НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

"На правах рукопису"

УДК 004.89

ДО ЗАХИСТУ ДОПУЩЕНО

Завідувач кафедри

_____ Олександр РОЛІК

“ ____ ” _____ 2023 р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за освітньо-науковою програмою

«*Інформаційні управляючі системи та технології*»

зі спеціальності 126 «*Інформаційні системи та технології*»

на тему:

«Генерація аватарів за категоріями з використанням мережі GAN»

Виконав:
студент VI курсу, групи ІС-11мн
Іванов Анатолій Ігорович _____

Керівник:
д.т.н., професор кафедри ІСТ
Онищенко Вікторія Валеріївна _____

Рецензент:
д.фіз.-мат.н., снс, професор кафедри ЦТЕ ННІАТЕ
Матичин Іван Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-наукова програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Олександр РОЛІК

«__» _____2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Іванов Анатолій Ігорович

1. Тема дисертації «Генерація аватарів за категоріями з використанням мережі GAN», науковий керівник дисертації Онищенко Вікторія Валеріївна, д.т.н., професор кафедри ІСТ, затверджені наказом по університету від «20» березня 2023 р. № 1275-с

2. Строк подання студентом дисертації “12” травня 2023 р.

3. Об’єкт дослідження: процес генерації зображень у вигляді аватарів у соціальні мережі за категоріями.

4. Предмет дослідження: методи генерації зображень з випадкового (гаусівського) шуму за категоріями.

5. Перелік завдань, які потрібно розробити
Виконати огляд існуючих методів та засобів генерації зображень; виконати огляд моделей GAN для генерації зображень за категоріями; здійснити порівняльний аналіз різних методів та моделей; проаналізувати алгоритми для перевірки моделей на якість генерації; модифікувати існуючі моделі для генерації аватарів за категоріями; розробити програмну реалізацію отриманої моделі; провести

експериментальні дослідження для розробленої моделі; виконати аналіз отриманих результатів.

6. Перелік графічного (ілюстративного) матеріалу

Архітектура створеної моделі GAN; Архітектура генератору; Архітектура дискримінатору; Діаграма послідовностей процесу навчання моделі; Діаграма станів процесу перетворення та класифікації даних; Схема моделі генератору; Схема моделі дискримінатору.

7. Орієнтовний перелік публікацій
Image generations techniques using Generative adversarial networks / V. Onyshchenko, A. Ivanov. // Adaptive systems of automatic control. – 2023. – №1(42).

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання “ 31 ” січня 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Огляд існуючих методів та засобів генерації зображень.	28.02.2023	
2	Огляд існуючих моделей GAN для генерації зображень за категоріями.	05.03.2023	
3	Аналіз оглянутих методів та моделей	21.03.2023	
4	Визначення та аналіз алгоритмів перевірки якості моделей.	24.03.2023	
5	Розробка модифікації моделі для генерації аватарів за категоріями.	13.04.2023	
7	Проведення експериментальних досліджень модифікованої моделі	16.04.2023	
8	Аналіз отриманих результатів	18.04.2023	
9	Оформлення документації	19.04.2023	
10	Подання роботи на попередній захист	20.04.2023	
11	Подання роботи на основний захист	12.05.2023	

Студент _____ *Анатолій ІВАНОВ*

Науковий керівник _____ *Вікторія ОНИЩЕНКО*

РЕФЕРАТ

Магістерська дисертація: 126 с., 39 рис., 13 табл., 45 джерел, 4 додатки.

Актуальність. Швидкий ріст кількості користувачів соціальних мереж, а також веб-сервісів, що мають кабінет користувача та його взаємодію з іншими людьми у мережі створюють проблему їх розпізнавання. І, оскільки, багато людей візуали, то спілкуючись з іншими вони по-перше запам’ятовують їх аватари. Через це багато систем використовують ініціали користувача або ж певне стандартне зображення для тих, хто не бажає додавати власне фото. При цьому, існуючі сервіси та алгоритми надають можливість генерації лише за існуючим зображенням, перетворюючи його у певному стилі. Таким чином, генерація зображень за категорією може вирішити проблему генерації аватарів за певною тематикою, що не буде містити ніякої відсилки до зовнішнього вигляду користувача та буде обиратися лише за його вподобаннями.

Мета дослідження – модифікація моделі GAN для її використання у генерації аватарів за певними категоріями. При цьому, підвищення її ефективності за рахунок зниження часу навчання моделі, не знижуючи якості отриманого зображення.

Для досягнення мети необхідно виконати наступні **завдання**:

- виконати огляд існуючих методів та засобів генерації зображень;
- виконати огляд моделей gan для генерації зображень за категоріями;
- здійснити порівняльний аналіз різних методів та моделей;
- проаналізувати алгоритми для перевірки моделей на якість генерації;
- модифікувати існуючі моделі для генерації аватарів за категоріями;
- розробити програмну реалізацію отриманої моделі;
- провести експериментальні дослідження для розробленої моделі;
- виконати аналіз отриманих результатів.

Об’єкт дослідження – процес генерації зображень у вигляді аватарів у соціальні мережі за категоріями.

Предмет дослідження – методи генерації зображень з випадкового (гаусівського) шуму за категоріями.

Наукова новизна отриманих результатів

Розробка моделі генерації зображень, що є модифікацією існуючих моделей, та показує кращі результати, а також у програмній реалізації даної моделі для генерації аватарів, що можуть бути використані у подальшому на веб-ресурсах для анонімної ідентифікації користувачів.

Публікації. Onyshchenko V. Image generations techniques using Generative adversarial networks / V. Onyshchenko, A. Ivanov. // Adaptive systems of automatic control. – 2023. – №1(42).

ГЕНЕРУВАННЯ ЗОБРАЖЕНЬ, АВАТАРИ, GENERATIVE ADVERSARIAL NETWORK, MIN MAX OPTIMIZATION, WASSERSTEIN LOSS FUNCTION, CGAN, КАТЕГОРІЇ.

ABSTRACT

Master’s Thesis: 126 pages, 39 figures, 13 tables, 45 references, 4 appendices.

Relevance. The rapid growth of social media users, as well as web services in which users have their own accounts and interact with others, creates a problem of their identification. Since many people are visually learned, they will first remember each other by their own avatars. Therefore, many systems have them in a way of user initials for those who don’t wish to set it. Existing services and algorithms provides the ability to generate avatars only from an existing photo, like user photos by transformations of it into a certain style. Thus, generating images by category can solve the problem of generating avatars via a specific category, which will not contain any reference to the user's appearance and will be selected only by their preferences.

Objective of the research – modification of the GAN model for its usage in a avatars generation process by specific categories. At the same time, improving its efficiency by reducing the speed of generation and model training without compromising the quality of the resulting image.

To achieve the goal, the following tasks must be performed:

- conduct a review of existing methods of image generation;
- conduct a review of GAN algorithms for generating images by categories;
- perform a comparative analysis of different methods and models;
- find algorithms to verify the quality of model generation;
- modify existing models for generating avatars by categories;
- develop a software implementation of the obtained model;
- conduct experimental studies for the algorithm modification;
- analyze the obtained results.

The object of the research is the process of generating images in the form of avatars in social media by categories.

The subject of the research are image generation methods from random (Gaussian) noise by categories.

Scientific novelty of the obtained results

The existing image generation models for the investigated problem were modified based on the GAN model and their training speed was optimized. The use of this algorithm allows for an improvement in both the quality and speed of image generation.

Publications. Onyshchenko V. Image generations techniques using Generative adversarial networks / V. Onyshchenko, A. Ivanov. // Adaptive systems of automatic control. – 2023. – №1(42).

IMAGE GENERATION, AVATARS, GENERATIVE ADVERSARIAL NETWORK, MIN MAX OPTIMIZATION, WASSERSTEIN LOSS FUNCTION, CGAN, CATEGORIES.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GAN – Generative adversarial network, підхід до генеративного моделювання, що використовує методи глибокого навчання, такі як згорткові нейронні мережі.

CGAN – Conditional generative adversarial network, тип GAN, який включає умовну генерацію зображень за допомогою моделі генератору.

FID – The Fréchet Inception Distance, одна з метрик оцінки мережі GAN, що відображає наскільки сильно згенеровані зображення відрізняються від реальних. Менше значення відображає кращий результат роботи генератору мережі.

KID – Kernel Inception Distance, модифікація метрики FID, що допомагає подолати її головний недолік – неможливість застосування на зображеннях, згенерованих за допомогою не гаусівського розподілу.

IS – Inception Score, ще один тип метрік, який відображає наскільки сильно згенеровані зображення відрізняються від реальних, але більші значення якого відповідають за кращу модель генератору.

TF – Tensorflow, бібліотека глибокого навчання, що була використана у роботі. Має підтримку мови програмування Python, працює на основі мови C++.

ВСТУП

Інтернет з'явився лише декілька десятиліть тому і з кожним роком активно збільшував свою популярність. Через це, сьогодні майже немає людей, які б ні разу не чули та не використовували його. З розвитком глобальної мережі інтернет, розширювалося і покращувалося його наповнення. Наразі, існує безліч різноманітних веб-сайтів, що пропонують користувачам різноманітні послуги.

Деякі з цих сайтів в дійсності надають, що вони пропонують, інші надають частково, але майже у всіх із них присутня можливість, що зазвичай стає необхідністю, зареєструватися та створити власний кабінет чи профіль для отримання повного спектру послуг, що надається даним веб-ресурсом. При цьому, однією з основних та стандартних частин цього кабінету є фотографія або зображення користувача, яке зазвичай залишається порожнім, або ж містить певні стандартні значення, такі як перші літери ім’я та прізвища або зображення контуру уявного користувача.

З іншого боку, приблизно з середини минулого десятиріччя почала стрімко розвиватися сфера використання нейронних мереж та глибокого навчання для генерації контенту у текстовому, графічному, аудіо та відео-форматі. І хоча, більшість із запропонованих на ринку рішень генерують лише певні базові елементи, які неможливо використовувати без втручання користувача, існують деякі продукти, що можуть спокійно замінювати працю людину, наприклад, генерувати зображення, що є дуже схожими на реальні витвори мистецтва відомих авторів, які досить складно відрізнити від оригіналів.

А, отже, існує можливість об’єднання даних нейронних мереж для генерації зображень та необхідності «ідентифікації» користувача за допомогою додавання власної фотографії або зображення. Для цього, необхідно модифікувати мережу настільки, щоб вона могла згенерувати зображення аватарів, що не буде відображати реальних людей, а буде чимось більш абстрактним, при цьому дані зображення, могли б використовуватися людьми для своїх профілів, що будуть

достатньо унікальними для використання та дозволять додати можливість легкої ідентифікації користувачів на певних форумах/коментарях та інших веб-ресурсах.

Отже, дана робота є актуальною, оскільки у ній розглядається модифікація існуючої моделі генерації зображень, що, як мінімум, буде ще одним кроком у розвитку даної сфери та може сприяти вирішенню досить великої кількості проблем у галузі самоідентифікації користувачів, що не бажають завантажувати власні фото на веб-ресурсах.

Також, виконана програмна реалізація даної моделі та її навчання на обраному наборі даних. Це надає можливість використовувати дану мережу у реальних застосунках для подальшої генерації аватарів користувачів, що містяться майже на кожному веб-застосунку, а отже, необхідні кожному користувачу.

Об’єктом дослідження є процес генерації зображень у вигляді аватарів у соціальні мережі за категоріями.. Предметом дослідження є методи генерації зображень з випадкового (гаусівського шуму) за категоріями.

Метою даної роботи є модифікація моделі GAN для її використання у генерації аватарів за певними категоріями. При цьому, підвищення її ефективності за рахунок зниження часу навчання моделі, не знижуючи якості отриманого зображення.

З практичної точки зору розроблена та навчена модель може стати корисним кроком для її використання на веб-ресурсах, або ж, створення певних сервісів для генерації аватарів за певними специфічними запитами.

Пояснювальна записка магістерської дисертації складається з наступних розділів: вступ, основні, висновки, список використаних джерел із 45 найменувань та три додатки. Графічна частина містить 39 рисунків та 7 креслеників. Загальний обсяг 126 сторінок.

1 СУЧАСНИЙ СТАН ПРОБЛЕМИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

1.1 Генерація зображень

Розглядаючи стан проблеми, необхідно більш чітко окреслити її. В даному випадку, розглянемо генерацію зображень, що можуть бути використані користувачами соціальних мереж як фото свого профілю у персональних кабінетах на сайтах, як генерацію даних з умовами, що будуть відповідати за те, який саме тип аватару бажає отримати користувач.

1.2 Огляд існуючих алгоритмів генерації зображень

Наразі, існує досить багато різноманітних алгоритмів для генерації зображень. Деякі з них можуть бути використані для отримання певних типів зображень, інші ж мають більш загальну структуру та можуть працювати з різноманітними типами вхідних, а отже і вихідних даних. Такі моделі зазвичай видають результат, що в більшості залежить від того, на яких даних дана модель навчалася. При цьому, всі моделі генерації використовують deep neural network для отримання результатів.

Найбільш популярними алгоритмами генерації зображення є:

- GAN, або Generative Adversarial Networks;
- варіаційні автокодери (VAE);
- Deep Dream;
- фрактальні алгоритми;
- стільникові автомати.

Generative Adversarial Networks – нейронна мережа, що складається з двох мереж – генератора та дискримінатора [24]. Генератор створює зображення, а дискримінатор оцінює їх, намагаючись визначити справжні вони або ж підроблені. Так, оцінюючи один одного, дві мережі навчаються.

Варіаційні автокодери (VAE) – ще один тип нейронної мережі, яку можна використовувати для створення зображень [25]. Вона працюює, розглядаючи низьковимірне представлення зображення, після чого намагається згенерувати нові зображення, що будуть містити частини цього представлення.

Deep Dream – ще один алгоритм генерації, що заснований на використанні комп'ютерного зору, який був розроблений компанією Google у 2015 році [26]. Принцип роботи полягає у аналізі зображенні та застосуванні шаблонів, що є в інших зображеннях. У результаті виконання генерації отримується зображення, що містить елементи першого та переймає стиль іншого наданих заздалегідь.

Фрактальні алгоритми – створюють зображення через повторення певного простого шаблону на одному полотні в різних масштабах [27]. Результатом відпрацювання даного алгоритму є зображення з високою деталізацією, яке часто використовується в цифровому мистецтві.

Стільникові автомати – тип алгоритмів, які створюють зображення шляхом імітації поведінки окремих комірок у сітці [28]. Поведінка кожної клітини визначається набором правил, які з часом можуть створювати складні та досить складно передбачувані моделі.

Інші алгоритми зазвичай виконують лише частину роботи, так як додаткове перетворення та використовуються у розглянутих алгоритмах як певні модифікації для покращення результату або пришвидшення часу виконання.

Як можна побачити з [8] більшість існуючих алгоритмів, що надають найкращий результат побудовані саме на основі моделі GAN. Через це, в даній роботі розглянемо генерацію саме використовуючи перший алгоритм, оскільки саме він надає найбільшу варіативність результатів та дозволяє отримувати зображення у багатьох різноманітних стилях без необхідності повної зміни моделі та в цілому залежить лише від набору даних.

1.3 Огляд модифікацій моделі GAN

Існує досить велика кількість різних модифікацій GAN, що використовуються для генерації зображень [12]. Вони включають у себе зміни, що були зроблені задля покращення моделі для виконання певних більш специфічних задач, таких як покращення зображень [13], генерація облич або їх частин [14-15], створення номерних знаків [16] та вивчення наркотичних елементів [17].

При цьому, існує деяка кількість стандартних моделей, що використовуються для вирішення даної проблеми. Так, WGAN [17] допомагає вирішити стандартну проблему спадання градієнту, оскільки дискримінатору потрібно навчатися набагато довше ніж генератору для того щоб натренувати модель. Але, з іншого боку, дана модель продукує гірші результати [19] та може випадково зійтися. Через це було запропонована деяка кількість різноманітних покращень [20-21], що намагаються подолати проблему сходження додаючи додаткові обчислення.

В цілому ж, модифікації GAN розділяються на чотири основні категорії, до яких вже потім додаються певні зміни. Даними категоріями є:

- CGAN, що використовує додатковий вхід для додавання мітки до кожного входу та результату генерації. [22], що дозволяє отримувати більш детерміновані результати [23] та надати можливість генерувати однією моделлю великої кількості екземплярів різноманітних даних;
- Progressive GAN (PGAN) – модифікація моделі, який створює зображення з високою роздільною здатністю, починаючи із тих самих зображень, але з низьким розширенням, що дозволяє значно покращити якість зображення та отримати більше розширення без використання певних складних та потужних алгоритмів, загальний принцип роботи якої зображено на рис. 1.1;

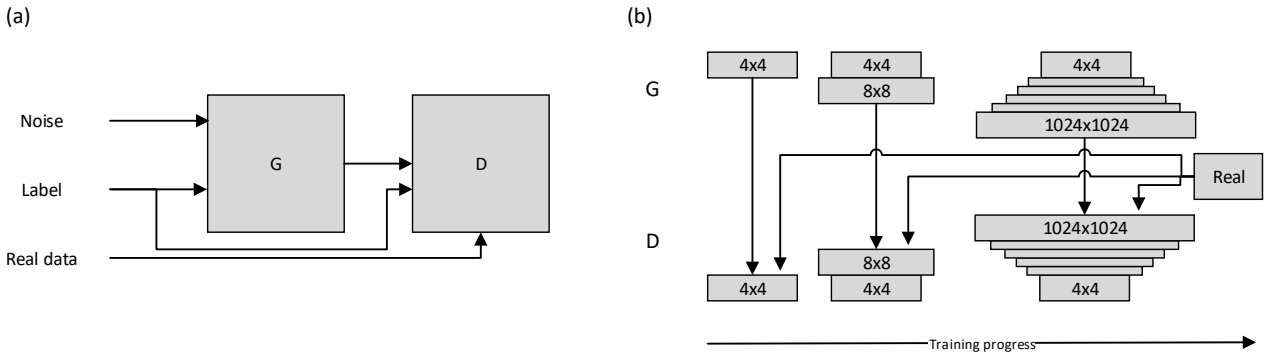

Рисунок 1.1 — Принципи роботи CGAN(a) та PGAN (b) [36]

- StyleGAN – ще один тип GAN, який генерує зображення через передачу стилю одного зображення на інший, що надає можливість отримувати зображення у певному стилі (наприклад акварелю чи лише у темних тонах);
- CycleGAN – тип GAN, який можна використовувати для перекладу зображення в зображення.

Модель CycleGAN надає можливість додавати додаткові атрибуту до зображення, наприклад окуляри або шапку, що змінює лише певні частини зображення, тобто не працює з ним повністю. Дана модель самостійно визначає куди необхідно додати той чи інший предмет, що надає можливість з легкістю додавати певні елементи. Загальний принцип роботи мережі CycleGAN зображено на рис. 1.2.

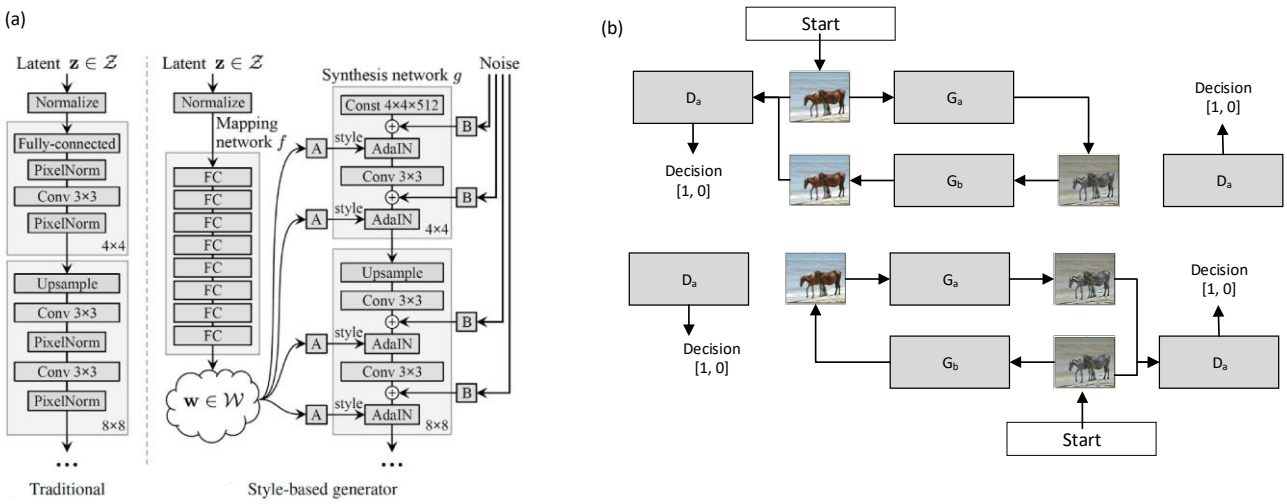

Рисунок 1.2 — Принципи роботи StyleGAN(a) та CycleGAN(b) [36]

Як можна побачити, в цілому існує не так багато стандартних модифікацій і кожна з них має певну чітку ціль та зазвичай виконує саме її. Далі розглянемо модифікації описаних моделей, що надають можливість покращувати якість, швидкість виконання або ж додають певні специфічні елементи до отриманого результату.

1.3 Transformer based GAN для зображень великого розширення

У роботі [9] описується використання GAN для покращення генерації, використовуючи модель трансформеру та алгоритм Swin.

Оскільки дані, що подають на вхід мають досить велику кількість параметрів, що в свою чергу призводить до великої складності їх обрахувань в цілому зображенні, воно розбивається на локальні фрагменти, які обчислюються незалежно один від одного. Саме для цього, був адаптований алгоритм Swin, що додав обчислення до генерації, що відображені у формулі (1).

$$\square^{l} = W\_MSA\left(LN(x^{l})\right) + x^{l}\ x^{l+1} = MLP\left(LN(\chi^{l})\right) + \chi^{l} \text{ – звичайне вікно}$$
$$\square^{l+1} = SW\_MSA\left(LN(x^{l+1})\right) + x^{l+1}\ x^{l+2} = MLP\left(LN(\chi^{l+1})\right) + \chi^{l+1} \text{ – вікно з сувом} \quad (1)$$

При цьому, дискримінатор був покращений до Conv-based версії, для покращення загальної стабільності системи. Це дозволило отримувати більш стабільний вихід, через підвищену потужність моделі.

Також, для покращення результату, було додано style injection, що було додано до моделі генератору. Це дозволило обрати найбільш ефективні моделі, що в даному випадку надало три варіанти та додати їх до моделі. При цьому, не усі моделі можно було застосувати через генерацію зображень високого розширення.

Double attention – через використання Swin алгоритму, частини зображення обраховуються окремо, що не надає можливості отримати повний результат у

коректному вигляді, оскільки великі об’єкти можуть трохи відрізнятися у його частинах. Через це, після обробки попереднього блоку модель мала доступ до його сусідів, через що, для їх генерації вже була можливість використати присутні частини зображення.

Оскільки модель навчалася на зображеннях меншого формату, то на зображеннях більшого виникали певні артифакти, для зменшення яких було використані наступні рішення:

- Artifact-free generator – для цього було використано декілька алгоритмів, що, в цілому дозволяють знизити кількість аномалій шляхом перегляду та обробки не лише поточної частини зображення, а і певних сусідніх частин;
- Artifact-supression discriminator – були використані алгоритми, що дозволяють виявляти локальні аномалії зображень та значно підвищити «програш» моделі через їх невиявлення.

Для отримання результати, наведених у таблиці 1.1 були використані згенеровані зображення з роздільною здатністю 256x256 пікселів використовуючи метріку FID, для якої нижче значення є кращим.

Таблиця 1.1 — Порівняння моделей генерації зображень на наборах даних FFHQ, CelebA-HQ, LSUN Church

Метод	**FFHQ**	**CelebA-HQ**	**LSUN Church**
StyleGAN2	3.62	-	3.86
PG-GAN	-	8.03	6.42
U-Net GAN	7.63	-	-
INR-GAN	9.57	-	5.09
MSG-GAN	-	-	5.20
CIPS	4.38	-	**2.92**
TransGAN	-	9.60	8.94
VQGAN	11.40	10.70	-

HiT-B	2.95	3.39	-
StyleSwin	**2.81**	**3.25**	2.95

Як можна побачити в результаті експериментів, була отримана модель, що має значне кращі результати генерації ніж багато інших стандартних моделей, що, в свою чергу, дозволяє їй отримувати кращі зображення.

