Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Формування альтернативних підходів до генерації шумів у GAN-мережах
(тема)

Виконав:
студент 2 курсу, групи СШІм-22-2
Білоконь В.А.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)
Керівник проф. Рябова Н.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)
Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____20 ___р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Білоконю Василю Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Формування альтернативних підходів до генерації шумів у GAN-мережах

затверджена наказом університету від 1 квітня 20 24 р. № 260Ст
2. Термін подання студентом роботи до екзаменаційної комісії 5 червня 20 24 р.
3. Вихідні дані до роботи науково-технічні публікації, дані Інтернет-джерел та відомих наукових проектів щодо розробки та дослідження генеративних моделей, набори даних зображень з підписами

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі і постановка задачі дослідження
2) Огляд генеративних моделей
3) Огляд стандартних та альтернативних видів шумів
4) Програмна реалізація і порівняння розроблених моделей

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	01.04.2024	виконано
2	Аналіз предметної галузі	01.04.2024 – 05.04.2024	виконано
3	Постановка задачі	08.04.2024 – 12.04.2024	виконано
4	Порівняльний аналіз існуючих підходів щодо	15.04.2024 – 19.04.2024	виконано
	вирішення проблеми, що досліджується		
5	Пропонований підхід до вирішення досліджуваної	22.04.2024 – 03.05.2024	виконано
	проблеми та її основних складових задач		
6	Експериментальне моделювання та навчання моделей	06.05.2024 – 17.05.2024	виконано
7	Оцінка результатів	20.05.2024 – 24.05.2024	виконано
8	Написання пояснювальної записки	01.05.2024 – 31.05.2024	виконано
9	Попередній захист	31.05.2024	виконано
10	Захист перед ЕК	05.06.2024	

Дата видачі завдання 1 квітня 20 24 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Рябова Н.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 103 с., 32 рис., 7 табл., 2 дод., 32 джерела.

ВИДИ ШУМІВ, ГЕНЕРАТИВНО-ЗМАГАЛЬНА МЕРЕЖА, ГЕНЕРАЦІЯ ЗОБРАЖЕНЬ, ГЕНЕРАЦІЯ ШУМІВ, СИНТЕТИЧНЕ ЗОБРАЖЕННЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об'єкт дослідження – процес генерації зображень генеративно-змагальними мережами, особливості і процес навчання в залежності від різних видів шумів.

Предмет дослідження – методи моделювання генеративно-змагальних мереж і їх навчання, порівняння, метрики оцінки якості генерацій.

Мета роботи – вивчення та застосування генеративно-змагальних мереж для генерації зображень, а також порівняння їх ефективності при використанні різних підходів при формуванні шумів.

Методами дослідження є методи машинного та глибинного навчання стосовно досліджуваної проблеми, аналіз науково-технічної літератури в області генерації зображень, порівняльний аналіз найбільш відомих та успішних результатів досліджень.

У цій дослідницькій роботі розглядається процес генерації зображень за допомогою генеративно-змагальних мереж, а також розгляд стандартних та альтернативних видів шумів. Зрозуміти в яких завданнях краще використовувати ті чи інші види вхідних шумів.

ABSTRACT

Master’s thesis contains: 103 pp., 32 fig., 7 tabl., 2 ann., 32 sources.

ARTIFICIAL INTELLIGENCE, GENERATION OF IMAGES, GENERATION OF NOISES, GENERATIVE ADVERSARIAL NETWORK, SYNTHETIC IMAGE, TYPES OF NOISES.

The object of the study is the process of image generation by generative competitive networks, features and the learning process depending on different types of noise.

The subject of the research is the methods of modeling generative-competitive networks and their training, comparison, metrics for assessing the quality of generations.

The purpose of the work is to study and apply generative competitive networks for image generation, as well as to compare their effectiveness when using different approaches to noise generation.

The research methods are machine and deep learning methods in relation to the researched problem, analysis of scientific and technical literature in the field of image generation, comparative analysis of the most famous and successful research results.

This research work examines the process of image generation using generative adversarial networks, as well as consideration of standard and alternative types of noise. Understand in which tasks it is better to use certain types of input noise.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень та термінів 9
Вступ 10
1 Аналіз предметної галузі 12
1.1 Генерація даних за допомогою GAN-мереж 12
1.2 Опис завдання генерації зображень 13
1.3 Загальне визначення генеративно-змагальних мереж 17
1.3.1 Історія розвитку GAN 17
1.3.2 Загальна архітектура GAN 18
1.3.3 Згорткові нейронні мережі (CNN) 21
1.3.4 Архітектура генератора 22
1.3.5 Архітектура дискримінатора 23
1.3.6 Функція втрат 25
1.4 Аналіз існуючих рішень 27
1.4.1 Оригінальна немодифікована генеративно-змагальна мережа 27
1.4.2 Варіаційні автокодувальники 29
1.4.3 Генеративні моделі на основі потоку 31
1.4.4 Дифузійні моделі 32
1.4.5 Авторегресійні моделі 34
1.5 Еволюція моделей генерацій зображень за текстовим описом 36
1.6 Постановка задачі дослідження 38
2 Огляд генеративних моделей 40
2.1 Генеративно-змагальні мережі 40
2.1.1 Базова генеративно-змагальна мережа 40
2.1.2 Додавання умови до генерації в GAN (CGAN) 42
2.1.3 Генеративна змагальна мережа з глибокими згортками (DCGAN) 44

2.1.4 Стекова генеративна змагальна мережа (StackGAN) 46
2.1.5 Циклічна генеративна змагальна мережа (CycleGAN) 48
2.1.6 StyleGAN 50
2.2 Дифузійні моделі 52
2.2.1 Ймовірнісна дифузійна модель з усуненням шуму (DDPM) 53
2.3 Переваги та недоліки GAN-мереж та дифузійних моделей 57
3 Основні підходи до формування та генерації шумів 60
3.1 Стандартні типи вхідного шуму 60
3.1.1 Нормальний розподіл (Gaussian Distribution) 60
3.1.2 Рівномірний розподіл (Uniform Distribution) 61
3.1.3 Логнормальний розподіл (Lognormal distribution) 62
3.1.4 Експоненційний розподіл (Exponential Distribution) 63
3.1.5 Розподіл Лапласа (Laplace) 64
3.2 Альтернативні види генерації шуму 66
3.2.1 Перлін-шум (Perlin Noise) 66
3.2.2 Шум з фрактальної геометрії (Fractal Noise) 67
3.2.3 Шум зображення (Image Noise) 68
3.2.4 Додавання випадкових векторів (Random Vectors) 68
3.2.5 Випадкові спотворення (Random Distortions) 69
3.3 Використання стандартних та альтернативних шумів 69
4 Програмна реалізація 73
4.1 Обґрунтування обраних програмних засобів 73
4.1.1 Порівняння PyTorch і TensorFlow 73
4.1.2 Набір бібліотек для проведення експериментів 75
4.2 Огляд набору даних 75
4.2.1 Опис набору даних MNIST 75
4.2.2 Обґрунтування вибору набору даних MNIST 77
4.3 Огляд метрик для оцінки моделей 77
4.3.1. Функція втрат на основі бінарної крос-ентропії (BCE) 78

4.3.2 Frechet Inception Distance (FID) 79
4.3.3 Inception Score (IS) 80
4.4 Реалізація і тренування моделей 81
4.5 Оцінка якості результатів 84
4.5.1 Аналіз графіків втрат 84
4.5.2 Візуальний аналіз результатів 87
4.5.2 Аналіз результатів під час використання різних шумів 88
4.5.3 Кількісний аналіз якості моделей 90
4.6 Перспективи розвитку 91
Висновки 93
Перелік джерел посилання 95
Додаток А Вихідний код програми для досвідчених експериментів 99
Додаток Б Відомість кваліфікаційної роботи 103

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

CNN – Convolutional Neural Network – згорткова нейронна мережа, призначена для обробки даних із сітчастою топологією, таких як зображення;

DCGAN – Deep Convolutional Generative Adversarial Network – архітектура генеративної змагальної мережі, що використовує глибокі згорткові нейронні мережі для генерації реалістичних зображень;

DNN – Deconvolutional Neural Network – зворотна згорткова нейронна мережа, яка використовує деконволюцію для збільшення роздільної здатності вхідних даних, зокрема для генерування зображень або візуалізації;

DM – Diffusion Model − дифузійна модель, що використовується для аналізу та моделювання процесів дифузії в системах, де речовина рухається від висококонцентрованої області до менш концентрованої;

DDPM – Denoising Diffusion Probabilistic Models – ймовірнісна дифузійна модель з усуненням шуму, яка використовує процес дифузії для поступового додавання та видалення шуму з даних, щоб навчитися їх відновлювати;

GAN – Generative Adversarial Network − генеративна змагальна мережа, це тип штучної нейронної мережі, яка використовується для генерації нових даних;

MNIST – Modified National Institute of Standards and Technology database – це велика база даних рукописних цифр, яка зазвичай використовується для навчання різних систем обробки зображень;

VAE – Variational Autoencoder − варіаційний автокодувальник, це тип нейронної мережі, яка використовується для генерації нових даних, навчаючись репрезентувати та генерувати зображення в латентному просторі.

ВСТУП

В останні роки генеративні моделі здійснили справжній прорив, досягнувши небаченого раніше рівня реалістичності у генерації людиноподібного тексту, зображень, музики та інших типів матеріалів. Ці моделі відкривають найширший спектр можливостей, від створення зображень текстових описів до розробки алгоритмів стиснення даних.

Незважаючи на вже досягнутий рівень реалістичності, генеративні моделі постійно удосконалюються, залишаючи за собою величезний потенціал для перетворення різних галузей, таких як графічний дизайн, ігрова індустрія, музичне виробництво, промисловий дизайн.

Одним із найбільш вражаючих досягнень у галузі генеративних моделей є перетворення тексту на зображення. Це завдання, що лежить на стику комп'ютерного зору та обробки природної мови, є нетривіальною проблемою, вирішення якої знаменує собою важливий крок на шляху до створення штучного інтелекту загального призначення.

Моделі для перетворення тексту на зображення, як правило, навчаються на масивних наборах даних, що містять зображення та їх текстові описи. У процесі навчання модель освоює асоціації між словами та фразами, з одного боку, та візуальними характеристиками (форма, колір, текстура) – з іншого.

Найбільш поширеною архітектурою для цього завдання є генеративно-змагальна мережа (GAN). GAN складається з двох нейронних мереж: генератора та дискримінатора. Генератор «навчається» створювати зображення з текстових описів, а дискримінатор – відрізняти згенеровані зображення від реальних.

Незважаючи на вражаючі результати, GAN та інші моделі перетворення тексту на зображення все ще мають ряд недоліків, що обмежують їх масштабованість та застосовність у різних доменах. Це стимулює активні дослідження альтернативних архітектур, таких як

дифузійні моделі, а також пошук нових підходів до створення шумів для створення зображень.

Генеративні моделі – це область, що динамічно розвивається, має величезний потенціал для перетворення різних сфер людської діяльності. Удосконалення цих моделей та пошук нових рішень у галузі генерації контенту – це ключ до створення по-справжньому «розумних» систем, здатних допомагати людям.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Генерація даних за допомогою GAN-мереж

Генеративно-змагальні мережі є потужним інструментом у галузі глибокого навчання, що дозволяє генерувати дані, які можуть бути зовні невиразні від реальних. З моменту свого введення в 2014 році Гудфеллоу та інших [1], GAN-мережі стали об'єктом інтенсивних досліджень у різних галузях завдяки своїй здатності до створення нових даних на основі статистичних закономірностей навчального набору.

У контексті роботи GAN-мереж широкий спектр даних може бути успішно згенерований. Деякі з найважливіших напрямів можливих генерацій даних [2]:

а) GAN-мережі можуть використовуватися для створення реалістичних зображень осіб, тварин, природи та інших об'єктів. Це знайшло застосування у різних галузях, включаючи комп'ютерну графіку, рекламу, розважальну індустрію та фотообробку:

– реалістичні обличчя – фотореалістичні зображення людей;

– художні твори – зображення у різних стилях, від імпресіонізму до поп-арту;

– предмети – 3D-моделі об'єктів, таких як меблі, одяг та навіть їжа;

– ландшафти – реалістичні зображення пейзажів, яких немає у світі;

б) GAN-мережі також можуть застосовуватися для генерації текстових даних, таких як вірші, рецепти, музичні тексти тощо. Це може бути корисним у завданнях автоматичного створення контенту та генерації нових ідей:

– статті та тексти на різні теми, від новин до наукових статей;

– вірші у різних стилях, від сонетів до репу;

– вихідний код для різних програм;

в) GAN-мережі можуть створювати звукові ефекти, музичні композиції та голосові дані. Це може бути використане в музичній галузі, аудіо-обробці та створенні звукових ефектів для фільмів та ігор:

– генерація мелодій у різних жанрах, від класики до електронної музики;

– звуки – такі як шум дощу, спів птахів чи гул машин;

г) GAN-мережі також можуть створювати реалістичні відео, включаючи анімацію, візуалізації та спецефекти. Це має застосування у кінематографі, анімації, медичній візуалізації та інших областях:

– відеоролики – короткі відеоролики, наприклад, анімація або відео згенерованих осіб;

– спецефекти – можуть використовуватись для створення реалістичних спецефектів для фільмів та відеоігор.

Цей різноманітний спектр застосувань GAN-мереж підкреслює їх значущість та потенціал у галузі штучного інтелекту та глибокого навчання. У міру розвитку технології можливості GAN тільки розширюватимуться.

Варто зазначити, що GAN не завжди генерують бездоганний контент. Зображення можуть бути розмитими або мати спотворення (артефакти), текст може бути безглуздим, а музика – какофонічною. Тим не менш, GAN є потужним інструментом, який може використовуватись для створення нового та оригінального контенту.

1.2 Опис завдання генерації зображень

З розвитком технологій аналізу даних у різних галузях стає викликом, особливо у контексті обробки візуальної інформації. Одним із ключових напрямів є застосування аналітичних методів, таких як машинне та глибинне навчання, для вирішення проблем, що раніше не могли бути вирішені за допомогою традиційних статистичних методів. Це стимулює

розвиток різноманітних напрямків досліджень, включаючи генерацію зображень з різних даних.

Підходи до генерації зображень можна класифікувати за різними категоріями, такими як перетворення одного типу зображення в інший, створення ескізів зображень, зображення з врахуванням умов, конвертація тексту в зображення, обробка кількох зображень, розпізнавання облич, створення макетів, зображення з урахуванням позицій, а також генерація відео, панорамних виглядів та графів сцен.

На рисунку 1.1 представлено опис цих категорій [3].

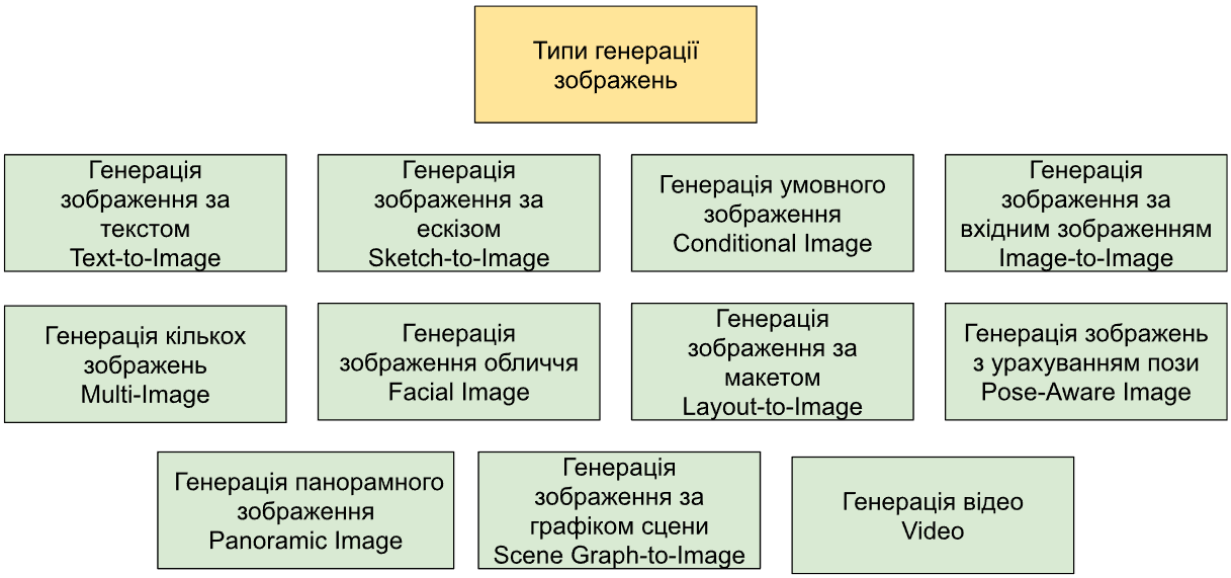


Рисунок 1.1 − Види техніки генерації зображень

Типи генерації зображень:

– генерація зображення за текстом (Text-to-Image Generation) передбачає створення зображень на основі текстових описів. Моделі машинного навчання навчені розуміти зв’язок між текстом і відповідними зображеннями, що дозволяє їм генерувати зображення, які відповідають наданим описам;

– генерація зображення за ескізом (Sketch-to-Image Generation) у цьому підході грубі ескізи або контури перетворюються на детальні

зображення. Алгоритми машинного навчання аналізують вхідні дані ескізу та генерують реалістичні зображення на основі наданих контурів;

– генерація умовного зображення (Conditional Image Generation) передбачає створення зображень на основі конкретних умов або атрибутів. Наприклад, створення зображень різних об’єктів або сцен на основі факторів, таких як колір, розмір або форма;

– генерація зображення за вхідним зображенням (Image-to-Image) у цьому підході система приймає на вхід одне зображення і генерує нове зображення на основі цього вхідного зображення. Це може включати перетворення зображення одного типу на зображення іншого типу як перетворення чорно-білого зображення на кольорове, стилювання фотографії або створення мистецьких ефектів;

– генерація кількох зображень (Multi-Image Generation) ця техніка передбачає створення серії пов’язаних зображень або складеного зображення з кількох вхідних зображень. Його можна використовувати для таких завдань, як завершення зображення, коли відсутні частини зображення генеруються на основі навколишнього контексту;

– генерація зображень обличчя (Facial Image Generation) ця техніка, спеціально зосереджена на створенні зображень людських облич, спрямована на створення реалістичних зображень обличчя, які часто використовуються в таких програмах, як створення зображень для соціальних сетей (аватарів) або системи розпізнавання облич;

– генерація зображення за макетом (Layout-to-Image Generation) це передбачає створення зображень на основі попередньо визначених макетів або структур. Наприклад, створення дизайну інтер’єру на основі планування кімнат або архітектурних креслень;

– генерація зображень з урахуванням пози (Pose-Aware Image Generation) у цьому підході зображення генеруються з урахуванням конкретних поз або орієнтацій. Це зазвичай використовується в таких програмах, як оцінка пози або віртуальна примірка одягу;

– генерація панорамних зображень (Panoramic Image Generation) – створення панорамних зображень передбачає зшивання кількох зображень для створення ширококутного огляду сцени. Ця техніка зазвичай використовується у фотографії та віртуальних турах;

– генерація зображення за графіком сцени (Scene Graph-to-Image Generation) графіки сцени представляють об’єкти сцени разом із їхніми зв’язками та атрибутами. Генерація графіка сцени до зображення передбачає створення зображень на основі цих структурованих представлень, що дозволяє створювати складні сцени з кількома об’єктами та взаємодіями;

– генерація відео (Video Generation) – генерація відео зосереджені на створенні послідовності зображень для створення реалістичних відео. Ці методи часто передбачають створення послідовних кадрів на основі початкового кадру або ключових кадрів.

Ці методи демонструють універсальність і можливості сучасних алгоритмів машинного навчання для створення різноманітних і реалістичних зображень у різних областях і програмах.

Перетворення тексту в зображення відоме як унікальна та складна задача в галузі глибокого навчання. Ця задача відрізняється від інших завдань, таких як класифікація зображень або виявлення об'єктів, через деякі особливості:

– неоднозначність і суб'єктивність текстових описів – оскільки різні люди можуть розуміти один і той же текст по-різному, моделі повинні вміти точно інтерпретувати текстові описи, щоб генерувати відповідні зображення. Наприклад, опис «жовта квітка» може мати різні візуальні інтерпретації, такі як ромашка або соняшник;

– великі обсяги даних – текстові описи можуть бути довгими і різноманітними, що ускладнює засвоєння моделлю складних взаємозв'язків між текстом і зображеннями;

– обмеженість маркованих даних – для синтезу текст-зображення доступно обмежена кількість маркованих даних через складність анотування текстових описів;

– вимоги до реалістичності – згенеровані зображення повинні відтворювати деталі та характеристики, описані в тексті, і виглядати природно і реалістично.

Незважаючи на ці складності, синтез тексту в зображення має великий потенціал у різних сферах, таких як ігри, електронна комерція, реклама і медицина.

1.3 Загальне визначення генеративно-змагальних мереж

1.3.1 Історія розвитку GAN

Генеративно-змагальні мережі були представлені в роботі Іена Гудфеллоу та його колег у 2014 році [1]. Вони запропонували новий підхід до генерації даних, заснований на концепції «змагальності» двох нейронних мереж – генератора та дискримінатора. Ця концепція була натхненна ігровою теорією, де два гравці, генератор та дискримінатор, змагаються один з одним. У роботі була продемонстрована прикладі генерації зображень осіб.

У наступні роки були розроблені різні методи та техніки для покращення стабільності навчання GAN. Це включало використання різних функцій втрат, нормалізацію даних і модифікації архітектури мереж.

З часом GAN стали широко застосовуватися в різних галузях, включаючи комп'ютерний зір, обробку природної мови, аудіообробку, геноміку та інші. Кожна область має свої унікальні складності та вимоги, що призвело до розробки спеціалізованих варіантів GAN [4].

Пізніше були запропоновані різні варіації GAN із складнішими архітектурами та функціями втрат. Деякі з них включають умовні

GAN (Conditional GAN), де генератор та дискримінатор мають доступ до додаткової інформації, та автокодувальник-GAN (AE-GAN), що поєднує концепції GAN та автокодувальника.

1.3.2 Загальна архітектура GAN

Генеративно-змагальні мережі являють собою модель глибоких нейромереж, що складається з двох основних компонентів: генератора та дискримінатора. Генератор відповідає створення нових зразків даних, імітуючи розподіл навчального набору, тоді як дискримінатор намагається розрізнити згенеровані дані від реальних. Ще одна важлива частина цієї структури є функцією втрат. Функція втрат дає зворотний зв'язок генератору та дискременатору для оновлення ваги та забезпечує критерії зупинки дій навчання.

