатор також складається з повнозв’язаних шарів і функції активації.

Для реалізації моделі дискримінатора було використано три приховані слої які обробляють картинку та визначають точність її створення з урахуванням початкових, вхідних картинок на яких виконувалось навчання. Після перевірки виводимо згенероване зображення. Та зберігаємо нову модель з метою подальшого навчання. Приклад коду на мові Python наведено в додатку А з назвою «Модель дискримінатора»;

– модель GAN: Об'єднує генератор і дискримінатор для створення моделі GAN. При тренуванні GAN, генератор намагається згенерувати зображення, які дискримінатор не зможе відрізнити від реальних зображень.

Для моделі GAN потрібно об’єднати модель генератора та дискримінатора в одне ціле для їх одночасного навчання і отримання послідовного результату при навчанні без помилок та перенавчання однієї з моделей. Далі скомпілювати модель для подальшого її використання. Приклад коду на мові Python наведено в додатку А з назвою «Об’єднання генератора та дискримінатора в модель GAN»;

– функції для тренування: Використовуються функції для генерації реальних і фейкових зразків, обчислення втрати і точності, а також для тренування генератора і дискримінатора за допомогою градієнтного спуску.

Тренування моделі найважливіша частина реалізації програмного застосунку, сюди передаємо всі дані для навчання, а саме: модель генератора, модель дискримінатора, скомпільовану модель GAN, шум який буде використано для вводу в генератор, кількість епох навчання та кількість картинок для навчання за одну епоху. Ці всі дані нам необхідні для навчання системи.

Спочатку створюємо цикл в якому будемо рахувати кількість епох які були пройдені при навчанні. В циклі спочатку підготуємо картинки реальні та згенеровані поміщаючи їх до відповідних масивів. Потім передамо картинки дискримінатору на яких він зможе навчитися та почати визначення реальних та згенерованих зображень. Після цього тренуємо генератор, де передаємо йому шум з якого він намагається створити картинку для обману генератора. Та в цей же час визначаємо які картинки були використані для генератора і тренуємо його за допомогою скомпільованої моделі GAN. Приклад коду на мові Python наведено в додатку А з назвою «Тренування моделі GAN»;

– функції для візуалізації: Використовуються функції для відображення та збереження згенерованих зображень, а також для відображення процесу тренування, що включає перевірку на точність роботи генератора та дискримінатора.

При візуалізації використовуємо навчену модель для генерації зображень, тож перевіряємо точність дискримінатора на реальних даних та на згенерованих. Цю інформацію виводимо в консоль для відображення поточних результатів системи. Налаштовуємо вивід згенерованих картинок на екран користувача для візуального їх відображення за допомогою бібліотеки pyplot в цьому випадку. Для фінального відображення всіх епох в кінці роботи програми треба на цьому етапі також зберегти згенероване зображення як файл, набір яких потім буде сформовано в єдиний файл generated_gif.gif а наступних кроках. Приклад коду на мові Python наведено в додатку А з назвою «Функція візуалізації згенерованих зображень».

На рисунку 3.3 зображено отримані після навчання дані які були повністю згенеровані за допомогою шуму як вхідних даних.