1.4 Генерація зображення використовуючи покращену cGAN

У роботі [10] описується генерація зображень використовуючи Conditional GAN з певними модифікаціями. Для цього додають додаткові входи до генератору та дискримінатору *y*, що приймають параметри, які допомагають направити генерацію у потрібну сторону.

На рис. 1.3, зображена саме така модель генерації, що містить дві основні частини – генератор, на вхід якого подається умова (бажаний тип результату) та певний шум (зазвичай використовується гаусівський шум) та дискриминатор, що отримує на вхід реальні зображення з набору даних та згенеровані генератором, а також умову, яка в результаті перевіряє, чи є дане зображення реальним.

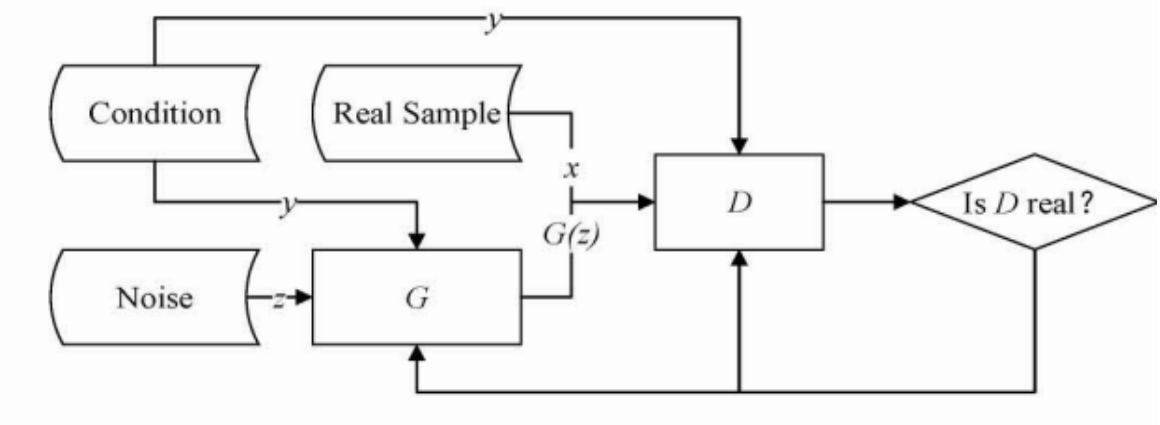

Рисунок 1.3 — Архітектура мережі CGAN [10]

З іншого боку, було використана Wasserstein GAN, що надає у результаті виходу не звичайний булевий результат, а проміжок, наскільки результат є

реальним або згенерованим. Її перевагами є те, що вона надає можливість вирішити проблему нестабільного навчання, але, з іншого боку не є цілком контрольованою.

У роботі пропонується схрестити моделі cGAN та WGAN, в результаті чого отримана модель CWGAN. В ней було додано наступні зміни:

- інформація про стан *y* додається в обидві моделі, через що керування створенням зображення відбувається з використанням *y*;
- оскільки дискримінатор у WGAN виконує завдання регресії, останній шар дискримінатора - сигмоподібний;
- розрахунок втрат моделей відбувається без використання логарифмічної функції;
- кожен раз, коли параметри дискримінатору оновлюються, їх абсолютні значення обмежуються фіксованою константою *c*;
- при оптимізації втрат, більше не використовуються стандартні алгоритми, такі як momentum і Adam, а використані RMSProp та стохастичний градієнтний спуск.

Результат був перевірений, використовуючи різні набори даних, один з прикладів яких зображений на рис. 1.4.

(a) CGAN (b) CWGAN

Рисунок 1.4 — Приклад результатів, отриманих з використанням мережі CGAN(a) та CWGAN (b) [10]

Отриманий результат є набагато більш стабільний за стандартні мережі, оскільки лінії є набагато більш чіткими та отримані фігури мають кращу якість.

1.5 Генерація зображень використовуючи Layered Recursive GAN

У роботі [11] описується ще один варіант модифікації моделі GAN, що генерує окремо фон та передній план зображення використовуючи рекурсію, а потім об’єднує їх для отримання натурального зображення. При цьому, для кожного переднього плану модель навчається генерувати його вигляд, форму та позу.

На рис. 1.5 зображена архітектура даної моделі, що складається з декількох типів шарів – convolutional, fractional convolutional, лінійні та нелінійні. Модель складається з двох основних частин – генератор фону і переднього плану, що не мають спільних частин та параметрів один з одним.

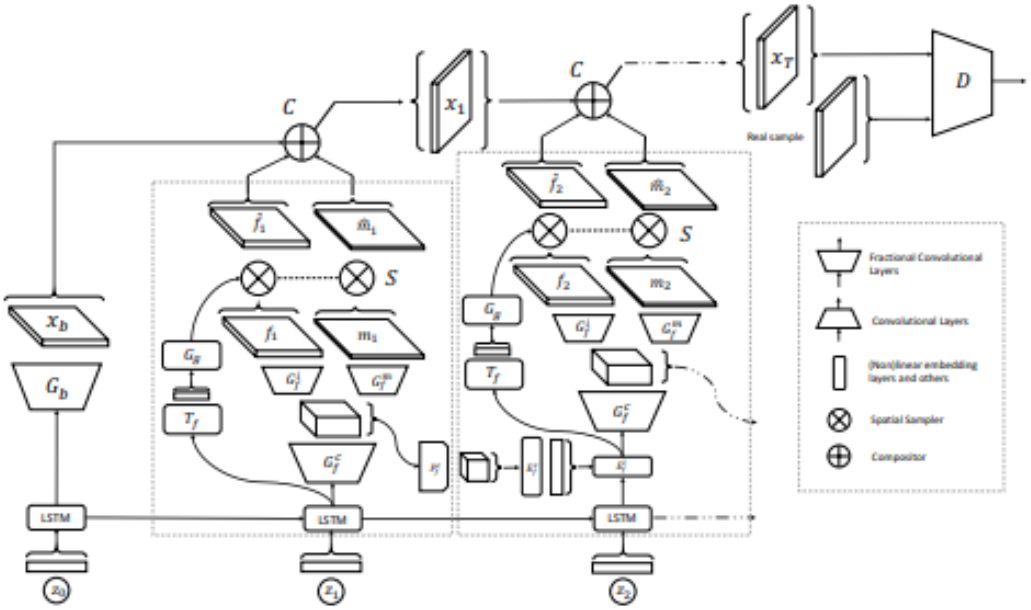

Рисунок 1.5 — Архітектура моделі Layered Recursive GAN [11]

При цьому, генерація фону відбувається лише один раз, в той час як генерація переднього плану може відбуватися багато разів, що будуть мати спільні параметри між собою, але можуть мати різні вхідні, а отже і вихідні дані.

Також, у моделі присутні тимчасові зв’язки, що використовуються для додавання шуму та об’єднання частин зображення. Також, при генерації багатьох об’єктів це надає можливість мати доступ генератору до попередніх результатів.

Генератор переднього плану у свою чергу складається з трьох частин, що генерують власне сам передній план, маску для нього та певну основу.

Мережа, що перетворює результати генерації у формат, що можна використати для подальшого об’єднання отриманих частин. Для цього будується матриця трансформацій, що використовуючи ще один генератор дозволяє об’єднати та трансформувати частини зображення.

В результаті цих змін була отримана модель, що має ще один досить унікальний спосіб трансформацій та може бути використана для додавання декількох об’єктів до одного фону.

1.6 Генерація зображень, використовуючи асиметричне навчання

У роботі [12] було розглянуто ще одну модифікацію GAN, а саме CVAE-GAN, що додає auto-encoder (AE) до звичайної GAN моделі, що дозволяє генерувати зображення у певній категорії.

При цьому генерація зображення відбувається як композиція мітки та латентних атрибутів моделі, що попередньо закодовуються, використовуючи окрему мережу для цього.

Таким чином, при додавання різноманітних міток категорій, можна отримувати різні варіації зображень та контролювати вихід. Через це дана модель може бути навчена генерувати досить велику кількість різноманітних зображень з будь-яких сфер, що використовувалися при її навчанні.

Особливістю даної моделі є використання cross entropy loss для дискримінатору та mean discrepancy objective для генератору. Її архітектура представлена на рис 1.6.

Така асиметрична функція дозволяє зробити навчання моделі набагато більш стабільним. По-друге, в даній мережі, була адаптована encoder network для

вивчення зв’язків між латентним простором та реальними зображеннями та використане pairwise feature matching для збереження структури згенерованих зображень.

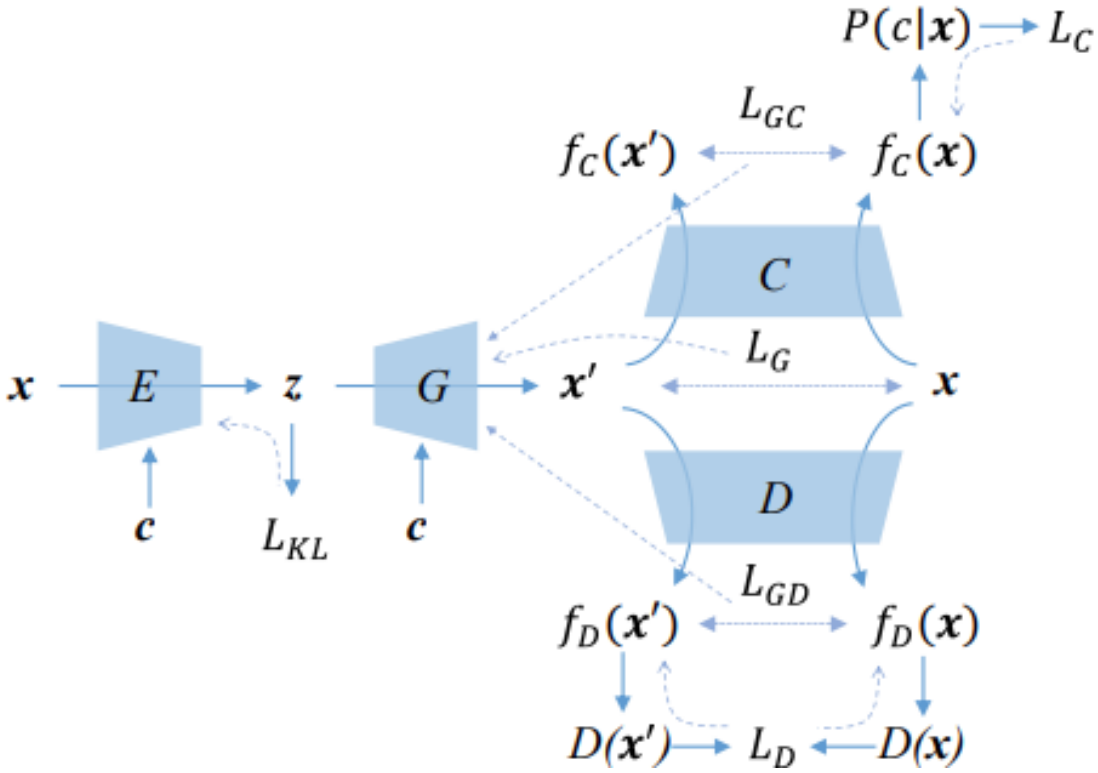

Рисунок 1.6 — Архітектура моделі, що використовує асиметричне навчання [13]

В цілому, дана модель складається з чотирьох частин:

- encoder network – додає дата семпли *c* до латентної репрезентації *z*;
- generative network – генерує зображення *x’* використовуючи вихідні дані з попередньої мережі. Дана мережа намагається вивчити реальний дозволіл даних, що надається їй дискримінатором;
- discriminative network – звичайна мережа, що відрізняє фейкові та реальні зображення;
- classification network – використовується для обрахування posterior P(c|x).

В результаті використання мережі, отримані зображення, деякі приклади яких відображені на рис. 1.7, що мають значно вищу якість ніж більшість звичайних мереж, що можуть генерувати дані.

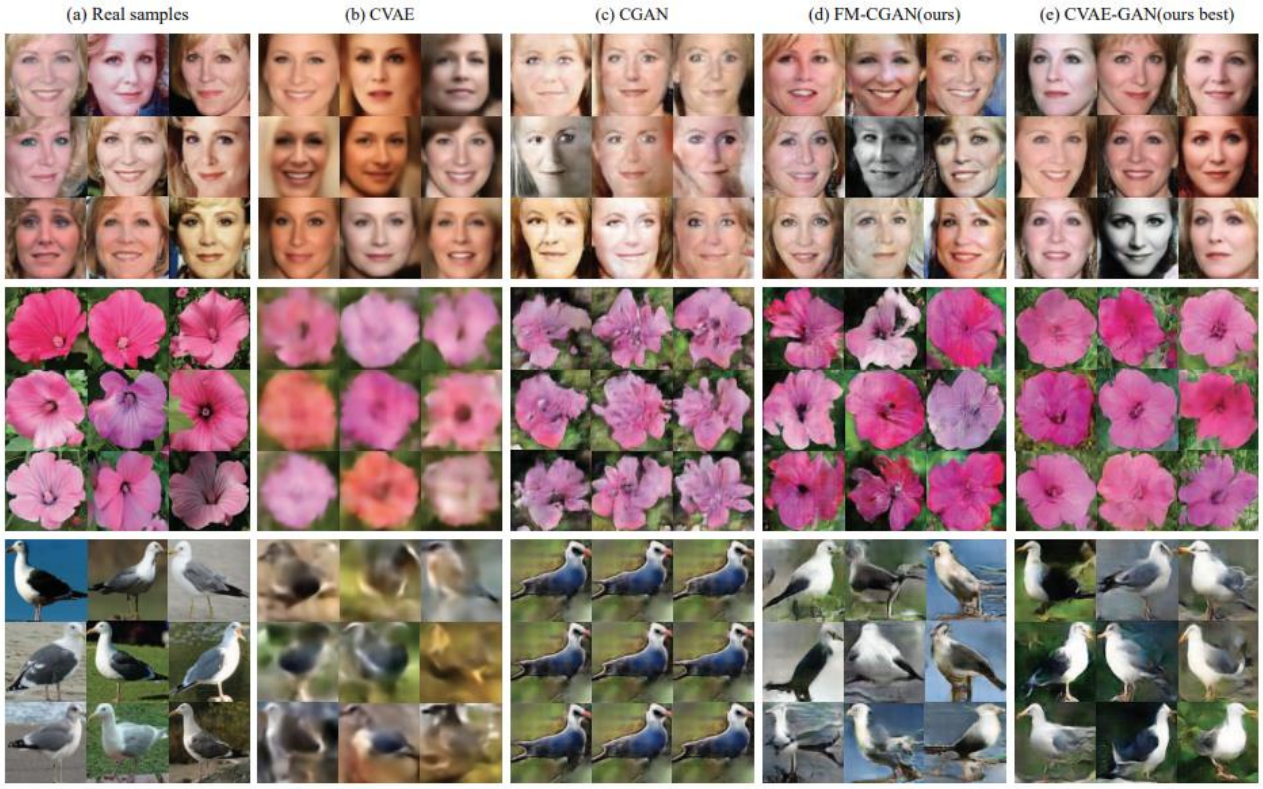

Рисунок 1.7 — Результати генерації, використовуючи модель асиметричного навчання

При цьому, у колонці *a* наведено приклади реальних зображень, що використовувалися для навчання моделі, у колонці *b* – результати мережі CVAE, *c* – результати мережі CGAN, *d* – загальні результати даної мережі, *e* – найкращі результати даної мережі, що були отримані.

Як можна побачити, хоч і не всі семпли є ідеальними, а деякі з них є досить далекими від ідеалу, в цілому отримана якість є досить високою у порівнянні з іншими моделями та достатньою у порівнянні з реальними прикладами зображень.

1.7 Навчання GAN для створення природніх зображень високої роздільної здатності

Ще одним варіантом модифікації моделі є така, що описана у статті [29]. Вона використовує набір даних ImageNet [30], який містить найбільшу кількість зображень за усіма можливими темами та багатьма різноманітними розширеннями,

розбитими за певними основними категоріями. Це дало змогу дуже добре заскейлити дану модель, через підвищення кількості параметрів у 2-8 разів, що дозволило отримати зображення, порівнянні з витворами мистецтва, або принаймні реальними фотографіями, які неможливо вже відрізнити.

На рис. 1.8 зображено декілька прикладів зображень, які були згенеровані даною модифікацією моделі GAN.

Рисунок 1.8 – Результати генерації, використовуючи набір даних ImageNet

Як можна побачити, їх якість неможливо відрізнити від реальних фотографій або картин.

Для отримання даного результату було додано дві прості архітектурні зміни моделі, що допомогли покращити масштабованість та підвищити швидкість відпрацювання та навчання.

Також, це додатково надало можливість використати техніку, яку дозволила чітко та детально контролювати різницю між різноманітністю вибірки та точністю фінальної генерації. При цьому, мінусом є існуюча нестабільність, що характерна для великих мереж даного типу. Завдяки використанню даних модифікацій існуючі методи можуть зменшити дану нестабільність, але повноцінна тренувальна стабільність може бути досягнута лише у варіанті значного збільшення часу та якості навчання.

Також, в даній роботі було розглянуто результат використання звичайної мережі GAN для великих зображень (результати наведені в таблиці 1.2).

Таблиця 1.2 – Метрики запропонованої моделі GAN в залежності від використаного розширення та кількості параметрів.

Batch	Ch.	Param (M)	Shared	Skip-z	Ortho.	Itr (10^3)	FID	IS
256	64	81.5	SA-GAN Baseline			1000	18.65	52.52
512	64	81.5	-	-	-	1000	15.30	58.77 (±1.18)
1024	64	81.5	-	-	-	1000	14.88	63.03 (±1.42)
2048	64	81.5	-	-	-	732	12.39	76.85 (±3.83)
2048	96	173.5	-	-	-	295	9.54	92.98 (±4.27)
2048	96	160.6	+	-	-	185	9.18	94.94 (±1.32)
2048	96	158.3	+	+	-	152	8.73	98.76 (±2.84)
2048	96	158.3	+	+	+	165	8.51	99.31 (±2.10)
2048	64	71.3	+	+	+	371	10.48	86.90 (±0.61)

При цьому, менше значення FID є кращим, а менше значення IS є гіршим. Batch – роздільна здатність отриманого зображення, Param – кількість параметрів, що передавалася генератору, Ch – параметр, що відображував кількість елементів на кожному шарі. Shared – використання спільного простору виконання, Skip-z – використання пропускного з’єднання латентного простору у шарах, Ortho – Orthogonal Regularization, Its – Ітерація, на якій досягається найкращий коефіцієнт, що не покращується під час подальшого навчання.

Принцип роботи даної моделі була побудований наступним чином. Як базова архітектура була використана SA-GAN Zhang et al. (2018), що використовує об'єктив GAN з використанням петлі (Lim & Ye, 2017; Tran et al., 2017). До неї була надана класова інформація для генератору з умовної пакетної нормалізацією класу (Dumoulin et al., 2017; de Vries et al., 2017), а для дискримінатору - з проекцією (Miyato & Koyama, 2018). Налаштування оптимізації слідують за Zhang et al. (2018) (зокрема, використовують норму Спектра в генераторі), з тим, щоби зменшити швидкості навчання вдвічі та виконувати два кроки дискримінатору на крок генерації. Для оцінки були використані рухомі середні ваги, за Karras et al. (2018);

Mescheder et al. (2018); Yazc et al. (2018), зі зниженням 0.9999. Також, була використана ортогональна ініціалізація (Saxe et al., 2014). Кожна модель навчалася на 128-512 ядрах Google TPUv3 Pod (Google, 2018), та обчислювальна статистику BatchNorm на всіх пристроях, а не на кожному пристрої окремо, як відбувається зазвичай. В результаті було встановлено, що прогресивне збільшення (Karras et al., 2018) виявилося непотрібним для моделей розміром 512 × 512.

В результаті отримана модель отримала характеристику результатів, відображену у таблиці 1.3, де зазначено середні значення метрик і їх найбільші та найменші значення, що були отримані в результаті декількох навчань моделей. Як можна побачити, вона показала набагато кращі результати на тому самому ж розширенні, значно знизивши показник FID та підвищивши IS майже у 2 рази.

Таблиця 1.3 – Метрики запропонованої моделі GAN в залежності від розширення зображення.

Model	Res.	FID/IS	Min Fid / IS	FID (valid IS)	FID (max IS)
BigGAN	128	27.6/36.8	-	-	-
BigGAN	128	18.6/52.5	-	-	-
BigGAN	128	8.7/98.8	7.7/126.5	9.6/166.3	25/206
BigGAN	256	8.7/142.3	7.7/178.0	9.3/233.1	25/291
BigGAN	512	8.1/144.2	7.6/170.3	11.8/241.4	27.0/291
BigGAN-deep	128	5.7/124.5	6.3/148.1	7.4/166.5	25/253
BigGAN-deep	256	6.9/171.4	7.0/202.6	8.1/232.5	27/317
BigGAN-deep	512	7.5/152.8	7.7/181.4	11.5/241.5	39.7/298

1.8 Порівняльний аналіз проаналізованих моделей

Після аналізу деяких основних модифікацій мережі виконаємо порівняльний аналіз їх між собою та спробуємо виділимо сильні та слабки сторони кожної з них.

Першою розглянутою мережею була Transformer based GAN для зображень великого розширення – її перевагою є те, що вона генерує зображення не повністю в одній мережі, а частинами ітеративно, додаючи все нові і нові частини, що дозволяє значно знизити складність обрахувань. При цьому її перевагами над звичайним GAN є:

- інжекція стилів;
- генерація локальних частин;
- додаткова робота з позиціонуванням пікселів;
- обробка аномалій.

Другою розглянутою мережею була CWGAN. У ній додається додатковий вхід до генератора та дискримінатора, що надає можливість генерувати зображення за категоріями. Це надає наступні переваги:

- певна детермінованість виводу;
- можливість зміни параметрів обох мереж одночасно через використання функції Вассерштайну;
- додавання проміжків на яких може змінюватися параметри.

Ще однією такою мережею була Layered Recursive GAN, яка генерує окремо фон та окремо саме зображення, після чого об’єднує їх. З її переваг можна виділити:

- генерація частинами;
- використання однієї з частини для декількох результатів;
- використання частини попередньо згенерованих результатів для отримання наступних;
- можливість генерації схожих зображень.

І останньою розглянутою мережею є CVAE-GAN. Вона генерує зображення не через використання явної категорії, а через додавання її за допомогою додаткового модулю – encoder. Перевагами над звичайною GAN є:

- більш стабільне навчання;
- вивчення зв’язків між реальними та згенерованими зображеннями;
- генерація за категоріями.

Виконуючи порівняльний аналіз цих мереж (за основу взята стандартна мережа GAN), результати, зображені у таблиці 1.4. Як можна побачити, в цілому кожна мережа націлена на певний сегмент генерації та має свої переваги.

Таблиця 1.4 – Порівняння проаналізованих модифікацій моделей GAN.

Критерій порівняння	**TB GAN**	**CWGAN**	**LR GAN**	**CVAE-GAN**
Генерація за категорією	-	++	+	+++
Генерація великого розширення	+++	-	+	-
Локальна генерація	++	-	++	+
Додаткова стабільність	+	+	++	+++
Детермінованість результату	-	++	+	++
Пост обробка	+++	-	+++	+
Генерація базуючись на попередніх ітераціях	-	-	+++	+
Підвищення швидкості навчання	-	+++	-	++
Підвищена якість	+++	-	+	++

1.9 Висновки до розділу

У даному розділі було зроблено огляд та розглянуто існуючі алгоритми генерації зображень. Було знайдено 5 основних алгоритмів, які можуть виконувати задачі, схожі на задачі генерації, хоча вони і використовують різні методи та деяким з них необхідні навіть певні реальні вхідні дані, а не елементи латентного пространства.

Стосовно ж мережі GAN, в цілому, існує велика кількість стандартних модифікацій даної мережі, усі з яких також можуть бути розділені на декілька груп. В даній роботі більш детально було розглянуто саме моделі, що надають можливість отримати зображення без або майже без вхідних даних та дозволяють отримати результати значно кращі ніж відсутність певних модифікацій.