Архітектура GAN-мережі складається з кількох шарів, які забезпечують виконання завдання генерації даних. Зазвичай генератор і дискримінатор є згортковими нейронними мережами (CNN) або зворотними згортковими нейронними мережами (DNN), хоча можуть використовуватися й інші архітектури, залежно від конкретної задачі.

Навчання GAN-мереж є процесом змагання між генератором та дискримінатором, і успішне навчання вимагає правильного балансу між різноманітністю згенерованих даних та їх реалістичністю [5].

Схематичне зображення архітектури GAN представлене на рисунку 1.2.

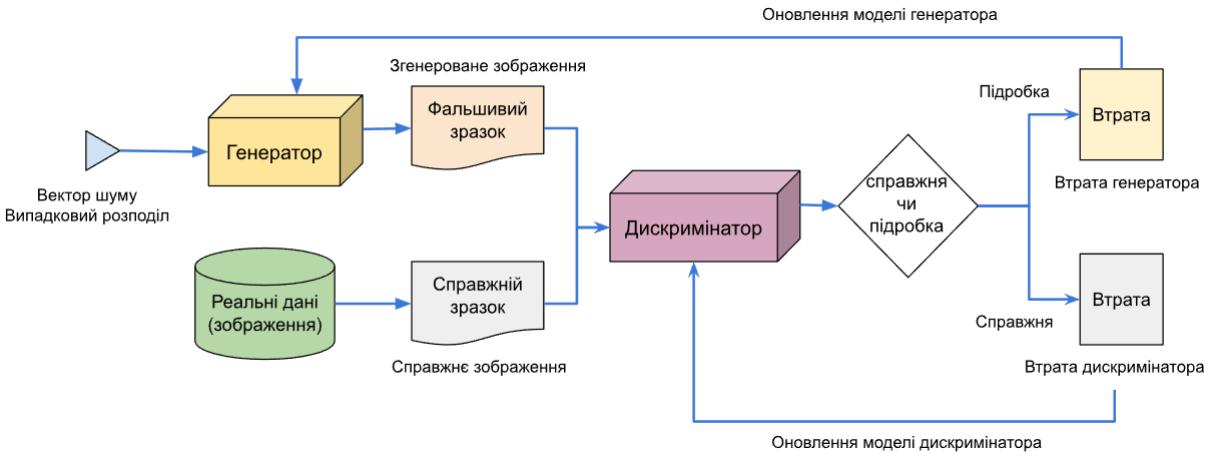


Рисунок 1.2 – Загальна архітектура генеративно-змагальної мережі

Для навчання генератора використовується вектор шуму, витягнутий із випадкового набору цифр (зазвичай розподіл Гауса). Результат роботи генератора – зображення певного розміру (ширина x висота x RGB колір). Генератор передає свій результат дискримінатору та коригує свої параметри на основі втрат дискримінатора з метою збільшення їх.

Для навчання дискримінатора використовуються як позначені зображення, згенеровані генератором, і справжні зображення. Дискримінатор вчиться визначати зображення як реальні чи згенеровані та навчається з використанням функції втрат. Навчання генератора та дискримінатора відбувається по черзі. Ця схема навчання схожа на змагальну гру для двох гравців, генератор прагне максимізувати втрати дискримінатора, а дискримінатор прагне мінімізувати власні втрати.

Алгоритм роботи генеративно-змагальних мереж включає наступні основні кроки:

а) ініціалізація. Створення двох нейронних мереж: генератора (Generator) та дискримінатора (Discriminator). Ініціалізація параметрів обох мереж є випадковими значеннями;

б) навчання:

– крок 1: навчання дискримінатора. Подача на вхід дискримінатора реальних зразків даних з навчального набору та синтетичних зразків, згенерованих генератором. Дискримінатор класифікує кожен зразок як реальний (що належить навчальному набору) або синтетичний (генерований генератором). Розраховується функція втрат для дискримінатора, яка оцінює, наскільки добре він розрізняє реальні та згенеровані зразки;

– крок 2: навчання генератора. Генератор створює синтетичні зразки даних з урахуванням випадкового шуму чи іншого вхідного сигналу. Згенеровані зразки подаються на вхід дискримінатору. Генератор прагне мінімізувати можливість, що дискримінатор зможе правильно класифікувати синтетичні зразки як «згенеровані», тобто він намагається обдурити дискримінатор. Розраховується функція втрат для генератора, яка оцінює, наскільки добре синтетичні зразки обманюють дискримінатор;

в) послідовний процес навчання. Кроки 1 і 2 повторюються чергово. Обидві мережі прагнуть знаходження рівноваги, у якому дискримінатор неспроможна відрізнити згенеровані зразки від реальних, а генератор створює зразки, які мають реальні дані.

Навчання зупиняється, коли досягається певний критерій зупинки, наприклад коли досягнуто певну кількість епох навчання або функція втрат перестає змінюватися.

Використання навченої мережі для створення даних. Після успішного навчання генеративно-змагальних мереж, генератор стає здатним створювати нові дані, які можуть бути невідмінними від реальних зразків у навчальному наборі. Для створення нових даних просто подається випадковий шум або інший вхідний сигнал на вхід генератора. Генератор трансформує цей вхідний сигнал на синтетичний зразок даних, використовуючи шаблони та закономірності, вивчені в процесі навчання.

Алгоритм GAN дозволяє навчати дві мережі паралельно, кожна з яких покращує свої навички за рахунок конкуренції з іншою. Це дозволяє генератору створювати більш реалістичні зразки даних, тоді як дискримінатор прагне стати дедалі точнішим у розрізненні реальних і згенерованих зразків.

1.3.3 Згорткові нейронні мережі (CNN)

Генеративно-змагальні мережі використовують як згорткові нейронні мережі (CNN), так і зворотні згорткові нейронні мережі (DNN) як свої основні компоненти, залежно від конкретної архітектури та завдання. CNN є основним вибором для роботи із зображеннями через їхню здатність до вивчення просторових закономірностей у даних.

Згорткові нейронні мережі складаються з декількох типів шарів, включаючи шари згортки, шари об'єднання і повнозв'язні шари.

Згорткові шари – ці шари виконують операцію згортки між вхідними даними та набором фільтрів, щоб отримати різні ознаки з вхідних зображень. Фільтри дозволяють виявляти різні характеристики, такі як грані, текстури та форми об'єктів.

Шари об'єднання (pooling) – після згортки використовуються шари об'єднання для зменшення розмірності ознакового простору та підвищення інваріантності до невеликих трансформацій вхідних даних.

Повнозв'язкові шари – в кінці CNN зазвичай слідують один або кілька повнозв'язкових шарів, які об'єднують одержані ознаки для прийняття остаточного рішення [6].

У генераторі – CNN може використовуватися для перетворення випадкового шуму або іншого вхідного сигналу в синтетичні зображення, які повинні бути невідмінними від реальних. У той самий час, у дискримінаторі – CNN можна використовувати класифікації між синтетичними і реальними зображеннями [7].

На рисунку 1.3 наведено базова архітектура згорткової нейронної мережі.

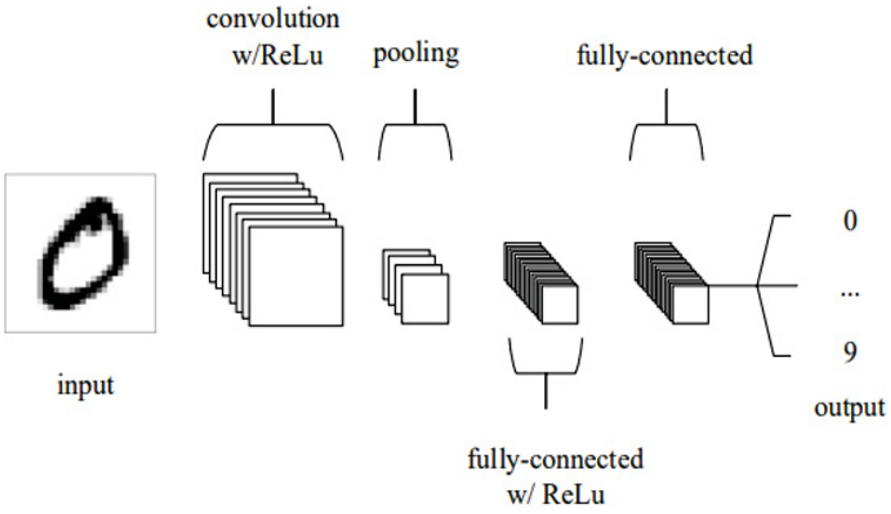


Рисунок 1.3 – Базова архітектура згорткової нейронної мережі

1.3.4 Архітектура генератора

Генератор – це ключовий компонент у генеративно-змагальній мережі. У контексті GAN, генератор діє протилежно до звичайних згорткових нейронних мереж (CNN), де зображення подається як вхід, а класифікація видається як результат. Натомість, у генератора випадковий шум стає вхідним сигналом, а саме зображення – вихідним. Його головна функція – створювати нові дані, спираючись на власну уяву.

Архітектура генератора складається з декількох компонентів, включаючи прихований простір, сам генератор та процес генерації зображення. Генератор працює наступним чином – спочатку, він виконує випадкову вибірку з прихованого простору, а потім використовує нейронну мережу для перетворення цієї вибірки у вихідні дані, що представляють собою нові зображення (рисунок 1.4).

Під час навчання модель генератора взаємодіє з дискримінатором, і разом вони утворюють змагальну систему, де генератор старається створити такі вихідні дані, щоб дискримінатор не міг відрізнити їх від реальних

зображень. Після завершення навчання модель генератора залишається для подальшого використання, і вона може генерувати нові зразки, що мають схожі характеристики з вихідними даними.

Використовуючи випадковий вектор фіксованої довжини як вхід, генератор створює нові вибірки в області даних. Цей вектор, зазвичай взятий з розподілу Гауса або інші альтернативні шуми та відображається на точки в прихованому просторі. Прихований простір представляє собою компактне уявлення розподілу даних і дозволяє моделі генератора створювати нові дані, що мають аналогічні характеристики з даними навчання.

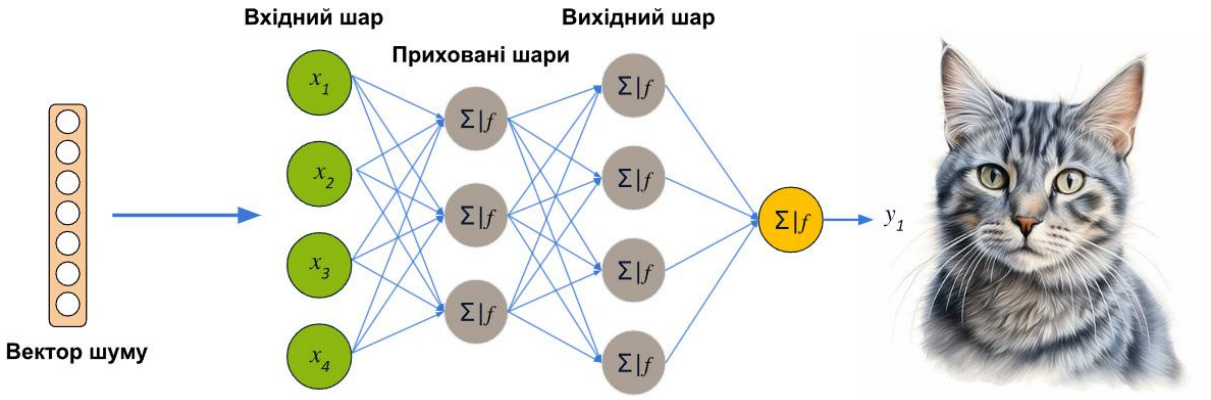


Рисунок 1.4 – Схема роботи генератора

У підсумку, генератор у системі GAN є ключовим інструментом для створення нових даних на основі навчального набору, що дозволяє моделі створювати реалістичні вихідні дані після завершення навчання.

1.3.5 Архітектура дискримінатора

Цей аспект GAN можна порівняти з тим, що робить звичайна згорткова нейронна мережа (CNN). Дискримінатор у GAN також є згортковою нейронною мережею, яка складається з численних прихованих

шарів та одного вихідного. Основна відмінність полягає у тому, що вихідний шар GAN має лише два виходи, відмінно від CNN, які можуть мати виходи в залежності від кількості класів, на які вони навчені. Вихід дискримінатора може приймати значення або 1, або 0 через певну функцію активації, що дозволяє визначити, чи є вхідні дані реальними чи сгенерованими. Дискримінатор навчається на реальних даних з метою визначення їхнього характеру та створення правил для класифікації.

Архітектура дискримінатора відповідає за розрізнення між реальними та сгенерованими зображеннями. На рисунку 1.5 наведено основні компоненти архітектури дискримінатора.

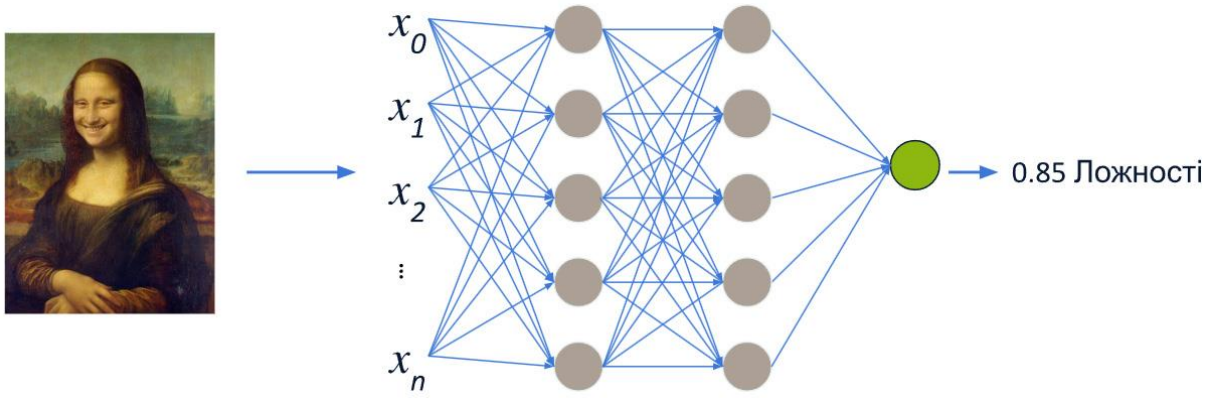


Рисунок 1.5 – Схема роботи дискримінатора

Для побудови дискримінатора потрібно пройти кілька кроків:

– спочатку створюється згорткова нейронна мережа для бінарної класифікації реальних та згенерованих даних;

– потім генерується набір реальних даних, а також сгенерований генератором;

– далі модель дискримінатора навчається на цих наборах;

– останнім кроком є поєднання процесу навчання дискримінатора з навчанням генератора – генератор підлаштовується на виході дискримінатора.

Дискримінатор може використовувати всі корисні функції, що характерні для моделей класифікації, і діяти як адаптивна функція втрат для GAN загалом. Це означає, що дискримінатор може адаптуватися до основного розподілу даних, що є однією з переваг сучасних моделей глибокого навчання. Завдяки цій адаптивності дискримінатор оцінює виведення як реальних, так і згенерованих зображень, що допомагає покращити якість генерованих даних.

1.3.6 Функція втрат

Раніше було розглянуто, як працює GAN нейронна мережа і тепер розглянемо функцію втрат, що використовується для навчання GAN. Ця функція втрат дуже важлива і допомагає тренувати компоненти GAN, надаючи зворотний зв'язок для покращення ваги генератора та дискримінатора. Стандартна функція втрат, розглянута у статті [1] і називається функцією мінімально-максимальних втрат, як показано нижче.

Втрати дискримінатора (D_loss):

$$L_D = -\mathbb{E}_x(log(D(x))) + \mathbb{E}_z(log(1 - D(G(z))). \quad (1.1)$$

Втрати генератора (G_loss):

$$L_G = -\mathbb{E}_z log(D(G(z)))). \quad (1.2)$$

Функція втрат (Binary Cross Entropy (BCE) Loss) складається з наступних частин:

− оцінка дискримінатора ймовірності того, що реальний екземпляр даних x є дійсним, визначається функцією $D(x)$;

− індекс $\mathbb{E}_x$ є очікуваним значенням для всіх екземплярів реальних даних;

− коли подано шум z, генератор виводить $G(z)$ як екземпляр даних;

− дискримінаційна оцінка ймовірності фальшивого екземпляра даних із шумом, оскільки z дійсне, визначається функцією $D(G(z))$;

− індекс $\mathbb{E}_z$ – це очікуване значення (розподіл латентного вектора) для всіх створених помилкових екземплярів $G(z)$ для всіх випадкових вхідних даних генератора;

− рівняння є результатом перехресної ентропії між дійсним і згенерованим розподілами.

Генератор намагається мінімізувати цю функцію втрат, а дискримінатор максимізувати. Для генератора мінімізація функції втрат аналогічна мінімізації $log(1 - D(G(z))$ оскільки генератор не може безпосередньо впливати на член $log(D(x))$ у функції. $log(D(x))$ відноситься до ймовірності того, що генератор правильно класифікує реальне зображення. Щоб точно маркувати підроблене зображення, створене генератором, необхідно максимізувати $log(1 - D(G(z)))$.

У генеративно-змагальних мережах функція втрати може дорівнювати 0 тільки у двох специфічних випадках коли є ідеально навчений генератор або якась помилка при навчанні яка призводить до нульового значення.

Основними проблемами функції втрат, які хотілося б взяти до уваги, є згортання моделі (model collapse) та зникнення градієнтів (vanishing gradients).

Основна мета GAN – генерувати різні типи вмісту, наприклад зображення, для різноманітного шуму, що подається в генератор. З іншого боку, генератор може навчитися створювати такий результат, лише якщо він забезпечує дуже правдоподібний результат. По правді кажучи, генератор завжди шукає рішення, яке дискримінатор вважає найбільш правдоподібним. Як наслідок, відхилення вихідного сигналу генератора завжди є найкращим варіантом для дискримінатора [8]. Однак, якщо наступна ітерація дискримінатора потрапить у локальний мінімум і не

зможе вирватися шляхом подальшої оптимізації своїх ваг, наступному генератору буде легко ідентифікувати найбільш вірогідний результат поточного дискримінатора. У результаті він продовжуватиме забезпечувати ті самі результати, ігноруючи додаткове навчання. Це може призвести до згортання моделі, коли подальше навчання перестає покращувати результати, і модель стає менш ефективною у генерації нових та реалістичних зразків.

Зникаючі градієнти відбувається, коли дискримінатор працює помітно краще, ніж генератор. Або оновлення дискримінатора неправильні, або вони зникають [9]. Одне із запропонованих пояснень цього полягає в тому, що генератор зазнає суворих штрафів, які призводять до того, що функція після активації значення стає насиченою, а градієнт зрештою зникає.

1.4 Аналіз існуючих рішень

Протягом останніх років було розроблено різноманітні підходи та моделі, які використовують глибоке навчання для створення, включаючи генеративно-змагальні мережі (GAN), моделі на основі потоків (Flow-based models), варіаційні автокодувальники (VAE) та дифузійні моделі (DM).

1.4.1 Оригінальна немодифікована генеративно-змагальна мережа

Оригінальна немодифікована генеративно-змагальна мережа (Vanilla GAN) – це найпростіший тип GAN, який складається з генератора та дискримінатора. Причому генератор та дискримінатор виконують внутрішню класифікацію та генерацію зображень з використанням багатошарових перцептронів. Генератор записує розподіл даних. Тим часом дискримінатор намагається визначити ймовірність того, що вхідні дані належать даному класу. Нарешті, після обчислення функції втрат зворотний зв'язок відправляється генератору та дискримінатору, і таким чином

намагається мінімізувати втрати. На рисунках 1.6 и 1.7 показано зворотне поширення (backpropagation) для дискримінатора та генератора.

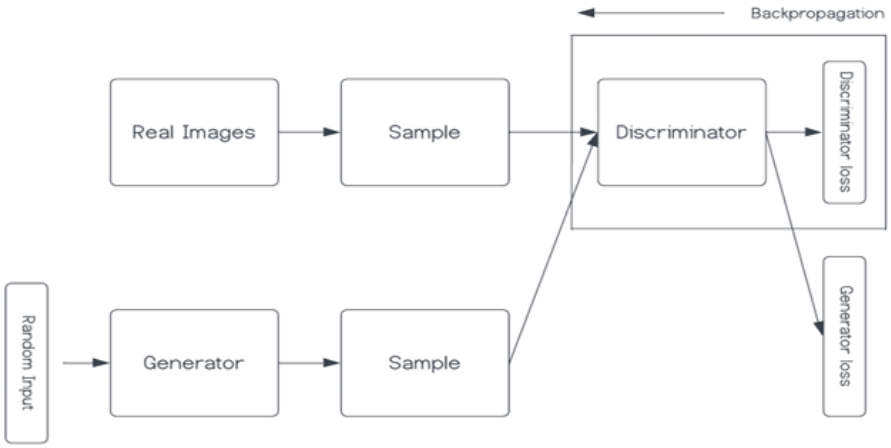


Рисунок 1.6 – Зворотне поширення в дискримінаторі з використанням дискримінаторних втрат

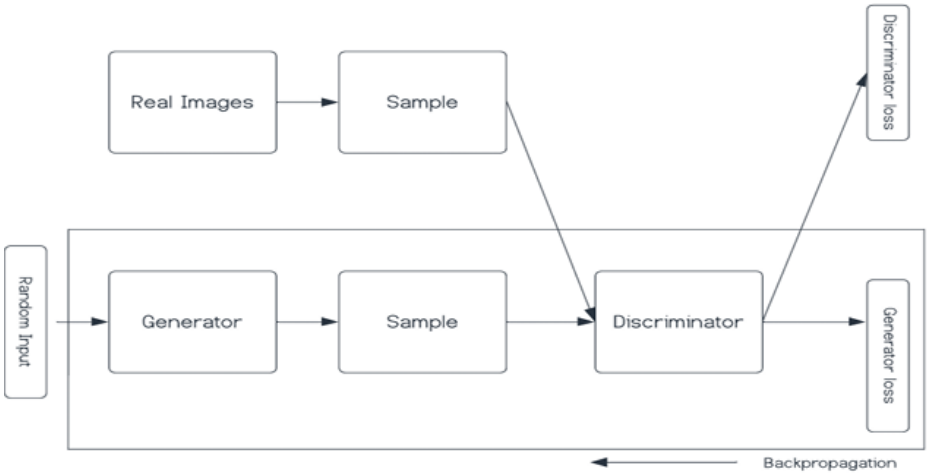


Рисунок 1.7 – Зворотне поширення в генераторі з використанням втрат генератора

Оригінальна генеративно-змагальна мережа (Vanilla GAN) є базовою моделлю, яка дала поштовх до розвитку багатьох інших варіацій і

вдосконалень генеративно-змагальних мереж. Вона заклала фундамент для подальших досліджень і застосувань у сфері генеративного навчання.