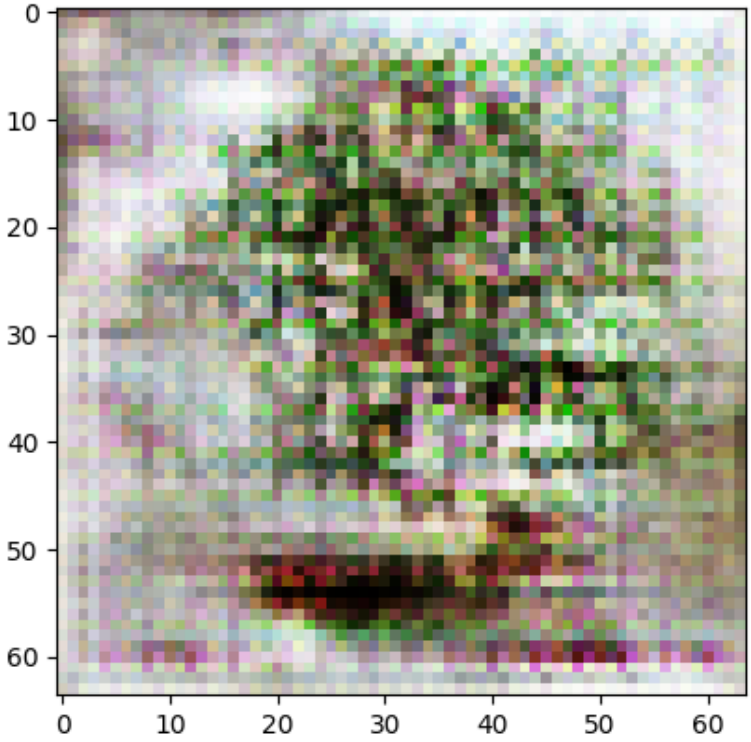

Рисунок 3.3 – Візуалізація згенерованих зображень

Для кращого візуального відображення було створено також generated_gif.gif файл який відображає всі епохи навчання одна за одною в порядку зростання та зберігає цей файл в папці з програмним застосунком.

Це було реалізовано за допомогою двох бібліотек imageio та glob де, бібліотека imageio в цьому проєкті допомагає створити gif файл за допомогою об’єднання різних зображень, а glob зчитує зображення з папки проєкту. Приклад коду на мові Python наведено в додатку А з назвою «Функція створення generated_gif.gif файлу».

Для візуалізації моделі GAN також було створено зображення з покроковою працею моделі. На рисунку 3.4 зображено вхідні дані їх приведення до однпкового формату та потім вивід згенерованих зображень.

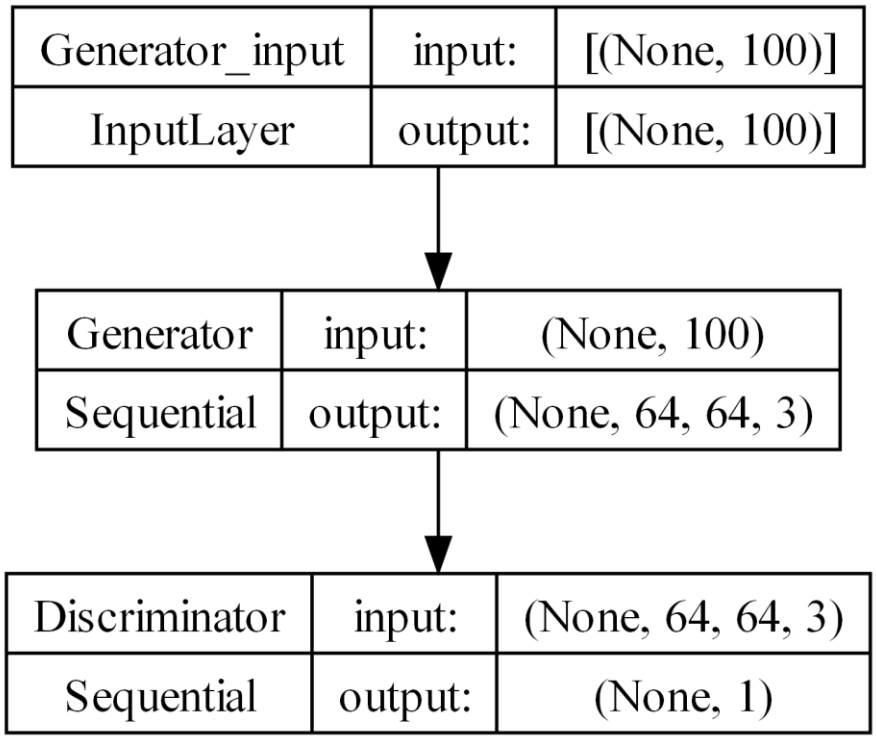

Рисунок 3.4 – Покрокова праця моделі

За моделлю програми також була створена сіквенс діаграма яка показана на рисунку 3.5 і зображує взаємодію всіх компонентів в системі.