При цьому, також, існують досить серйозні модифікації існуючих моделей, що надають можливість отримати результати високої якості та набагато більшої деталізації аніж стандартні моделі, але їх мінусом є необхідність використання великих обчислювальних потужностей протягом дуже довгого часу, що не є можливим в звичайних лабораторних умовах.

Як можна побачити з розглянутих джерел, питання генерації зображень є досить обширним та, хоч запропоновані варіанти і отримають гарні результати, їх алгоритми генерації є абсолютно різними, а отже і фокусування кожної моделі іде зачасту лише на один елемент мережі або її результату, а отже можна спробувати скомбінувати певні аспекти, що надають найкращий результат у кожній з них та об’єднати його для отримання покращеної генерації зображень.

2 ОГЛЯД ПРИНЦИПІВ РОБОТИ GAN ТА АЛГОРИТМІВ ПОРІВНЯННЯ ЗОБРАЖЕНЬ

2.1 Розподіл моделей машинного навчання

Усі алгоритми машинного навчання можна поділити на алгоритми навчання з учителем та без нього. Архітектура першого з них зображена на рис 2.1 та вимагає набір даних, що буде використаний для навчання моделі та буде містити велику кількість даних, що містить вхідні та вихідні значення. Модель при цьому тренується передбачати вихід та коректувати себе, щоб цей вихід був найбільш приближеним до реальності [3].

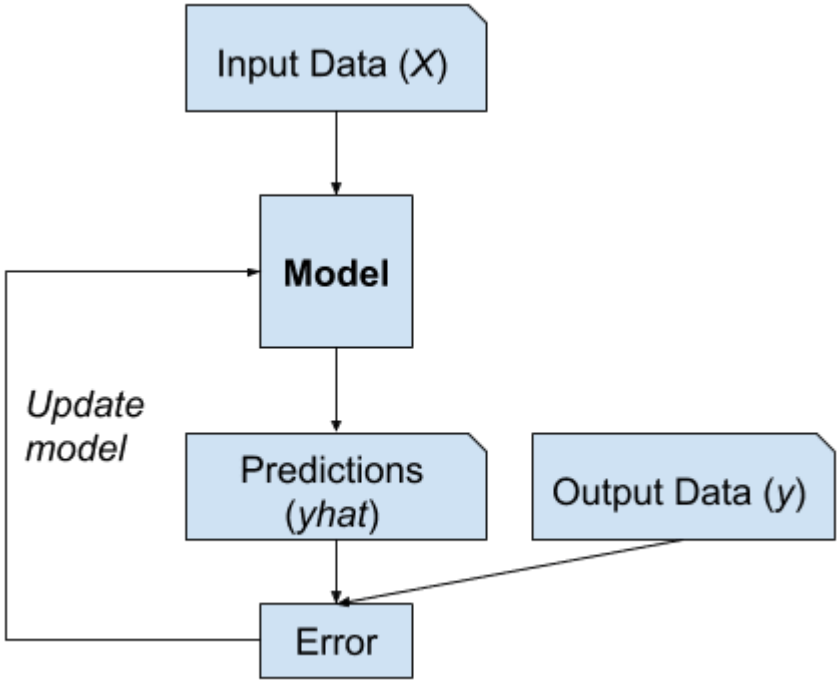

Рисунок 2.1 — Архітектура простої мережі GAN [3]

Прикладами проблем, що вирішує дана модель є проблеми класифікації та регресії. При цьому, типами алгоритмів, що використовуються є, наприклад, алгоритми регресії та випадкового лісу.

Іншою типовою проблемою, що вирішує машинне навчання є проблема, за якої в початкових даних відсутні результати, або їх кількість є дуже малою. Принцип роботи даної моделі зображений на рис. 2.2.

Прикладами проблем, що виконує дана модель є кластеризація та генеративне модулювання, а прикладами алгоритмів є алгоритми К-середнього та Generative Adversarial Networks.

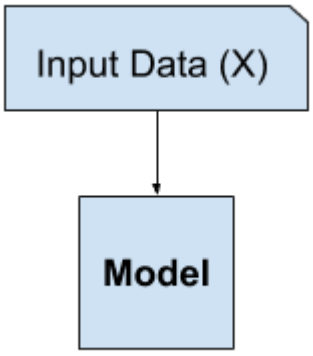

Рисунок 2.2 — Принцип роботи навченої мережі GAN для генерації результатів навчання моделей [3]

Розглядаючи модель GAN варто розглянути також задачу передбачення, що називається класифікацією та традиційно відноситься до дискрімітаційного моделювання. Це відбувається через те, що модель базуючись на декількох вхідних параметрах (змінних), що надаються, повинна зробити вибір визначити до якого самого кластеру/класу відноситься той чи інший об’єкт.

З іншого ж боку, моделі навчання без учителя, що розподіляють вхідні змінні можна використовувати для створення або генерації нових прикладів для розподілу. Такі моделі називаються генеративними.

2.2 Архітектура GAN

GAN – генеративна модель глубокого навчання. Більш загально, це модель архітектури, що використовується для навчання генеративної моделі та найчастіше використовується у архітектурі глибокого навчання. Вперше дана модель була описана у 2014 році Ian Goodfellow, у праці, що має назву “Generative Adversarial Networks.” [4].

Більш стабільна версія моделі була описана у наступному році Alec Radford, у праці “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks“. [5]

GAN використовується для покращення штучного інтелекту та автоматичної генерації об’єктів, що вимагають людську креативність [6].

Модель даної архітектури включає у собі дві підмоделі, а саме:

- генератор, що використовується для генерації нових прикладів/екземплярів;
- дискримінатор, що навчається класифікувати екземпляри як реальні (залежить від предметної області) та фейкові (згенеровані).

В цілому, дана модель побудована на сценарії теорії ігор, в якій генератор змагається зі своїм противником. Він повинен продукувати екземпляри, а дискримінатор намагається відрізнити екземпляри, що згенеровані, від даних, що були побудовані генератором. На рис. 2.3. відображено за допомогою блок-схеми приклад звичайної мережі GAN.

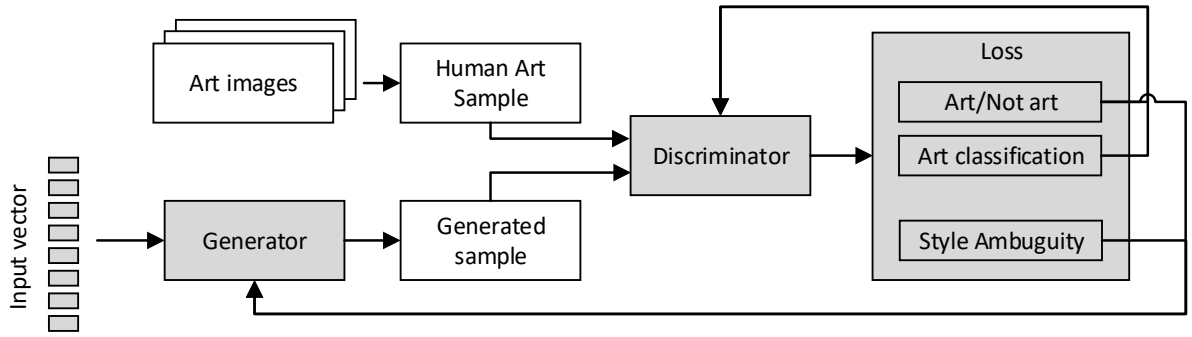

Рисунок 2.3— Блок-діаграма мережі GAN

На даній блок-схемі відображена архітектура моделі з приділенням більшої уваги дискримінатору, що визначає чи є дані, що були подані йому на вихід витвором мистецтва, або ж класифікує їх за певними параметрами, наприклад, надаючи їм значення від 0 до 1, де 1 – є реальне зображення.

2.3 Принципи роботи GAN

Модель генератора приймає випадковий вектор визначеної довжини та на основі нього генерує дані з визначеної предметної області. Даний вектор будується на основі гаусівського розподілу та використовується для усього процесу генерації. Після навчання, точки на багатовимірному векторі простору будуть відповідати відповідним точкам у проблемі предметної області, формуючи стиснуту репрезентацію розподілу даних.

Даний вектор простору відноситься до латентного простору та містить латентні (або сховані) змінні, що важливі для предметної області, але не можуть бути переглянуті та змінені ззовні.

Модель дискримінатору приймає приклад даних предметної області як вхід (причому даний приклад може бути як реальний, так і згенерований) та намагається визначити чи був даний екземпляр реальним або згенерованим. При цьому, дискримінатор є звичайною моделлю для класифікації.

Після закінчення процесу навчання, дискримінатор може бути відкинутий, оскільки вже більше не є потрібним.

При цьому, частини моделі, що створює генератор або дискримінатор може бути використана для отримання фіч інших алгоритмів машинного навчання для отримання кращих моделей і використання їх у інших сферах, наприклад навчання моделей з учителем.

Принцип роботи GAN є у тому, що моделі генератору та дискримінатору навчаються одночасно. Тобто, генератор створює певну кількість фейкових даних, до яких також додаються реальні екземпляри предметної області. Після цього ця

група екземплярів надається дискримінатору, що визначає чи є вони реальними або згенерованими.

Після цього дискримінатор оновлює себе для більш кращого розуміння де є які дані, і, що більш важливо, генератор оновлюється на основі того, як гарно згенеровані результати були оброблені дискримінатором.

В даному випадку дані дві моделі змагаються одна з одною, що у теорії ігор є грою з нульовою сумою. В даному випадку нульова сума означає, що якщо дискримінатор коректно визначає тип даних, то він не змінює свої параметри у той час коли генератор отримує великі оновлення у своїй моделі. При цьому, якщо ж генератор зміг створити дані, що дискримінатор не зміг відрізнити від реальних, генератор не змінює параметри, а дискримінатор, отримує досить великі зміни.

В результаті, у випадку коректного навчання моделі, генератор створює ідеальні репліки даних зі своєї моделі, а дискримінатор не може відрізнити та видає 50% результат для кожного варіанту даних. У такому варіанті навчання можна зупинити набагато раніше для отримання дуже гарних даних.

Модель роботи генератора та дискримінатора зображена на рис. 2.4. На ньому проілюстровано саме принцип навчання даної моделі, в результаті якої не відбувається отримання нового, згенерованого зображення, а відбувається налаштування моделі генератору або дискримінатору, в залежності від того, чи вдалося мережі другого компонента правильно визначити категорію, подану їх на вхід.

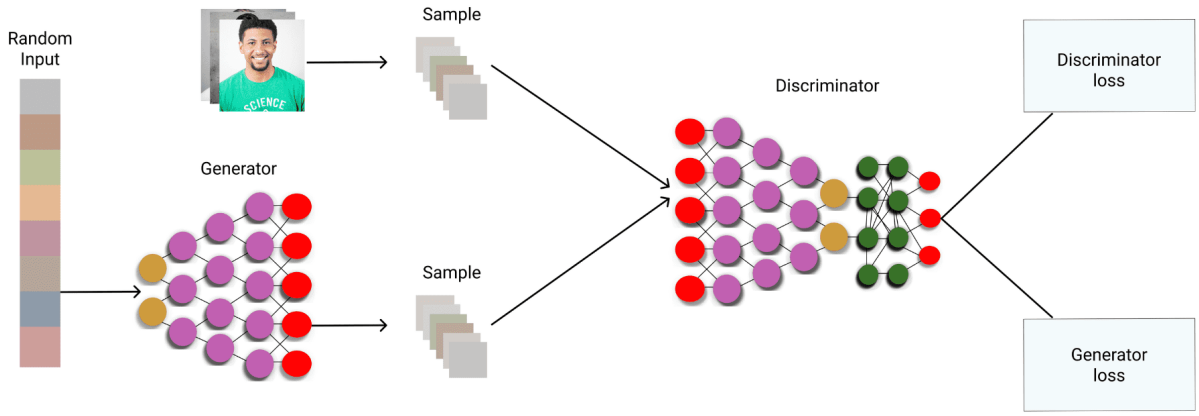

Рисунок 2.4 — Модель роботи мережі GAN [4]

2.4 Основа побудови GAN

Найчастіше GAN працює з графічними даними та використовує Convolutional Neural Networks, або CNNs як моделі генератора та дискримінатора. Даний тип моделі має назву DCGAN (Deep Convolutional GAN). Принцип перетворення графічних даних у результат дискримінатору та масиву шуму до зображення у генератору, наведений на рис. 2.5.

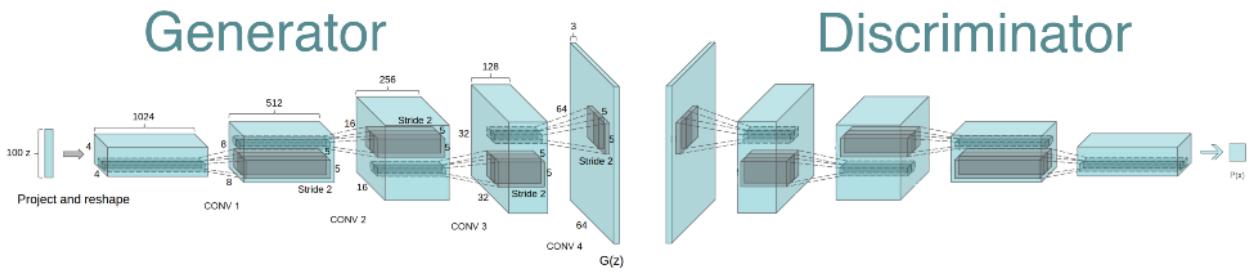

Рисунок 2.5 — Принцип перетворення та масштабування даних у генераторі та дискримінаторі [5]

Причиною чого є те, що вперше дана модель була застосована для комп'ютерного зору та використовувала CNN та графічні дані, а також, через те, що за останні роки CNNs в цілому досягали набагато більш кращих результатів у області комп'ютерного зору у задачах, таких як, виявлення об'єктів або розпізнавання лиць.

Моделювання зображень означає, що в латентному просторі на вхід генератора надається набір зображень або фотографій, що використовуються для тренування моделей. При цьому, це означає, що генератор створює нові дані у форматі, що може бути переглянутий або перевірений розробнику або користувачам моделі.

2.5 Conditional GAN

Conditional GAN – додавання додаткової інформації до генератору та дискримінатору може значно покращити саму модель. [6] В даному випадку їм обом надаються дані, що позначену *y*. Це може бути будь який тип інформації, наприклад додаткові дані або мітки. Це дозволяє створювати кращі зображення.

Це надає наступні переваги:

- швидше навчання, оскільки навіть випадковий розподіл має свій патерн;
- вихід генератору може контролюватися, надаючи під час виходу певну мітку.

На рис. 2.6 зображено архітектуру даної моделі, в якій, крім звичайного масиву даних x/z, додаються додаткові параметри y, що використовуються у обох елементах мережі.

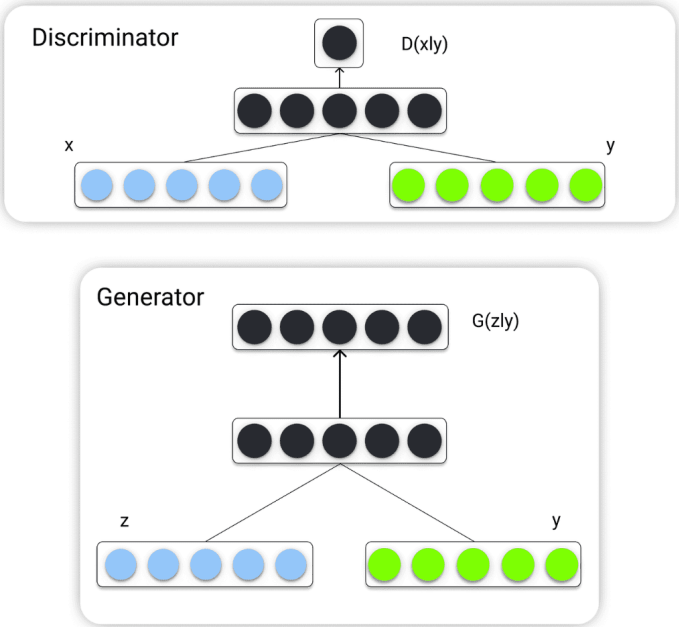

Рисунок 2.6 — Архітектура мережі CGAN [5]

Зробивши крок уперед, cGAN може бути наданий графічний елемент предметної області, що дозволить виконувати такі задачі як text-to-image, image-to-image перетворення. Це дозволяє створювати ще кращі застосунки, що

використовують GAN, наприклад переведення стилю, додавання кольорів на фото, трансформації фотографій або об’єктів на них (літо у зиму, день у ніч, тощо).

2.6 Принцип генерації та навчання

Для початку роботи генеруються дані, що будуть мати нормальний розподіл та передаються генератору. Оскільки генератор ще не має ніякого стану, то він повертає цей нормальний розподіл. [7]

При цьому, дискримінатору надається даний розподіл та реальне зображення для порівняння та евалюює його. Якщо даний розподіл вийде близьким до оригінального екземпляру, то повернеться «1» - реальні дані, якщо ж вони навіть близько не стоять, то буде повернено «0» - фейкові/згенеровані дані.

Для того, щоб визначити як буде еволюціонувати генератор, спочатку треба визначити як дискримінатор буде визначати реальність отриманих даних.

Відповідь полягає у функції втрат, що визначає відстань між розподілом у згенерованих та реальний даних. При цьому генератор намагається мінімізувати свою функцію, а дискримінатор намагається її максимізувати.

$$E_x\left[log\left(D(x)\right)\right] + E_z[log(1 - D\big(G(z)\big))] \quad (2)$$

При цьому генератор не має напряму свою функцію втрат, а використовує її через дискримінатор. Якщо він визначає, що дані фейкові, то дана функція штрафує генератор. Коли втрати вираховані, ваги генераторів оновлюються через обернений розподіл мережі дискримінатора до генератора.

Цикл процесу навчання дискримінатору можна навести у наступному вигляді:

- обирається m даних {z1,z2…zn}, що мають нормальний розподіл та трансформується через генератор;
- обирається m реальних даних { x1, x2… xn}, що мають нормальний розподіл;

- використовується функція втрат, щоб порахувати втрати;
- береться градієнт функції втрат та оновлюються параметри дискримінатора, при цьому для оновлення параметрів використовується підвищення градієнту, щоб максимізувати втрати.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [log\, D\left(x^{(i)}\right) + log(1 - D\left(G\left(z^{(i)}\right)\right))] \quad (3)$$

Цикл процесу навчання генератору можна навести у наступному вигляді:

- обираються m даних з нормального розподілу $\{z_1 \ldots z_n\}$ та трансформуються через генератор;
- оскільки в даному випадку важлива лише частина з генератором, то у функції витрат частина з дискримінатором дорівнює нулю;
- при цьому, є важливим, що генератор еволюціонує, використовуючи дискримінатор як гайд, що допомагає йому це робити.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} log(1 - D\left(G\left(z^{(i)}\right)\right)) \quad (4)$$

У якості функцій витрат зазвичай використовують наступні варіації:

- min-max loss;
- Wasserstein loss.

Min-max loss – заснований на підвищенні та зниженні помилок. Він був використаний із теорії ігор та заснований на змаганні гравців та ідеї, що, щоб виграти, гравець повинен максимізувати можливість виграшу та мінімізувати можливість цього для опоненту знайшовши найкращі для себе кроки.

$$R = \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [log\, D\left(x^{(i)}\right) + log(1 - D\left(G\left(z^{(i)}\right)\right))] \quad (5)$$

У даному випадку дискримінатор намагається збільшити шанс виграшу для себе D(x) = 1, та знизити його для генератору D(G(x)) = 0. Через це різниця між (1 – D(G(x)) повинна зростати та більша різниця індикує на більш кращу роботу дискримінатору.

Генератор, навпаки, намагається мінімізувати можливість виграшу дискримінатору (1 – D(G(x)))

Wasserstein loss – був розроблений для типу GAN, у якому дискримінатор видає не значення 0/1, а число на проміжку [0,1], хоча ідея, що реальні зображення видають більше значення, а фейкові - менше, залишається тою ж самою. Через це, вона не може виявити чи це є фейкові дані.

Через це, дискримінатор називають критиком та отримують наступну функцію втрат:

$$F(x) = C(x) - C(G(z)) \quad (6)$$

Дискримінатор намагається максимізувати її в той час як генератор – мінімізувати.

2.7 Приклади застосування GAN

Наразі даний тип нейронних мереж використовується у багатьох сферах життя, таких як [1-2, 6]:

- перетворення тексту у зображення, що історично було дуже складною задачею для обробки машиною;
- маніпуляція даними – наприклад, прибирання зайвого об’єкту на фотографії, такого як сонячні окуляр для покращення подальшої обробки або розпізнавання;
- створення картин за певним арт стилем, або на основі інших картин, для чого дана мережа використовує трансфер стилю, для додавання певних елементів, кольорів з обраного стилю до обраного зображення;

- навчання supervised моделі при наявності дуже малої кількості позначених даних та великої кількості непозначених, що можна виконати, використовуючи такий тип моделі як Semi-supervised GAN;
- створення даних для медичної сфери для більш кращого навчання інших типів моделей, що дозволяє підвищити кількість елементів у наборі даних, оскільки отримати нові зразки деяких медичних фотографій в великій кількості майже неможливо в обмежений час;
- перетворення чорно-білої у кольорову фотографію;
- перетворення супутникового знімку у карту місцевості;
- видалення шумів з фотографії, що надає можливість значно підвищити якість зображення;
- рух об’єктів на фотографії, наприклад, зображеної людину у певну позу тіла з можливістю її подальшого анімування;
- виявлення аномалій у аудіо, відео та файлах;
- створення, написання та тестування коду від багів, що значно поліпшує розробку девелоперів та надає можливість більш швидкого написання програмного забезпечення;
- генерація фейкових облич людей, що дозволяє їх використовувати для 3Д графіки, створення певного контенту. Для цього використовується модифікація, що має назву Deep Convolutional GAN;
- генерація музики – створення нових мелодій на основі певних параметрів.

2.8 Алгоритми оцінки зображення

2.8.1 Inception Score

Inception score (IS) – це широко використовуваний показник для оцінки якості зображень, згенерованих за допомогою GAN [31-32]. Дана оцінка була запропонована в 2016 році дослідниками Google під керівництвом Тіма Саліманса.

Вона дозволяє виміряти ступінь реалістичності, різноманітності та якості згенерованих зображень на основі прогнозів класифікатора.

Для її обчислення використовується початковий показник, що обчислюється за допомогою попередньо навченої мережі Inception-v3, яка є глибокою нейронною мережею, яка була розроблена для класифікації зображень. Ця мережа використовується як екстрактор ознак для обчислення ймовірностей класу згенерованих зображень. Після цього, класифікатор GAN застосовується до кожного згенерованого зображення, а бали усереднюються для обчислення середнього значення та стандартного відхилення балів. В даному випадку середній бал характеризує якість створених зображень, тоді як стандартне відхилення відповідає різноманітності створених зображень.

Даний показник став інтуїтивно зрозумілим, досить простим у використанні та може бути швидко обчислений. При цьому, він чутливий, як до якості, так і до різноманітності згенерованих зображень.

Одним з його обмежень є те, що він передбачає, що створені зображення взяті з одного розподілу, через це оцінка може не повністю відображати якість створених зображень, якщо для їх генерації було використано декілька різноманітних типів. Наприклад, якщо GAN генерує зображення двох різних типів, початкова оцінка може не точно відобразити різноманіття згенерованих зображень.

Також, він не враховує семантичну узгодженість створених зображень. Це означає, що може бути некоректно визначена якість створених зображень, якщо вони не відповідають призначеному розподілу. Наприклад, якщо GAN генерує зображення синього кота замість помаранчевого кота, початкова оцінка все ще може бути високою, якщо синій кіт класифікується як кіт класифікатором Inception-v3.

При цьому, навіть незважаючи на ці обмеження, даний показник є цінним показником для оцінки якості створених зображень, оскільки він став стандартним еталоном для порівняння різних моделей GAN і використовувався в багатьох наукових роботах.

2.8.2 The Fréchet Inception Distance

The Fréchet Inception Distance (FID) — ще одна метрика для оцінки якості згенерованих зображень створених за допомогою GAN. Вона була запропонована Martin Heusel у 2017 році як покращення існуючих показників на той час, таких як inception score [33-34].