1.4.2 Варіаційні автокодувальники

Автокодер (Autoencoder) – це тип нейронної мережі, який навчається відтворювати вхідні дані через навчання функції ідентичності в неконтрольований спосіб. Він стискає дані в процесі, щоб отримати більш ефективне та компактне представлення. Ця концепція виникла ще у 1980-х роках і була подальше розвинена у працях Дж. Хінтона та Р. Салахутдінова.

Автокодер складається з двох основних частин – кодера і декодера (рисунок 1.8). Кодер перетворює вхідні дані високої розмірності у скорочений код низької розмірності, тоді як декодер відновлює дані з цього коду, зазвичай збільшуючи їхню розмірність. Мережа кодера забезпечує зменшення розмірності, схоже на те, як працюють методи аналізу головних компонент або матрична факторизація. Більше того, автокодер прямо оптимізований для відтворення даних з їхнього кодування. Вдале проміжне представлення може не лише ухопити латентні змінні, але також покращити процес декомпресії [5].

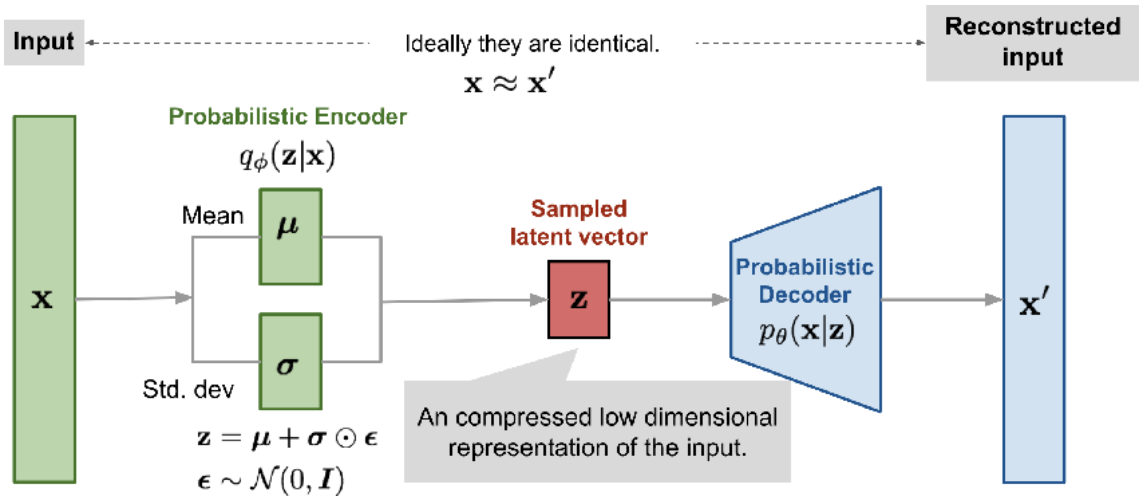


Рисунок 1.8 – Архітектура авто-кодувальника

Подібно до звичайного автокодера, варіаційний автокодер (VAE) – це архітектура, яка складається з кодувальника та декодувальника і навчається мінімізувати похибку відтворення між закодованими та відновленими даними та вхідними даними. Однак основна відмінність полягає в тому, що замість того, щоб кодувати вхідні дані як одну точку, вони кодуються як розподіл у латентному просторі (рисунок 1.9).

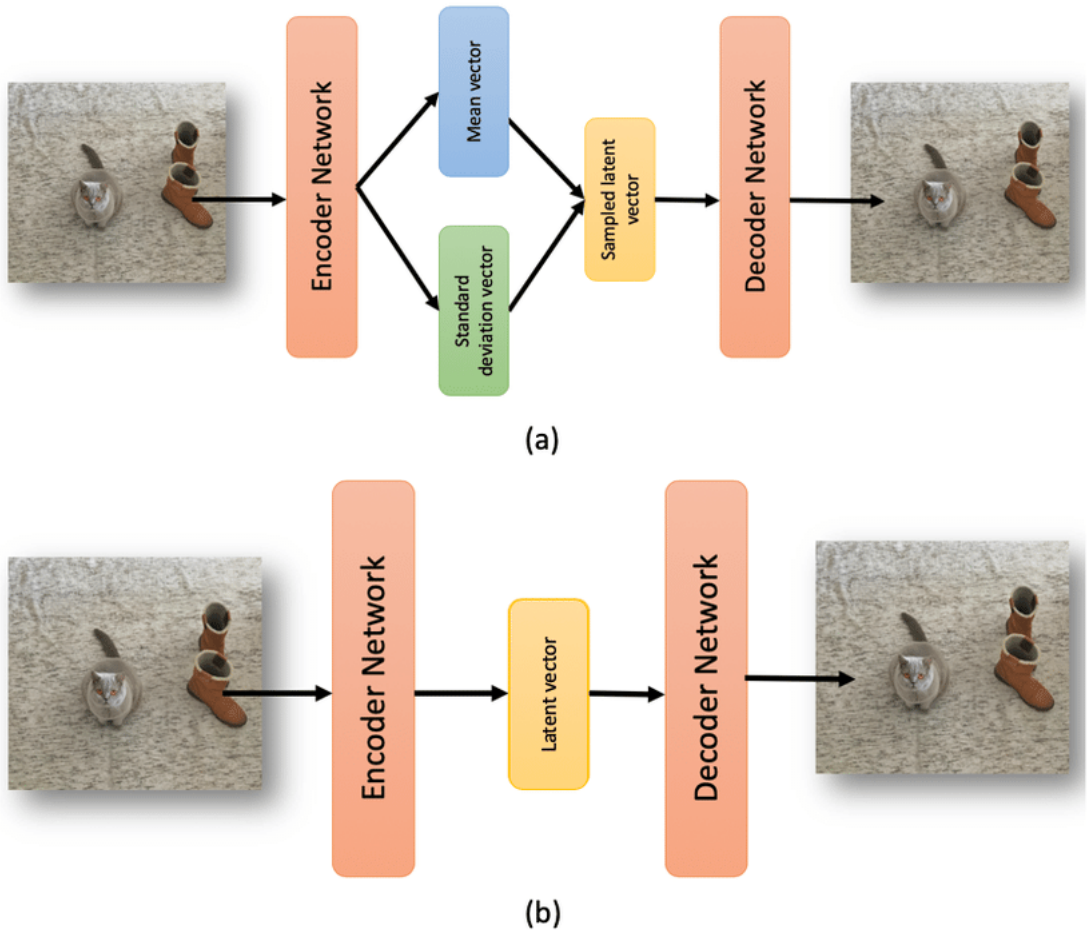


Рисунок 1.9 – Різниця між варіаційним автокодувальником (імовірнісним) та автокодувальником (детермінованим)

Модель навчається за такими кроками:

– спочатку вхідні дані кодуються у вигляді розподілу у латентному просторі;

– потім з цього розподілу вибирається точка у латентному просторі;

− обрана точка декодується, і обчислюється похибка реконструкції;

− похибка реконструкції розповсюджується через мережу.

На практиці, закодовані розподіли часто вибираються як нормальні, щоб кодувальник міг навчитися повертати середнє значення та матрицю коваріацій, які описують ці гаусіани. Причина, чому вхідні дані кодуються у вигляді розподілу з деякою дисперсією, а не як одна точка, полягає в тому, що це дозволяє дуже природно виразити латентну регуляризацію простору. Розподіли, які повертає кодувальник, повинні бути близькими до стандартного нормального розподілу.

Обмеження на латентний простір дозволяють отримувати вибірки, близькі до початкового розподілу даних, шляхом простої випадкової вибірки з латентного простору. Хоча VAE часто демонструють високі значення логарифмічної правдоподібності, важко досягти створення нерозмитих вибірок високої якості [10].

1.4.3 Генеративні моделі на основі потоку

Генеративні моделі на основі потоку представляють собою родину моделей, що складаються з послідовності інвертованих параметризованих функцій (рисунок 1.10). Основна ідея полягає в тому, щоб застосовувати ці інвертовані функції до вихідних даних з попередніх шарів для обчислення точної логарифмічної правдоподібності спостережень [6].

У порівнянні з GAN та VAE, генеративні моделі на основі потоку безпосередньо спрямовані на максимізацію точної логарифмічної правдоподібності (log-likelihood). Проте їхній успіх у генерації якісних зразків часто обумовлений складністю пошуку ефективних інвертованих архітектур.

Існують дві основні категорії потокових моделей – моделі з нормалізуючими потоками та моделі з авторегресійними потоками, кожна з яких спрямована на покращення продуктивності базової моделі. Моделі з

нормалізуючими потоками забезпечують наближення розподілу, застосовуючи послідовність інвертованих функцій перетворення. Авторегресійні потоки, у свою чергу, передбачають, що кожен вимір векторної змінної залежить від попередніх вимірів.

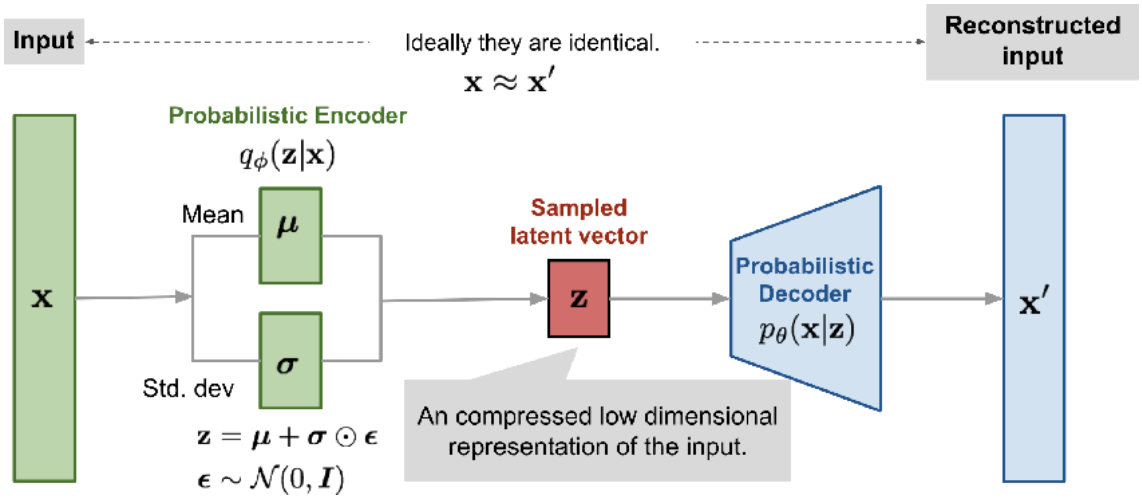


Рисунок 1.10 – Архітектура генеративних моделей на основі потоку

Хоча потокові моделі привабливі для моделювання складних розподілів, вони обмежені проблемами оцінювання щільності порівняно з авторегресійними моделями. Крім того, є значний розрив у обчислювальних витратах на навчання – потокові моделі вимагають більше часу для генерації зображень порівняно з GAN [11].

1.4.4 Дифузійні моделі

Дифузійні моделі (Diffusion Models, DM), відомі також як дифузійні ймовірнісні моделі, представляють собою сімейство генеративних моделей, які базуються на ланцюгах Маркова та навчаються за допомогою варіаційного висновку. Основна мета навчання DM полягає в тому, щоб моделювати процес дифузії даних зі шумом, з метою генерації нових вибірок [12].

Перші дифузійні моделі були запропоновані у 2015 році, вони базувалися на підході до моделювання молекулярних систем, відомому як динаміка Ланжевена. З тих пір вони стали значною альтернативою для традиційних генеративних моделей, таких як GAN або VAE, здобувши визнання в різних областях, включаючи перетворення тексту в зображення. Основна концепція полягає в створенні марковського ланцюжка, що поступово додає випадковий шум до даних, а потім вчиться відновлювати процес дифузії для генерації бажаних даних з шуму (рисунок 1.11).

Рисунок 1.11 − Принцип дифузних моделей із реверсом

Однією з найбільш значущих досягнень стала імовірнісна дифузійна модель з усуненням шуму (DDPM – Denoising Diffusion Probabilistic Model), яка з'явилася у 2020 році. Ця модель викликала великий інтерес в генеративній спільноті завдяки своїм високим результатам. DDPM визначається як параметризований марковський ланцюжок, який генерує зображення зі шумом в межах обмежених переходів під час виведення. Під час навчання ядра переходів вивчаються у зворотному напрямку, що дозволяє моделі ефективно видаляти шум із зображень [13].

Модель латентної дифузії (LDM) – це інша цікава розробка, яка запускає процес дифузії у латентному просторі, а не в піксельному, що дозволяє зменшити витрати на навчання та підвищити швидкість генерації [14].

Дифузійні моделі продовжують активно розвиватися і досліджуватися, привертаючи увагу дослідників у галузі генеративних

моделей, і вони вже показали великий потенціал у вирішенні різних завдань генерації та обробки даних.

1.4.5 Авторегресійні моделі

Авторегресійні моделі з’явилися як потужні інструменти для завдань генерації зображень, пропонуючи гнучку структуру для захоплення складних просторових залежностей у зображеннях. На відміну від традиційних генеративних моделей, які намагаються змоделювати весь розподіл ймовірностей зображення одночасно, авторегресійні моделі розкладають процес створення зображення на серію послідовних кроків, що дозволяє створювати високоякісні реалістичні зображення [15].

В основі авторегресійних моделей для створення зображень лежить концепція умовної ймовірності. Ці моделі вивчають умовний розподіл кожного пікселя з урахуванням його попередніх пікселів, що дозволяє їм фіксувати складні просторові залежності та кореляції на рівні пікселів у зображеннях. Ітеративно прогнозуючи кожен піксель на основі його контексту, авторегресійні моделі можуть генерувати зображення з дрібними деталями та когерентними структурами (рисунок 1.12).

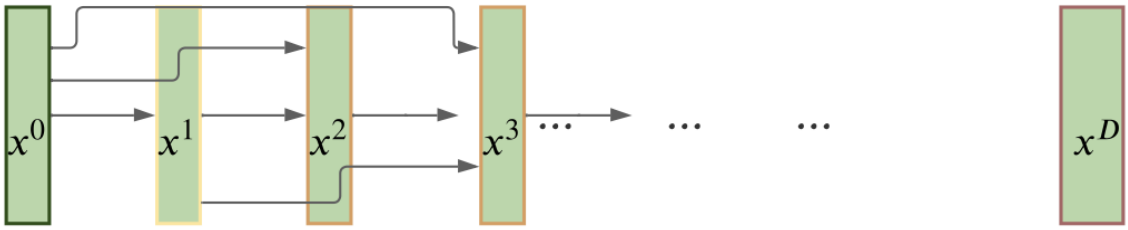


Рисунок 1.12 − Принцип авторегресійних моделей

Однією з піонерських моделей авторегресії для створення зображень є архітектура PixelCNN (Pixel Convolutional Neural Network), представлена ван ден Оордом та ін. у 2016 році. PixelCNN використовує замасковану

згортову нейронну мережу, щоб передбачити значення кольору кожного пікселя на основі попередньо згенерованих пікселів, забезпечуючи причинно-наслідковий зв’язок і запобігаючи витоку інформації з майбутніх пікселів. Укладаючи кілька шарів замаскованих звивин, PixelCNN може ефективно фіксувати ієрархічні особливості та створювати зображення високої роздільної здатності з надзвичайною точністю.

Спираючись на успіх PixelCNN, були запропоновані наступні варіанти та розширення для подальшого розширення можливостей авторегресійних моделей для створення зображень. PixelRNN розширює підхід авторегресійного моделювання до рекурентних нейронних мереж, уможливлюючи моделювання піксельних залежностей як у горизонтальному, так і у вертикальному напрямках. Крім того, архітектура WaveNet вводить розширені згортки, що дозволяє збільшити сприйнятливе поле без значного збільшення обчислювальної складності.

Іншим помітним прогресом у створенні авторегресійних зображень є використання механізмів уваги. Такі моделі, як авторегресійний трансформатор, використовують механізми самоконтролю для захоплення залежностей на великій відстані та глобального контексту в зображеннях, що призводить до покращення якості генерації та когерентності.

Незважаючи на свій успіх, авторегресійні моделі для створення зображень стикаються з проблемами при створенні зображень високої роздільної здатності через послідовний характер прогнозування пікселів і обчислювальних витрат, пов’язаних з обробкою великих зображень. Проте триваючі дослідницькі зусилля спрямовані на вирішення цих проблем шляхом вивчення методів паралелізації, підходів ієрархічного моделювання та нових архітектур, розроблених для ефективного створення авторегресійних зображень [15].

Підсумовуючи, авторегресійні моделі представляють багатообіцяючий підхід для генерації зображень, пропонуючи принципову основу для захоплення складних просторових залежностей і генерації

реалістичних зображень. Завдяки постійному вдосконаленню архітектури моделей і методів навчання авторегресійні моделі мають великий потенціал для розширення можливостей створення зображень у різних програмах, включаючи комп’ютерне бачення, графіку та творчий штучний інтелект.

1.5 Еволюція моделей генерацій зображень за текстовим описом

Концепція створення зображень за описом спочатку виникла у роботі alignDRAW. Проте якість отриманих зображень була дуже низькою, і згенеровані сцени та об'єкти мало було впізнати. Наступні п'ять років призвели лише до невеликих поліпшень, що були обумовлені розвитком технологій GAN (рисунок 1.13). Моделі, такі як Attentional Generative Adversarial Network (AttnGAN), Dynamic Memory Generative Adversarial Networks (DM-GAN) та Deep Fusion Generative Adversarial Network (DF-GAN), почали відображати частину контенту підписів, але зображення залишалися малореалістичними, за винятком деяких обмежених та простих наборів даних [5].

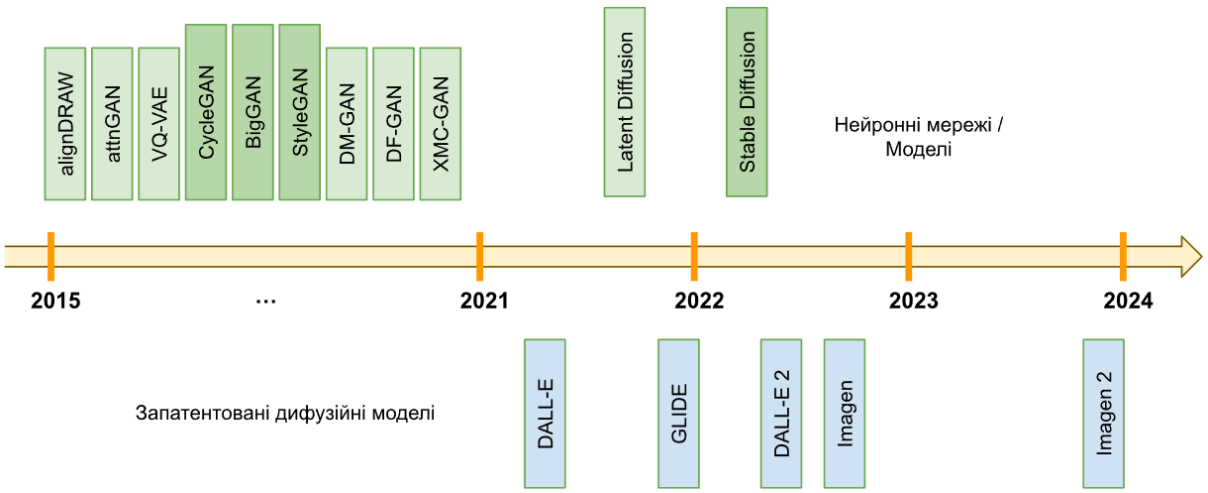


Рисунок 1.13 – Хронологія генеративних моделей «текст-в-зображення»

Введення контрастного навчання в процес і збільшення розміру набору даних сприяло створенню кращих зображень з більш чіткими сценами за допомогою Cross-Modal Contrastive Generative Adversarial Network (XMC-GAN) [16]. Також автори роботи DALL-E довели, що подальше збільшення обсягу набору даних до 250 мільйонів пар зображення-текст може дозволити навчання з «нульового пострілу» (zero-shot learning). Це означає, що модель DALL-E здатна поєднувати різні об'єкти, концепції та місця для створення незвичайних зображень, наприклад, крісла з авокадо. Проте DALL-E не використовує GAN, але замість цього використовує Vector Quantised-Variational AutoEncoder (VQ-VAE) [17] та два трансформери. Незважаючи на вражаючі результати DALL-E, це лише початок для нової категорії генеративних моделей.

Того ж року, коли було опубліковано роботу DALL-E, у роботі було показано, що дифузійні моделі можуть перевершити GAN у створенні зображень, обумовлених класом. Пізніше, у роботі про Guided Language to Image Diffusion for Generation and Editing (GLIDE) [18] було вперше застосовано дифузійні моделі для синтезу текст-зображення, отримуючи зображення вищої якості, ніж DALL-E, при навчанні на тому ж наборі даних. Автори GLIDE використовували мовну модель трансформеру для вбудовування текстових описів зображень, а потім дифузійні моделі для отримання зображень роздільною здатністю 256×256.

DALL-E 2 суттєво відрізняється від своєї попередньої версії. Так само, як і GLIDE, він використовує вкладання для створення зображень, але вкладання надходять від Contrastive Language-Image Pre-Training (CLIP), моделі мови зору, яка вивчає текстово-зображувальні представлення. DALL-E 2 використовує попередньо навчені кодувальники CLIP та вчить попередню модель для перетворення текстових вкладень CLIP на вкладення CLIP-зображень. Вони також використовують моделі каскадної дифузії для підвищення дискретності зображень з 64 × 64 до 1024 × 1024 [19].

Навпаки, Imagen [20] від Google схожий на GLIDE, але відрізняється тим, що автори не навчають мовну модель з нуля. Замість цього вони повторно використовують велику заморожену модель трансформера, яка була навчена лише на текстовому корпусі, відомому як модель T5 [21]. Вони продемонстрували, що збільшення розміру мовної моделі призводить до більших покращень, ніж збільшення розміру моделі дифузії. Автори стверджують, що їхня модель перевершує DALL-E 2, оскільки їхній підхід отримав кращий рейтинг продуктивності за критерієм оцінки зображень на валідаційному наборі даних Microsoft Common Objects in Context (MS-COCO).

1.6 Постановка задачі дослідження

Дана дослідницька робота буде спрямована на вивчення рішення традиційного завдання для GAN-мереж – генерація зразків даних, наприклад зображень. Оскільки процес навчання GAN-мереж часто стикається з проблемою генерації якісних шумів, то будуть розглянуті методи генерації шумів, які необхідні для різноманітності та стабільності навчання нейронної мережі.

Завдання кваліфікаційної роботи є теоретичне та практичне порівняння генеративно-змагальних нейронних мереж для завдання генерації зображень з текстового опису з різних сторін. Одним із варіантів є дослідження якості та можливостей досліджуваних моделей, в умовах різних підходів при генерації вхідних шумів.