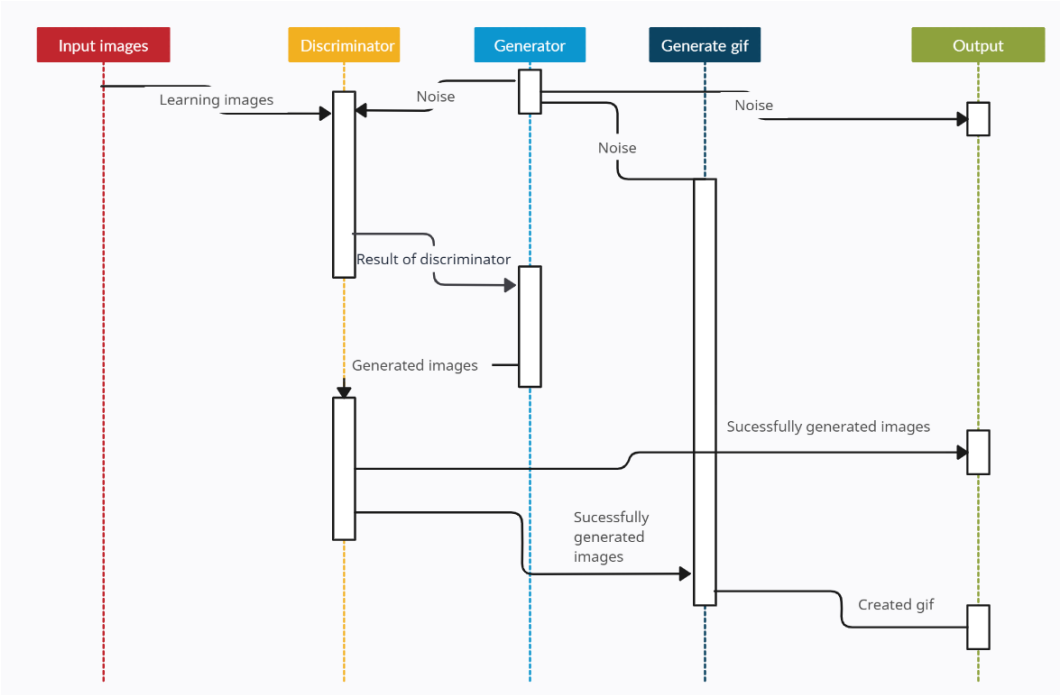

Рисунок 3.5 – Сіквенс діаграма моделі

Після завершення навчання будуть виведені кінцеві варіанти згенерованих зображень та створено файл generated_gif.gif який відобразить весь процес навчання системи.

3.4 Інструкція користувача

Перед запуском програми переконайтесь в наявності архіву з зображеннями або наявної папки з зображеннями в папці де розташований програмний застосунок (рис. 3.6).

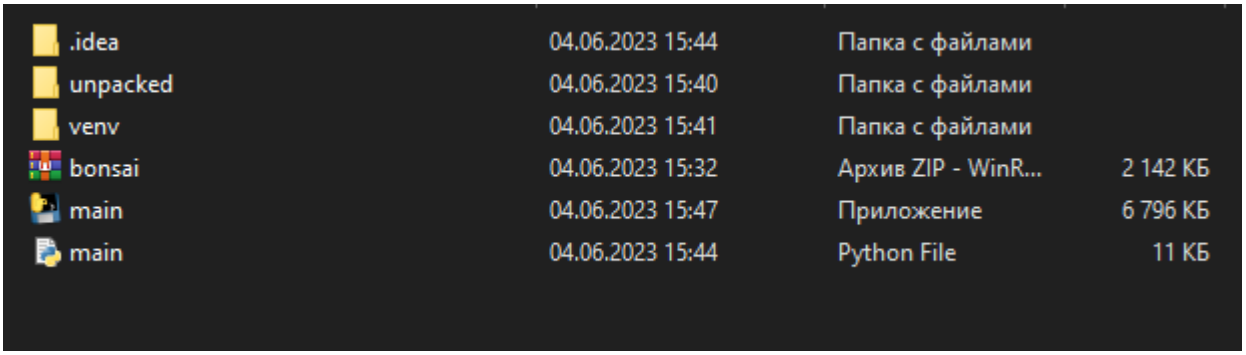

Рисунок 3.6 – Папка з файлами проєкту

Далі лишається тільки запустити програму яка розпакує архів, при необхідності, та почне навчання. При запуску будуть виведені зображення з вхідним шумом в систему та прикладами вхідних зображень в систему (рис. 3.7 – 3.9).