Дана метрика базується на відстані Фреше, яка є мірою відстані між двома розподілами ймовірностей у багатовимірному просторі. У випадку використання мережі GAN, одним з таких розподілів є набір реальних зображень, що використовуються для навчання дискримінатора, а іншим – набір згенерованих зображень.

Вона включає в себе два основні етапи обрахування: по-перше, обчислюється попередньо навчена мережа Inception-v3 як на реальних, так і на створених зображеннях. Це надає можливість отримати високорівневі характеристики зображень, що можуть бути обчислені в свою чергу на певних рівнях мережі Inception-v3.

Після цього, в даній матриці обчислюється відстань Фреше між двома наборами активацій. Відстань Фреше визначається як квадрат відстані між середніми значеннями двох розподілів плюс сума їхніх коваріацій. Таким чином, вона дозволяє вимірювати, наскільки подібні два набори даних з точки зору їх розподілу.

З математичної точки зору, оцінка FID розраховується за наступною формулою:

$$FID(P_r, P_g) = ||mu_r - mu_g||^2 + Tr(\sigma_r + \sigma_g - 2\sqrt{(\sigma_r * \sigma_g)}) \qquad (7)$$

де P_r і P_g — розподіли ймовірностей реальних і згенерованих зображень, mu_r і mu_g — засоби активації реальних і згенерованих зображень, $sigma_r$ і $sigma_g$ — їхні коваріаційні матриці, а Tr() — оператор трасування.

Таким чином, чим нижче оцінка FID, тим ближче згенеровані зображення до реальних з точки зору їх функцій високого рівня. На практиці оцінка FID менше 50 вважається дуже хорошою, тоді як оцінка більше 100 вказує на те, що створені зображення не дуже схожі на реальні зображення.

З мінусів даної оцінки варто виділити, що її розрахунок може бути дорогим у плані обчислень, особливо для великих наборів даних. При цьому існує певна кількість оптимізацій, які можна використати для прискорення обчислення, наприклад обчислення активацій і оцінки FID у пакетах.

При цьому, дана метрика не є досконалою та через це має деякі обмеження, як і попередня розглянута метрика IS. По-перше, вона базується на припущенні, що характеристики високого рівня реальних і згенерованих зображень відповідають розподілу Гаусса, що не завжди виконується для всіх зображень. По-друге, FID вимірює лише подібність між характеристиками високого рівня зображень і не враховує інші аспекти якості зображення, такі як різкість, баланс кольорів і текстура.

При цьому, незважаючи на ці обмеження, FID є досить широко використовуваною метрикою, що допомагає досить точно виміряти результати генеративного моделювання. При цьому, вона надає певний кількісний показник якості зображення, який можна використовувати для порівняння різних моделей і відстеження прогресу у порівнянні з часом їх навчання, використання різних наборів даних або ж просто зміни певних параметрів.

2.8.3 Kernel Inception Distance

Ще одним алгоритмом, що використовується для оцінки якості генерації за допомогою GAN є алгоритм Kernel Inception Distance (KID). На відміну від попередньо розглянутого алгоритму, KID вимірює подібність між характеристиками згенерованих і реальних зображень за допомогою методів ядра [35].

Він був вперше представлений Han Zhang та ін. у 2018 році як певна модифікація алгоритму, що базується на алгоритмі FID. При цьому, це було створено для того, щоб подолати обмеження, що має FID, наприклад такі, як чутливість до вибору мережевої архітектури та рівня, а також припущення гаусівського розподілу функцій.

Дана метрика вираховує відстань між матрицями Грама представлень ознак згенерованих і реальних зображень. Матриця Грама — це спосіб охоплення статистичних залежностей між ознаками. Тобто, KID вимірює, наскільки схожі залежності між функціями на згенерованих і реальних зображеннях.

При цьому, ядро, що використовується в KID, зазвичай є ядром Гаусса, також відомим як ядро радіальної базисної функції. Дане ядро використовується для обчислення попарної подібності між функціями. Після цього дана подібність зважується параметром пропускної здатності, який вже після цього визначає, наскільки ядро чутливе до відмінностей між функціями.

Математичне представлення, що використовується для обрахування KID відображено у наступній функції:

$$KID(P_r, P_g) = ||mu_r - mu_g||^2 + Tr(K_r + K_g - 2\sqrt{(K_r * K_g)}) \qquad (7)$$

де P_r і P_g — розподіли ймовірностей реальних і згенерованих зображень, mu_r і mu_g — засоби репрезентації ознак реальних і згенерованих зображень, K_r і K_g — матриці Грама реальних і згенерованих зображень, а Tr() є оператором трасування.

При цьому, першим параметром у оцінці KID є квадрат евклідової відстані між середніми значеннями представлень ознак, що вираховується відповідно до FID. Другий член, в свою чергу, вимірює різницю між матрицями Грама представлень ознак.

Для обчислення матриці Грама, відбувається зведення представлення ознак зображень у вектори. Після цього відбувається обрахунок зовнішнього добутку

векторів для отримання матриці. Дану матрицю, після її створення, пропускають через ядро Гаусса, для того щоб після виконання певних визначених операцій отримати матрицю Грама.

Іншим методом обчислення KID є обрахунок за допомогою зміщеної оцінки, яка передбачає віднімання поправочного члена від другого члена в оцінці KID. Зміщена оцінка є швидшою для обчислення та, як було показано, має досить високу емпіричну ефективність, хоча і показує трохи гіршу якість ніж стандартний алгоритм обчислень.

Також, як і в попередньо розглянутому алгоритмі, обчислення KID може бути досить дорогим з обчислювальної точки зору, особливо для великих наборів даних. Але, з іншого боку, також існують кілька оптимізацій, які можна використати для прискорення обчислення, наприклад обчислення представлень функцій і матриць Грама в пакетах.

Однією з переваг KID перед FID є те, що даний алгоритм є набагато більш стійким до змін архітектури та рівня мережі у алгоритмі GAN. Крім того, KID може обробляти негаусівські розподіли представлень ознак, що є головним мінусом FID.

Однак дана метрика також має деякі обмеження. Наприклад, вона може не обробляти та аналізувати абсолютно усі аспекти якості зображення, наприклад. такі як різкість і текстура. Крім того, вона може бути досить чутливою до вибору параметрів ядра та пропускної здатності.

При цьому, незважаючи на свої обмеження, KID стала широко використовуватися як показником оцінки ефективності роботи моделі GAN. Зазвичай даний показник використовується, як додатковий показник якості зображення та вираховується разом із FID, через те, що разом вони можуть надати більш повну картину продуктивності GAN.

2.9 Висновки до розділу

У другому розділі було розглянуто архітектуру GAN та принципи роботи. Було описано дві її основні складові – генератор та дискримінатор, що відповідають за генерацію та перевірку коректності зображень відповідно. Їх робота у парі дозволяє отримувати набагато кращі результати навчання ніж більшість інших моделей.

При цьому, під час генерації застосовується теорія ігор, а саме гра з нульовою сумою, в результаті чого, якщо дискримінатор розпізнає, що зображення фейкове, генератор отримує значне змінення своїх параметрів генерації. Якщо ж генератору вдається перехитрити та виграти даний раунд, то це відбувається з дискримінатором.

Також, було розглянуто основні метрики, що можуть бути використані для обчислення якості моделі GAN та результатів її операцій, а саме

- Inception Score – є найпростішим алгоритмом для порівняння, який видає досить чіткі результати якості моделі та заснований на порівнянні реальних та згенерованих зображень на основі нейронної мережі;
- The Fréchet Inception Distance – більш складний алгоритм, що також, оснований на нейронній мережі, але виконує певні обчислення для отримання більш кращих результатів, при цьому, його основним мінусом є те, що дана метрика може працювати лише з гаусівським розподілом;
- Kernel Inception Distance – створена метрика, як модифікація FID, що використовується зазвичай разом з нею та заснована на досить складних та операційно витратних обчисленнях, що дозволяє подолати основні мінуси попередньої метрики, а саме роботу з негаусівським розподілом даних.

Дані метрики дозволяють досить чітко оцінювати результат генерації та порівнювати різноманітні моделі GAN між собою. При цьому, кращою моделлю є та, яка має більше значення Inception score та менше значення FID та KID.

3 РОЗРОБКА ВЛАСНОЇ МОДЕЛІ

3.1 Розробка моделі

Після проведеного аналізу існуючих моделей та принципів роботи GAN, необхідно розробити її модифікацію, що буде мати змогу генерувати зображення за категоріями.

По-перше, необхідно визначити очікувані входи та виходи створюваної моделі генератору та дискримінатору. Оскільки, необхідно генерувати зображення за категоріями, то одним з входів обох моделей повинна бути, власне сама категорія, що буде означати тип обраного зображення.

Другий вхід буде відрізнятися у даних моделей:

- дискримінатор буде приймати на вхід кольорове зображення;
- генератор буде приймати на вхід випадковий гаусівський шум.

На вихід, в свою чергу, будуть йти відповідно:

- у дискримінатору – значення від 0 до 1, яке буде означати чи є зображення реальним або ж згенерованим;
- у генератору – кольорове зображення того ж розширення, що приймається дискримінатором.

Стосовно розширення зображення, воно повинно бути усереднене, тобто не дуже велике, оскільки це значно підвищиться складність та час навчання моделі та не дуже мале, оскільки через це можуть бути втрачені деякі елементи та деталі зображення, які варто відобразити у результаті.

Таким чином, оптимальним розширенням зображення будуть розміри у проміжку від 80x80 до, приблизно, 100x100, оскільки вони будуть мати досить непогану якість, як для звичайної аватарки на сайті, так і досить невелику кількість кольорів пікселів, або ж елементів моделі (до 30 тисяч).

Отже, зобразимо загальну архітектуру очікуваної моделі на рис. 3.1, в якому відображено модель дискримінатору та генератору та їх відповідні входи та виходи. Отримана модель є моделлю CGAN.

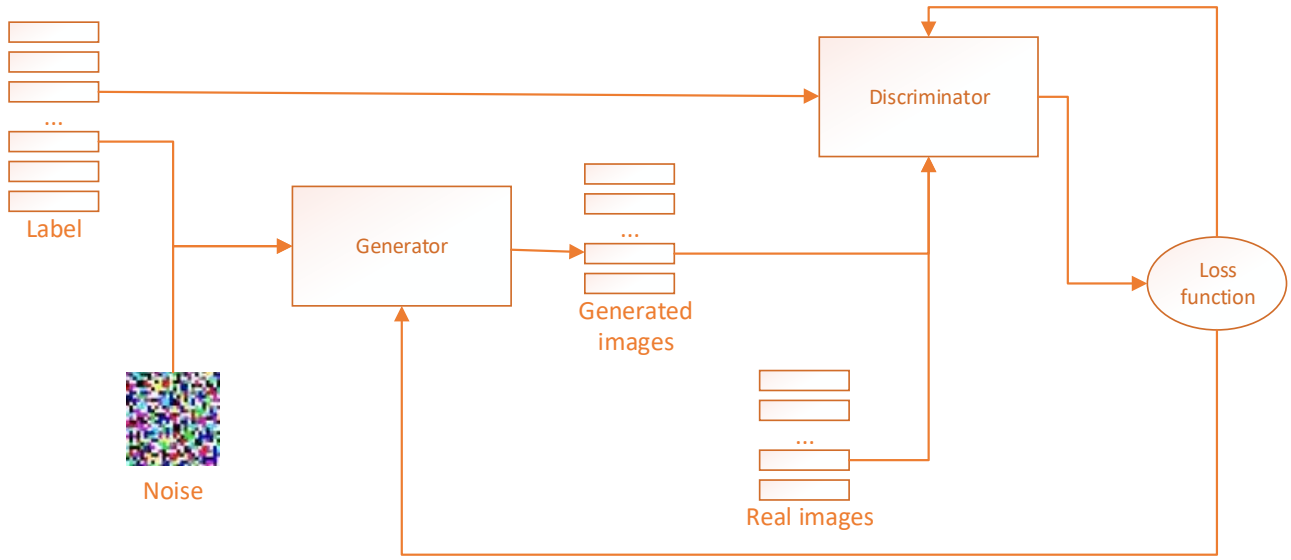

Рисунок 3.1 — Архітектура розробленої моделі

Для функції втрат використаємо Wasserstein loss function [37], або функцію витрат Вассерштейна, формула обрахування якої відображена у (8), де D_l – результат функції втрат дискримінатору, G_l – результат втрат генератору, D(x) – вихід дискримінатору на реальних даних, G(z) – вихід генератору, або згенеровані зображення, D(G(z)) – вихід дискримінатору на згенерованих генератором наборі даних.

$$\begin{cases} D_l = D(x) - D(G(z)) \\ G_l = D(G(z)) \end{cases} \quad (8)$$

Оскільки очікуваний вихід дискримінатору є числом від 0 до 1, то, втрати будуть обчислюватись та застосовуватися до обох моделей незалежно від того, чи є зображення згенерованим або реальним, різниця буде лише в абсолютному значенні змін параметрів, оскільки чим точніше дискримінатор визначить втрати, тим менше змін він отримає та більше змін отримають ваги та параметри генератору.

Також, для покращення результату генерації, використаємо частину моделі PGAN, але імплементуємо це у якості шарів нейронної мережі, кожен з яких буде підвищувати розширення попереднього шару, що, в свою чергу, повинно значно

зменшити необхідну кількість епох для навчання, при цьому підвищивши час навчання однієї епохи.

Відобразимо результуючу загальну архітектуру генератору та дискримінатору на рис. 3.2. На даному зображенні можна побачити, що моделі дискримінатору та генератору містять шари, що перетворюють не лише зображення, а і поступово змінюють її розширення для отримання кращих результатів генерації.

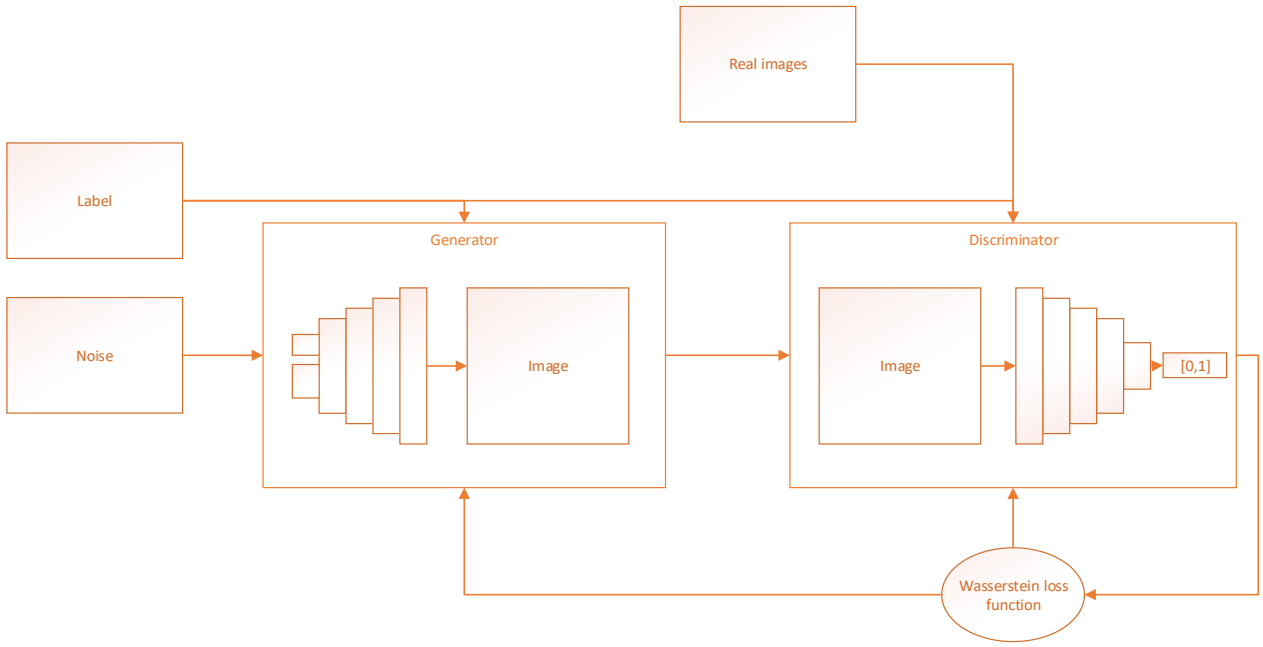

Рисунок 3.2 — Архітектура розробленої моделі

В результаті усіх цих перетворень отримаємо модель, що повинна містити у собі усі переваги CGAN, WGAN та PGAN, при цьому позбувшись деяких їх недоліків.

Далі буде розглянуто програмну реалізацію розробленої моделі та вибір набору даних, що буде використаний для її навчання та генерації подальших зображень аватарів.

3.2 Вибір набору даних

Не менш важливим етапом аніж розробка алгоритму є вибір даних [38-39], на яких буде навчатися отримана модель. Особливо це є важливим при навчанні великої та складної моделі. Через це, можна виділити декілька необхідних і достатніх умов для вибору набору даних, а саме, він повинен:

- бути репрезентативним для проблеми, що вирішується тобто містити достатню кількість прикладів, які охоплюють різні аспекти проблеми, бо в іншому випадку нейронна мережа може бути недостатньо навчена, або ж, взагалі, не мати достатньої точності у вирішенні поставленої задачі;
- бути якісним та збалансованим, тобто дані у ньому повинні мати певний стандартизований формат та не містити велику кількість шуму та аномалій, також, важливо, щоб була приблизно однакова кількість прикладів у кожному класі, бо інакше модель може краще навчитися виділяти лише певний з них;
- мати достатньо широкий спектр зображень, а не мати модифікацію одного і того ж у великій кількості разів, оскільки це дозволить значно швидше навчитися одній з моделей, при цьому залишивши іншу позаду;
- мати документацію та опис, а також йти від перевірених джерел, щоб уникнути зайвих проблем під час його аналізу та подальшого навчання, оскільки деякі приватні джерела можуть містити лише загальні дані в описаному форматі, а в середині мати будь-що інше;
- також, бажаною умовою є формат набору даних, тобто він повинен бути доступний для легкого аналізу та обробки звичайними моделями мови, що буде використовуватися.

Отже, засновуючись на вище описаних критеріях було знайдено декілька наборів даних, що стосуються тематики зображень облич або аватарів у різноманітних форматах. Розглянемо деякі з них.

Bitmoji faces [40] – набір даних, що містить близько 130 тисяч екземлярів намальованих лиць, що виражають певні емоції. Приклад елементів даного набору даних відображений на рис. 3.3.

Рисунок 3.3 — Елементи набору даних Bitmoji faces

Отже, виділимо основні плюси та мінуси даного набору даних. До його переваг можна віднести:

- достатньо велику детермінованість даних, тобто обличчя є достатньо чіткими та не містять зайвих елементів;
- велика різноманітність даних, бо більшість елементів є достатньо унікальними, що дозволяє отримувати зображення у будь-якому форматі;
- досить чітка документація, що саме міститься та у якому форматі.

До мінусів, з іншого боку, можна віднести такі параметри даного набору даних, як:

- відсутність розміченості даних, тобто неможливість вибору певних категорій даних, бо неможливо визначити перед навчанням моделей, що саме зображено на малюнку;
- дуже велика різноманітність даних, через що, моделі буде досить складно навчатися, оскільки кожне окреме зображення має досить багато відмінностей від усіх інших, а отже, буде складно зрозуміти якісь їх загальні риси.

Другим знайденим набором даних є Anime GAN [41], що містить більше 25 тисяч зображень різноманітних облич людей, згенерованих у стилі аніме. Приклад елементів даного набору даних відображений на рис. 3.4.

Рисунок 3.4 — Елементи набору даних Anime GAN

Знову ж, виділимо основні плюси та мінуси даного набору даних. До плюсів можна віднести:

- відносно велику кількість даних;
- велику різноманітність даних;
- наявність певних категорій у зображень, що дозволяє додавати елемент класифікації.

До мінусів:

- відсутність стандартизованого формату зображень, оскільки більшість з них мають якісь досить випадкові елементи, такі як фон або об’єкти заднього плану, що створить багато проблем під час передобробки даних;
- відсутність збалансованості, оскільки є дуже великий розкид за кількістю зображень у різних категоріях.

Ще одним розглянутим набором даних є Cartoons Faces [42], що містить близько 100 тисяч зображень та розмітки для них за декількома категоріями у кожному типі даних.

Приклади деяких з зображень, розміщених у наборі даних Cartoons Faces відображені на рис. 3.5.

Рисунок 3.5 — Елементи набору даних Cartoons Faces

Виділимо плюси та мінуси даного набору даних. До його переваг можна віднести:

- розміченість даних, а отже можливість створення певних категорій;
- Велика кількість даних;
- наявність меншого набору даних у 10 тисяч елементів, що можна буде використовувати для шивдкого тестування моделі;
- схожий формат даних, що складається з білого фону та елементів, що беруться з певного пулу елементів.

До мінусів можна віднести лише нереалістичність деяких елементів, оскільки їх частини поєднувалися випадковим чином при створенні набору даних, а отже деякі з них можуть мати некоректні пропорції з точки зору людини.

Існують ще багато інших наборів даних, але всі з них, мають певні недоліки, що не дозволяють їх використовувати, такі як:

- маленькі об’єми даних, а саме менше 5 тисяч зображень;
- відсутність розмітки даних;
- дуже багато категорій зображень, що дуже сильно відрізняються один від одного.

Отже, з усіх розглянутих наборів даних, найкраще підходить останній розглянутий, а саме - Cartoons Faces, оскільки він містить дані в одному форматі та надає можливість швидко та правильно розбити їх на певні категорії.

3.3 Висновки до розділу

В даному розділі було описано розроблюваний алгоритм, заснований на моделі GAN для генерації зображень та його зміни, а також було проаналізовано декілька наборів даних та обрано найбільш підходящий з них для вирішення поставленої задачі.

Під час опису алгоритму, що планується розробити та протестувати, було описано основні вимоги до входів та виходів моделей дискримінатору та генератору, а саме, було визначено, що генератор буде мати два входи, що будуть приймати випадковий шум та категорію даних. При цьому в результаті перетворень з них буде отримуватися очікуване кольорове зображення необхідного формату.

Дискримінатор, в свою чергу, буде також мати два входи, що будуть приймати зображення (реальне або згенероване) та категорію. На виході з нього буде отримуватися число від 0 до 1, що буде означати чи є дане зображення справжнім.

В залежності від виходу дискримінатору буде рахуватися функція втрат, як функція Вассерштейна, тобто буде врахований увесь можливий проміжок даних, а не лише 0 або 1.

Також, обидві моделі будуть перетворювати модель поступово, збільшуючи, або ж зменшуючи розширення зображення, що повинно дозволити зменшити кількість епох, необхідних для навчання моделі.

Після цього, було розглянуто деякі існуючі набори даних, що можуть бути використані у якості аватарів у соцмережі та їх плюси та мінуси. Основними мінусами більшості з них виявилась відсутність міток даних, що не надасть можливість якимось чином класифікувати зображення, або ж перенасиченість елементами, що значно ускладнить процес передобробки зображень.

В результаті був обраний набір даних від компанії Google, що містить близько 100 тисяч розмічених зображень облич від декількох художників, усі з яких складаються з приблизно однакового набору елементів, що обирається з певного пула.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНОЇ МОДЕЛІ

4.1 Вибір технологій

Оскільки була поставлена задача створення програмної реалізації застосунку для навчання та використання моделі, обов’язковим та досить критичним вибором є саме вибір мови програмування та фреймворку, на якому дана модель буде реалізована. Правильний вибір може значно зменшити кількість необхідного коду та допомогти навчити модель набагато швидше. Некоректний вибір, з іншого боку, може створити, в свою чергу, досить багато проблем, таких як час відпрацювання програми та неможливість знаходження методів та засобів, необхідних для реалізації моделі. Що, в свою чергу, буде потребувати набагато більше часу на написання та тестування коду, який можна було б витратити на тестування саме моделі.

4.1.1 Вибір мови програмування

Оскільки задачею, поставленою у роботі є створення моделі, що може генерувати зображення, а не побудова складного застосунку з великою кількістю шарів логіки, то вибір мови програмування варто робити з урахуванням даного обмеження.