Для успішного виконання даної роботи потрібно:

– детально вивчити наукові джерела та інформацію в Інтернеті, що стосується генеративних моделей;

– описати математичні основи кожної моделі;

– розглянути підходи формування вхідних шумів;

– створити та навчити моделі на однаковому наборі даних, використовуючи однакові параметри;

– провести аналіз отриманих результатів для кожної моделі;

– сформулювати висновки щодо порівняльного аналізу різних моделей.

2 ОГЛЯД ГЕНЕРАТИВНИХ МОДЕЛЕЙ

2.1 Генеративно-змагальні мережі

2.1.1 Базова генеративно-змагальна мережа

Концепція генеративно-змагальних мереж вже було розглянуто в першому розділі та в даній секції буде розглянута більш детально.

GAN нейронна мережа складається з двох взаємодіючих компонентів: генератора G та дискримінатора D. Вони беруть участь у міні-максній грі для двох гравців. Мета дискримінатора полягає в тому, щоб відрізняти справжні навчальні дані від штучно створених зображень, тоді як генератор прагне обдурити дискримінатор, генеруючи фальшиві зображення.

Дискримінатор і генератор грають у гру на $V(D, G)$ як наведено нижче:

$$min_G max_D\, V(D, G) = \\ = \mathbb{E}_{x \sim P_{data}(x)}[log(D(x))] + \mathbb{E}_{z \sim P_z(z)}[log(1 - D(G(z)))], \quad (2.1)$$

де x – це реальне зображення з навчального набору даних;

z – випадковий шумовий вектор, який використовує генератор;

$D(x)$ – ймовірність того, що дискримінатор вважає реальне зображення з навчального набору дійсно реальним;

$D(G(z))$ – ймовірність того, що дискримінатор вважає згенероване зображення реальним;

$\mathbb{E}$ – математичне очікування обчислюється як середнє значення випадкової величини.

Генератор G неявно визначає розподіл ймовірностей p_g як розподіл вибірок $G(z)$, отриманих при $z \sim p_z$. Як видно з формули, глобальний оптимум досягається, коли p_g збігається з p_{data}. Крім того, якщо G і D

мають достатню потужність, p_g збігається до p_{data} за умови виконання певних умов. На початковому етапі навчання дискримінатор схильний з високою впевненістю відхиляти погані зразки від генератора. Практика показала, що максимізація $log(D(G(z)))$ дає кращі результати для генератора порівняно з мінімізацією $log(1 - D(G(z)))$.

Для навчання GAN необхідно виконати кілька навчальних ітерацій, під час яких по черзі навчається дискримінатор, а потім генератор. Щоб навчити дискримінатор, для кожної ітерації ми маємо деякі реальні дані.

На кожному кроці з попереднього розподілу шуму $p_g(z)$ вибирається пакет з m зразків шуму $\{z(1), \dots, z(m)\}$, а інший міні-пакет з m реальних прикладів $\{x(1), \dots, x(m)\}$ береться з розподілу даних $p_{data}(x)$. Дискримінатор оновлюється шляхом збільшення його стохастичного градієнта, що передбачає обчислення градієнта функції втрат відносно його параметрів θ_d. Параметри оновлюються за градієнтом висхідним шляхом вказані у формулі 2.2:

$$max_D V(D;G) = \frac{1}{m}\sum_{i=1}^{m}[logD(x_i) + log(1 - D(G(z_i)))]. \qquad (2.2)$$

Перший доданок є логарифмом виходу дискримінатора на реальних навчальних прикладах, а другий – виходом дискримінатора на згенерованих генератором синтетичних даних.

Це також можна оптимізувати за допомогою існуючої двійкової перехресної втрати ентропії, яка зазвичай доступна в більшості систем глибокого навчання, встановивши цільові ймовірності для $x_0, x_1, \dots, x_m$ як 1.0 і цільові ймовірності для $G(z_0), G(z_1), \dots, G(z_m)$ як 0.0, де 1.0 означає, що дані були зі справжнього набору, а 0.0 означає, що дані були підробленими або зі згенерованого набору.

Після оновлення дискримінатора вибирається ще одна партія з m відліків шуму $G(z_0), G(z_1), \ldots, G(z_m)$, де $z_0, z_1, \ldots, z_m \sim p_z$ і генератор оновлюється за спаданням стохастичного градієнта вказані у формулі 2.3:

$$min_G V(G; D) = \frac{1}{m} \sum_{i=1}^{m} [log(1 - D(G(z_i)))]. \quad (2.3)$$

Це також можна оптимізувати, використовуючи наявні двійкові перехресні втрати ентропії, встановивши цільові ймовірності для $G(z_0), G(z_1), \ldots, G(z_m)$ як 1.0.

2.1.2 Додавання умови до генерації в GAN (CGAN)

Умовні генеративно-змагальні мережі (Conditional GAN, CGAN) є розширенням оригінальних генеративно-змагальних мереж, які дозволяють контролювати процес генерації шляхом введення додаткової інформації, яка може бути міткою класу або іншими даними моделі. Ця додаткова інформація підтримує дискримінатор у визначенні умовної ймовірності, а не спільної ймовірності. Цей підхід був запропонований для того, щоб зробити процес генерації більш керованим та цілеспрямованим, надаючи можливість задавати конкретні атрибути або властивості бажаних зразків даних.

Архітектура CGAN включає в себе дві основні мережі, як і в оригінальних GAN, але з додаванням умовної інформації на вході як до генератора, так і до дискримінатора.

На рисунку 2.1 наведена архітектура моделі Conditional GAN, де:

– G – позначає генератор;

– D – позначає дискримінатор;

– z – представляє вектор шуму;

– $y(z)$ – представляє функцію, яка може бути або міткою класу, або будь-якими даними моделі;

– $G(z)$ – є функцією, яка є результатом моделі генератора, по суті, це підроблений зразок.

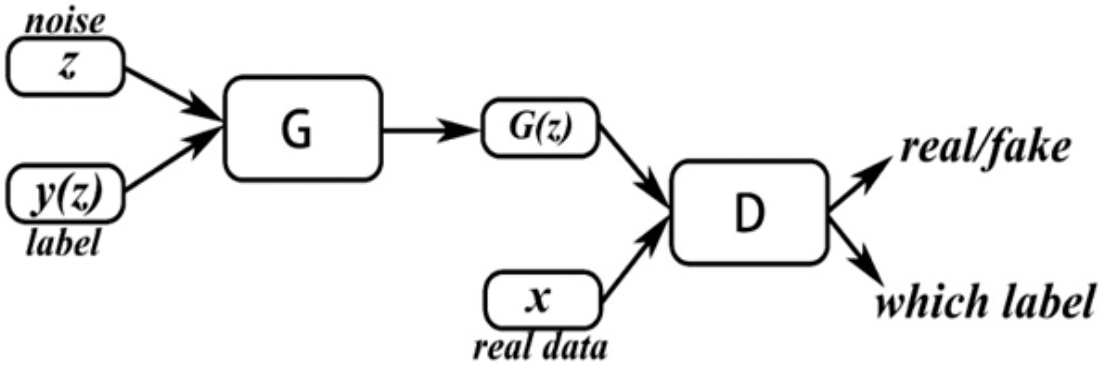


Рисунок 2.1 – Архітектура моделі Conditional GAN

CGAN може бути використаний для різних задач, таких як перетворення тексту на зображення або зображення на текст. Наприклад, навчання на парах текстів і зображень дозволяє CGAN створювати реалістичні зображення на основі текстових описів. Це відкриває широкий спектр застосувань, таких як передача стилю, додавання кольору до чорно-білих зображень, або зміна параметрів зображення, таких як пора року або час доби.

Крім того, CGAN може бути корисним у медицині для створення синтетичних зображень медичних сканів на основі текстових описів або даних пацієнтів, що допомагає в навчанні моделей діагностики. У сфері розваг CGAN можна використовувати для створення анімацій або персонажів, які відповідають певним описам. Завдяки умовній природі CGAN, моделі стають більш гнучкими та здатними враховувати додаткову інформацію, що підвищує якість та реалістичність згенерованих даних.

2.1.3 Генеративна змагальна мережа з глибокими згортками (DCGAN)

Генеративна змагальна мережа з глибокими згортками (Deep Convolutional GAN, DCGAN) є одним із найвідоміших і найпопулярніших варіантів генеративно-змагальних мереж, запропонованим Алеком Редфордом, Лукасом Метцером та Суміт Чінталою у 2015 році [22]. DCGAN використовує глибокі згорткові нейронні мережі (CNN) для покращення якості згенерованих зразків та стабільності навчання.

Архітектура DCGAN будується на базі традиційних GAN, але з декількома ключовими відмінностями, які роблять її більш потужною для роботи з зображеннями. Основними компонентами DCGAN є генератор та дискримінатор, які використовують згорткові нейронні мережі.

Шари генератора складаються з наступних елементів:

– генератор у DCGAN використовує транспоновані (або зворотні) згорткові шари (transposed convolutional layers), також відомі як дезактиваційні шари (deconvolutional layers). Ці шари дозволяють збільшувати розмір вхідного латентного вектора, перетворюючи його в зображення високої роздільної здатності;

– кожен транспонований згортковий шар зазвичай супроводжується нормалізацією партій (Batch Normalization) і нелінійною активаційною функцією ReLU (Rectified Linear Unit), за винятком останнього шару, де зазвичай використовується активація tanh (гіперболічний тангенс).

Архітектура генератора DCGAN може виглядати наступним чином:

– вхідний латентний вектор (звичайно з розподілу нормального шуму);

– повнозв'язний шар, перетворюючий вектор у багатовимірний тензор;

– кілька транспонованих згорткових шарів з нормалізацією партій та активацією ReLU;

– остаточний транспонований згортковий шар з активацією tanh для створення зображення.

На рисунку 2.2 показано архітектуру мережі генератора, где DCGAN перетворює вектор шуму розміром 100x1, позначений z, у вихідний сигнал $G(z)$ розміром 32x32x3. Якщо потрібно отримати більше розмір згенерованого зображення, додаткові внутрішні шари для масштабування зображення буде потрібно додати. Випадковий шум буде масштабовано за допомогою функції Conv2DTranspose, що виконує векторне перетворення в напрямку, протилежному звичайному пакунку.

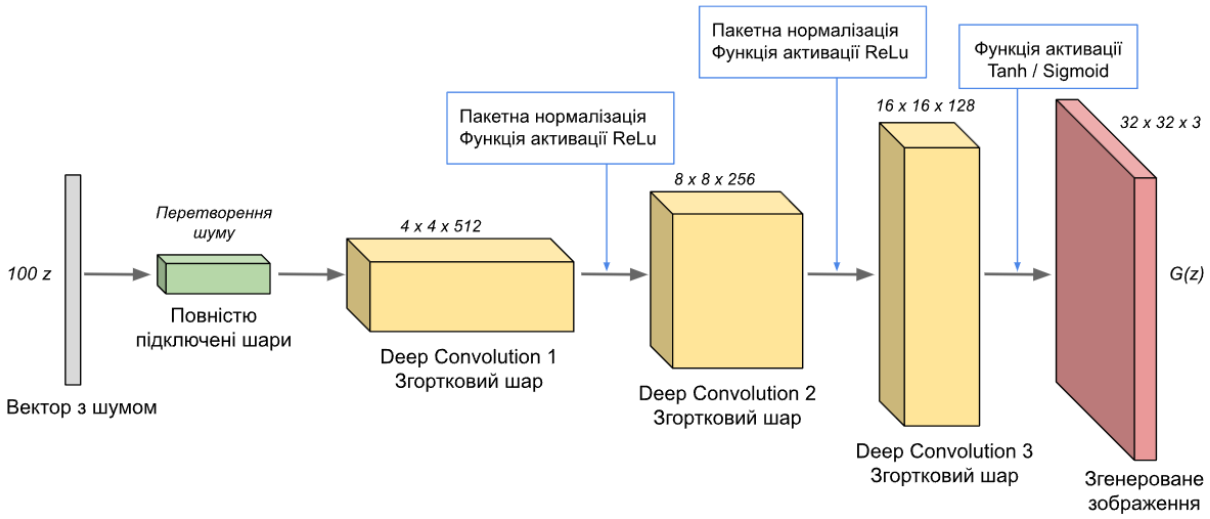


Рисунок 2.2 – Архітектура мережі з глибокими згортками (DCGAN)

Шари дискримінатора складаються з наступних елементів:

– дискримінатор у DCGAN використовує звичайні згорткові шари для обробки зображень, вивчаючи їх особливості та визначаючи, чи є зображення реальним чи згенерованим;

– замість нормалізації партій у дискримінаторі часто використовуються шари відсіву (dropout layers) для запобігання перенавчанню;

– активаційна функція Leaky ReLU використовується після кожного згорткового шару, за винятком останнього шару, де зазвичай використовується сигмоїдна активація.

Архітектура дискримінатора DCGAN може виглядати наступним чином:

– вхідне зображення;

– кілька згорткових шарів з активацією Leaky ReLU;

– шари відсіву для запобігання перенавчанню;

– остаточний згортковий шар з сигмоїдною активацією для видачі ймовірності, що зображення є реальним.

DCGAN має кілька переваг у порівнянні з класичними GAN, що робить їх особливо корисними для задач, пов'язаних з обробкою зображень:

− покращена стабільність навчання – використання нормалізації партій та спеціальних активаційних функцій сприяє стабільнішому і швидшому навчанню моделей;

− краща якість зображень – згорткові шари ефективно вивчають просторові залежності у зображеннях, що призводить до створення більш реалістичних і чітких зображень;

− більш глибока архітектура – DCGAN дозволяють створювати глибші архітектури з більшою кількістю параметрів, що підвищує виразну здатність моделі.

2.1.4 Стекова генеративна змагальна мережа (StackGAN)

Стекова генеративна змагальна мережа (StackGAN) є просунутою архітектурою генеративно-змагальних мереж, яка була запропонована для генерації високоякісних зображень з текстових описів. Основна ідея StackGAN полягає в поетапній генерації зображень, де кожен етап генерує зображення з поступово покращуваною роздільною здатністю та деталізацією [23].

StackGAN складається з двох основних етапів генерації: Stage-I і Stage-II. Кожен з цих етапів має свою окрему генеративно-змагальну пару, що складається з генератора та дискримінатора (рисунок 2.3).

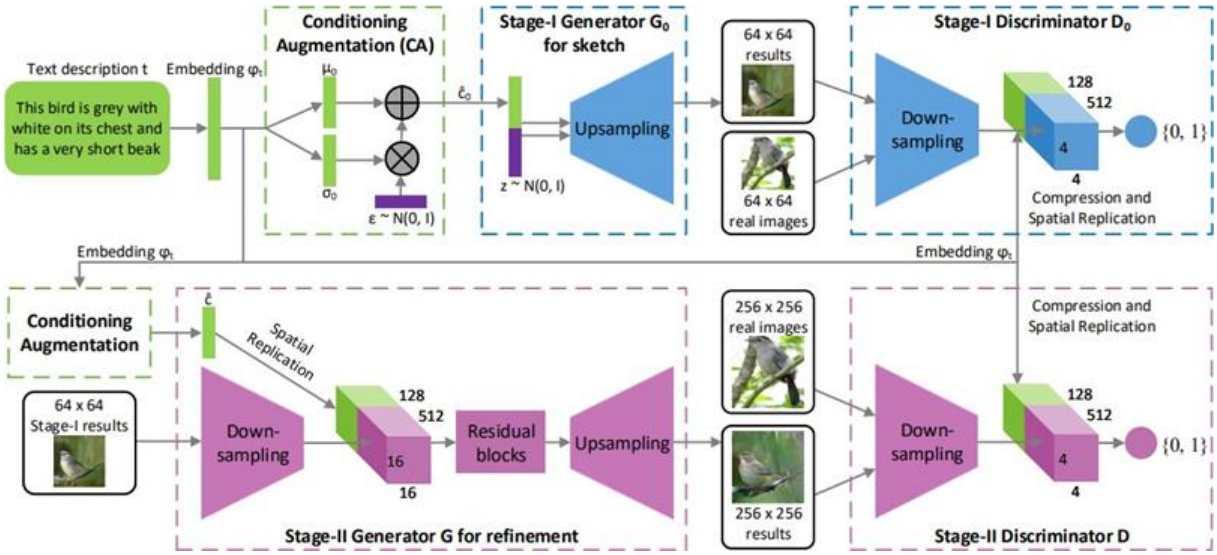


Рисунок 2.3 – Архітектура StackGAN

Stage-I Generator. Перший генератор приймає на вхід умовний вектор (текстовий опис) та випадковий латентний вектор. На цьому етапі генерується базове зображення з низькою роздільною здатністю (наприклад, 64x64 пікселів), яке передає основні форми та кольори, але не має достатньої деталізації.

Stage-I Discriminator. Перший дискримінатор оцінює згенеровані зображення низької роздільної здатності на відповідність текстовому опису. Його завдання полягає в тому, щоб відрізняти реальні зображення низької роздільної здатності від згенерованих, враховуючи умовний текстовий опис.

Stage-II Generator. Другий генератор приймає на вхід зображення, згенероване на першому етапі, разом з тим самим умовним вектором (текстовий опис). Він вдосконалює зображення, додаючи більше деталей і підвищуючи роздільну здатність (наприклад, до 256x256 пікселів). Цей етап використовує структуру та колірну інформацію зображення,

згенерованого на першому етапі, для створення більш реалістичного та деталізованого зображення.

Stage-II Discriminator. Другий дискримінатор оцінює згенеровані зображення високої роздільної здатності на відповідність текстовому опису. Він повинен відрізняти реальні зображення високої роздільної здатності від згенерованих, враховуючи умовний текстовий опис.

StackGAN представляє собою потужний підхід до генерації зображень з текстових описів, який покращує якість та реалістичність згенерованих зразків шляхом поетапної генерації та умовного навчання. Цей підхід знаходить застосування в різних областях, включаючи комп'ютерну графіку, візуалізацію даних, створення контенту для розваг і багато іншого.

2.1.5 Циклічна генеративна змагальна мережа (CycleGAN)

CycleGAN призначений для перекладів із зображення в зображення (image-to-image), які включають відображення одного зображення в інше. Наприклад, припустимо, що ми піддаємо перетворення одного зображення на інше (Image-Image). У цьому випадку ми можемо знайти функцію відображення, яка може перетворювати одне зображення (кінь) на інше зображення (зебра), як показано на рисунку 2.4, і навпаки, додаючи або видаляючи функції на основі відображення, щоб прогнозований результат та фактичний результат мали найменшу кількість втрати.

На рисунку 2.5 показано архітектуру CycleGAN, де:

– зображення X передається через генератор G, який дає згенероване зображення $\hat{Y}$;

– згенероване зображення $\hat{Y}$ передається через генератор F, який дає циклічне зображення $\hat{X}$;

– середня абсолютна похибка обчислюється між X і $\hat{X}$.

Рисунок 2.4 – Приклад перетворення за допомогою CycleGAN

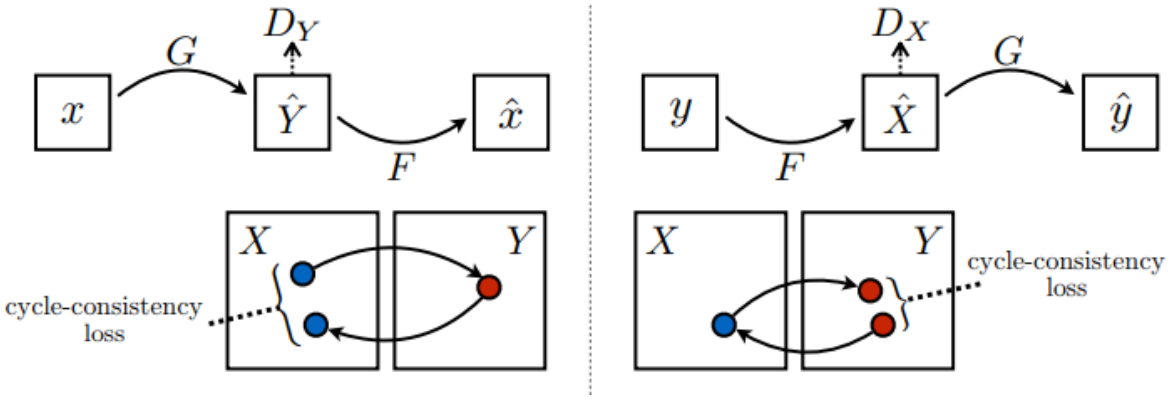


Рисунок 2.5 – Процедура навчання CycleGAN

Маючи два набори зображень, скажімо, коней і зебр, один генератор перетворює коней на зебр, а інший перетворює зебр на коней. Під час фази навчання присутні дискримінатори, щоб вирішити, чи є створені генератором зображення справжніми чи підробленими.

Генератори можуть вдосконалюватися за допомогою цього процесу за допомогою відповідних дискримінаторів. Один генератор отримує додатковий зворотний зв’язок від іншого у випадку CycleGAN. Цей зворотний зв’язок гарантує, що зображення, сформоване генератором, є циклічним, що означає, що застосування обох генераторів до того самого зображення в тому самому порядку має створити подібне зображення.

2.1.6 StyleGAN

StyleGAN є одним з найвідоміших та найуспішніших підходів у галузі генеративно-змагальних мереж. Він був запропонований дослідниками з NVIDIA і здобув велику популярність завдяки здатності генерувати високоякісні зображення людей, які виглядають надзвичайно реалістично. Основна ідея StyleGAN полягає в розділенні латентного простору на простір стилів, що дозволяє контролювати різні аспекти згенерованих зображень незалежно один від одного. Це досягається через використання спеціальної архітектури, зображеної на рисунку 2.6, що включає кілька ключових компонентів.