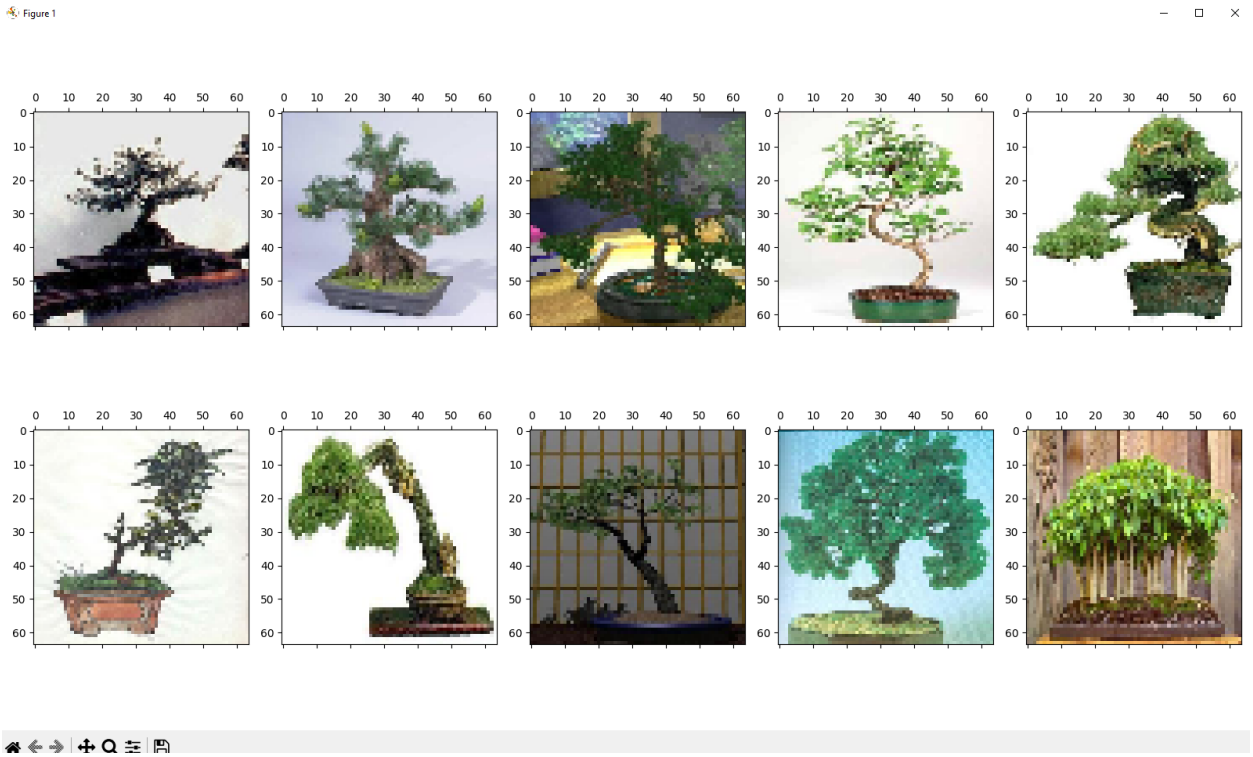

Рисунок 3.7 – Приклад вхідних зображень

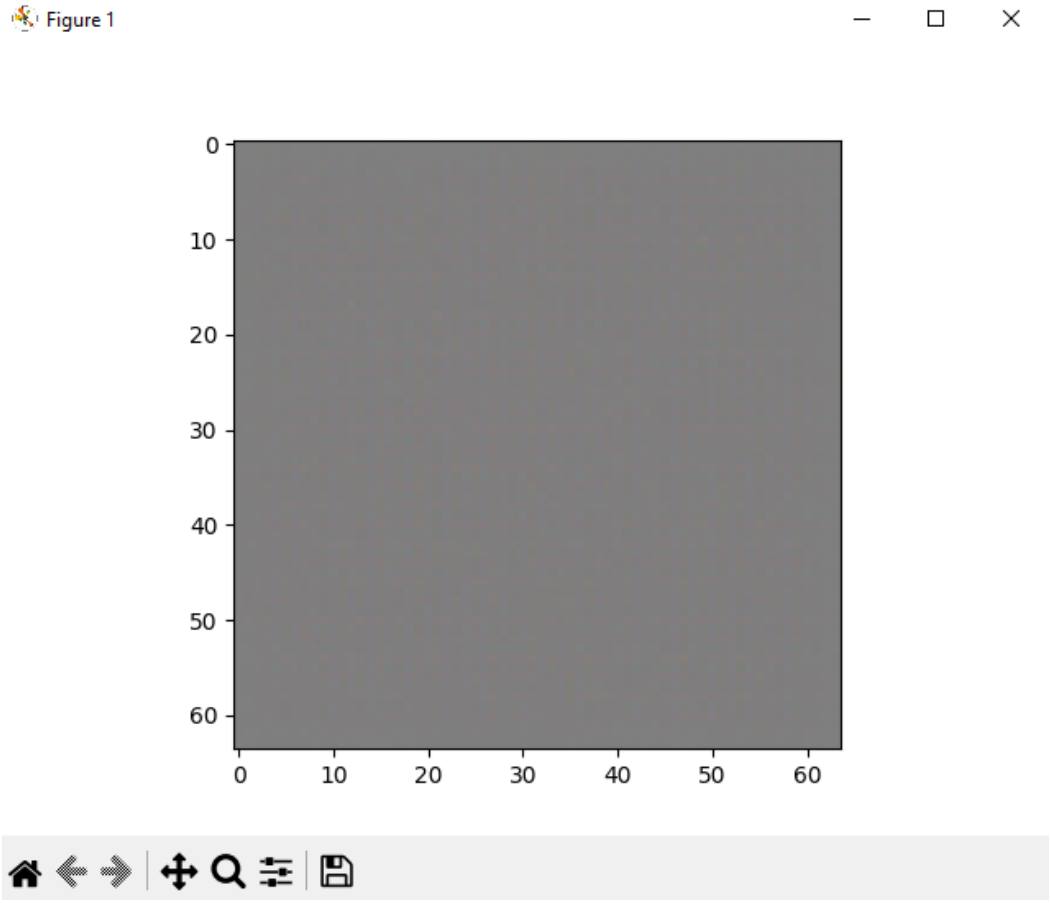

Рисунок 3.8 – Приклад шуму який обробляє шенератор

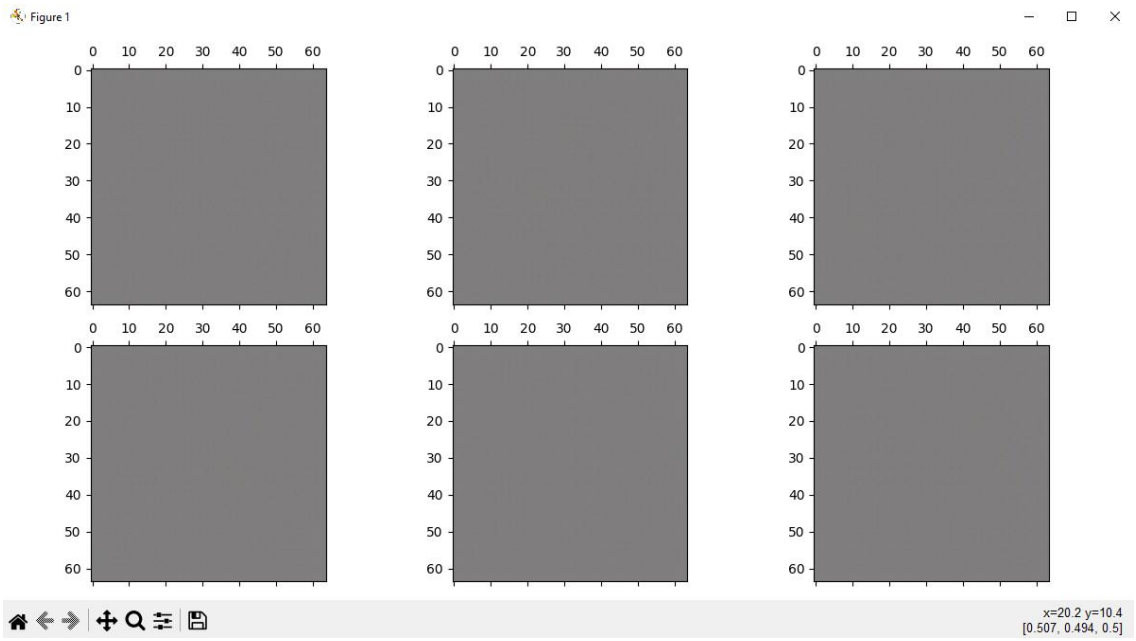

Рисунок 3.9 – Приклад зображень першої епохи навчання

Після закриття вікон з зображеннями буде розпочато навчання системи. Навчання може зайняти деякий час, але в процесі будуть виводитися проміжні результати (рис. 3.10) та зберігатися в вигляді прикладів в папці з програмним застосунком [22 – 34].