Отже, найбільш швидкими мовами програмування, що, в цілому, також мають певні бібліотеки для роботи з нейронними мережами та є такі мови, як C, C++, Python та Java. Коротко розглянемо кожну з них та виділимо основні переваги та недоліки.

Розпочнемо огляд з мови C, яка вважається, напевно, найшвидшою мовою програмування, оскільки вона взаємодіє майже безпосередньо з процесором, забезпечуючи дуже високу швидкість виконання програм. Однак, її основний недолік полягає в складному синтаксисі, що ускладнює розуміння коду. Тому мову C рідко використовують для проектів, де зрозумілість коду має пріоритет перед

швидкістю виконання. Крім того, у даній мові програмування немає загальновідомих фреймворків або бібліотек для реалізації алгоритмів машинного навчання.

Наступною мовою, що буде розглянута, є мова C++. Незважаючи на те, що вона має дуже високу продуктивність, через те що запускається на нижніх рівнях операційної системи, вона не є дуже популярною для роботи з даними. Так само, як і C, C++ синтаксис даної мови для розуміння звичайної людини не є легким, тому без достатнього досвіду програмування на C++, написання програм може зайняти набагато більше часу, ніж при використанні високорівневих мов програмування. Незважаючи на те, що швидкість виконання програми має значення, важливіше все ж таки зручність модифікації програми, тому C++ може не бути найкращим варіантом. Стосовно ймовірних фреймворків для побудови та навчання моделей, дана мова має деякі варіанти, які, теоретично цілком можливо використовувати для цього, хоча будуть значні проблеми з документаціює.

Наступна мова, що буде розглянута є Python. Ця мова є, напевно, найбільш популярною на теперішній час, але, при цьому має свої переваги та недоліки. Перевагами Python є його поширеність та величезна кількість пакетів для виконання різних завдань, починаючи з роботи з мікроконтролерами і, закінчуючи створенням веб-додатків з великим навантаженням.

Крім того, Python має зрозумілий та простий синтаксис, завдяки чому поріг входу у програмуванні на даній мові є низьким та майже не потребує специфічних знань для початку розробки. З недоліків цієї мови можна виділити відсутність строгої типізації даних, але, в даному випадку це не буде дуже серйозною проблемою, оскільки буде матися справа в більшості зі звичайними числами та масивами тих самих чисел. Крім того, це є далеко не найшвидшою мовою, якщо порівнювати із C або C++. Незважаючи на це, Python має всі необхідні засоби для створення моделі нейронної мережі та її навчання. Ще однією перевагою даної мови є наявність середовищ розробки, в яких код можна компілювати та запускати частинами, що дозволяє запускати та тестувати функції на попередньо обрахованих результатах.

Останній розглядаємий варіант – Java. Як і Python, це досить популярна мова загального призначення, яка має високий час виконання та зручний синтаксис для розробки. В порівнянні з Python, вона має строгу типізацію, що є важливою для розробки великих застосунків. Однак, її мінусом є обробка помилок, через що можуть виникати випадкові проблеми з компілятором, середовищем розробки або самим кодом. Дана мова програмування має бібліотеки, на яких можна створювати моделі нейронних мереж та їх навчати, що однозначно є плюсом.

Отже, було розглянуто чотири високошвидкісних мов програмування різного призначення. Основними критеріями, крім високої швидкості обробки даних для фінального вибору є:

- зручність написання коду;
- можливість скомпілювати частину та зберегти результат у пам’яті/файлі, а не перезапускати весь застосунок;
- наявність фреймворку для машинного навчання, що буде містити більшість або усі необхідні компоненти для побудови мережі GAN;
- стабільність роботи (за власною думкою).

У таблиці 4.1 виставимо оцінки від 1 до 5 кожній мові за кожним з критеріїв та підрахуємо сумарне значення.

Таблиця 4.1 — Порівняльний аналіз мов програмування.

	C	**C++**	**Python**	**Java**
Зручність написання коду	2	3	4	5
Компіляція частинами	0	2	5	4
Фреймворк для машинного навчання	2	3	5	4
Стабільність	5	4	5	3
Результат	9	12	19	16

В результаті, оскільки мова програмування Python набрала найбільшу кількість балів та вона має всі необхідні елементи для застосування для її створення, вона була обрана для подальшої розробки.

4.1.2 Вибір фреймворку машинного навчання

У Python існує досить багато різноманітних фреймворків машинного навчання, кожен з яких має свої переваги та недоліки. Розглянемо декілька найбільш популярних з них та оберемо найбільш підходящий з них.

По-перше, розглянемо бібліотеки Scikit-learn [43], що є одним з найпопулярніших відкритих фреймворків машинного навчання у Python. Він має дуже великий спектр моделей для різноманітних алгоритмів класифікації, регресії, кластеризації, зменшення розмірності та вибірки даних. Даний фреймворк є простим у використанні та має дуже велику кількість документації. Однак, він не має можливостей глибокого навчання, через що не може використаний для створення моделі GAN.

З іншого боку, існує PyTorch, що є фреймворком з відкритим вихідним кодом, що використовується для глибокого навчання та створений компанією Meta. Він має простий синтаксис і дозволяє легко візуалізувати графи обчислень. Даний фреймворк дозволяє налагоджувати та налаштовувати моделі, що дозволяє розробникам швидко їх створювати та оптимізовувати. Мінусом, PyTorch є неможливість підтримки розпаралелювання на багатьох пристроях.

Tensorflow - ще одним фреймворк, що використовується для глибокого навчання та створений корпорацією Google. Він має багатофункціональну архітектуру та велику кількість алгоритмів та можливостей для роботи з великими обсягами даних. Даний фреймворк дозволяє використовувати графи обчислень, що дозволяє розпаралелювати операції на багатьох пристроях, включаючи CPU, GPU та TPU. Однак, TensorFlow у своєму чистому вигляді має досить складний синтаксис та вимагає великої кількості коду для налаштування моделей.

У певній мірі, гілкою Tensorflow є фреймворк Keras, що працює на його основі та може бути використаний для створення складних моделей з меншим обсягом коду. Він має досить простий та інтуїтивний API, що дозволяє швидко створювати та навчати моделі. Але, він не надає такої гнучкості, як Tensorflow або

PyTorch, через що для більш складних завдань може знадобитись використання базового фреймворку.

Так само, як і для мов програмування виділимо основні критерії вибору фреймворку та проаналізуємо найкращий з них. До таких критеріїв можна віднести:

- наявність глибокого навчання;
- відсутність необхідності підключення додаткових сторонніх бібліотек;
- наявність усіх необхідних компонентів;
- швидкодія.

До переліку для аналізу також включимо додатково комбінацію Tensorflow та Keras, оскільки вони досить часто використовуються разом та в цілому, чистий Keras може не мати великої кількості необхідних бібліотек. У таблиці 4.2 зображено відповідний порівняльний аналіз обраних фреймворків.

Таблиця 4.2 — Порівняльний аналіз обраних фреймворків.

	Scikit-learn	**PyTorch**	**Tensorflow**	**Keras**	**Keras + TF**
Наявність моделей глибокого навчання	0	5	4	5	5
Підключення сторонніх бібліотек	5	4	5	3	5
Наявність усіх необхідних моделей	3	5	3	5	5
Швидкодія	3	4	5	5	5
Результат	11	18	16	18	20

В результаті, найбільшу кількість балів було отримано комбінацією Keras + Tensorflow, оскільки вона набагато краще паралелиться та в цілому має набагато більше можливостей для запуску, аніж PyTorch, що є її основним конкурентом.

Отже, в результаті проведеного аналізу мов програмування та їх фреймворків для глибокого навчання, було прийняте рішення використовувати мову Python та фреймворк Keras у поєднанні з Tensorflow

4.2 Передобробка даних

Процес передобробки даних є дуже важливим елементом створення та навчання моделі, оскільки під час цього процесу аналізуються використовувані дані та відбуваються певні перетворення усього набору даних. Через це може підвищитися точність моделі та швидкість її навчання.

До основних категорій передобробки даних можна віднести:

- видалення шуму, або ж даних, що містять не дуже коректні елементи, такі як порожні значення у деяких моментах або ж певні критичні значення, що дуже виходять за межі більшості масиву даних;
- обрізка та масштабування даних, під час якого змінюються та зменшуються розміри даних, що дозволяє значно покращити швидкість навчання трохи поступившись якістю отриманих результатів;
- балансування класів, що дозволяє отримати приблизно однакову кількість прикладів у кожному класі, щоб у моделі не було направленості на один клас, або певну групу з усього можливого масиву результатів;
- нормалізація даних, що дозволяє значно зменшити вплив великих значень на модель, що покращує, у свою чергу, якість отриманих результатів.

4.2.1 Аналіз обраного набору даних

Для початку обробки набору даних, проведемо попередній аналіз даних, щоб подивитися існуючі елементи та їх розподіл.

По-перше, оскільки усі зображення у обраному наборі даних [42] містять набір певних категорій, то розглянемо усі існуючі категорії за кожною ознакою та спробуємо вивести з них певні закономірності. Дані категорії відображені у таблиці 4.3.

Таблиця 4.3 — Категорії елементів у наборі даних Cartoon Faces.

Назва ознаки	**Кількість підкатегорій**	**Опис ознаки**
chin_length	3	Висота підборіддя
eye_angle	3	Нахил ока всередину або назовні
eye_lashes	2	Видно вії чи ні
eye_lid	2	Зовнішній вигляд повік
eyebrow_shape	14	Форма брів
eyebrow_weight	2	Товщина лінії брів
face_shape	7	Форма лиця
facial_hair	15	Тип волосся на обличчі (борода, вуса)
Glasses	12	Тип окулярів
Hair	111	Тип волосся
eye_color	5	Колір райдужки очей
face_color	11	Колір шкіри
glasses_color	7	Колір окулярів, якщо присутні
hair_color	10	Колір усього волосся
eye_eyebrow_distance	3	Відстань між оком і бровами
eye_slant	3	Схоже до eye_angle, але відповідає саме за погляд, а не поворот ока
eyebrow_thickness	4	Розмір брів по вертикалі
eyebrow_width	3	Розмір брів по горизонталі, або в ширину

З усіх існуючих ознак найбільш помітними будуть наступні:

- face_shape – форма обличчя, що містить лише 7 підкатегорій;
- facial_hair – наявність бороди та її стиль, містить 15 підкатегорій;
- glasses – наявність окулярів та їх стиль, містить 12 підкатегорій;
- hair – тип волосся, або ж зачіска, містить 111 підкатегорій;

- face_color – колір шкіри обличчя, містить 15 підкатегорій;
- hair_color – колір волосся, містить 10 підкатегорій.

Під час проведеного аналізу було виявлено, що деякі категорії, такі як face_shape мають досить незначний вплив на загальне зображення, що можна побачити на рис. 4.1.

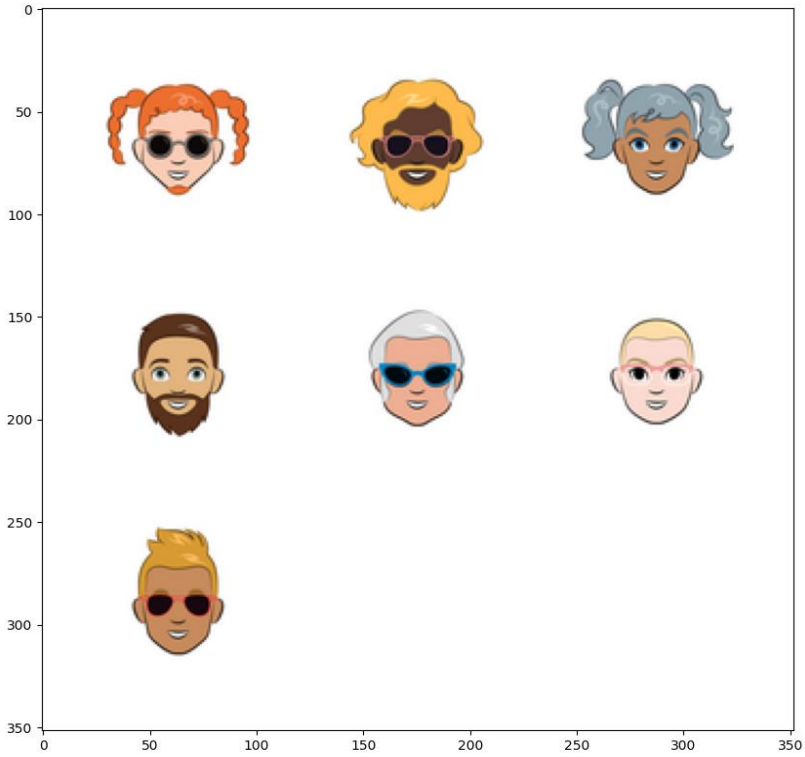

Рисунок 4.1 — Існуючі підкатегорії ознаки face_shape

З іншого ж боку, така категорія як hair, або тип волосся, має досить велику кількість різноманітних елементів. Усі їх можливі типи зображені на рис. 4.2.

Через це, дуже складно поєднати між собою, навіть значно зменшивши кількість категорій. Навіть при видаленні деяких, все ще досить багато елементів залишаться без конкретної категорії та їх розподіл не буде рівним, а отже, коли один елемент буде мати 100 зображень, інший може мати 300, 500, або навіть 1000.

Отже, можливо їх розбити лише на існуючі, без особливого об’єднання. Через це на кожну з яких в результаті не вийде набрати достатню кількість даних, оскільки навіть в наборі даних на 100 тисяч елементів, при розбиті на 100 категорій лише по типу волосся вийде не більше 1000 елементів на кожен тип.

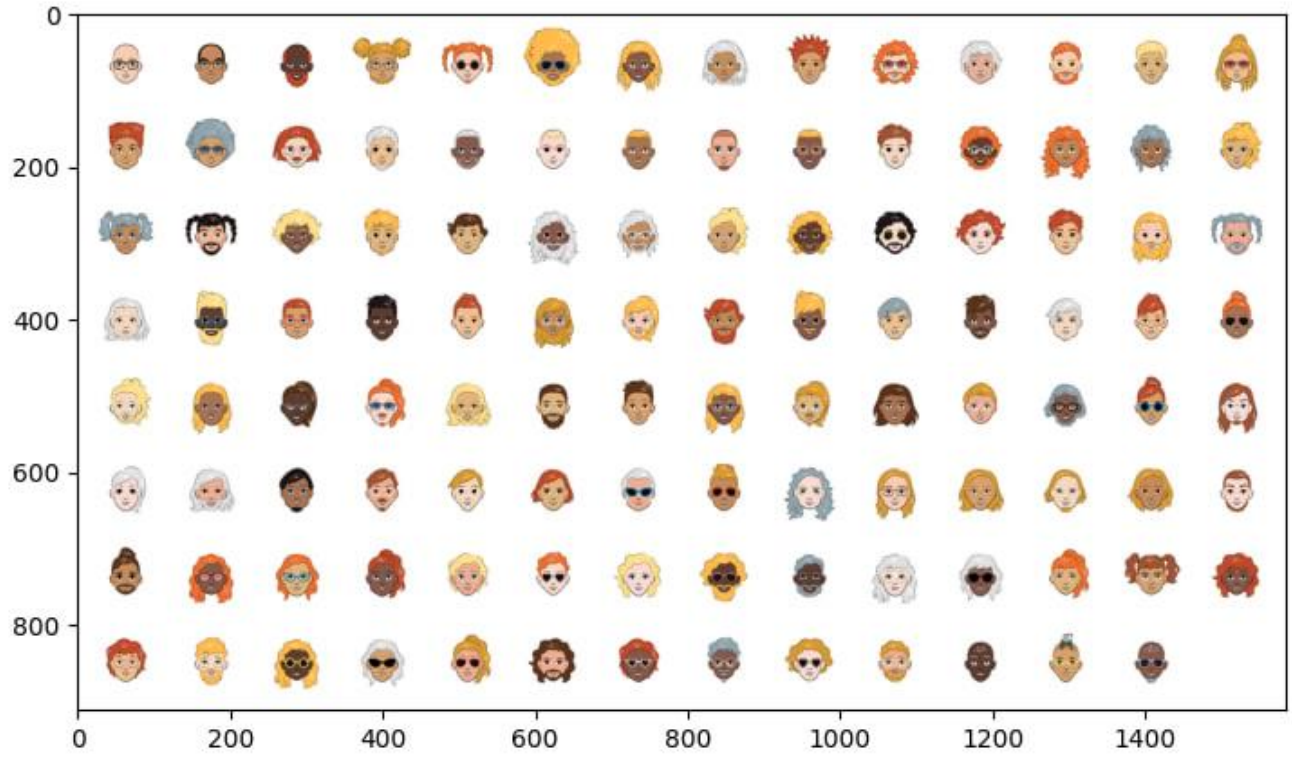

Рисунок 4.2 — Існуючі підкатегорії ознаки hair_color

Інші 4 категорії містять у собі досить адекватні дані, що будуть розглянути далі та дозволяють розбити себе на певні підкатегорії.

4.2.2 Розподіл набору даних за категоріями

Першою спробою розбиття на категорії було розбиття – як воно є, тобто створення категорій за кожною підкатегорією ознаки. Але, обрані чотири ознаки містять 15, 12, 15 та 10 підкатегорій, що при перемноженні дає:

$$15 * 12 * 15 * 10 = 27000$$

27000 можливих комбінацій. Оскільки набір даних містить 100 тисяч елементів, отже виходить, що кожна категорія буде містити від 3 до 5 зображень, що не є валідним для повноцінного навчання моделі.

Другою спробою було групування даних підкатегорій у більш загальні підкатегорії, але основною проблемою, що виникла під час даної обробки, була

неоднакова кількість даних за певними критеріями. Результати наявності підкатегорій за ознаками розміщені на рис. 4.3.

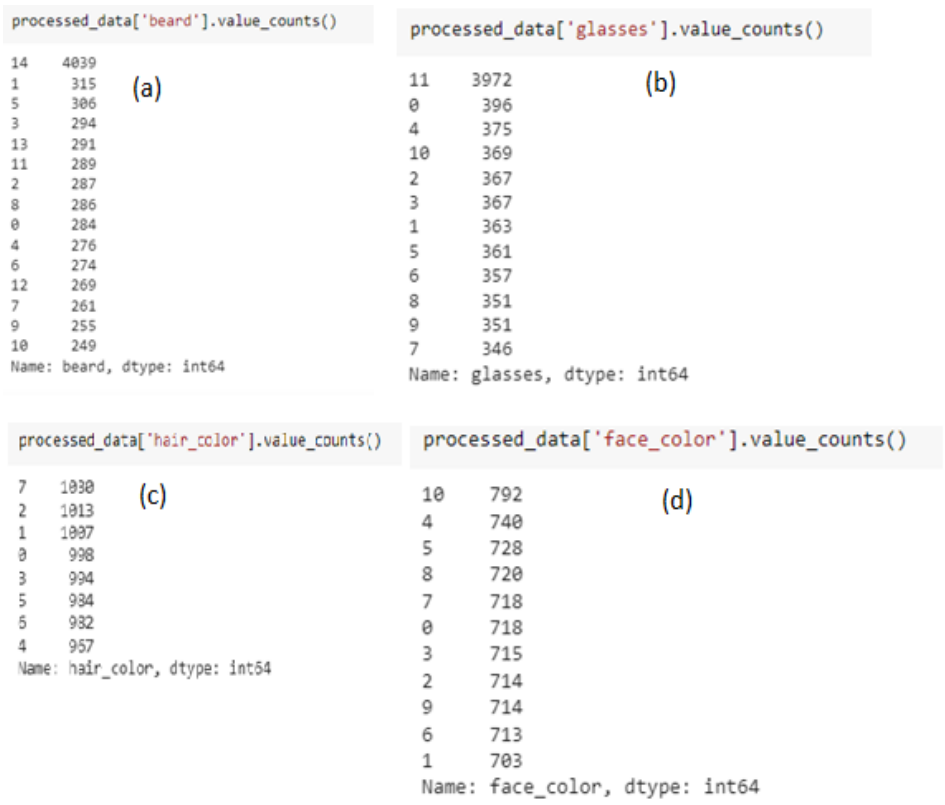

Рисунок 4.3 — Розподіл категорій за ознаками бороди (a), окулярів (b), кольору волосся (c) та кольору шкіри (d)

Отже, як можна побачити, останні дві ознаки мають приблизно однаковий розподіл за кожною з підкатегорій, а от перші дві містять деяких елементів набагато більше ніж інших.

Розглянемо кожну з ознак більш детально для того, щоб об’єднати можливі елементи у категорії, враховуючи їх кількість у наборі даних.

Почнемо розгляд з кольору шкіри. Його розподіл зображений на рис. 4.4. Як можна побачити, його досить легко можна розподілити на три основні категорії, що відобразимо у таблиці 4.4. Індекси ознак починаються з 0 та ідуть горизонтально починаючи з верхнього лівого кута.

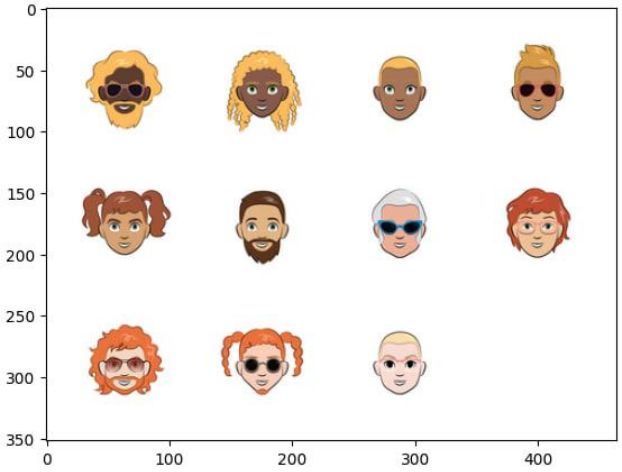

Рисунок 4.4 — Можливі варіації кольору шкіри

Таблиця 4.4 — Об’єднання підкатегорій ознак кольору шкіри.

Категорія	**Індекси підкатегорій**
Темна шкіра	[0, 1, 2]
Звичайна шкіра	[3, 4, 5, 6]
Світла шкіра	[7, 8, 9, 10]

Розподіл можливих кольорів волосся зображений на рис. 4.5. В цілому, його можна розподілити за кількома можливими категоріями, але, для того, щоб не створювати їх велику кількість, обмежимось двома. Відобразимо їх у таблиці 4.5. Індекси ознак починаються з 0 та ідуть горизонтально починаючи з верхнього лівого кута.

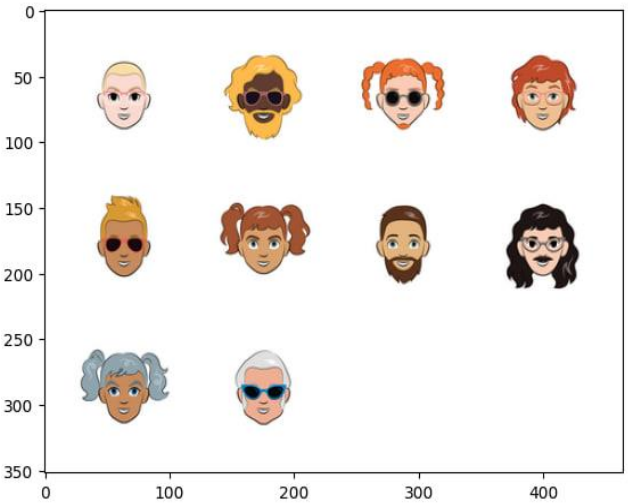

Рисунок 4.5 — Можливі варіації кольору волосся

Таблиця 4.5 — Об’єднання підкатегорій ознак кольору волосся.

Категорія	**Індекси підкатегорій**
Світле волосся	[0, 1, 2, 4]
Темне волосся	[3, 5, 6, 7]

При цьому у даному випадку є деякі кольори ([8,9]), що не мають дуже великого контрасту з фоном зображення, через що в результаті було вирішено видалити їх. Також, вони породжували проблему, при якій навчена модель не могла чітко визначити де закінчуються контури зображення, а де починається вже його фон, що значно знижувало якість самого зображення.