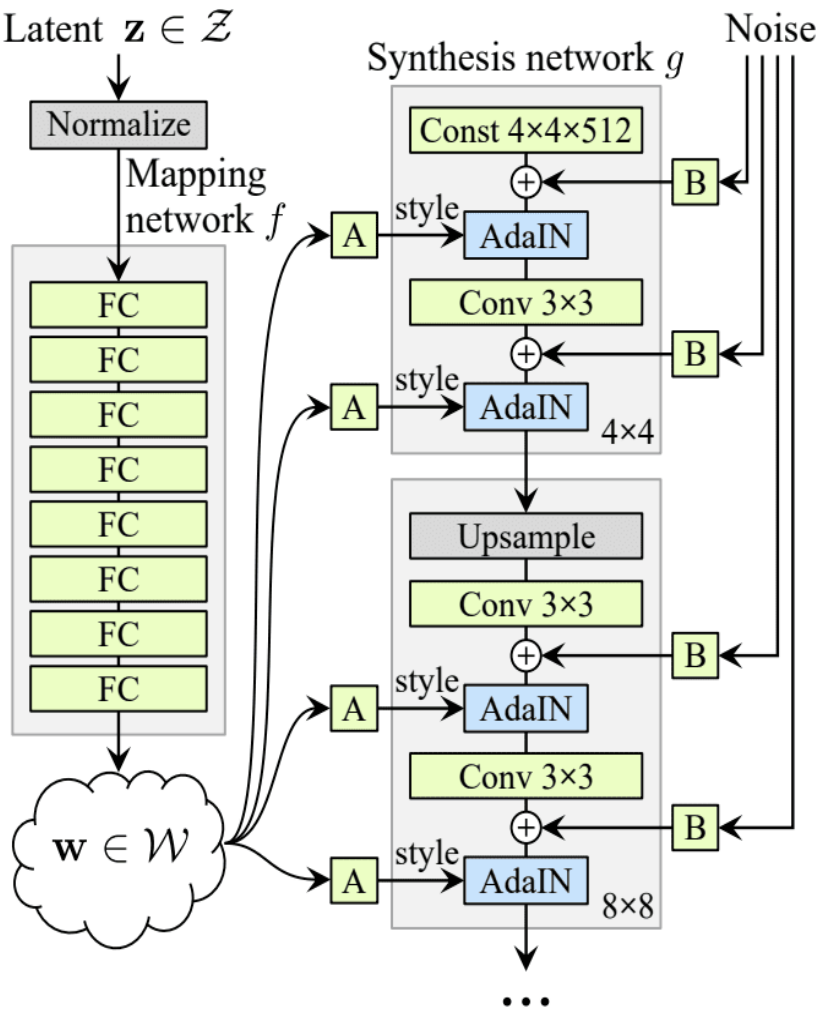


Рисунок 2.6 – Архітектура моделі генератора StyleGAN

Основні компоненти StyleGAN:

– латентний простір (Latent $z \in \mathcal{Z}$). Як і в інших GAN, StyleGAN використовує латентний вектор z, який вибирається з простого розподілу, зазвичай зі стандартного нормального розподілу як $z \sim \mathcal{N}(0, \boldsymbol{I})$;

– складач (mapping network f). Латентний вектор z проходить через складач f, що складається з кількох повнозв'язних шарів, перетворюючи його в інший простір латентних векторів w як $w = f(z)$. Це перетворення дозволяє розділити фактори варіації, що покращує контроль над генерованими зображеннями;

– адаптація стилів (AdaIN). Вектор w використовується для адаптації параметрів нормалізації на різних рівнях генератора через механізм адаптивної нормалізації (Adaptive Instance Normalization, AdaIN). Нормалізація кожного шару здійснюється за наступною формулою:

$$AdaIN(x, w) = \gamma(w)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta(\boldsymbol{w}) \qquad (2.4)$$

де $\mu(x)$ і $\sigma(x)$ – середнє і стандартне відхилення по каналах для активацій x;

$\gamma(w)$ і $\beta(w)$ – параметри, що визначаються вектором стилю w;

– генератор. Генератор StyleGAN складається з кількох блоків, кожен з яких додає деталі на різних рівнях просторової роздільної здатності. Вхідний вектор проходить через початковий повнозв'язний шар, який перетворює його в початковий тензор активацій, після чого ці активації модифікуються вектором стилю w через AdaIN на кожному шарі:

$$x_{i+1} = G_i(x_i, w); \qquad (2.5)$$

– прогресивне зростання. Генерація зображень у StyleGAN здійснюється через поступове збільшення роздільної здатності. Мережа спочатку навчається на низьких роздільних здатностях, додаючи нові шари

і збільшуючи роздільну здатність до досягнення кінцевого розміру зображення. Це дозволяє покращити стабільність тренування і якість кінцевих зображень.

StyleGAN є потужною та інноваційною архітектурою в галузі генеративно-змагальних мереж, що пропонує нові методи контролю та генерації високоякісних зображень. Використання механізму адаптивної нормалізації (AdaIN) та прогресивного зростання роздільної здатності дозволяє досягти значних успіхів у створенні реалістичних зображень. Завдяки цим інноваціям, StyleGAN став ключовим інструментом для багатьох застосувань у комп'ютерному баченні та інших областях.

2.2 Дифузійні моделі

Дифузійні моделі займають центральне місце у сучасних дослідженнях в області генеративного машинного навчання. Зокрема, дифузійні моделі є класом генеративних моделей, що використовуються для моделювання складних розподілів ймовірностей. Ці моделі представляють собою ланцюги Маркова, які навчаються за допомогою варіаційного висновку та функціонують шляхом багаторазового застосування дифузійних кроків до даних, наприклад, до зображень.

Дифузійні моделі можна класифікувати принаймні на три основні категорії. Першу категорію складають імовірнісні моделі дифузії з усуненням шуму (DDPM). Основою для DDPM є теорія нерівноважної термодинаміки. В цих моделях розподіл ймовірностей оцінюється за допомогою латентних змінних, тому вони можуть розглядатися як особливий тип варіаційних автокодувальників (VAE), де процес прямої дифузії є кодуванням, а зворотної дифузії – декодуванням.

Друга категорія включає мережі оцінок з умовою шуму (NCSN). У цих моделях вибірки генеруються динамікою Ланжевена з використанням градієнтів розподілу даних, що оцінюються через зіставлення оцінок. Коли

дані розташовані на низьковимірних многовидах, вони зазнають гаусівського шуму різного рівня, що дозволяє спільно оцінити відповідні оцінки – векторні поля градієнтів розподілу збурених даних для всіх рівнів шуму. Це робить можливим більш точне моделювання складних розподілів, навіть якщо градієнти важко оцінити точно.

Третя категорія – це моделі, що базуються на стохастичних диференціальних рівняннях (SDE). Ці моделі пропонують альтернативний підхід до дифузії, де використовується континуум розподілів, що змінюються з часом згідно з процесом дифузії. На відміну від моделей з фіксованою кількістю шумових розподілів, SDE моделюють процес за допомогою заданого стохастичного диференціального рівняння, яке не потребує параметрів для навчання і не залежить від даних. Генерація нових зразків здійснюється шляхом інверсії процедури дифузії. Таким чином, SDE можна вважати узагальненням для DDPM та NCSN.

Ці моделі, разом з GAN, забезпечують потужні інструменти для генерації високоякісних синтетичних даних, що відкриває нові можливості в різних галузях, включаючи обробку зображень, створення мультимедійного контенту та наукові дослідження.

2.2.1 Ймовірнісна дифузійна модель з усуненням шуму (DDPM)

Ймовірнісні дифузійні моделі з усуненням шуму (Denoising Diffusion Probabilistic Models, DDPM) є новітнім підходом у галузі генеративного моделювання, що відзначається високою якістю створюваних зображень та стабільністю тренування. У цьому розділі розглянемо основні концепції DDPM, математичні основи моделі, процес тренування та генерації даних [24].

DDPM базуються на поступовому додаванні випадкового шуму до даних протягом багатьох кроків, що утворює марковський процес. На кожному кроці дані стають дедалі більш зашумленими, і в кінцевому

підсумку перетворюються на чистий шум. Зворотній процес полягає у поступовому відновленні даних, видаляючи шум на кожному етапі. Ці два процеси прямої дифузії та параметризованої зворотної зображені на рисунках 2.7 та рисунку 2.8.

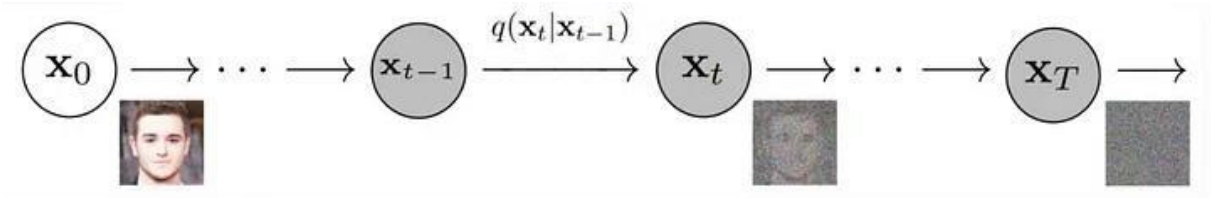


Рисунок 2.7 – Процес прямої дифузії

Дифузійний (прямий) процес (на рисунку 2.7 справа наліво) є стохастичним процесом, який стартує з вибірки даних і поступово створює все більш зашумлені вибірки за допомогою гаусівського дифузійного ядра. Це являє собою ланцюг Маркова, оскільки на момент часу t зображення залежить не від усіх попередніх зображень, а тільки від зображення на кроці $t = 1$. На кожному кроці t додається шум за допомогою наступной формули:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t x_{t-1}}, \beta_t I), \tag{2.4}$$

де $\mathcal{N}(x; \mu, \sigma)$ – нормальний розподіл, що видає x з середнім значенням μ і дисперсією σ;

β_t – невеликі додатні величини, що визначають величину шуму на кроці;

I – одинична матриця тотожності, що має розміри як у вхідному зображенні x_t.

Після T кроків початкове зображення x_0 перетворюється на майже чистий шум x_t.

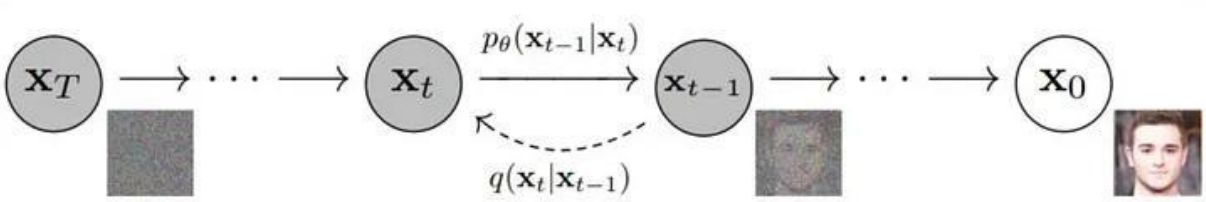


Рисунок 2.8 – Процеси зворотної дифузії

Зворотній процес полягає у поступовому видаленні шуму з даних, відновлюючи початкове зображення. Для цього використовується модель параметризована нейронною мережею θ, що наближає умовний розподіл:

$$p_\theta(\boldsymbol{x_{t-1}} \mid \boldsymbol{x_t}) = N(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \Sigma_\theta(\boldsymbol{x}_t, t)), \tag{2.5}$$

де $\mu_\theta(\boldsymbol{x}_t, t)$ і $\Sigma_\theta(\boldsymbol{x}_t, t)$ – параметри, що прогнозуються нейронною мережею.

Процес прямої і зворотної дифузії наглядно можна побачити на рисунку 2.9 [25].

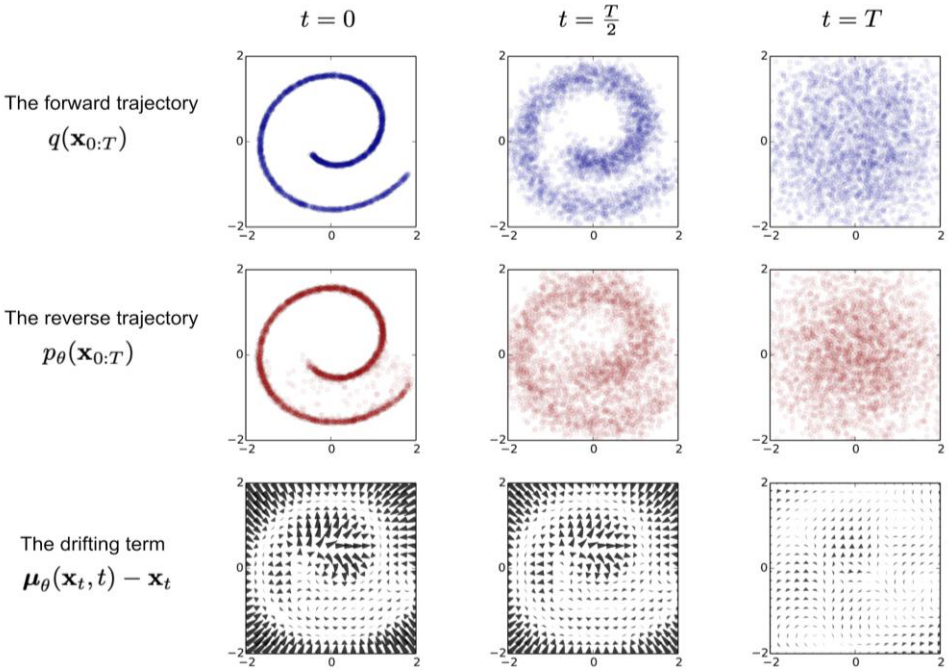


Рисунок 2.9 – Приклад навчання дифузійної моделі для моделювання даних 2D рулету

Мета тренування полягає у мінімізації різниці між справжнім розподілом і наближеним розподілом, використовуючи негативний варіаційний нижній кордон (Negative Variational Lower Bound, NVLB):

$$L_{vlb} = \mathbb{E}_q[\sum_{t=1}^{T} D_{KL}[q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)] - log\, p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)]. \quad (2.6)$$

Враховуючи властивості гаусівського розподілу, цей вираз можна спростити і використовувати для тренування нейронної мережі:

$$L_{simple} = \mathbb{E}_{x_0,\epsilon,t}[\parallel \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) \parallel^2] \quad (2.7)$$

де $\boldsymbol{\epsilon}$ – випадковий шум, доданий на кожному кроці;

$\boldsymbol{\epsilon}_\theta$ – прогнозований шум нейронною мережею.

Для генерації нового зображення спочатку створюється випадковий шум x_T, який проходить через зворотній процес дифузії, поступово видаляючи шум:

– задати $x_T \sim \mathcal{N}(0, I)$;

– для $t = T, T-1, \ldots, 1$ обчислити як:

$$x_{t-1} = \frac{1}{\sqrt{\alpha^t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x_t}, \boldsymbol{t})\right) + \sigma_t \boldsymbol{z} \quad (2.8)$$

де $\alpha_t = 1 - \beta_t, \sigma_t$ – додатковий шум, що моделює невизначеність.

Ймовірнісні дифузійні моделі з усуненням шуму (DDPM) є ефективним підходом до генеративного моделювання, що забезпечує високу якість створюваних зображень. Вони базуються на процесах поступового додавання та видалення шуму, що дозволяє відновлювати початкові дані з високою точністю. Незважаючи на обчислювальну складність, цей метод демонструє значний потенціал у різних галузях застосування.

2.3 Переваги та недоліки GAN-мереж та дифузійних моделей

Генеративно-змагальні мережі (GAN) та ймовірнісні дифузійні моделі (DDPM) є двома провідними методами в галузі генеративного моделювання. Вони мають свої унікальні переваги та недоліки, які роблять їх придатними для різних завдань і сценаріїв застосування. У цьому розділі розглянемо ключові особливості цих методів.

Переваги GAN:

− висока якість генерації – GAN здатні генерувати дуже реалістичні зображення, що важко відрізнити від справжніх. Це досягається завдяки змагальному характеру тренування, де генератор постійно вдосконалюється, щоб обманути дискримінатор;

− швидкість генерації – після тренування генератора, процес генерації нових зображень відбувається дуже швидко, оскільки це просто проходження одного зразка через нейронну мережу;

− гнучкість – GAN можуть бути використані для різних типів даних, включаючи зображення, текст і звук. Це робить їх універсальним інструментом для різних застосувань.

Недоліки GAN:

− складність тренування – тренування GAN є складним завданням через нестабільність, пов'язану з змагальним процесом. Це може призводити до таких проблем, як колапс моди, коли генератор навчається створювати лише обмежену кількість зразків;

− відсутність контрольованості – у GAN відсутні явні механізми контролю над процесом генерації. Це означає, що складно задати конкретні властивості згенерованих зразків;

− високі обчислювальні витрати – тренування GAN вимагає значних обчислювальних ресурсів, оскільки необхідно тренувати одночасно дві мережі генератор і дискримінатор;

− шуми – незважаючи на важливість початкового шуму, GAN зазвичай використовують стандартні види шуму без значних модифікацій, що може обмежувати контрольованість генерованого процесу.

Переваги дифузійних моделей (DDPM)

− стабільність тренування – процес тренування дифузійних моделей є більш стабільним порівняно з GAN, оскільки він не базується на змагальній схемі. Це дозволяє уникнути багатьох проблем, пов'язаних із колапсом моди;

− висока якість реконструкції – DDPM здатні відтворювати високоякісні зображення завдяки поступовому процесу генерації, де на кожному етапі використовується усунення шуму;

− явний контроль над процесом генерації – у дифузійних моделях є можливість більш точно контролювати процес генерації, що дозволяє задавати специфічні властивості для згенерованих даних;

− шуми – дифузійні моделі використовують складний процес додавання та усунення шуму для генерації зображень. Початкові дані поступово зашумлюються протягом декількох кроків, а потім модель навчається відновлювати дані, усуваючи шум на кожному етапі. Це дозволяє використовувати як стандартні, так і нестандартні види шуму, що додає гнучкості в моделюванні складних розподілів даних.

Недоліки дифузійних моделей (DDPM):

− повільність генерації – процес генерації в дифузійних моделях займає більше часу порівняно з GAN, оскільки він включає багато етапів поступового усунення шуму;

− складність архітектури – дифузійні моделі мають складнішу архітектуру і процес тренування, що може вимагати значної кількості обчислювальних ресурсів та спеціальних знань для їх реалізації;

− обчислювальні витрати на тренування – тренування DDPM є обчислювально інтенсивним через необхідність виконання великої кількості кроків для моделювання дифузійного процесу.

Обидві моделі мають свої унікальні переваги і недоліки, що робить їх придатними для різних задач. GAN забезпечують швидку генерацію високоякісних зображень, але можуть бути складними в тренуванні і контролі. Натомість дифузійні моделі забезпечують стабільніший процес тренування і високу якість реконструкції, але на шкоду швидкості генерації і складності архітектури. Використання шумів у обох моделях також має значення – GAN зазвичай використовують стандартний шум для початкового вектора, тоді як дифузійні моделі включають складніший процес додавання та усунення шуму, що дозволяє експериментувати з різними типами шумів для досягнення бажаних результатів.

Узагальнюючи, вибір між GAN і DDPM залежить від конкретних вимог до завдання генеративного моделювання. Якщо потрібна швидка генерація і високоякісні результати, GAN можуть бути кращим вибором. Якщо ж важлива стабільність тренування і контроль над процесом генерації, дифузійні моделі можуть бути більш підходящими. Значення шумів у цих моделях також є ключовим аспектом, який впливає на їхню ефективність і результати.

3 ОСНОВНІ ПІДХОДИ ДО ФОРМУВАННЯ ТА ГЕНЕРАЦІЇ ШУМІВ

Генеративні змагальні мережі використовують вхідний шум для ініціалізації процесу генерації зразків. Цей шум зазвичай походить з певного розподілу і допомагає генератору створювати різноманітні та реалістичні зразки. Розглянемо стандартні типи вхідного шуму, такі як гауссівський, рівномірний, логнормальний і експоненційний розподіл, а також альтернативні методи генерації шуму, які можуть бути використані для навчання генератора та створення нових зображень.

3.1 Стандартні типи вхідного шуму

3.1.1 Нормальний розподіл (Gaussian distribution)

Один з найпоширеніших типів вхідного шуму, який використовується у GAN, є гауссівський або нормальний розподіл (рисунок 3.1).

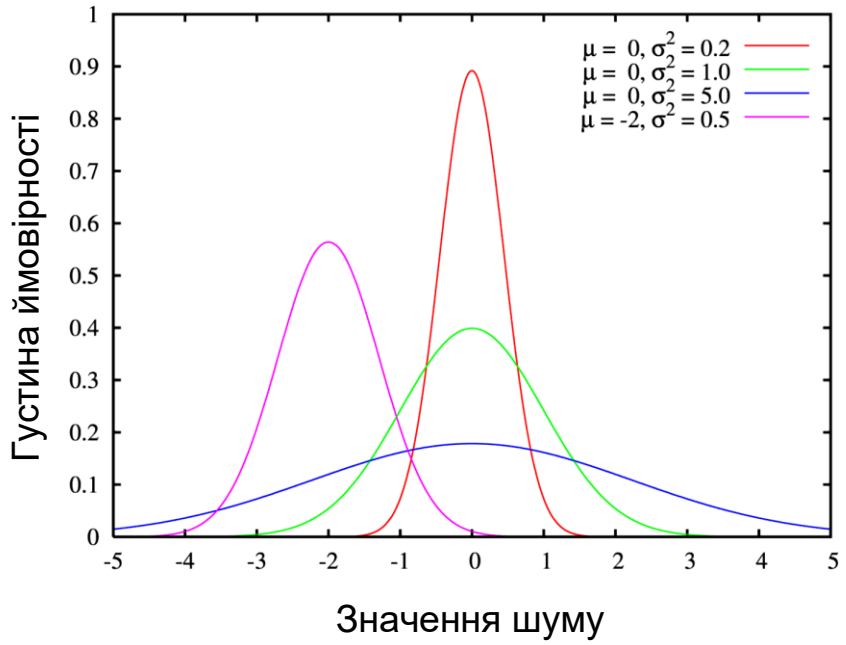


Рисунок 3.1 – Нормальний розподіл або розподіл Гауса

Розподіл має форму дзвона і характеризується середнім значенням (μ) та стандартним відхиленням (σ). Вектор шуму z зазвичай вибирається з розподілу з середнім значенням $\mu = 0$ та стандартним відхиленням $\sigma = 1$.

Нормальний розподіл часто позначається як:

$$z \sim N(\mu, \sigma^2). \tag{3.1}$$

Формула густини ймовірності для нормального розподілу:

$$P(z) = \frac{1}{\sigma\sqrt{2\pi}}\ exp(-\frac{(z-\mu)^2}{2\sigma^2}). \tag{3.2}$$

Перевага гауссівського розподілу полягає в його простоті та теоретичній обґрунтованості, що робить його зручним для використання в нейронних мережах та дозволяє створювати безперервний і плавний простір латентних змінних.

3.1.2 Рівномірний розподіл (Uniform distribution)

Постійне рівномірне розподілення (рисунок 3.2) також часто використовується як вхідний шум для GAN.

У цьому випадку вектор шуму z має рівну ймовірність приймати будь-яке значення в заданому інтервалі [a, b]. Найчастіше використовується інтервал від -1 до 1 або від 0 до 1.

Рівномірне розподілення часто позначається як:

$$z \sim U(a, b). \tag{3.3}$$

Формула густини ймовірності для рівномірного розподілу:

$$P(z) = \frac{1}{b-a}, \quad z \in [a, b]. \tag{3.4}$$

Рівномірний розподіл забезпечує рівну ймовірність для всіх значень в заданому діапазоні, що може бути корисним для створення широкого спектру різних варіантів.