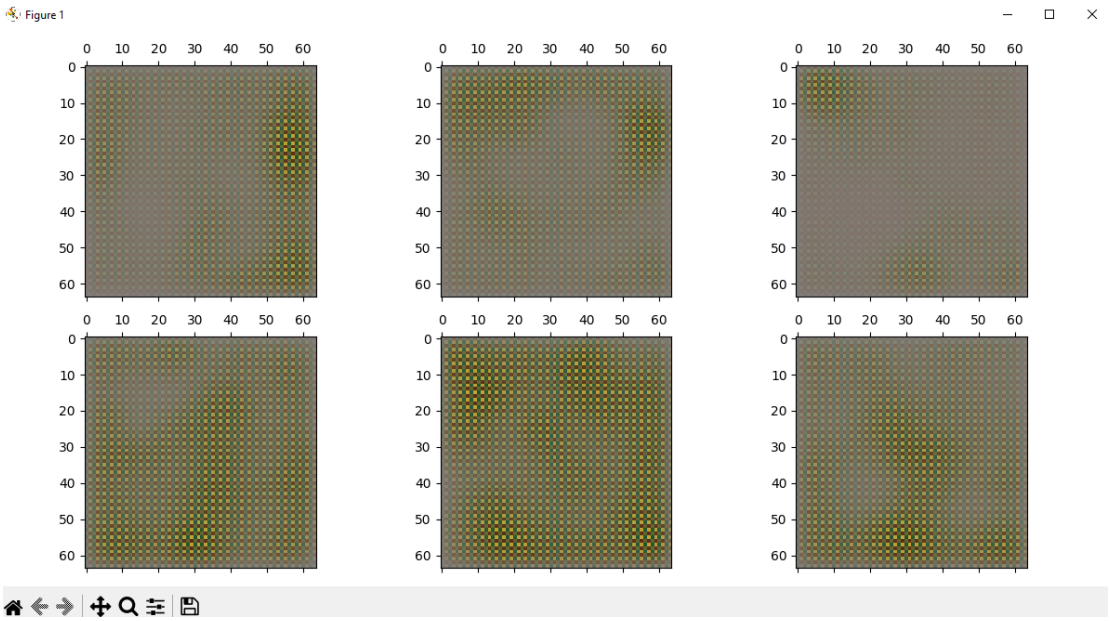

Рисунок 3.10 – Приклад зображень сотої епохи навчання

ВИСНОВКИ

У рамках кваліфікаційної роботи була проаналізована область нейронних мереж та генерації зображень, розроблений і реалізований програмний застосунок для генерації зображень.

Програмний застосунок генерує дані після навчання на зображеннях з архіву. Це все створюється за допомогою генератора та дискримінатора які впродовж навчання намагаються покращити себе змагаючись один з одним в одній моделі, тим самим покращуючи один одного і завдяки цьому створюють з кожною ітерацією кращий результат, що є картинка з зображенням маленького дерева у нашому випадку. Отримані картинки після приблизно тисячі циклів навчання стає важко відрізнити від початкових прикладів, що є корисно для отримання розширеного набору даних або створення нового на основі цих даних. Для генерації зображень можна використовувати майже будь-які дані які будуть в процесі оброблятися системою та генерувати нові зображення.

Завдяки створенню таких даних можна навчати інші програми або нейромережі наприклад для розпізнавання рослин або для будь-яких інших цілей.

Результати роботи апробовано у вигляді тези доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [35].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Oleksandr Zeleniy, Diana Rudenko, Valentin Lyubchenko, Vyacheslav Lyashenko (2022). Image Processing as an Analysis Tool in Medical Research.

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

3. Vyacheslav Lyashenko, Valentin Lyubchenko, Ayaz Mohammad, Khan Alveera, Oleg Kobylin (2016). The methodology of image processing in the study of the properties of fiber as a reinforcing agent in polymer compositions.

4. Yu-Wing Tai (2017). Conditional CycleGAN for Attribute Guided Face Image Generation.

5. Т.А. Самолюк (2019). Нейромережі GAN у створенні нових моделей.

6. How to spot the realistic fake people creeping into your timelines. URL: https://www.fastcompany.com/90332538/how-to-spot-the-creepy-fake-faces-who-may-be-lurking-in-your-timelines-deepfaces (дата звернення 08.05.2023).

7. I.J. Goodfellow (2014). Generative adversarial networks. Universite de Montreal.

8. Generative Adversarial Networks (GAN) Serve Safety and Cybersecurity. URL: https://towardsdatascience.com/generative-adversarial-networks-gan-serve-safety-and-cybersecurity-34278ce7a31d (дата звернення 10.05.2023).