При аналізі наступної ознаки – окулярів, варто врахувати їх розподіл за кількістю у кожній категорії, через що було вирішено розбити усі можливі дані лише на дві основні категорії, що відображені у таблиці 4.6. Розподіл усіх можливих підкатегорій даної ознаки зображений на рис 4.6.

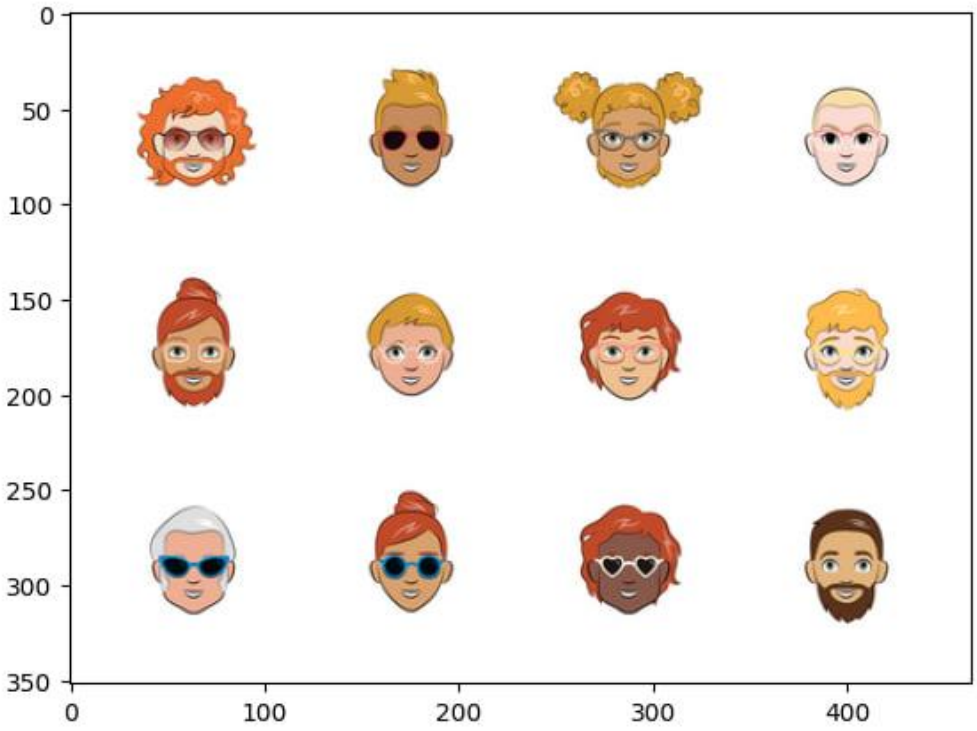

Рисунок 4.6 — Можливі варіації форми окулярів

Таблиця 4.6 — Об’єднання підкатегорій ознак окулярів.

Категорія	**Індекси підкатегорій**
Немає окулярів	[11]
Є окуляри	[0, 1, 2, …, 9, 10]

Останньою обраною ознакою є борода. Розподіл її підкатегорій зображений на рис. 4.7. Враховуючи кількість елементів у кожній з категорій було також вирішено розбити її лише на дві основні категорії за наявністю бороди. Даний розподіл відображено у таблиці 4.7.

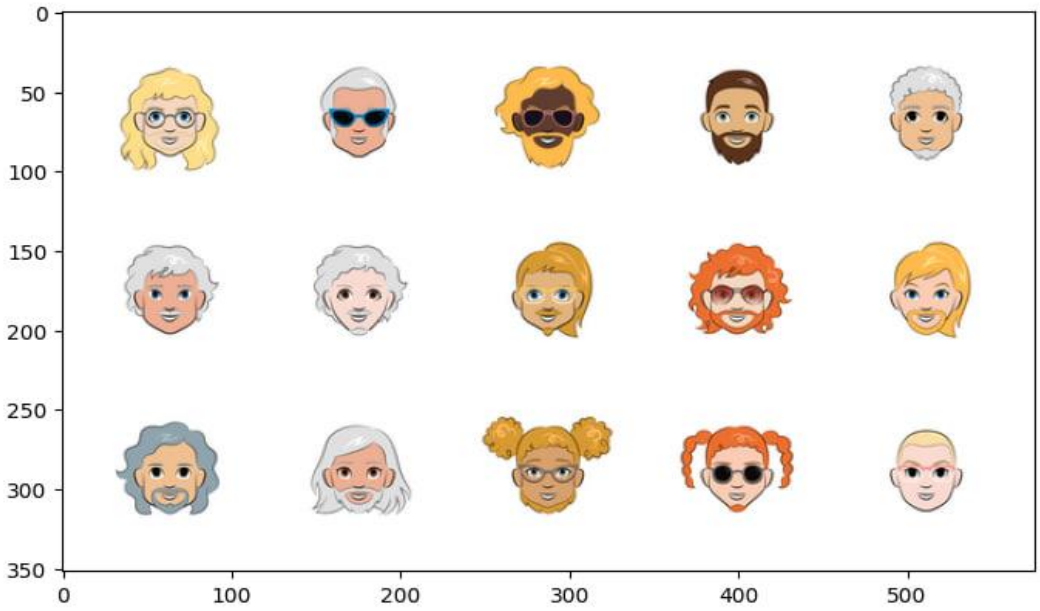

Рисунок 4.7 — Можливі варіації бороди

Таблиця 4.7 — Об’єднання підкатегорій ознак бороди.

Категорія	**Індекси підкатегорій**
Немає бороди	[14]
Є волосся на обличчі	[0, 1, 2, …, 11, 12, 13]

Отже, в результаті проведеного аналізу набору даних та його розподілу було отримано 24 категорії. Їх опис відображено у таблиці 4.8 та містить перелік ознак, що відповідає кожній з них.

Таблиця 4.8 — Отриманні категорії та їх індекси, що були використані для збереження та категоризації даних.

№	Індекс категорії	Кількість елементів	Опис категорії
1	2222	353	Звичайна шкіра | Борода | Окуляри | Темне волосся
2	1111	300	Темна шкіра | Немає бороди | Немає окулярів | Світле волосся
3	2223	359	Світла шкіра | Борода | Окуляри | Темне волосся
4	1113	336	Світла шкіра | Немає бороди | Немає окулярів | Світле волосся
5	1122	320	Звичайна шкіра | Борода | Немає окулярів | Світле волосся
6	2211	281	Темна шкіра | Немає бороди | Окуляри | Темне волосся
7	1221	263	Темна шкіра | Борода | Окуляри | Світле волосся
8	1223	364	Світла шкіра | Борода | Окуляри | Світле волосся
9	1222	356	Звичайна шкіра | Борода | Окуляри | Світле волосся
10	2121	261	Темна шкіра | Борода | Немає окулярів | Темне волосся
11	2112	350	Звичайна шкіра | Немає бороди | Немає окулярів | Темне волосся
12	2123	380	Світла шкіра | Борода | Немає окулярів | Темне волосся
13	1123	376	Світла шкіра | Борода | Немає окулярів | Світле волосся
14	1213	382	Світла шкіра | Немає бороди | Окуляри | Світле волосся
15	1211	254	Темна шкіра | Немає бороди | Окуляри | Світле волосся
16	2213	398	Світла шкіра | Немає бороди | Окуляри | Темне волосся
17	2113	349	Світла шкіра | Немає бороди | Немає окулярів | Темне волосся
18	1212	374	Звичайна шкіра | Немає бороди | Окуляри | Світле волосся
19	1121	246	Темна шкіра | Борода | Немає окулярів | Світле волосся
20	2212	345	Звичайна шкіра | Немає бороди | Окуляри | Темне волосся
21	1112	414	Звичайна шкіра | Немає бороди | Немає окулярів | Світле волосся
22	2122	384	Звичайна шкіра | Борода | Немає окулярів | Темне волосся
23	2111	256	Темна шкіра | Немає бороди | Немає окулярів | Темне волосся
24	2221	274	Темна шкіра | Борода | Окуляри | Темне волосся

При цьому, отримана вибірка даних містить приблизно однаковий розподіл елементів за кожною з категорій, а саме від 250 до 400 елементів. Це є досить невеликим розкидом на 10 тисяч елементів, що стане ще меншим у відсотковому відношенні при використанні більшого набору даних.

4.2.3 Нормалізація зображень

Важливим кроком для передачі моделі до моделі є нормалізація даних [44], що їй передаються, оскільки під час цього етапу відбувається перетворення даних таким чином, щоб їх розподіл став нормальним або близьким до нього. Це надає змогу зробити більш точні висновки з даних та використовувати більш точні статистичні методи.

Для нормалізації зображень, що необхідна у даному випадку, можуть бути використані такі методи, як:

- середнє значення, тобто нормалізація шляхом віднімання від кожного пікселя середнього значення пікселів зображення;
- стандартне відхилення, тобто нормалізація шляхом ділення кожного пікселя на стандартне відхилення пікселів зображення;
- нормалізація значень пікселів до діапазону від 0 до 1 або ж до діапазону від -1 до 1;
- зміщення яскравості або контрастності зображення.

Оскільки зображення з обраного набору даних містять, в цілому, більшість спектру можливих кольорів, то підрахунок відхилення або ж зміщення контрастності та яскравості не надасть великих покращень у результатів.

При цьому зменшення можливого розподілу пікселів від 0 до 255 до меншого значення дозволить нормалізувати дані та значно покращити отримані результати. Через це було вирішено привести вхідні дані зображень до розподілу від -1 до 1 за наступною формулою.

$$p_n = (p_m - 127.5)/127.5$$

Де p_n – значення отриманого пікселя після проведеного процесу нормалізації, p_m – значення відповідного пікселя до проведеного процесу нормалізації (число від 0 до 255), 127.5 – середина значень пікселів, що дозволяє їх привести до діапазону від -1 до 1.

Дана нормалізація дозволила привести вхідні дані моделі до нормалізованого формату для подальшої роботи з ними.

4.3 Створення моделей GAN

Оскільки описана модель складається з двох основних елементів – дискримінатору та генератору, далі буде розглянуто архітектуру обох з них окремо.

Після цього, розглянемо архітектуру самої моделі та її процес навчання та як саме воно відбувалося, необхідні параметри та алгоритми, що були використані для цього.

4.3.1 Розбиття даних

Перед процесом створення необхідно було розбити дані на масиви, що містять менші кількості елементів та зможуть бути навчені під час одного кроку однієї епохи навчання.

Під епохою розуміється один прохід через весь набір даних. Кожен етап навчання мережі складається з серії епох, під час яких нейронна мережа отримує на вхід певний набір даних та коригує свої ваги, щоб зменшити функції втрат та підвищити точність передбачень.

Оскільки під час кожної епохи обраховуються функції втрат, то під час закінчення кожної з них відповідні моделі оновлюються свої ваги, засновуючись на результатах тренування під час даної епохи. Її ж величина може варіюватися в залежності від розміру набору даних, складності моделі та ресурсів, що доступні для навчання. Якщо ж епоха занадто мала, то навчання буде відбуватися дуже повільно, якщо ж занадто велика – це може дуже швидко привести до

перенавчання. Оптимальною величиною кількості даних у одній епосі є значення, що знаходяться емпірично або ж за допомогою певних додаткових методів.

Під час навчання та тренування моделі було виявлено, що оптимальним розміром кроку епохи під час навчання є кількість зображень, що складає близько 64 зображень. Це дозволяє навчати модель досить швидко, при цьому уникаючи можливого перенавчання її на початкових етапах.

Ще одним моментом є зміна розмірності даних, оскільки з одного боку зображення мають розмір 256x256, що є досить великим для моделі навчання та значно збільшить кількість використовуваних коефіцієнтів. Через це воно було зменшене приблизно у три рази до розширення 96x96.

Другим моментом є кольоровість зображення, що означає, що кожне зображення має декілька слоїв, що комбінуються та в результаті надаються на вихід очікувані результати.

Розглянемо основні формати збереження та використання зображеннь. RGB (Red, Green, Blue) – формат зображення, що складається з трьох каналів: червоного, зеленого та синього. Кожен з даних каналів містить значення яскравості пікселів від 0 до 255. Найбільш стандартний формат для зберігання та обробки більшостів цифрових зображень.

HSV (Hue, Saturation, Value) – ще один інший формат кольорів, який використовується для представлення кольорів у зображеннях. У форматі HSV, колір представлений трьома значеннями: відтінок (hue), насиченість (saturation) та значення (value). Відтінок визначає колір самого пікселя, насиченість визначає яскравість даного кольору, а значення визначає яскравість самого пікселю.

GRAYSCALE – це є найбільш звичайним форматом зображення для збереження чорно-білих фотографій, що складається з одного каналу яскравості, де кожен піксель має значення відтінку від 0 до 255.

CMYK (Cyan, Magenta, Yellow, Key) – ще один, досить популярний формат кольорових зображень, що використовується в друкованій продукції. У даному форматі, кожен піксель представлений чотирма значеннями: фіолетовий (cyan),

рожевий (magenta), жовтий (yellow) та їх ключ (key), що відповідає за яскравість зображення.

Для використання у даній моделі найбільше підходять формати RGB та HSV, але в результаті декількох проведених експериментів було виявлено, що формат RGB є набагато легшим для моделі та видає набагато менше аномалій в результатів, аніж інші формати.

Таким чином, перед початку навчання моделі необхідно виконати наступні етапи передобробки вхідних даних:

- завантажити зображення;
- зробити перетворення, що зменшить загальний розмір зображення від 256x256 пікселів до розширення у 96x96;
- розбити кожне зображення на три шари, що будуть зберігатися та оброблятися певним чином окремо один від одного;
- завантажити категорію під кожне з очікуваних зображень та поєднати їх для подальшого використання.

При цьому, алгоритм для перетворення великої кількості даних займає досить багато часу та вимагає великих обчислювальних потужностей для обробки близько 80 тисяч зображень, що були отримані в результаті процесу класифікації.

Через це, його результати зберігаються окремо на диску після виконання даного процесу, що дозволяє значно прискорити усі наступні запуски моделі, оскільки отримані дані є статичними та не вимагають додаткових змін для свого використання.

4.3.2 Модель дискримінатору

В цілому, модель дискримінатору має загальну архітектуру, що зображена на рис. 4.8. Він приймає на свій вхід категорію та відповідне зображення, після чого намагається визначити чи є вхідні дані реальними або ж згенерованими. Оскільки було вирішено використовувати функцію витрат Вассерштейна, то на своєму

виході він видає значення у проміжку від 0 до 1, де 0 означає згенероване зображення, а 1 – реальне.

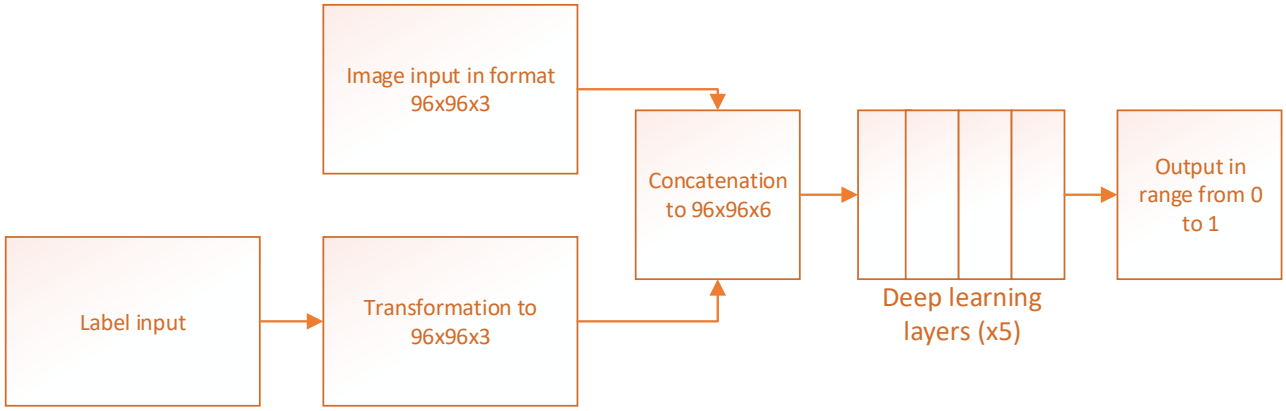

Рисунок 4.8 — Архітектура створеної моделі дискримінатору

Розглянемо більш детально, що відбувається у самій нейронній мережі дискримінатору та з яких елементів вона складається. Усі шари та їх елементи даної моделі відображено у таблиці А.1.

Повна архітектура створюваної моделі дискримінатору зображена на рис. Б.1. Розглянемо дві основні частини дискримінатору трохи більш детально, а саме перетворення даних та один крок, що відбувається під час навчання моделі та перетворення даних для вирішення, чи є зображення, подане на вхід, реальним чи згенерованим.

На рис 4.9 відображено частину моделі, що приймає дані. Як можна побачити з даного малюнку, модель дискримінатору має два входи:

- вхід input_2, що приймає зображення з розширенням 96x96 у форматі RGB, тобто містить три шари для відповідних кольорів;
- вхід input_1, що приймає категорію (число від 0 до 23 включно) та перетворює її спочатку до формату одновимірного масиву, що містить 96x96x3 елементів, а потім перетворює ці дані до тривимірного масиву.

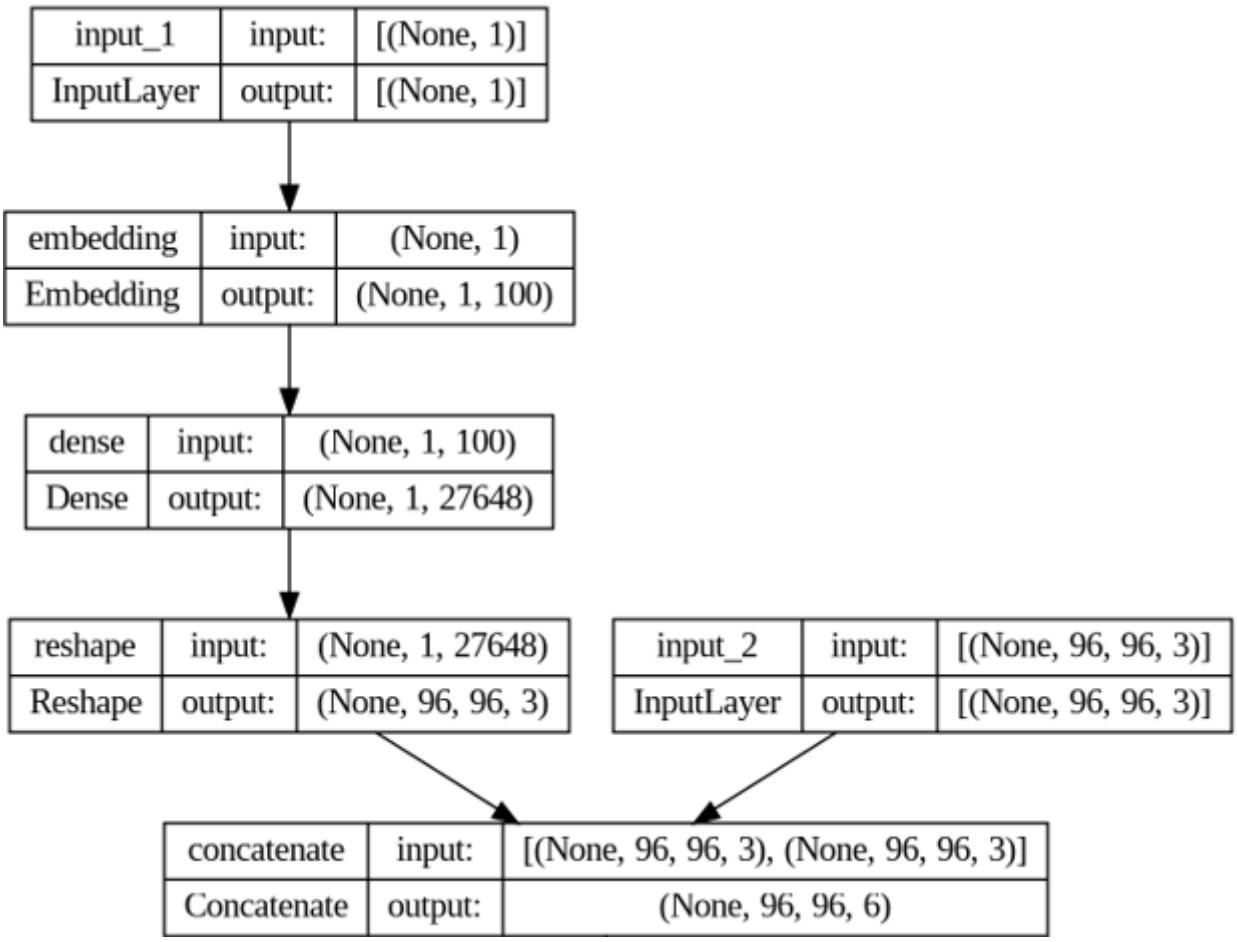

Рисунок 4.9 — Частина моделі дискримінатору, що відповідає за отримання даних

Після відповідних перетворень, дані входи об’єднуються та їх результат об’єднується, тобто в результаті отримаємо масив розмірністю 96x96x6, що вже і використовується у самій моделі для виконання усіх подальших перетворень даних.

На рис. 4.10 відображено частину моделі дискримінатору, що використовується як один з етапів перетворення даних. У створеній моделі додано п’ять таких шарів, що, відповідно, один за одним змінюються розмірність зображення для приведення його одного значення, що буде відповідати за вихід моделі.

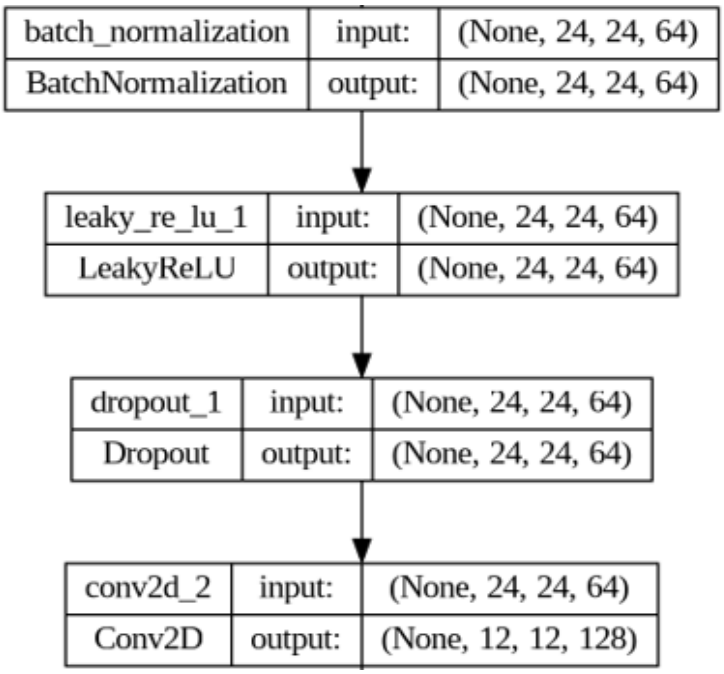

Рисунок 4.10 — Частина моделі дискримінатору, що працює як шар для перетворення отриманих даних

Для виконання даного перетворення, отриманий шар містить чотири наступні перетворення:

- нормалізація даних попереднього входу, для приведення їх до однакового формату;
- функція активації LeakyRelu (більш детально описано у пункті 4.3.4), що приводить значення попередньої операції до проміжку значень від 0 до 1;
- Dropout – що використовується для встановлення нульових значень деяким елементам моделі та дозволяє підвищити час навчання моделі, щоб мати відповідний час для навчання генератору;
- Conv2d шар, що використовується для здійснення згорткових операцій над зображеннями у двовимірному просторі, що дозволяє виявляти локальні шаблони, що, в свою чергу, допомагає розпізнати певні патерни на зображенні.

В результаті виконання усіх цих операцій та виводу результату моделі отримується число від 0 до 1 з двох входів – категорії та власне самого зображення.

4.3.3 Модель генератору

Модель генератору, в свою чергу, у даній моделі має загальну архітектуру, що зображена на рис. 4.11. Він приймає на свій вхід категорію та шум, що перетворюються до формату зображення з низьким розширенням, яке потім, поступово і потрохи збільшується з кожним наступним шаром мережі. Це надає змогу отримати кращі результати, оскільки зображення генерується не цілком та повністю для усього масиву пікселів, а лише для певних секторів у ньому, що поступово зменшують свої розміри, покращуючи відповідно якість отримуваного зображення.