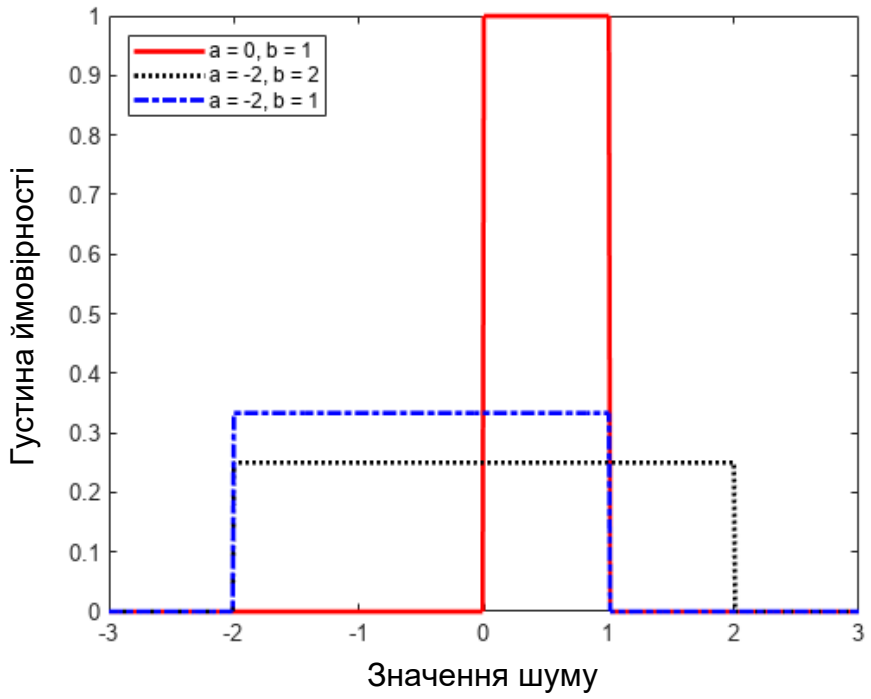


Рисунок 3.2 – Постійне рівномірне розподілення

3.1.3 Логнормальний розподіл (Lognormal distribution)

Логнормальний (логарифмічно-нормальний) розподіл є ще одним варіантом для генерації вхідного шуму для GAN-мереж (рисунок 3.3).

Якщо випадкова величина має нормальний (гауссівський) розподіл, то експонента цієї величини має логарифмічно-нормальний розподіл.

Логнормальний розподіл часто позначається як:

$$z \sim LogN(\mu, \sigma^2). \tag{3.5}$$

Формула густини ймовірності для логнормального розподілу наведена нижче:

$$P(z) = \frac{1}{z\sigma\sqrt{2\pi}}\ exp(-\frac{1}{2}[\frac{(ln(z)-\mu)}{\sigma}]^2). \quad (3.6)$$

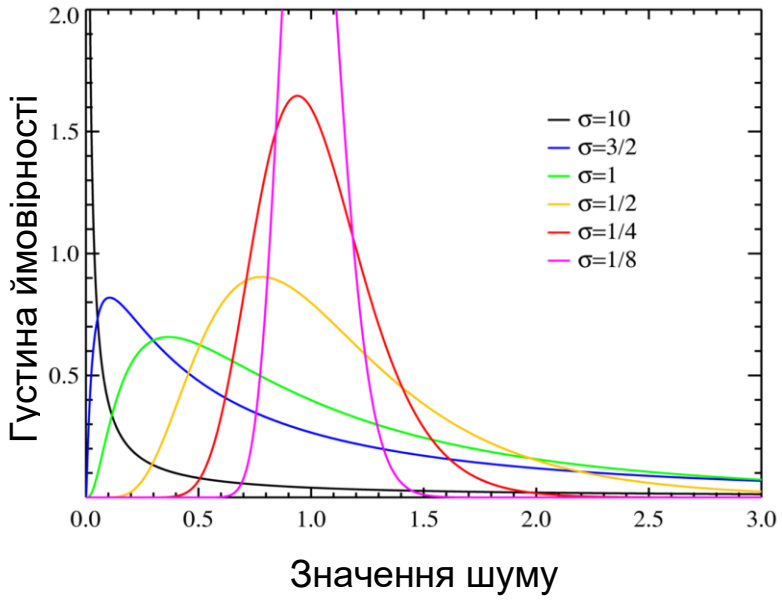


Рисунок 3.3 – Логарифмічно-нормальний розподіл

Логанормальний розподіл часто використовується в задачах, де необхідно моделювати позитивні величини, такі як доходи або значення фінансових активів, які не можуть бути від'ємними.

Використання логанормального розподілу в GAN може бути корисним для генерації даних, які природним чином слідують асиметричному розподілу.

3.1.4 Експоненційний розподіл (Exponential distribution)

Експоненційний розподіл (рисунок 3.4) використовується рідше, але також може бути застосований в деяких випадках у GAN. Цей розподіл характеризується параметром λ, який визначає інтенсивність розподілу.

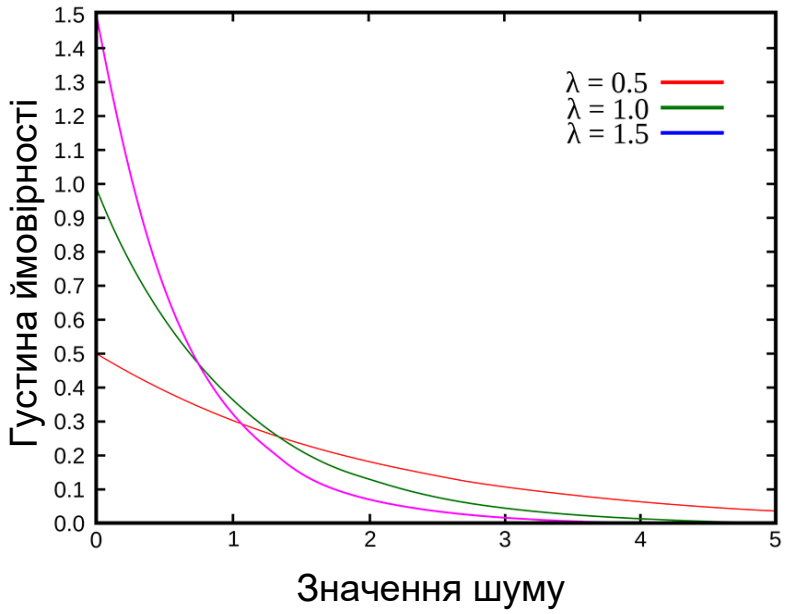


Рисунок 3.4 – Експоненційний розподіл

Експоненціального розподілу часто позначається як:

$$z \sim Exp(\lambda) \tag{3.7}$$

Формула густини ймовірності для експоненціального розподілу наведена нижче:

$$P(z) = \lambda\, exp(-\lambda z), \quad z \geq 0. \tag{3.8}$$

Експоненціальний розподіл може бути корисним для моделювання процесів, де ймовірність значення швидко зменшується з відстанню від початкової точки.

3.1.5 Розподіл Лапласа (Laplace distribution)

Цей тип шуму характеризується «гострими» пиками біля середнього значення та важкими хвостами. Розподіл Лапласа (рисунок 3.5)

використовується для задач, де важливі великі відхилення або для моделювання даних із різкими змінами.

Розподіл Лапласа часто позначається як:

$$z \sim Laplace(\mu, b). \tag{3.9}$$

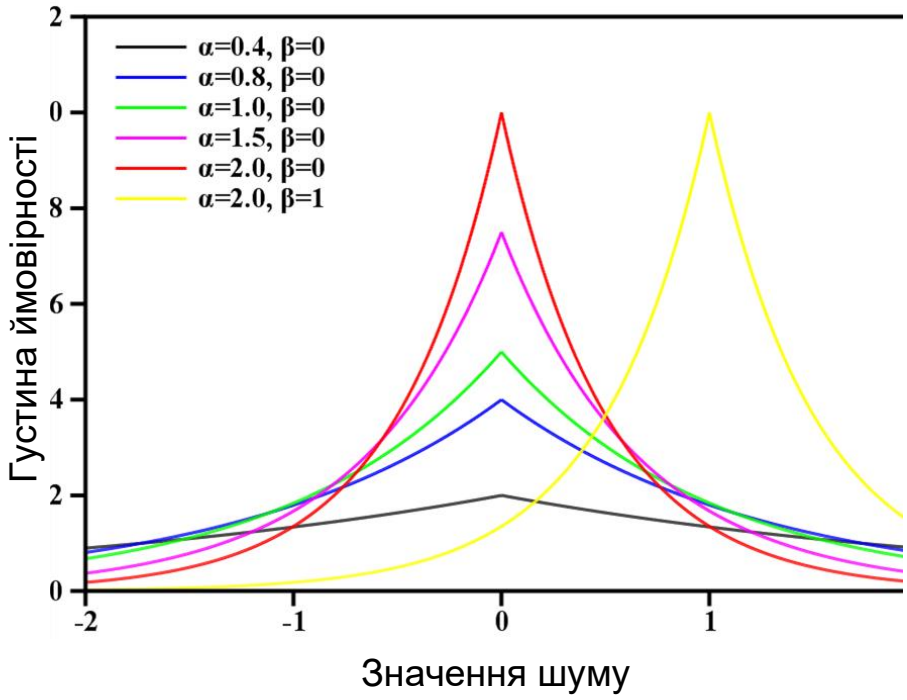


Рисунок 3.5 – Розподіл Лапласа

Розподіл Лапласа, також відомий як дво-експоненційний розподіл, характеризується наступною формулою щільності ймовірності для змінної x:

$$P(z) = \frac{1}{2b} exp(-\frac{|x-\mu|}{b}) \tag{3.10}$$

де μ – місцева середина розподілу (медіана);

b – масштабний параметр, який визначає розкид розподілу.

Для стандартного розподілу Лапласа зазвичай використовуються параметри $\mu = 0$ і $b = 1$.

Завдяки високій концентрації значень поблизу медіани та наявності більш довгих «хвостів» у порівнянні з нормальним розподілом, шум з розподілу Лапласа може сприяти генерації більш різноманітних зображень або інших даних. Також вектор шуму з розподілу Лапласа може мати структуру, яка сприятиме кращому навчання генератора у деяких задачах.

3.2 Альтернативні види генерації шуму

Крім стандартних розподілів, існують інші методи генерації вхідного шуму, які можуть бути використані для навчання генератора та створення нових зразків даних [26].

3.2.1 Перлін-шум (Perlin Noise)

Перлін-шум є видом градієнтного шуму, який генерується шляхом інтерполяції між випадковими градієнтами (рисунок 3.6). Використовується у комп'ютерній графіці для створення природних текстур, таких як ландшафти, хмари, вода, земля.

Шум Перліна найчастіше реалізується як дво-, три- чи чотиривимірна функція, але може бути визначений для будь-якої кількості вимірювань. Реалізація зазвичай включає три етапи:

– задаються випадкові градієнти на точках сітки;

– виконується інтерполяція між цими градієнтами для обчислення значень шуму в проміжних точках;

– значення шуму комбінуються для створення гладкої текстури.

Функція шуму Перлина зазвичай видає значення діапазону [–1,0, 1,0] і можуть відповідним чином масштабуватися. Математично, перлін-шум визначається як:

$$Perlin(x, y) = \sum_{i,j} G(i, j) \cdot (x - i, y - j) \cdot f(x - i) \cdot f(y - j) \qquad (3.11)$$

де $G(i,j)$ – випадкові градієнти на точках сітки;

$(x-i, y-j)$ – вектор від точки сітки до проміжної точки;

f – функція згладжування, наприклад, функція скінченних різниць.

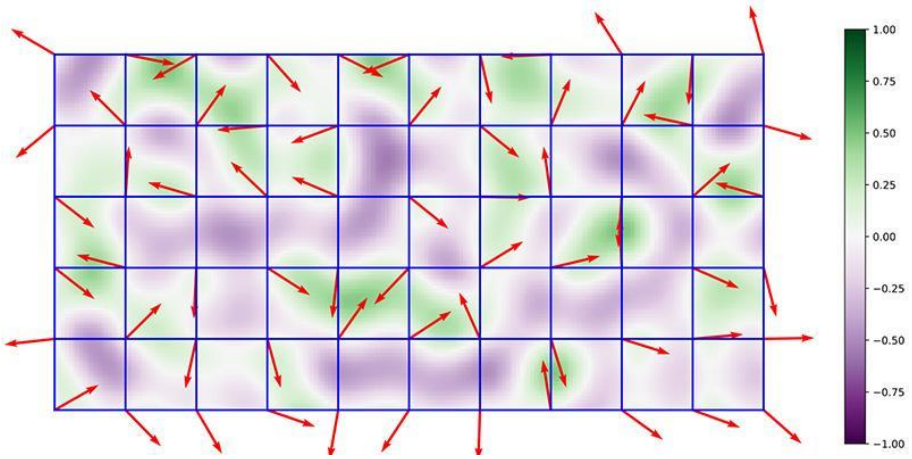


Рисунок 3.6 – Інтерполований результат для двомірної сітки векторів градієнта

Використання шуму Перлін в GAN може допомогти створювати більш структуровані та реалістичні зображення [27].

3.2.2 Шум з фрактальної геометрії (Fractal Noise)

Фрактальний шум базується на фрактальних функціях і може використовуватися для створення складних, самоподібних структур. Використовується у випадках, коли потрібно генерувати природні або органічні структури.

Математично, фрактальний шум можна визначити як суму перлін-шумів на різних масштабах, кожен з яких має різну амплітуду та частоту. Це створює багатошарову текстуру з самоподібними властивостями [28].

Формула фрактального шуму:

$$Fractal(x, y) = \sum_{i=0}^{n} \frac{1}{2^i} \cdot Perlin(2^i x, 2^i y)^h \quad (3.12)$$

де n – кількість октав (рівнів деталізації), кожна з яких додає все більш дрібні деталі до загальної текстури;

h – параметр гладкості.

3.2.3 Шум зображення (Image Noise)

Інший підхід полягає у використанні реальних зображень як вхідного шуму. Цей метод включає додавання випадкового шуму до реальних зображень та їх подальше використання як вхідних даних для генератора.

Це може допомогти генератору вивчати реалістичні текстури та деталі зображень.

3.2.4 Додавання випадкових векторів (Random Vectors)

У цьому підході кожен вхідний вектор генератора z доповнюється випадковим вектором ϵ, що представляє шум:

$$x_{input} = z + \epsilon. \quad (3.13)$$

Цей метод може бути використаний як при навчанні генератора, так і при генерації даних із вже навченою моделлю. Шум додається до випадкового вектора на вході генератора, щоб стимулювати створення різноманітних зразків.

Даний метод дуже простий у реалізації, а також є можливість керувати різноманітністю згенерованих даних. З недоліків можлива втрата семантичного зв'язку між вхідними даними та вихідними даними.

3.2.5 Випадкові спотворення (Random Distortions)

У цьому підході вихідні дані зазнають випадкових спотворень, наприклад, зміни яскравості, контрасту, поворотів, масштабування, горизонтальне перевертання тощо:

$$x_{distorted} = f(x_{original}, \epsilon). \quad (3.14)$$

де f – функція спотворення;

$x_{original}$ – вихідні дані;

ϵ – параметри випадкового спотворення.

Цей метод переважно використовується під час навчання генератора.

Випадкові спотворення застосовуються до вихідних даних, щоб збільшити обсяг та різноманітність навчального набору для підвищення стійкості моделі та її здатності обробляти різноманітні вхідні дані, а також покращує узагальнюючу здатність моделі.

З недоліків можна назвати можливість спотворювати семантичний зміст даних, особливо у разі складних об'єктів.

3.3 Використання стандартних та альтернативних шумів

Генеративно-змагальні мережі є потужним інструментом для генерації зображень та інших даних.

Однією з ключових складових GAN є вхідний шум, який впливає на якість і різноманітність згенерованих даних.

Нижче наведено порівняння популярних видів GAN-мереж та типів шумів, які використовуються на їхній вхід (таблиця 3.1).

Таблиця 3.1 – Види вхідного шуму використовувані GAN-мережами

GAN-мережа	Вид шума	Використання
Vanilla GAN	Нормальний (гаусівський)	Стандарт через його властивості, що сприяють стабільному процесу навчання
Conditional GAN (CGAN)	Гаусівський Рівномірний	До вхідного шуму додається умова (label), що допомагає контролювати генерацію
Deep Convolutional GAN (DCGAN)	Гаусівський	Нормальний шум сприяє стабільності генерації
Wasserstein GAN (WGAN)	Гаусівський	Може працювати з різними типами шумів
Enhanced GAN (EGAN)	Шум зображення (альтернативний)	При генерації зображень у реальних умовах, де дані можуть бути забруднені шумом, використання шуму зображення допомагає генератору навчитися створювати більш стійкі до шуму зображення
BigGAN	Додавання випадкових векторів (альтернативний)	Генерація зображень з високою роздільною здатністю, стилізація зображень
StyleGAN	Випадкові спотворення (альтернативний)	Генерація портретів, створення зображень для розширеної реальності (AR) та віртуальної реальності (VR).

Дифузійні моделі (Diffusion Models) використовуються для генерації зображень шляхом додавання шуму до зображень і потім поступового видалення, щоб створити нові зображення з випадкового шуму. У цих моделях застосовуються різні види шумів для різних етапів навчання та генерації. Основні та альтернативні види шумів, що використовуються в дифузійних моделях, представлені в таблиці 3.2.

Таблиця 3.2 – Види шуму у дифузійних моделях

Вид шума	Описание
Гаусівський шум	Гаусівський шум має нормальний розподіл із певним середнім значенням та стандартним відхиленням.
Шум зображення (Image Noise):	Шум зображення відноситься до шумів, які можуть виникати в процесі формування зображення, наприклад, через сенсори камери, умови освітлення або інші зовнішні фактори. Цей шум включає комбінацію різних типів шумів, таких як гаусовский, Пуассона, сіль-перець та інші.
Додавання випадкових векторів (Random Vectors)	Додавання випадкових векторів передбачає додавання випадкових значень у кожну точку зображення. Ці випадкові значення може бути як гауссовскими, і мати будь-який інший розподіл.
Випадкові спотворення (Random Distortions)	Випадкові спотворення включають різні форми деформацій зображення, такі як повороти, розтягування, спотворення форми, шуми з різними просторовими кореляціями тощо.
Шум білого шуму (White Noise)	Білий шум є випадковими значеннями, розподіленими рівномірно по всьому діапазону інтенсивностей.
Шум Пуассона (Poisson Noise)	Шум Пуассона виникає внаслідок дискретних подій, таких як лічильники фотонів у зображенні. Це шум, який найчастіше зустрічається у завданнях медичної та наукової візуалізації.
Шум сіль-перець (Salt-and-Pepper Noise)	Цей вид шуму характеризується випадковими яскравими та темними пікселями (сіль – білі пікселі, перець – чорні пікселі), що з'являються на зображенні.
Шум Лапласа (Laplacian Noise)	Шум із розподілом Лапласа має подвійний експоненційний розподіл, що робить його більш гострим у порівнянні з гауссівським шумом.

Гаусівський шум є найбільш поширений вид шуму, що використовується у дифузійних моделях. Шум додається до зображення на кожній ітерації процесу дифузії, поступово перетворюючи його на чистий шум.

Шум зображення у дифузійних моделях може бути використаний для симуляції реальних умов зйомки та навчання моделей, стійких до різних видів спотворень та шуму. Це допомагає зробити моделі адаптованішими і застосовнішими в реальних сценаріях.

Додавання випадкових векторів використовується для дослідження впливу різних випадкових факторів на процес створення зображень. Цей метод дозволяє експериментувати з різними розподілами шуму та вивчати їх вплив на якість та різноманітність згенерованих зображень.

Випадкові спотворення у дифузійних моделях можуть бути використані для створення різноманітних та реалістичних наборів даних для навчання. Це допомагає моделям навчитися краще справлятися з різними видами спотворень, які можуть зустрічатися у реальних зображеннях, та покращує загальну стійкість та генеративну здатність моделей.

Шум білого шуму у деяких варіаціях дифузійних моделей може використовуватися для різних експериментів та аналізу, хоча частіше використовується гаусівський шум.

Шум Пуассона може бути використаний у спеціалізованих дифузійних моделях, де імітується процес формування зображень, що залежить від лічильників.

Шум сіль-перець рідко використовується в стандартних дифузійних моделях для генерації зображень, але може бути корисним у контексті досліджень стійкості моделей до різних типів спотворень (артефактів).

Шум Лапласа може бути використаний у дифузійних моделях для вивчення впливу різних статистичних властивостей шуму на процес генерації зображень.

Альтернативні види шуму можуть значно покращити продуктивність як GAN-мереж, так і дифузійних моделей у різних задачах. Використання таких шумів, як шум зображення, додавання випадкових векторів та випадкові спотворення, дозволяє досягти більш високої якості та різноманітності згенерованих зображень, а також підвищити стабільність процесу навчання. Дослідження та експерименти з різними типами шумів допомагають знайти оптимальні підходи для конкретних задач та розширюють можливості застосування генеративних моделей.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Обґрунтування обраних програмних засобів

Вибір програмних засобів для розробки та навчання генеративно-змагальних мереж є критично важливим аспектом дослідження, який впливає на ефективність, швидкість розробки та якість результатів. У цьому розділі ми порівняємо дві найпопулярніші платформи для глибокого навчання: PyTorch та TensorFlow. Також розглянемо стандартний набір бібліотек, які використовуються для написання та навчання GAN-нейронних мереж на Python.

4.1.1 Порівняння PyTorch і TensorFlow

PyTorch – це бібліотека машинного навчання на основі бібліотеки Torch, яка використовується для таких програм, як комп’ютерне бачення та обробка природної мови та являється частиною Linux Foundation.

PyTorch має наступні переваги:

– PyTorch використовує динамічний граф обчислень, що робить його більш інтуїтивно зрозумілим для дослідників та розробників. Це дозволяє змінювати граф на льоту, що спрощує відладку та експерименти з моделями;

– PyTorch має простий та зрозумілий інтерфейс, який більше схожий на стандартний Python-код. Це робить його більш зручним для новачків та швидкої розробки прототипів;

– PyTorch швидко став популярним серед дослідників та має активну спільноту, що забезпечує велику кількість навчальних матеріалів та прикладів;

– PyTorch добре інтегрується з іншими бібліотеками для наукових обчислень, такими як NumPy і SciPy.

До недоліків можна віднести те, що PyTorch менш оптимізований для використання на робочих сервах (production) у порівнянні з TensorFlow, хоча ситуація покращується з появою PyTorch Serve та інших інструментів для розгортання моделей.

TensorFlow – це безкоштовна бібліотека програмного забезпечення з відкритим кодом для машинного навчання та штучного інтелекту. Він може бути використаний у ряді завдань, але має особливу увагу на навчанні та висновках глибоких нейронних мереж. Він був розроблений командою Google Brain для внутрішнього використання Google у дослідженнях і для робочих продуктів.