9. Adversarial Training Methods for Deep Learning: A Systematic Review. URL: https://www.mdpi.com/1999-4893/15/8/283#:~:text=Adversarial%20training%20is%20one%20of,adversarial%20data%20and%20clean%20data. (дата звернення 10.05.2023).

10. A Detailed Explanation of GAN with Implementation Using Tensorflow and Keras. URL: https://www.analyticsvidhya.com/blog/2021/06/a-detailed-

explanation-of-gan-with-implementation-using-tensorflow-and-keras/ (дата звернення 10.05.2023).

11. Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, Honglak Lee (2016). Learning What and Where to Draw.

12. Alec Radford(2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,

13. MNIST-GAN: Detailed step by step explanation & implementation in code. URL: https://medium.com/intel-student-ambassadors/mnist-gan-detailed-step-by-step-explanation-implementation-in-code-ecc93b22dc60 (дата звернення 12.05.2023).

14. Image Generation in 10 Minutes with Generative Adversarial Networks. URL: https://towardsdatascience.com/image-generation-in-10-minutes-with-generative-adversarial-networks-c2afc56bfa3b (дата звернення 25.05.2023).

15. Alec Radford (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,.

16. How to Develop a GAN for Generating MNIST Handwritten Digits. URL: https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-an-mnist-handwritten-digits-from-scratch-in-keras/ (дата звернення 12.05.2023).

17. 18 Impressive Applications of Generative Adversarial Networks (GANs). URL: https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/ (дата звернення 12.05.2023).

18. Generative Adversarial Network (GAN). URL: https://www.geeksforgeeks.org/generative-adversarial-network-gan/ (дата звернення 12.05.2023).

19. Що таке GAN - генеративно-змагальні нейронні мережі і як їх застосовувати для генерації зображень. URL: https://evergreens.com.ua/ua/articles/gan.html (дата звернення 13.05.2023).

20. Machine learning Life cycle. URL: https://www.javatpoint.com/machine-learning-life-cycle (дата звернення 18.05.2023).

21. Генеративна змагальна мережа. URL: https://uk.wikipedia.org/wiki/Генеративна_змагальна_мережа (дата звернення 13.05.2023).

22. Create Data From Random Noise With Generative Adversarial Networks. URL: https://www.toptal.com/machine-learning/generative-adversarial-networks (дата звернення 13.05.2023).

23. How to get datasets for Machine Learning. URL: https://www.javatpoint.com/how-to-get-datasets-for-machine-learning (дата звернення 21.05.2023).

24. A Gentle Introduction to Generative Adversarial Networks (GANs). URL: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (дата звернення 13.05.2023).

25. How to Develop a Conditional GAN (cGAN) From Scratch. URL: https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/ (дата звернення 13.05.2023).

26. GANs from scratch. URL: https://towardsdatascience.com/gans-from-scratch-8f5da17b3fb4 (дата звернення 13.05.2023).

27. Building a GAN's in 8 steps from scratch. URL: https://www.kaggle.com/code/adusumilligkrishna/building-a-gan-s-in-8-steps-from-scratch/notebook#Step1:-Importing-the-modules (дата звернення 15.05.2023).

28. Generative Adversarial Networks: Build Your First Models. URL: https://realpython.com/generative-adversarial-networks/ (дата звернення 15.05.2023).

29. Overview Of Generative Adversarial Networks. URL: https://www.c-sharpcorner.com/article/overview-of-generative-adversarial-networks/ (дата звернення 15.05.2023).

30. Implementing GANs from Scratch. URL: https://www.section.io/engineering-education/implementing-gan-from-scratch/ (дата звернення 18.05.2023).

31. Machine learning. URL: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML (дата звернення 18.05.2023).

32. What is machine learning? URL: https://www.ibm.com/topics/machine-learning (дата звернення 18.05.2023).

33. Machine Learning Tutorial. URL: https://www.javatpoint.com/machine-learning (дата звернення 24.05.2023).

34. A Detailed Explanation of GAN with Implementation Using Tensorflow and Keras URL: https://www.kaggle.com/general/247477 (дата звернення 25.05.2023)

35. Шемаєв Д. І. (2023) Програмний застосунок для генерації датасетів за допомогою тестових даних: 27-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті». Зб. матеріалів форуму. Т. 7. Харків: ХНУРЕ. С. 14-15.