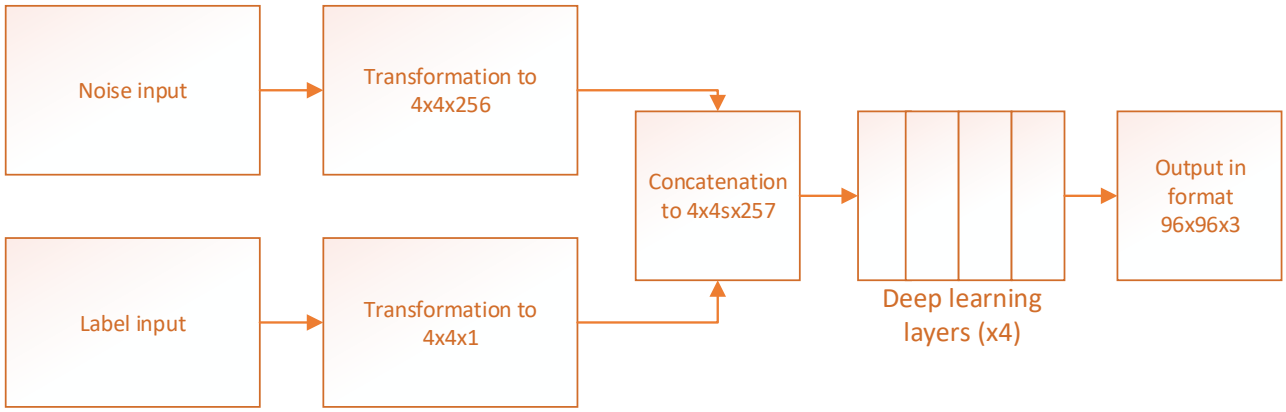

Рисунок 4.11 — Архітектура створеної моделі генератору

Так само, як і з моделлю дискримінатору, розглянемо більш детально, що відбувається у самій нейронній мережі та з яких елементів вона складається. Усі шари та їх елементи даної моделі відображено у таблиці А.2.

Повна архітектура генератору зображена на рис. Б.2. Розглянемо дві основні частини її трохи детальніше, а саме перетворення даних та один крок у їх генерації.

На рис. 4.12 зображено вхідні дані, що передаються до дискримінатору. Як можна побачити з малюнку модель має два основні входи:

- шум, що представлений у форматі масиву, що має 100 елементів, що в цілому складають гаусівський розподіл даних, після цього, він

перетворюється до формату 4x4x256, тобто зображення 4x4, що містить 256 шарів, кількість яких потім буде зменшуватися, збільшуючи розширення;

- мітка, що складається з масив, що містить один елемент, який перетворюється до формату 4x4x1, для приведення до такого самого розширення, як і шум.

Після цього, дані шари об’єднуються та отримується розмірність 4x4x257, що вже потім передається у наступні шари, де, власне, і відбувається генерація даних.

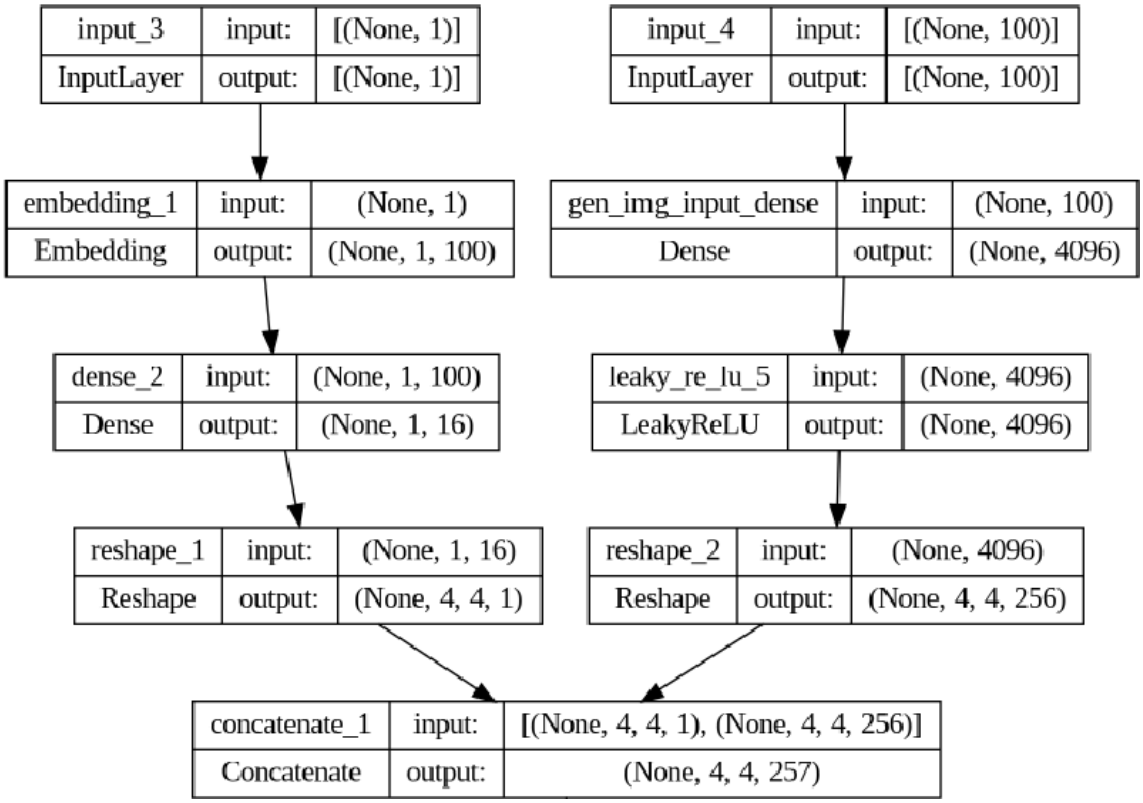

Рисунок 4.12 – Частина моделі генератору, що відповідає за отримання даних.

Далі розглянемо один з шарів генератору, в яких і відбувається перетворення. Його задача, в цілому, полягає у підвищенні якості зображення у 2 рази, при цьому додаючи якість відповідним елементам, присутнім у моделі.

Один з прикладів такого шару зображено на рис. 4.13, де можна побачити, що він містить 3 елементи:

- нормалізація входів – приведення їх до однакового розподілу даних;
- функція активації, що має форму LeakyReLU та буде розписана у пункті 4.3.4;
- Conv2DTranspose – власне шар нейронної мережі, що використовується для здійснення деконволюційних операцій, тобто він бере на вхід зображення та збільшує його розмір, зменшуючи при цьому кількість каналів.

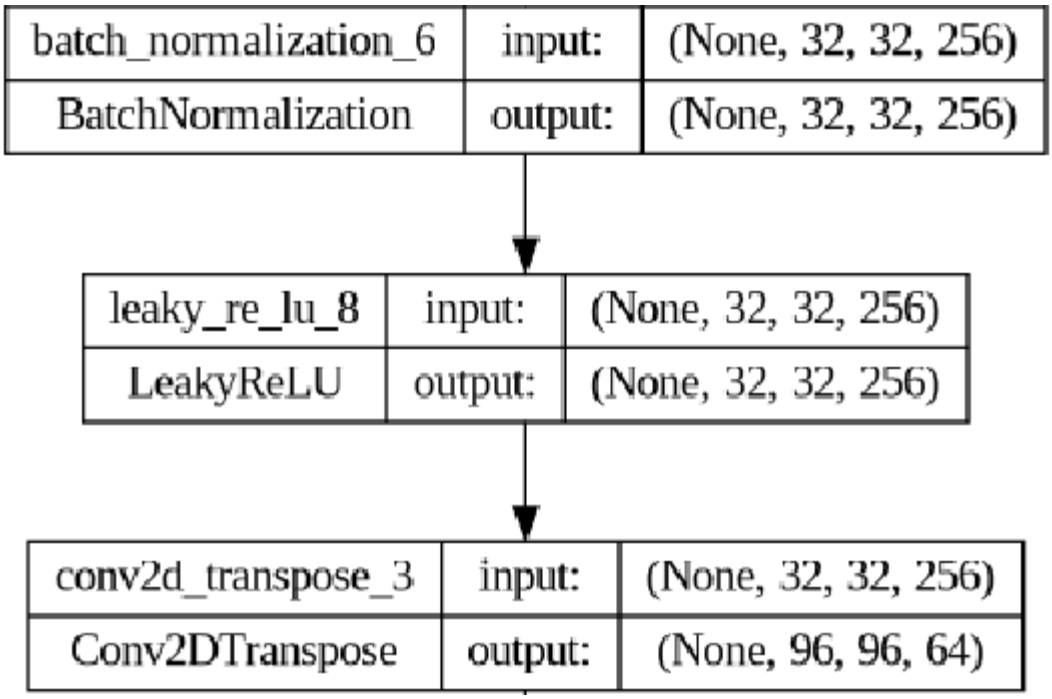

Рисунок 4.13 – Частина моделі генератору, що відповідає за перетворення даних або ж відображує один шар нейронної мережі.

В результаті, після проходження через низку таких шарів, зображення перетворюється від розширення 4x4 до 96x96x64, після чого зменшує кількість каналів до трьох та отримується фінальне результуюче зображення з розширенням 96x96x3.

4.3.4 Параметри навчання моделі

В цілому результуючі моделі дискримінатору та генератору мають близько п’яти мільйонів параметрів кожен. І всі ці параметри необхідно не лише ініціалізувати, а ще й навчати та потрохи їх змінювати для отримання певних значущих результатів, що будуть схожі на реальні зображення.

Під час навчання моделі для коригування її коефіцієнтів було використано деяка кількість функцій. Розглянемо їх трохи детальніше.

По-перше, була використана функція активації. Це є нелінійною функцією, що застосовується до вихідного значення нейрона з метою введення нелінійності. Без її використання, нейронна мережа буде здатна моделювати лише лінійні відношення між вхідними і вихідними значеннями, що значно обмежує її потенціал.

Розглянемо функції активації, що використовуються у розробленій мережі. Leaky ReLU (Rectified Linear Unit), що приймає значення від мінус нескінченності до нескінченності. Її плюсом є дуже велика швидкість відпрацювання, що дуже добре підходить для великих мереж з багатьма шарами. Дана функція зазвичай використовується при обробці зображень та звукових даних. Її можливість видавати від’ємні значення допомагають уникнути «мертвих» нейронів, що мають нульові ваги під час навчання, які значно підвищують шанс збіжності моделі. Її формула має наступний вигляд:

$$\{F(x) = x, x > 0\ F(x) = 0.01x, x \leq 0$$

Tanh (гіперболічний тангенс) – ще одна функція активації, що приймає значення від -1 до 1 та зазвичай використовується всередині мережі для стабілізації вихідних значень. Має трохи більший час відпрацювання ніж ReLU, що не дозволяє її використовувати у всіх шарах складних моделей. Її формула має наступний вигляд:

$$tanh\,(x)\ = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Ще одним типом функцій, що була використана для навчання моделі була функція втрат. В цілому, вона мала вигляд функції втрат Вассерштейна, але, також, додатково були використані функції витрат, що вираховують ентропію між фактичними та прогнозованими мітками, де кожна з них може приймати значення від 0 до 1.

4.4 Принцип навчання моделі

Як вже було зазначено навчання моделі відбувалося групами по 64 зображення в результаті яких, змінювалися відповідні коефіцієнти у дискримінаторі та генераторі. При цьому, усі дані зображення мали розширення 96x96 пікселів.

Сам процес навчання складається з наступних кроків:

- генерується набір зображень з випадкового шуму генератором, що потім буде використано дискримінатором;
- даний набір зображень передається дискримінатору та обраховується функція втрат в залежності від його відповідей;
- використовується функція Вассерштейна для обрахування коефіцієнтів з даних втрат та змінення коефіцієнтів моделей;
- обирається набір реальних зображень, що передаються до дискримінатору, з яких він рахує їх приналежність до реальних або згенерованих та обраховується функція втрат для реальних зображень;
- використовується функція Вассерштейна для обрахування коефіцієнтів з даних втрат та змінення коефіцієнтів моделей;
- генеруються нові зображення, з яких потім обраховується функція втрат, що йде у загальну статистику по завершенню обрахунків.

В результаті під час навчання відбувся прохід по всьому набору даних зображень 40 разів. При цьому, він містить близько 1200 груп зображень, що в результаті призводить до 50 тисяч епох навчання, в результаті яких отримуються зображення.

4.5 Висновки до розділу

В даному розділі було описано весь процес побудови програмної реалізації моделі, що була розроблена, що складався з генератору та дискримінатору.

При цьому, спочатку було описано технології, що були використані та обґрунтовано їх вибір. Отже, було вирішено розробляти дану модель на мові програмуванні Python, що є найбільш популярною мовою для розробки моделей нейронних мереж та глибокого навчання. Також, було вирішено використати модель TensorFlow, що має інтерфейс для створення та роботи з цими самими мережами, оскільки вона може компілюватися у код на мові C++, що працює значно швидше, ніж звичайний код, написаний на мові Python.

Після цього, було описано як відбувалася категоризація та передобробка даних, а саме виділення 24 категорій даних та зменшення розмірів зображення з 256x256 пікселів до 96x96 пікселів. Значення кольорів вже зменшеного зображення було нормалізоване до розподілу [-1, 1] для того, щоб модель могла працювати з нормалізованими даними, що дозволило підвищити її якість та швидкість навчання.

Також, було розглянуто основні елементи генератору та дискримінатору, що були застосовані у моделі та приклад одного з їх шарів, що використовується у них. Генератор має два входи, один з яких є випадковим гаусівським шумом, а другий – номером категорії. Він містить чотири основні скриті шари, що перетворюють шум у зображення. Дискримінатор, у свою чергу, приймає зображення у форматі 96x96x3 та категорію, що потім перетворює їх у значення від 0 до 1.

У кінці було описано як відбувається процес навчання та що за функції втрат та активації використовуються у шарах нейронної мережі. Сам процес навчання складається з 6 кроків, що відбуваються у кожній епосі та були запущені близько 50 тисяч разів.

5 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

5.1 Проблеми, що виникають при навчанні моделі

Кожну модель машинного навчання можна навчати досить довго, але, при цьому можуть виникнути три основні проблеми під час даного процесу навчання, що будуть впливати на результат:

По-перше, модель почне перенавчатися, тобто вона повністю вивчить патерни очікуваних даних, через що почне або працювати лише з ними та видавати на них 100% значення, при цьому на окремому текстовому наборі вона почне працювати все гірше і гірше.

По-друге, модель недонавчиться, або ж вона дійде до певного свого локального критичного значення, у якому буде видавати досить непогані результати, але не ідеальні. При цьому спроби змінити значення коефіцієнтів не будуть мати особливого впливу на модель через те, що вони не дадуть змогу вийти з цього локального максимуму або ж мінімуму, що залежить від моделі.

По-третє, модель досягає свого максимального потенціалу та ані процес подальшого навчання, ані навчання з іншими коефіцієнтами ніяк не зможуть покращити результат.

Якщо перші дві проблеми можна вирішити зберігаючи модель після певної кількості епох та запускаючи її навчання з іншим розподілом тестових даних або сідом для певних генерацій відповідно, то третя проблема може бути вирішена лише шляхом прямого втручання у саму модель та зміну певних її частин, таких як шарів, коефіцієнтів або ж функцій активації або втрат, як у випадку даної моделі.

Далі будуть розглянуті різноманітні методи тестування нейронних мереж, що можна використовувати для виявлення та визначення першої та третьої проблеми.

5.2 Методи тестування моделі

Розглянемо кожну з проблем окремо та можливі способи їх виявлення. Почнемо з проблеми перенавчання, або ж моменту, коли після певного етапу навчання модель починає видавати результати на випадкових даних набагато гірші аніж на тих самих даних декілька епох тому.

Найпростіший спосіб виявити це є побудова графіків витрат моделей. Оскільки у даній роботі будується моделі дискримінатору та генератору, то їх графіки втрат варто будувати окремо, при цьому, можна будувати втрати дискримінатору також окремо за реальними та згенерованими даними. По ним можна буде досить чітко зрозуміти момент, у якому модель буде навчена. Це буде відбуватися у критичному значенні, при якому значення обох моделей досягнуть свого локального максимуму, подальше ж навчання може зменшувати втрати однієї з моделей, при цьому збільшуючи втрати іншої.

Стосовно другої проблеми з максимальним потенціалом моделі досить гарно допоможе впоратись по-перше мануальне тестування, а саме перевірка отриманих результатів та їх якості за допомогою звичайного зору. Оскільки в цілому генерується обмежена кількість результатів через кожні N запусків моделі, то можна подивитись динаміку цих зображень та зрозуміти приблизно чи було досягнуто свого максимуму моделлю у моменті, коли модель все повністю навчилася.

По-друге, для цього існують коефіцієнти, що розглядалися у пункті 2.8 даної роботи, а саме IS, FID та KID. Їх обрахування надасть можливість порівняти моделі з точки зору результатів та виявити ту, що показує краще у тих чи інших умовах.

5.3 Мануальне тестування отриманих результатів

Почнемо тестування моделі з мануального тестування, оскільки це є найпростішим варіантом для визначення загальної якості моделі що генерує зображення.

Отже, на рис. 5.1 зображено декілька найкращих та зображень за епохами. Це надасть змогу зрозуміти у загальному процес навчання моделі та проблеми, що виникають у неї.

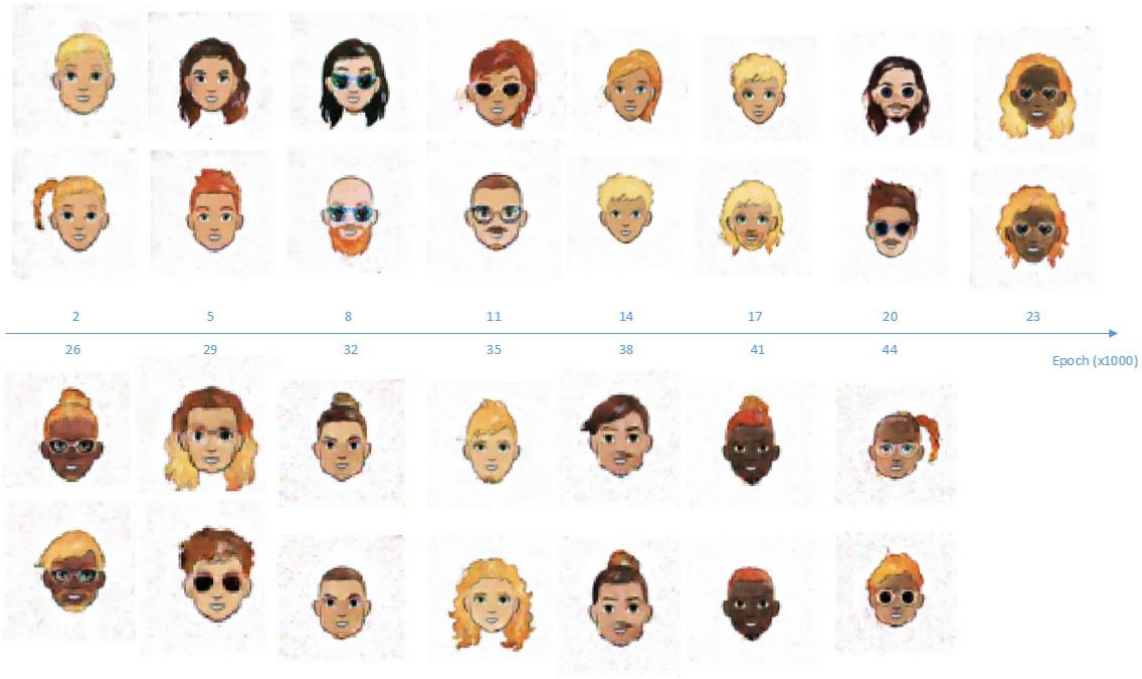

Рисунок 5.1 — Результати навчання створеної моделі за епохами, найкращі екземпляри

Як можна побачити з даного рисунку, приблизно до 30-35 тисячі епох модель активно навчалася та отримувала все кращі і кращі результати у кожній з епох.

Після цього, модель генератору зрозуміла, що не обов’язково видавати відмінні зображення, оскільки основним параметром є категорія і почала видавати найбільш прості зображення, що можна було створити.

Ближче до зупинки навчання, дискримінатор зміг визначити і ці зображення як згенеровані, через що генератор почав пробувати змінювати коефіцієнти, але, оскільки він потрапив у свій локальний максимум, то не зміг цього зробити та почав видавати все гірші та гірші результати.

Також, розглянемо найгірші результати, що видавалися моделлю генератору на тих самих епохах. На рис. 5.2 зображено деякі з прикладів таких зображень.

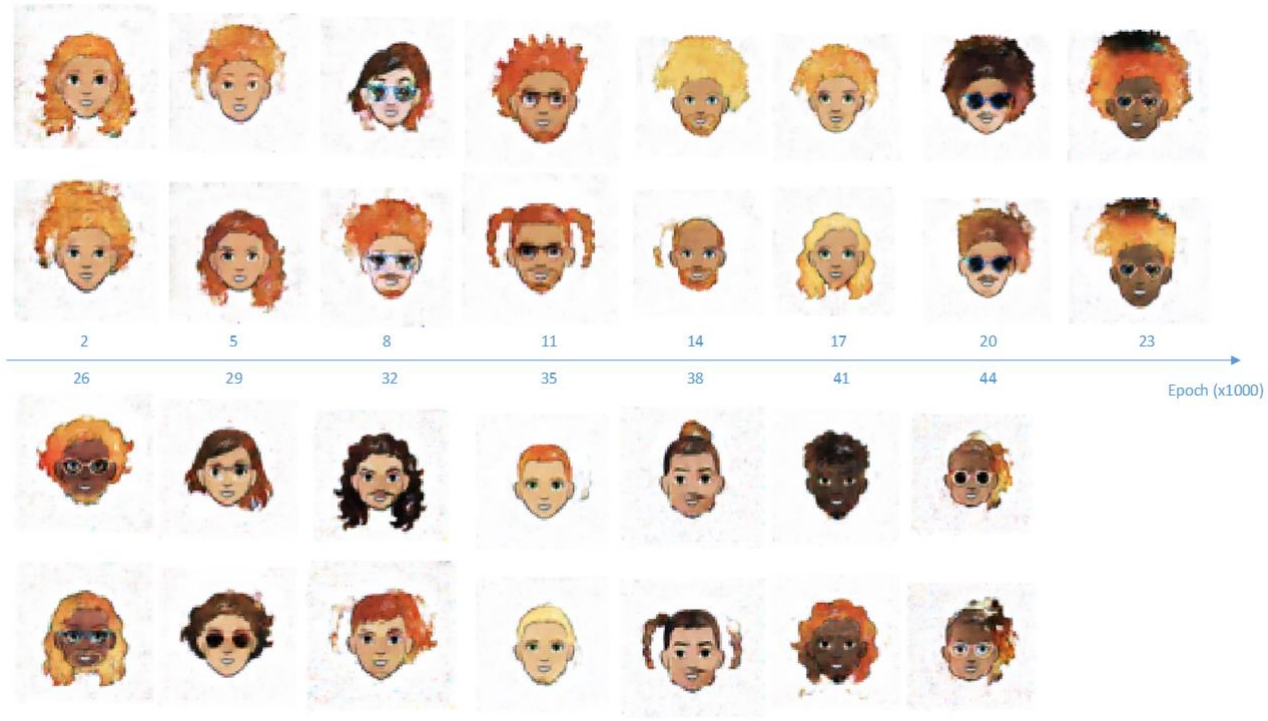

Рисунок 5.2 — Результати навчання створеної моделі за епохами, найгірші екземпляри

Як можна побачити з даного рисунку, основні дві проблеми, що виникають під час навчанні на даному наборі даних – досить великий рендж категорій, що дуже схожі, але мають мінімальні відмінності, такі як колір волосся, через що деякі зображення містять їх з кольором градієнту можливих кольорів даної категорії.

Також, другою проблемою, що неможливо вирішити на даному наборі даних є дуже велика кількість різноманітних складних зачісок, що дуже сильно виділяються з загального пула усіх можливих зачісок, через що, іноді, генератор не може зрозуміти як саме її побудувати через невелику кількість таких елементів у моделі дискримінатору та в результаті створює комбінацію з декількох, що видає досить незвичайний результат.