Бібліотека TensorFlow має наступні переваги:

– TensorFlow використовує статичний граф обчислень, який дозволяє виконувати оптимізації на рівні графу, що може призвести до покращення продуктивності;

– TensorFlow розроблений з урахуванням масштабованості, що дозволяє легко розгортати моделі на різних платформах, від мобільних пристроїв до великих кластерів серверів;

– TensorFlow має вбудовані інструменти для розгортання моделей на робочих серверах, наприклад TensorFlow Serving, що робить його зручним для використання для комерційних продуктів;

– TensorFlow має велику екосистему інструментів і бібліотек, включаючи; TensorFlow Lite, TensorFlow.js, TensorFlow Extended (TFX) та інші, що робить його універсальним рішенням для різних задач машинного навчання.

До недоліків можна віднести те, що TensorFlow має більш складний інтерфейс у порівнянні з PyTorch, що може ускладнити його використання для новачків. Також статичний граф обчислень ускладнює відладку та експерименти з моделями, оскільки зміни у графі потребують його повторного створення.

4.1.2 Набір бібліотек для проведення експериментів

Для розробки та навчання GAN-нейронних мереж було обрано мову програмування Python та були використані наступні бібліотеки:

− TensorFlow – для побудови моделей. Ця бібліотека має всі необхідні реалізовані компоненти, інтерфейси потоків для побудови моделей та зручні інструменти обробки даних;

− Keras – додаткові бібліотеки для роботи із зображеннями. tf.keras.preprocessing забезпечують інструменти для завантаження, перетворення та збільшення зображень (augmentation);

− NumPy – базова бібліотека для роботи з багатовимірними масивами та матрицями. Забезпечує велику кількість функцій для математичних операцій над масивами;

− Matplotlib.PyPlot – бібліотеки для візуалізації даних, забезпечує базову функціональність для створення графіків.

4.2 Огляд набору даних

Для тренування моделей генеративно-змагальних мереж в рамках даного дослідження було обрано набір даних рукописних цифр MNIST. Цей набір даних є одним із найпопулярніших та найвідоміших в галузі машинного навчання, що робить його ідеальним вибором для експериментів та порівнянь моделей. У цьому розділі розглянемо деталі набору даних MNIST, його структуру, а також причини вибору цього набору для тренування моделей GAN.

4.2.1 Опис набору даних MNIST

Набір даних MNIST (Modified National Institute of Standards and Technology) містить зображення рукописних цифр від 0 до 9. Він

складається з 60,000 зображень для тренувального набору та 10,000 зображень для тестового набору (рисунок 4.1). Кожне зображення представлено у відтінках сірого, має розмір 28x28 пікселів та містить лише одну цифру:

– розмір зображень: 28x28 пікселів;

– кількість класів: 10 (цифри від 0 до 9);

– кількість зображень у тренувальному наборі: 60,000;

– кількість зображень у тестовому наборі: 10,000.

Рисунок 4.1 – Приклади з набору даних MNIST

Зображення представлені як матриці розміром 28x28, де кожен елемент матриці є значенням яскравості пікселя (від 0 до 255). Значення 0 відповідає чорному кольору, а 255 – білому кольору. Таким чином, кожне зображення може бути представлене як вектор з 784 значень (28x28), що спрощує обробку даних для нейронних мереж.

4.2.2 Обґрунтування вибору набору даних MNIST

Причини з яких був обраний набір даних для проведення дослідницької роботи полягає в тому, що це проста і зручна вибірка для експериментів:

− MNIST є стандартним набором даних для тестування алгоритмів машинного навчання. Завдяки своїй простоті, його можна швидко завантажити та використовувати без додаткової попередньої обробки;

− MNIST широко використовується в академічних дослідженнях та промислових проектах, що дозволяє легко порівнювати результати різних моделей та методів;

− набір даних містить рівномірну кількість зображень для кожного з 10 класів, що дозволяє уникнути проблем, пов'язаних з дисбалансом класів під час тренування моделей;

− набір даних MNIST доступний у багатьох популярних бібліотеках машинного навчання, таких як TensorFlow та PyTorch, що спрощує його завантаження та використання.

MNIST є ідеальним вибором для початкового тестування та налаштування моделей GAN.

4.3 Огляд метрик для оцінки моделей

Оцінка якості моделей генеративно-змагальних мереж є важливим аспектом для розуміння їх ефективності та порівняння з іншими моделями. Використання лише однієї певної метрики не дасть повну картину для порівняння моделей. Тому поєднання різнобічних метрик зможе дати всебічну оцінку навчених моделей.

У цьому розділі розглянемо три основні метрики, які використовувались у роботі для оцінки моделей GAN: функція втрат на основі бінарної крос-ентропії (Binary Cross-Entropy Loss, BCE), яка буде

вимірюватиметься під час навчання моделей; Frechet Inception Distance (FID) та Inception Score (IS), які дають числові значення, що вимірює схожість між розподілами згенерованих зображень і реальних зображень в наборі даних. Відповідно FID та IS будуть використані вже для оцінки якості згенерованих зразків даних. Кожна з цих метрик має свої переваги та особливості, які дозволяють комплексно оцінити роботу навчених моделей GAN-мережі.

4.3.1. Функція втрат на основі бінарної крос-ентропії (BCE)

Функція втрат на основі бінарної крос-ентропії є стандартною метрикою для оцінки моделей GAN під час навчання. Вона вимірює різницю між передбачуваними та фактичними значеннями для класифікації на дві категорії (реальні та згенеровані зображення). Ця метрика використовується як для генератора, так і для дискримінатора.

Для дискримінатора функція втрат BCE виглядає наступним чином:

$$L_D = -\frac{1}{m}\sum_{\cdot}[log(D(x^{(i)})) + log(1 - D(G(z^{(i)})))], \quad (4.1)$$

де m – кількість прикладів у міні-партії (mini-batch);

$D(x)$ – ймовірність, яку дискримінатор присвоює реальному зображенню x;

$G(z)$ – згенероване зображення, отримане генератором на основі вхідного шуму z.

Для генератора функція втрат BCE виглядає наступним чином:

$$L_G = -\frac{1}{m}\sum_{i=1}^{m} log(D(G(z^{(i)}))), \quad (4.2)$$

де $D(G(z))$ – ймовірність, яку дискримінатор присвоює згенерованому зображенню $G(z)$.

4.3.2 Frechet Inception Distance (FID)

Метрика FID (початкова відстань Фреше) вимірює відстань між розподілами характеристик реальних та згенерованих зображень, використовуючи попередньо натреновану модель Inception, яка навчена класифікувати зображення. FID обчислює різницю між статистичними властивостями двох розподілів. Ця різниця виражається як евклідова відстань між двома наборами чисел, які описують властивості зображень з цих розподілів [29].

Frechet Inception Distance розраховується наступним чином [30]:

$$FID = || \mu_r - \mu_g ||^2 + Tr(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r \cdot \Sigma_g}), \quad (4.3)$$

де μ_r та μ_g – вектори середніх значень характеристик реальних та згенерованих зображень відповідно;

Σ_r та Σ_g – коваріаційні матриці характеристик реальних та згенерованих зображень відповідно;

Tr – слід матриці (сума діагональних елементів).

На рисунку 4.2 наведено приклад FID оцінок для можливих спотворень згенерованих зображень.

Відповідно до графіків на рисункку 4.2 можна побачити що чим менше значення FID, то більше схожі розподілу зображень, тобто краще якість генерації даних моделлю. Вважається, що значення FID менше 50 є дуже хорошим, а все, що більше 100 – поганим.

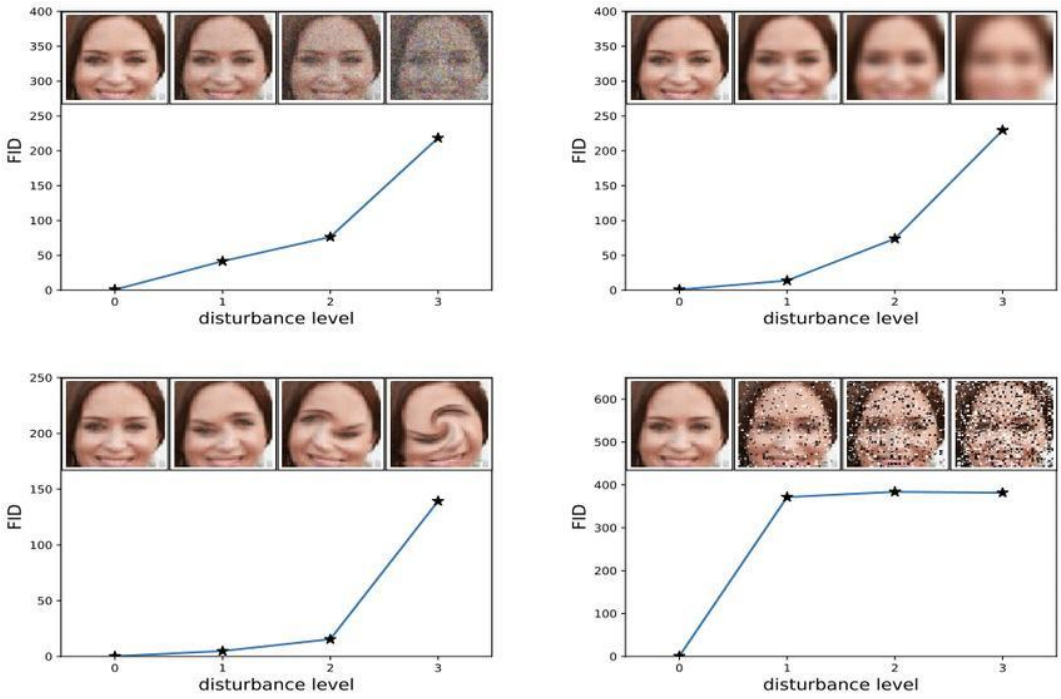


Рисунок 4.2 – Приклад кореляції спотвореного зображення з FID оцінкой

4.3.3 Inception Score (IS)

Начальный балл або Inception Score (IS) оцінює якість згенерованих зображень за допомогою попередньо натренованої моделі Inception. Метрика IS вимірює різноманітність та якість згенерованих зображень, обчислюючи середнє значення KL розбіжності між умовним розподілом класів та граничним розподілом класів згенерованих зображень. Більше значення IS вказує на те, що зображення виглядають реалістично і належать різним класам [31].

Inception Score розраховується наступним чином [32]:

$$IS(G) = exp(E_{x \sim D_G}[D_{KL}(p_G(y \mid x) \;||\; p_G(y))]), \qquad (4.4)$$

де $p(y|x)$ – умовний розподіл ймовірностей для передбачуваних класів y для зображення x, обчислений за допомогою *softmax* виходу мережі Inception;

$p(y)$ – розподіл граничної ймовірності класів реальних зображень у наборі даних, обчислений як частота кожного класу в навчальній вибірці $p(y) = E_x[p(y \mid x)]$;

D_{KL} – розбіжність або дивергенція Кульбака-Лейблера (KL) між двома розподілами ймовірністю $p(y|x)$ і граничною ймовірністю $p(y)$;

E – математичне очікуване значення на згенерованих зображеннях x.

4.4 Реалізація і тренування моделей

Суть проведеного експерименту полягає у дослідженні впливу різних типів вхідного шуму на процес навчання генеративно-змагальних мереж (GAN), а також на якість згенерованих зображень для вже навченої моделі. Для цього було вибрано три різні типи шуму для порівняння:

- гаусівський розподіл шуму (Normal/Gaussian Distribution);
- рівномірний розподіл шуму (Uniform Distribution);
- експоненційний розподіл шуму (Exponential Distribution).

Для кожного з цих типів шуму було навчено окрему модель GAN з використанням набору даних MNIST. Архітектура та реалізація навчання моделей зображена на рисунку 4.3.

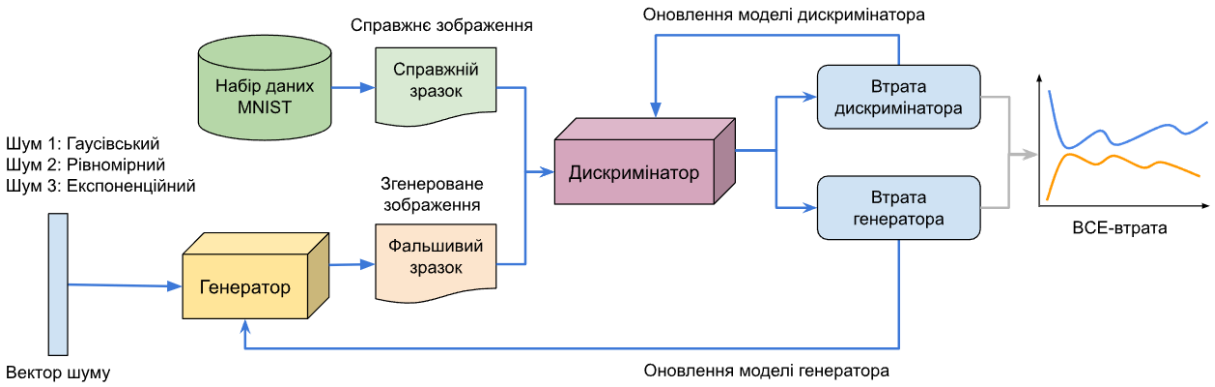


Рисунок 4.3 – Реалізація та тренування моделей

Характеристики тренувань моделей представлені у таблиці 4.1.

Таблиця 4.1 – Характеристика тренувань реалізованих моделей

Характеристика	Нормальний розподіл шуму	Рівномірний розподіл шуму	Експоненційний розподіл шуму
Тривалість однієї епохи	~2 хвилини	~2 хвилини	~2 хвилини
Кількість епох	50	50	50
Загальна кількість зразків для навчання	60 000	60 000	60 000
Розмір даних партии (batch)	256	256	256
Кількість класів для навчання	10	10	10
Загальний час навчання моделі	~2 години	~2 години	~2 години

Короткий алгоритм навчання генератора та дискримінатора, що використовуються в даному дослідження, наведено нижче:

− визначення моделей генератора та дискримінатора. Генератор приймає зумовлений шум (нормальний розподіл, рівномірний розподіл, експоненційний розподіл) плюс мітку класу та створює зображення. Дискримінатор приймає зображення та мітку класу, і визначає, справжнє воно чи фальшиве;

− визначається функція втрат. Втрати дискримінатора обчислюються як сума втрат реальних і фальшивих зображень. Втрати генератора обчислюються з урахуванням того, наскільки дискримінатор вважає фальшиві зображення справжніми;

− використовуються оптимізатори Adam з навчальним кроком 0.0001 (1e-4) для оновлення ваг генератора та дискримінатора;

− основний цикл навчання відбувається через кілька епох. У кожній епосі для кожної партії даних викликається функція навчання, що оновлює

ваги генератора та дискримінатора. Втрати записуються та моделі періодично зберігаються.

Даний підхід є традиційним і дозволяє поступово навчати генератор створювати все більш реалістичні зразки, а дискримінатор ставати все краще розрізнення фальшивих і справжніх зображень.

В результаті отримали 3 навчені моделі GAN-мережі з використанням різних типів вхідних шумів при навчанні. Таким чином ми зможемо порівняти, як кожен тип вхідного шуму впливає на якість навчання моделі та якість генерації зразків.

Також був проведений експеримент на вже навчених моделях GAN-мережі з використанням вхідного шуму, відмінного від того, який був використаний для навчання моделі. Цей експеримент цікавий з погляду розуміння – впливає вхідний шум на генерацію даних для вже навчених моделей чи ні. Реалізація цього експерименту зображено на рисунку 4.4.

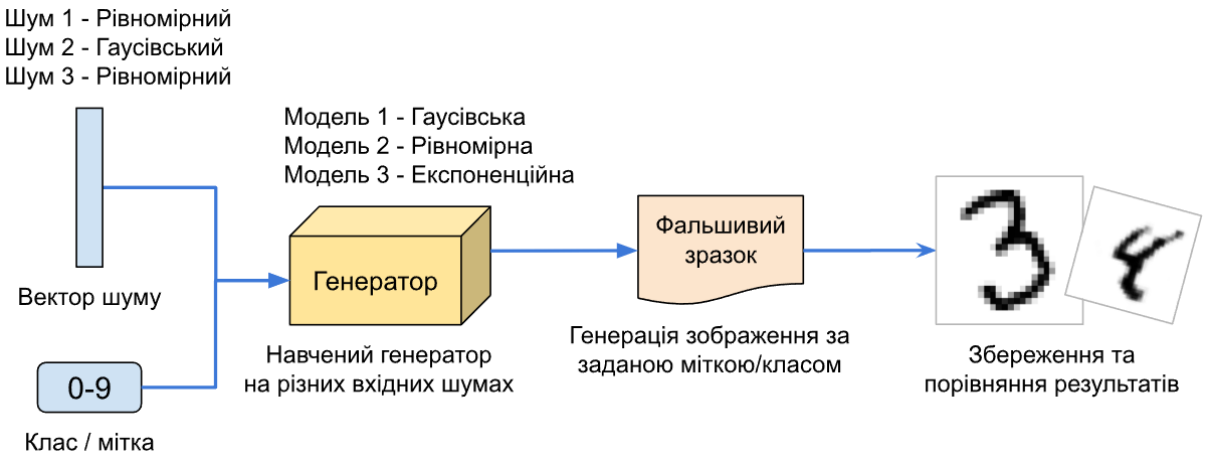


Рисунок 4.4 – Реалізація генерації зразка з різними вхідними шумами

В результаті були отримані згенеровані зразки для кожної моделі з відмінним вхідним шумом, які можна буде порівняти між собою і визначити яка модель найкраще впоралася з цим завданням, і яка найбільш схильна до впливу відмінних вхідних шумів.Вихідний код скрипту на Python для

навчання моделей GAN-мережі та проведення експериментів представлений у додатку А.

4.5 Оцінка якості результатів

Для порівняння результатів буде використано якісну та кількісну оцінку.

4.5.1 Аналіз графіків втрат

На рисунках 4.5, 4.6, 4.7 представлені функції втрат для генератора і дискримінатора під час навчання моделей для однакового набору навчальних даних MNIST і протягом 50 епох.

Давайте проаналізуємо графіки втрат для GAN, навчених із різними розподілами шуму, щоб визначити, який із них може генерувати кращі результати.

Нормальний / гаусівський шум (рисунок 4.5).

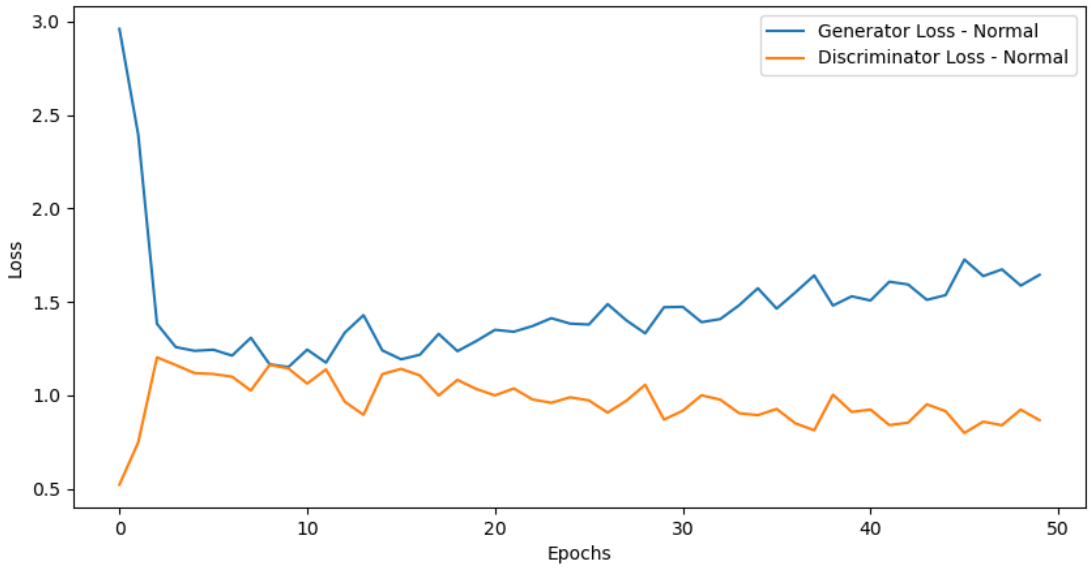


Рисунок 4.5 – Функція втрат – шум із нормальним (гауссівський) розподілом

Втрати генератора спочатку зменшуються, що вказує на покращення у створенні реалістичних зображень. Він показує коливання та загальну тенденцію до зростання, але дещо стабілізується.

Втрати дискримінатора спочатку зменшуються, а потім стабілізуються з коливаннями, що вказує на ефективне навчання та стабільність.

Рівномірний шум (рисунок 4.6).

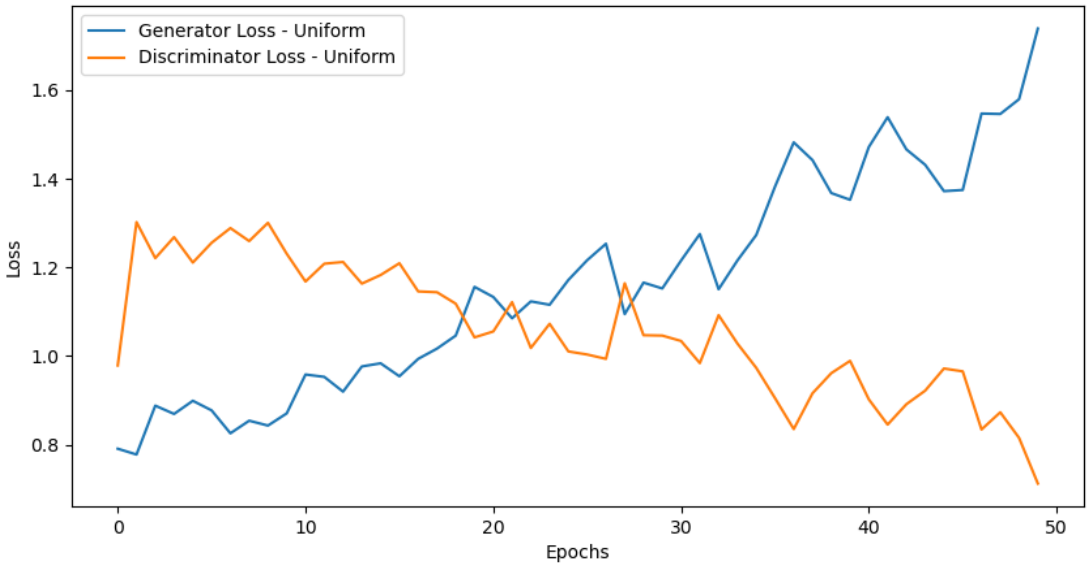


Рисунок 4.6 – Функція втрат – шум із рівномірним розподілом

Втрати генератора починаються з низького рівня, спочатку зменшуються, а потім коливаються. Він демонструє тенденцію до зростання до кінця, що вказує на певні труднощі у створенні незмінно реалістичних зображень.

Втрати дискримінатора починаються з високого рівня, зменшуються та коливаються. Тенденція до зниження вказує на ефективне навчання, але коливання свідчать про постійну адаптацію.

Експоненційний шум (рисунок 4.7).

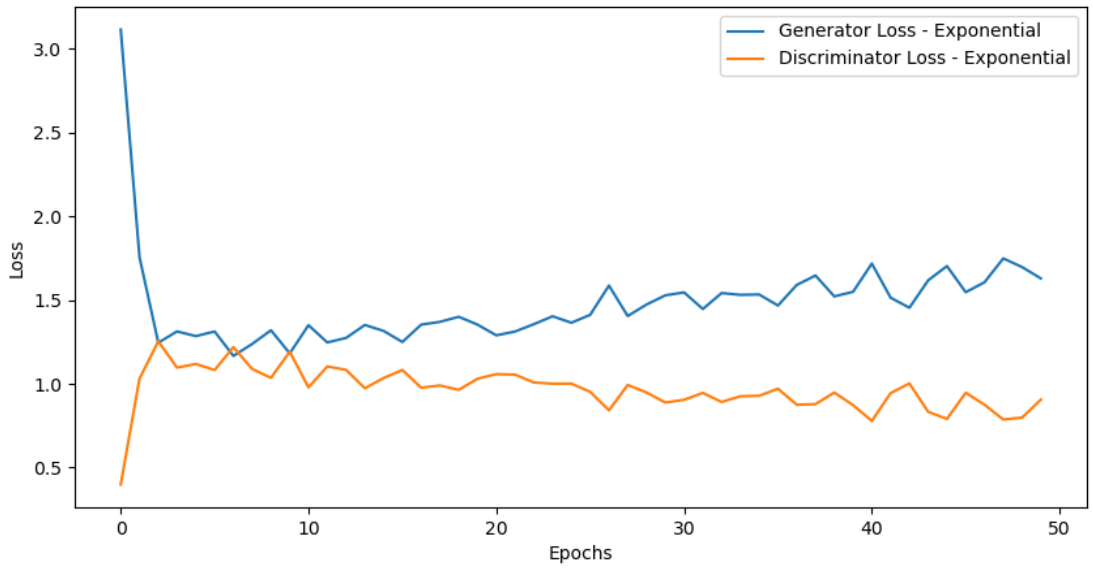


Рисунок 4.7 – Функція втрат – шум із експоненційним розподілом

Втрати генератора починаються з високого рівня, що вказує на початкові труднощі в обмані дискримінатора. Він зменшується, а потім демонструє деякі коливання, але має загальну тенденцію до зростання.

Втрати дискримінатора починаються з низького рівня, що вказує на те, що спочатку він працює добре. Він стає відносно стабільним із невеликими коливаннями, що означає, що дискримінатор залишається ефективним протягом навчання.

На основі аналізу графіків втрат нормальний шум і експоненційний шум видається найбільш перспективним для отримання кращих результатів. Втрати генератора стабілізуються після початкового зменшення, а втрати дискримінатора також демонструють постійну тенденцію до зниження, що вказує на ефективне навчання та створення реалістичних зображень. Рівномірний шум показують більше коливань і тенденцію до зростання втрат генератора до кінця, що вказує на потенційні труднощі при постійному створенні високоякісних зображень.

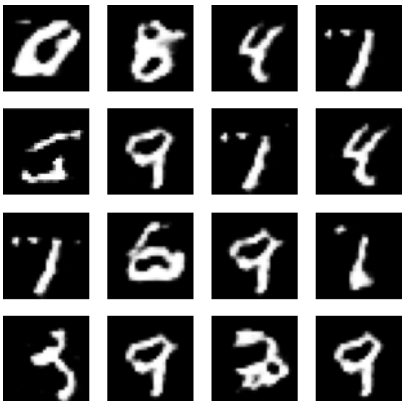
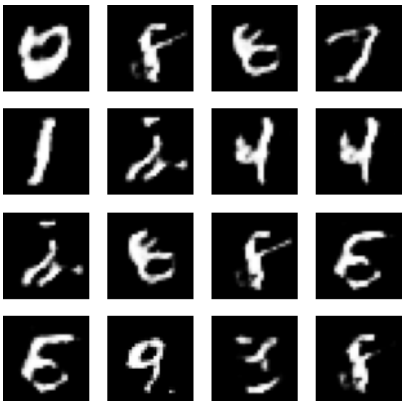
Щоб підтвердити ці спостереження, наступним оцінимо згенеровані зображення візуально. Також можна буде обчислити кількісні показники,

такі як показник початкової відстані Фреше (FID) для кожного типу шуму. Це дозволить отримати більш конкретну оцінку якості зображення.

4.5.2 Візуальний аналіз результатів

Для кожної навченої моделі були згенеровані випадкові зразки, представлені для порівняння в таблиці 4.2.

Таблиця 4.2 – Приклади генерації цифр навчених моделей

Модель 1. Розподіл шуму гауссівський	Модель 2. Розподіл шуму рівномірний	Модель 3. Розподіл шуму експоненціальний
		

Модель 1. Розподіл шуму гауссівський. Загалом зразки непогано генеруються. Візуально спостерігаються недоліки для генерацій зразків із мітками 1, 2, 5, 7.

Модель 2. Розподіл шуму рівномірний. Результати залишають бажати кращого. Генератор впорався більш менш тільки з мітками 0, 1, 3, 5, 7. Це свідчить про те, що з даним видом шуму генератору потрібно більше часу для навчання на даному наборі даних.

Модель 3. Розподіл шуму експоненційний. Загалом зразки непогано генеруються. Є питання щодо генерацій зразків з мітками 2, 6, 8.

На основі візуального огляду проведений аналіз графіків функцій втрат підтверджується. Нормальний шум і експоненційний шум видається

найперспективнішим для отримання кращих результатів. З деякими мітками краще впоралася модель навчена на шумі з нормальним розподілом, а для інших метох із експоненційний розподілом. Модель із рівномірним шумом показала найгірший результат.

4.5.2 Аналіз результатів під час використання різних шумів

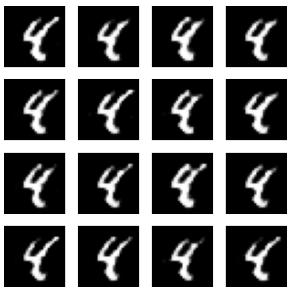
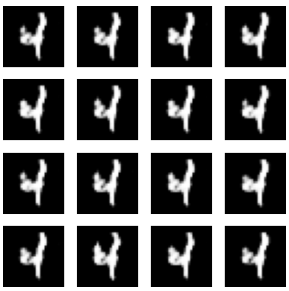
Попередні проведені есперименти використовували для генерації нових зразків той же шум на вході генераторі, що й використовувався під час навчання генератора.

В даному експерименті буде подаватися на вхід вже навченого генератора відмінний шум від того, що був використаний при навчанні. Як відмінний шум був обраний розподіл шуму рівномірний, так як модель з цим шумом сильно відрізняється від інших моделей. А для моделі з рівномірним шумом буде використаний шум із нормальним/гаусівським розподілом:

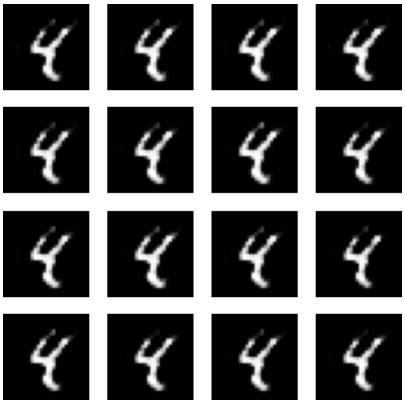
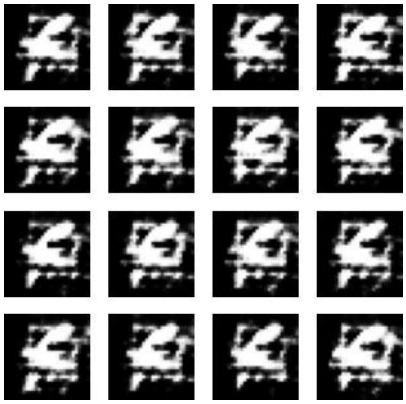
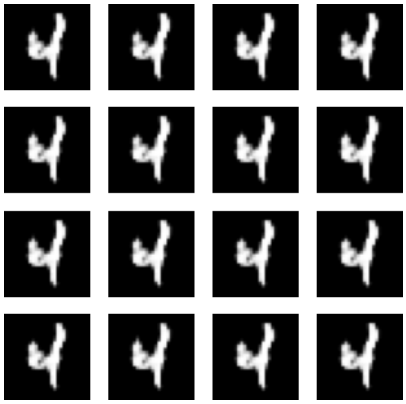
- модель 1 (гауссівська) – шум рівномірний;
- модель 2 (рівномірна) – шум – гауссівський;
- модель 3 (експоненційна) – шум рівномірний.

З візуальними результатами проведеного експерименту можна ознайомитись у таблицях 4.3 та 4.4.

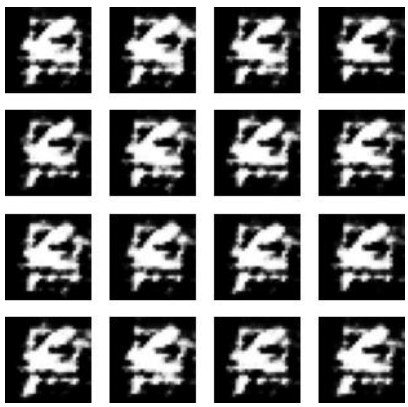
Таблиця 4.3 – Приклади генерації цифри 4

Модель 1. Гауссівська	Модель 2. Рівномірна	Модель 3. Експоненційна
 Оригінальний шум – гауссівський	 Оригінальний шум – Рівномірний	 Оригінальний шум – Експоненційний

Продовження таблиці 4.3

Модель 1. Гауссівська	Модель 2. Рівномірна	Модель 3. Експоненційна
		
Шум рівномірний	Шум – гауссівський	Шум рівномірний

Таблиця 4.4 – Приклади генерації цифри 5

Модель 1. Гауссівська	Модель 2. Рівномірна	Модель 3. Експоненційна
		
Оригінальний шум – гауссівський	Оригінальний шум – Рівномірний	Оригінальний шум – Експоненційний
		
Шум рівномірний	Шум – гауссівський	Шум рівномірний

На основі візуального огляду попередні аналізи повністю підтверджуються. А також було виявлено, що модель навчена з рівномірним розподілом шуму схильна до сильного впливу від використання інших видів шуму при генерації зразків.

4.5.3 Кількісний аналіз якості моделей

Для кількісного порівняння було пораховано метрики FID та IS описані в підрозділах 4.3.2 та 4.3.3. Було згенеровано 1000 зображень з різними мітками для кожної моделі і для прорахунку FID використовувалися реальні дані навчального набору даних по однакових мітках. Результати представлені у таблиці 4.5.

Таблиця 4.5 – Кількісна оцінка якості моделі

Метркиа	Нормальний розподіл шуму	Рівномірний розподіл шуму	Експоненційний розподіл шуму
Початкова відстань Фреше (FID)	156.272	257.465	168.474
Начальный балл (IS)	2.42	1.57	2.27

Приблизно оцінити початкову оцінку (IS) на основі оцінок FID непросто, оскільки ці два показники оцінюють різні аспекти створених зображень. Однак вони певною мірою корелюють – нижчі показники FID зазвичай вказують на зображення вищої якості, що може відповідати вищим значенням IS.

Згідно з отриманими результатами зазначені в таблиці 4.4 висока оцінка FID та низькі початкові оцінки IS вказує на низьку якість та різноманітність згенерованих зображень. Це означає, що поточні моделі GAN мають можливості для покращення, щоб створювати більш реалістичні та різноманітні зображення.

Отже, кількісні метрики також показали, що нормальний та експоненційний розподіли шуму дають кращі результати, ніж рівномірний при навчанні моделей з набором даних MNIST.

4.6 Перспективи розвитку

У майбутній дослідницькій роботі можливими напрями для подальшої роботи можуть бути:

– одним із перспективних напрямків розвитку є проведення експериментів з навчальними наборами даних, які містять фотографії людей, наприклад, набором даних CelebA (CelebFaces Attributes Dataset). CelebA складається з більш ніж 200 тисяч зображень обличчя знаменитостей, що робить його чудовим вибором для тестування генеративно-змагальних мереж. Застосування GAN до набору даних CelebA дозволяє генерувати високоякісні зображення облич, що має численні застосування у різних галузях, включаючи розваги, безпеку та охорону здоров'я. Використання цього набору даних дозволить дослідити, як різні типи вхідного шуму впливають на генерацію зображень складніших і реалістичних об'єктів. Зокрема, можна оцінити якість згенерованих зображень за допомогою метрик Inception Score (IS) та Frechet Inception Distance (FID), а також візуальної оцінки;

– ще одним важливим напрямком є дослідження дифузійних моделей та проведення експериментів з шумами для цих моделей. Дифузійні моделі, такі як DDPM, представляють новий підхід до генерації зображень, який відрізняється від традиційних GAN. Дифузійні моделі використовують процес додавання шуму до зображень і поступового його видалення для відновлення вихідного зображення. Ці моделі демонструють високу якість згенерованих зображень і можуть бути більш стабільними під час навчання. Дослідження різних типів шумів для дифузійних моделей може допомогти оптимізувати цей процес і покращити результати;

– однією з ключових задач у розвитку генеративних моделей є виявлення оптимальних вхідних шумів, які зможуть прискорити процес навчання нових моделей GAN. Пошук таких шумів може суттєво знизити час та ресурси, необхідні для тренування моделей, а також покращити якість згенерованих зображень. Для цього необхідно провести комплексні експерименти з різними типами шумів включаючи фрактальний, перлін-шум та інші альтернативні підходи. Під час цих експериментів слід вимірювати функції втрат (наприклад, BCE), а також використовувати додаткові метрики (FID, IS) для об'єктивного порівняння ефективності навчання та якості згенерованих зображень.

Перспективи розвитку генеративно-змагальних мереж охоплюють широкий спектр напрямків, включаючи експерименти з новими наборами даних, дослідження дифузійних моделей та оптимізацію вхідних шумів для прискорення процесу навчання. Ці дослідження сприятимуть покращенню якості згенерованих зображень та розширенню застосувань GAN у різних галузях.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено всебічний аналіз архітектури та принципів роботи генеративно-змагальних нейронних мереж. Дослідження охоплювало різноманітні аспекти створення та функціонування цих мереж, що дозволило глибше зрозуміти їх потенціал та можливі практичні застосування.

Було проведено детальний аналіз генеративних моделей, серед яких особливу увагу приділено генеративним змагальним мережам (GAN), варіаційним автокодувальникам (VAE), моделям на базі потоку (Flow-based models) та іншим. Кожна з цих моделей має свої унікальні характеристики та застосування, що робить їх важливими для різних дослідницьких і практичних завдань.

У роботі також розглянуто стандартні види шумів, такі як гаусівський, рівномірний, логнормальний, експоненційний та розподіл Лапласа. Окрім того, було досліджено альтернативні види шумів, включаючи шум Перліна, фрактальний шум, шум зображення, додавання випадкових векторів та випадкові спотворення. Це дозволило розглянути різні типи шумів, які можуть використовуватися в процесі навчання та впливають на якість генерації зображень у GAN-мережах та дифузійних моделях (DDPM).

Було проведено навчання генеративно-змагальних мереж на наборі даних MNIST для різних видів вхідного шуму. Під час експерименту було використано гаусівський, рівномірний та експоненційний шуми. Порівняння навчених моделей показало, що гаусівський та експоненційний шуми впоралися з навчанням GAN-мережі найкращим чином, демонструючи високу якість згенерованих зображень та стабільний процес навчання.

У результаті дослідження було виявлено, що гаусівський шум є найоптимальнішим для навчання GAN-мережі на наборі даних MNIST, що підтверджується низькими значеннями функцій втрат та високою якістю

згенерованих зображень. Експоненційний шум також показав добрі результати, хоча його застосування може потребувати додаткової оптимізації.

В результаті дослідження було виявлено, що альтернативні види шумів частіше використовуються в дифузійних моделях, оскільки вони дозволяють створювати більш реалістичні та детальні зображення. Водночас, у GAN-мережах за умовчанням використовується гауссівський розподіл шуму, що забезпечує стабільне навчання та високоякісну генерацію зображень.

Майбутній розвиток та напрямок досліджень у цій галузі можуть включати проведення експериментів з більшими та складнішими наборами даних, такими як CelebA, дослідження дифузійних моделей та оптимізацію вхідних шумів для прискорення процесу навчання нових моделей GAN. Ці перспективи дозволять покращити якість згенерованих зображень та розширити застосування генеративних моделей у різних сферах, включаючи медицину, розваги, безпеку та багато інших.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Generative Adversarial Networks / Ian J. Goodfellow et al.. June 2014. URL: https://arxiv.org/abs/1406.2661 (дата звернення: 10.05.2024).

2. A review of Generative Adversarial Networks (GANs) and its applications in a wide variety of disciplines - From Medical to Remote Sensing / Ankan Dash et el., October 2021. URL: https://arxiv.org/abs/2110.01442 (дата звернення: 10.05.2024).

3. Text-to-Image Synthesis with Generative Models: Methods, Datasets, Performance Metrics, Challenges, and Future Direction / Sarah K. Alhabeeb et al., DOI: 10.1109/ACCESS.2024.3365043 (дата звернення: 10.05.2024).

4. Generative adversarial network: An overview of theory and applications / Alankrita Aggarwal et al., April 2021. DOI: 10.1016/j.jjimei.2020.100004 (дата звернення: 10.05.2024).

5. Ten Years of Generative Adversarial Nets (GANs): A survey of the state-of-the-art / Tanujit Chakraborty et al., August 2023. URL: https://arxiv.org/abs/2308.16316 (дата звернення: 10.05.2024).

6. O'Shea, K. (2015, November 26). An Introduction to Convolutional Neural Networks. arXiv.org. https://arxiv.org/abs/1511.08458 (дата звернення: 10.05.2024).

7. Scene Reconstruction From 4D Radar Data with GAN and Diffusion. https://www.diva-portal.org/smash/get/diva2:1799731/FULLTEXT01.pdf (дата звернення: 10.05.2024).

8. Durall, R. (2020, December 17). Combating Mode Collapse in GAN training: An Empirical Analysis. UR: https://arxiv.org/abs/2012.09673 (дата звернення: 10.05.2024).

9. Arjovsky, M. (2017, January 17). Towards Principled Methods for Training Generative Adversarial Networks. arXiv.org. https://arxiv.org/abs/1701.04862 (дата звернення: 10.05.2024).

10. Kingma D. P., Welling M. Auto-Encoding Variational Bayes. https://arxiv.org/abs/1312.6114 (дата звернення: 10.05.2024).

11. Tomczak J. M. Flow-Based Models. Deep Generative Modeling. Cham, 2021. P. 27–56.

12. Prafulla Dhariwal, Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. June 2021. https://arxiv.org/pdf/2105.05233.pdf (дата звернення: 10.05.2024).

13. Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. April 2022. https://arxiv.org/pdf/2112.10752.pdf (дата звернення: 10.05.2024).

14. Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, Tim Salimans. Cascaded Diffusion Models for High Fidelity Image Generation. April 2022. URL: https://cascaded-diffusion.github.io/assets/cascaded_diffusion.pdf (дата звернення: 10.05.2024).

15. Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba & Ruslan Salakhutdinov. Generating Images From Captions With Attention. ICLR 2016. https://arxiv.org/pdf/1511.02793.pdf (дата звернення: 10.05.2024).

16. Han Zhang, Jing Yu Koh, Jason Baldridge, Honglak Lee, Yinfei Yang. Cross-Modal Contrastive Learning for Text-to-Image Generation. April 2022. https://arxiv.org/pdf/2101.04702.pdf (дата звернення: 10.05.2024).

17. Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, Ilya Sutskever. Zero-Shot Text-to-Image Generation. February 2021. https://arxiv.org/pdf/2102.12092.pdf (дата звернення: 10.05.2024).

18. Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, Mark Chen. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. March 2022. https://arxiv.org/pdf/2112.10741.pdf (дата звернення: 10.05.2024).

19. Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. April 2022. https://arxiv.org/pdf/2204.06125.pdf (дата звернення: 10.05.2024).

20. Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, Amnon Shashua, Kevin Leyton-Brown, Yoav Shoham. Standing on the Shoulders of Giant Frozen Language Models. April 2022. https://arxiv.org/pdf/2204.10019.pdf (дата звернення: 10.05.2024).

21. Blog: Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer. February 2020. URL: https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html (дата звернення: 10.05.2024).

22. Radford, A. (2015, November 19). Unsupervised Representation Learning with Deep Convolutional. arXiv.org. https://arxiv.org/abs/1511.06434 (дата звернення: 10.05.2024).

23. Dhivya K., Sharfaras Navas N. Text to Realistic Image Generation Using Stackgan. 2020 7th International Conference on Smart Structures and Systems (ICSSS), Chennai, India, 23–24 July 2020. 2020

24. Denoising Diffusion Probabilistic Models. URL: https://arxiv.org/pdf/2006.11239 (дата звернення: 10.05.2024).

25. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. URL: https://arxiv.org/pdf/1503.03585 (дата звернення: 10.05.2024).

26. Formation of alternative approaches to noise generation in gan networks. https://doi.org/10.30837/IYF.IIS.2024.094 (дата звернення: 10.05.2024).

27. Generating Perlin Noise / Andreas Jönsson, February 2002. URL: https://www.angelcode.com/dev/perlin/perlin.html (дата звернення: 10.05.2024).

28. Value Noise and Procedural Patterns. URL: https://www.scratchapixel.com/lessons/procedural-generation-virtual-

worlds/procedural-patterns-noise-part-1/simple-pattern-examples.html (дата звернення: 10.05.2024).

29. Frechet Inception Distance (FID) for Evaluating GANs, Yu Yu, Weibin Zhang, Yun Deng, China University of Mining and Technology-Beijing. September 2021

30. GANsTrained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. https://arxiv.org/pdf/1706.08500 (дата звернення: 10.05.2024).

31. Evaluation Metrics for Conditional Image Generation. URL: https://arxiv.org/abs/2004.12361 (дата звернення: 10.05.2024).

32. Improved Techniques for Training GANs. URL: https://arxiv.org/pdf/1606.03498 (дата звернення: 10.05.2024).