Але, хоч проблеми, що виникли при порівнянні найкращих зображень можна досить легко вирішити, проблеми, що виникли при порівнянні найгірших можна лише спробувати мінімізувати на даному наборі даних за допомогою зміни параметрів моделі. Приклади результатів таких змін відображено у пункті 5.5 даної роботи.

5.4 Графіки навчання моделі

Для визначення епохи, на якому обидві моделі досягли свого локального потенціалу побудуємо лінійні графіки втрат генератору та дискримінатору. Результат даної побудови відображений на рис. 5.3 та 5.4 відповідно.

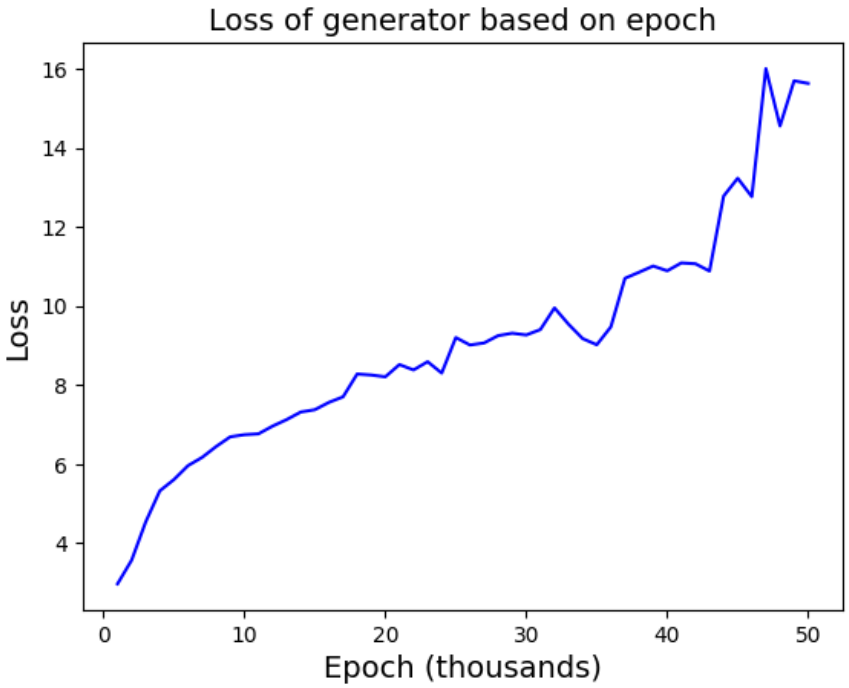

Рисунок 5.3 — Графік втрат моделі генератору

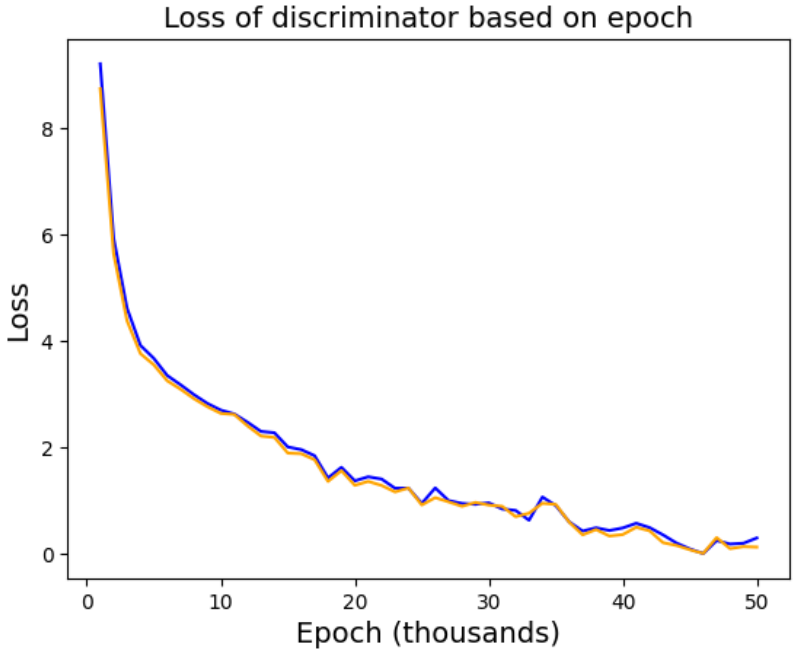

Рисунок 5.4 — Графік втрат моделі дискримінатору

Як можна побачити на даному графіку, модель дискримінатору змогла зменшити та стабілізувати свої втрати приблизно на 36 тисячі епох, в той час як модель генератору досягла свого локального мінімуму втрат біля 34 епохи, після чого графік дуже різко почав рости вгору через те, що дискримінатор навчився повністю відрізняти типи зображень.

Отже, таким чином, можна виявити, що обидві моделі досягли свого найкращого результату приблизно на 35 епосі, після чого модель дискримінатору стала дуже швидко покращувати свої результати, певними скачками, в той час як модель генератору почала різко збільшувати свої втрати. Отже, оптимальною епохою для зупинки навчання моделі є саме 35, що і буде обрана для наступних тестів та саме її результати відображені у додатку Б даної роботи.

5.5 Варіанти, отримані при зміні коефіцієнтів

Оскільки було зроблено досить багато спроб навчання моделі з різними коефіцієнтами, то у даному розділі наведемо деякі екземпляри створених зображень під час цих спроб та опишемо загальний процесу підбору цих самих коефіцієнтів.

На рис. 5.5 відображена спроба зменшення коефіцієнту dropout у моделі дискримінатору. Це призвело до того, що дана модель дуже швидко навчилася та отримала змогу виявляти усі основні елементи зображень та вже через 10-15 тисяч епох генератор перестав навчатися та просто пробував підібрати необхідні коефіцієнти.

Через це усі подальші зображення отримували досить випадкові аномалії, які ніяк не допомагали подальшому навчанню, а лише продовжували навчати дискримінатор все краще та краще.

Варто зазначити, що відображені згенеровані зображення є обраними з набору з 64 зображень на кожній епосі, а отже мають найкращі результати з отриманих. Усі інші згенеровані зображення, які не відображені на даному рисунку мали гіршу якість.

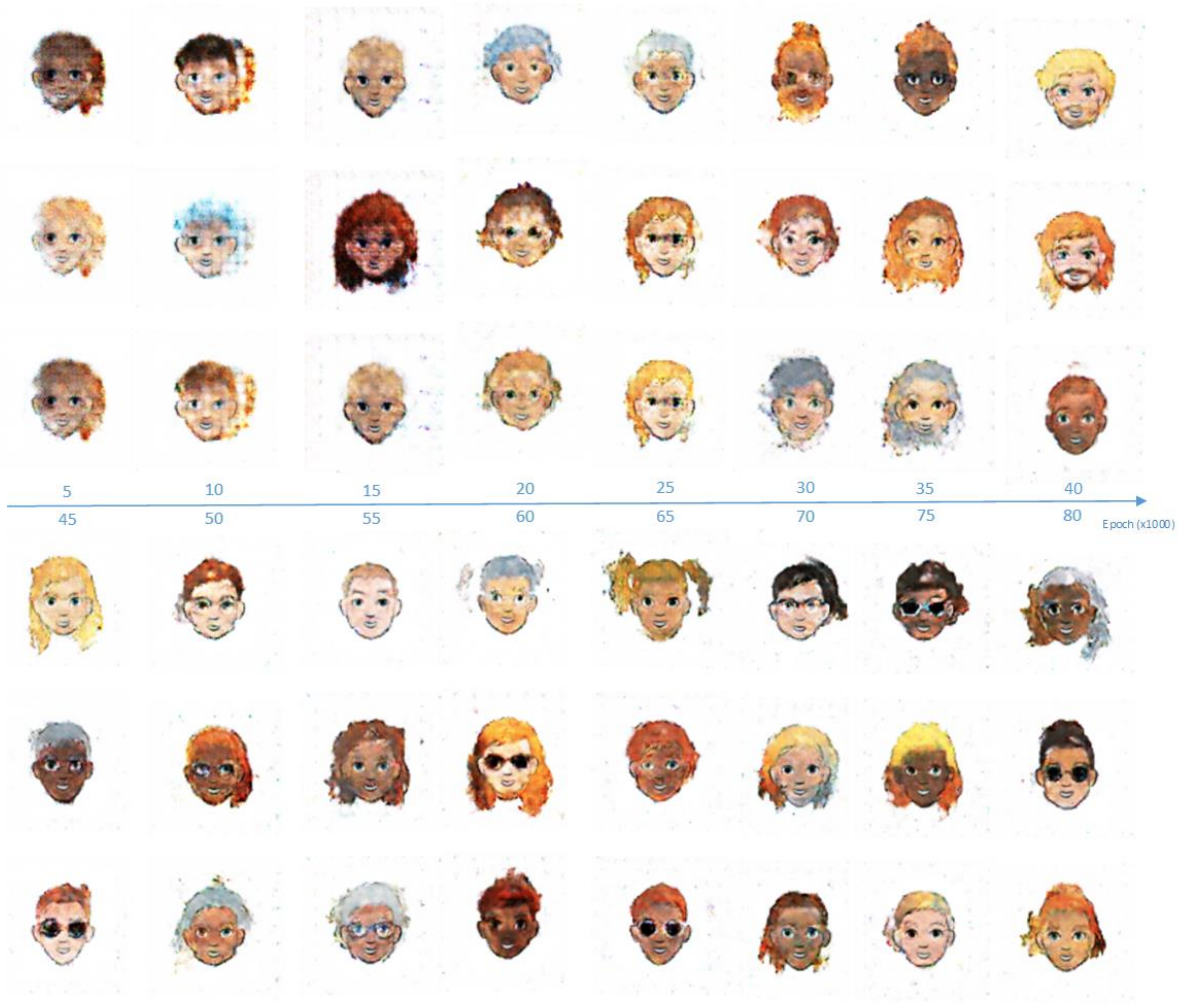

Рисунок 5.5 — Зменшення dropout за епохами

На рис. 5.6 відображено спробу збільшити коефіцієнти функції dropout до більш високих значень. В результаті це призвело до того, що навіть через 70 тисяч епох генератор не зміг повністю зрозуміти, що йому необхідно генерувати.

Як можна побачити з результату навчання, генератор поступово генерував все кращі і кращі результати, але, навіть через 80 тисяч ітерацій не зміг створити зображення, близьке до реального. Це відбувалося через те, що дискримінатор, втрачаючи велику кількість свої коефіцієнтів кожного разу намагався навчатися наново, а отже не міг визначити в результаті з високою точністю яким саме є дане зображення.

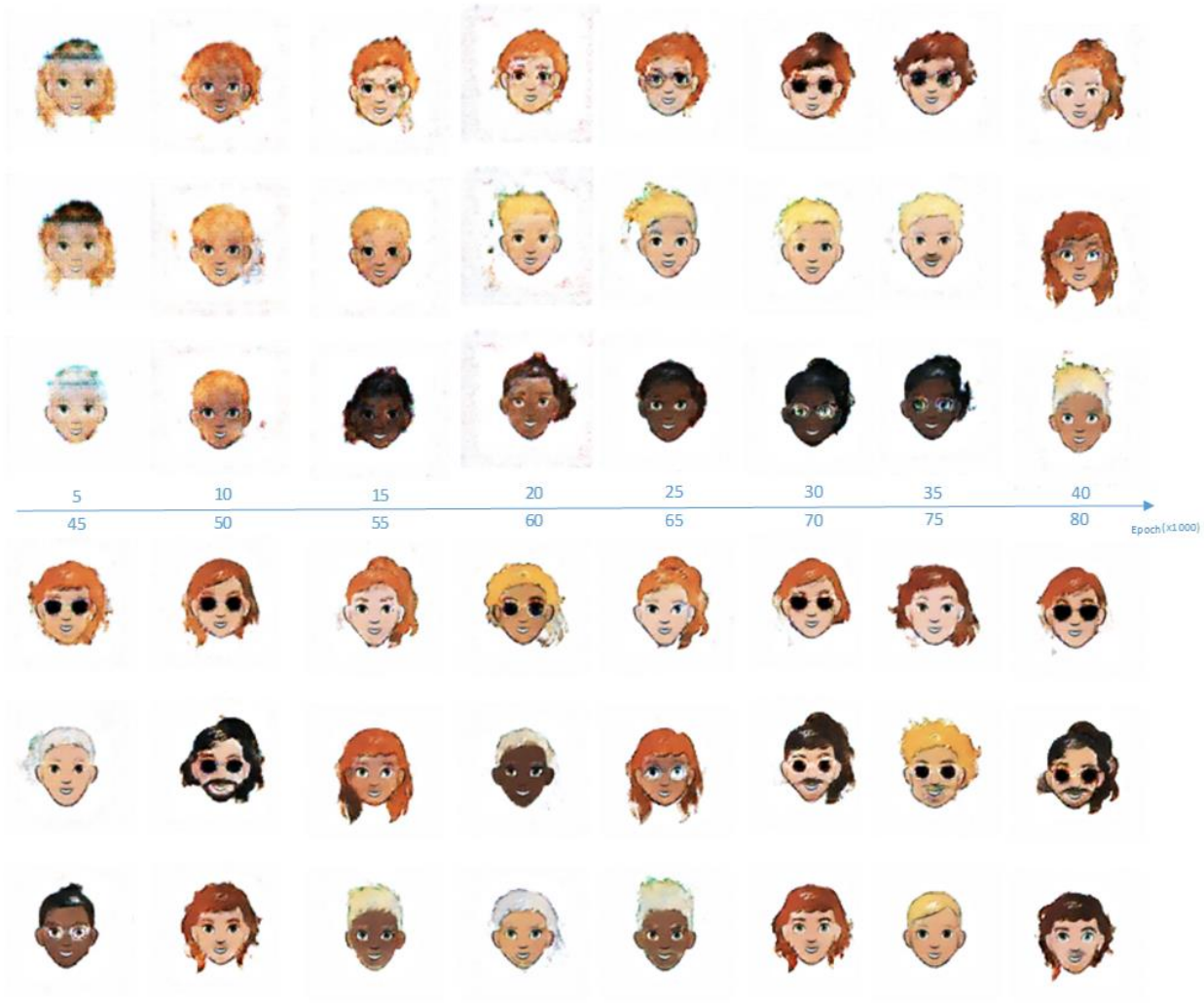

Рисунок 5.6 — Збільшення dropout за епохами

На рис. 5.7 відображено спробу зміни коефіцієнтів, що використовуються для навчання, а саме коефіцієнту, що змінює крок змін параметрів. У даному прикладі відбулося його зменшення.

Це знову ж призвело до того, що навіть після 70 тисяч епох генератор не зміг дійти до значень, близьких до реальних зображень через те, що кожна наступна епоха не мала великих змін у порівнянні із попередньою, хоч на кожній з них і відбувалося незначне покращення у порівнянні із попередньою.

Хоч, можливо, дана модель і досягла б значущих результатів через ще декілька сотень тисяч епох, але її навчання та фінальне налаштування для отримання найкращих результатів займає дуже багато часу, через що було вирішено зупинити навчання даної моделі.

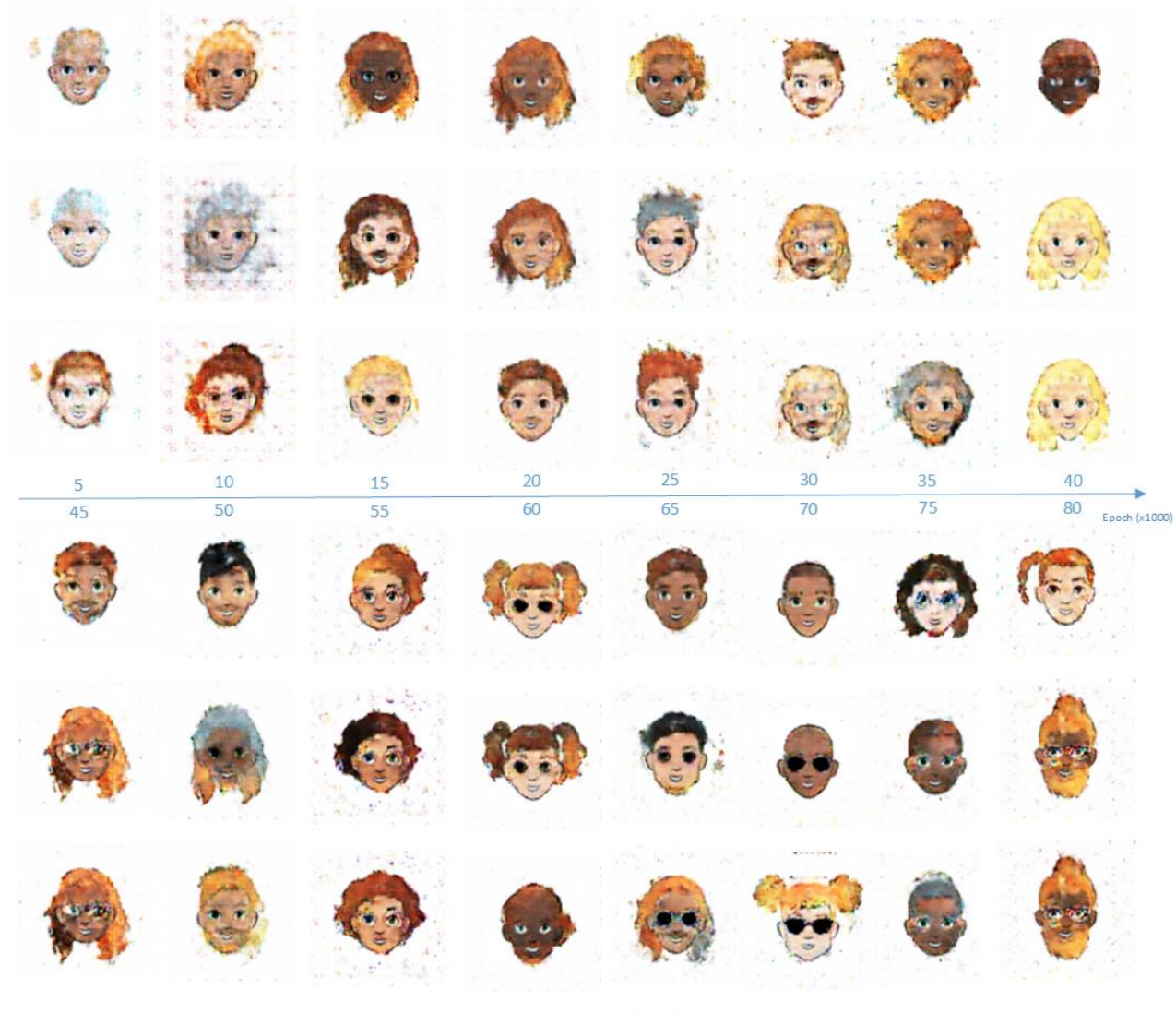

Рисунок 5.7 — Зміна коефіцієнту кроку навчання

Ще було зроблено досить багато різноманітних спроб підбору параметрів, з яких лише декілька було наведено вище. Але, більшість з цих спроб приводили до погіршення результатів генерації. Ті ж, які покращували результат було реалізовано та застосовано у фінальному алгоритмі.

Сумарно кількість спроб для запуску та налаштування моделей зі зміною різноманітних параметрів склала близько 30, кожна з яких займала від години навчання та більше.

5.6 Тестування за допомогою коефіцієнтів IS, FID та KID

В пункті 2.8 даної роботи було описано коефіцієнти для оцінки моделей GAN за їх результатами генерації. До даних коефіцієнтів входять Inception Score, Frechet Inception Distance та Kernel Inception Distance.

Використаємо їх для порівняння розробленої моделі генерації зображень аватарів за категоріями (M1_GAN), генерації зображень аватарів без категорій (M2_CWGAN), що була також розроблена під час виконання даної роботи та звичайної GAN, що описана у статті [45]. Результуючі значення даних коефіцієнтів відображені у таблиці 5.1.

Таблиця 5.1 — Порівняння коефіцієнтів IS, FID, KID моделей GAN.

Model name	**IS**	**FID**	**KID*10^3**
GAN_MNIST	37 ± 0.3	32 ± 1.1	47 ± 0.9
WGAN_Celeb10	18 ± 0.1	205 ± 2.1	220 ± 1.7
M1_CWGAN	23 ± 0.8	180 ± 3.2	213 ± 2.3
M2_GAN	38 ± 1.3	134 ± 2.7	113 + 1.0

Як можна побачити з даної таблиці, розроблена модель має значно кращі коефіцієнти за звичайну модель GAN, запущену на іншому, але досить схожому за типами наповненості набору даних, а отже буде продукувати, як мінімум, не гірші, а зазвичай, навість кращі. При цьому розроблена модель трохи поступається результом WGAN, а отже, в деяких випадках її результати не є ідеальними та очікуваними. При цьому, варто зазначити, що дані коефіцієнти допомагають лише порівнювати продуктивність моделей але мають дуже низький вплив на результати їх виходу.

5.7 Результати створення моделі

В результаті створення та навчання моделі було отримано її фінальну версію. Засновуючись на даній моделі було згенеровано приклади зображень за кожною з категорій, усі з яких були додані до додатку Б.

Сумарно було отримано 24 категорії та по 25 тестових зображень у кожній з них, що складає біля 600 згенерованих аватарів.

Аналізуючи дані зображення, можна побачити, що якість зображень є достатньо високою, хоч і існують деякі аномалії. Серед них можна виділити:

- наявність некоректних кольорів біля очей або носа, що відбувається через неможливість генератору визначити одну з моделей окулярів, що він намагається відобразити;
- некоректно побудовані волосся, особливо при їх великій кількості, що відбувається через достатньо невелику кількість зображень кожного окремого типу волосся окремо;
- частковий градієнт кольорів у волоссі, що створює досить унікальні переходи кольорів у обраній категорії та, в цілому, хоч і є аномалією, але, може вважатися і повноцінною унікальною частиною, що була додана мережею.

В цілому, це є усіма аномаліями, отримані зображення ж мають досить чіткі контури та повністю відповідає за стилем заданій категорії. Отже, таким чином, було отримано модель, що в більшості випадків (приблизно 80%) генерує валідні та якісні зображення, що є близькими до оригінального набору даних, а отже можуть бути використаними у майбутньому на сторінках веб-ресурсів для ідентифікації користувачів.

5.8 Висновки до розділу

В даному розділі було описано методи, що були використані для перевірки та оцінки створеної моделі генерації аватарів.

Отже, спочатку було розглянуто та описано основні проблеми, що можуть виникати під час створення та тестування нейронної мережі, в особливості GAN. До таких проблем входять:

- потрапляння моделі у локальний максимум, з якого вона не може вибратися, через що далі не відбувається процес навчання;
- перенавчання однієї з моделей, через що інша не розуміє в яку сторону та наскільки варто змінювати коефіцієнти;
- модель досягає свого максимального потенціалу та не може видавати ніякі кращі результати без зміни коефіцієнтів.

Після цього також було проведено тестування отриманої моделі, у наступних форматах:

- мануальне тестування, під час якого було перевірено та описано загальні результати генерації за епохами;
- побудова графіків функцій витрат генератору та дискримінатору, через яку було визначено найоптимальнішу епоху, при якій модель досягає свого максимального потенціалу та на основі якої були згенеровані подальші зображення;
- тестування моделі за допомогою алгоритмів, що вираховують коефіцієнти в залежності від різниці згенерованого та реального зображення, а саме IS, FID, KID.

Також, було описано результати генерації моделі та проаналізовано її результат у обраній епосі на наявність аномалій, некоректно згенерованих зображень і т.д.

ВИСНОВКИ

В результаті виконання даної магістерської дисертації було розроблено алгоритм генерації аватарів за категоріями. Для цього було розглянути існуючі алгоритми, засновані на мережі GAN та їх модифікації та обрано декілька, що можуть бути використані для створення моделі за поставленої задачі.

Під час реалізації моделі було обрано набір даних Cartoon Faces (Google's Cartoon Set), що містить близько 100 тисяч екземплярів різноманітних аватарів, намальованих у декількох різних стилях. Його особливістю є наявність розмітки, через яку були обрано та описано основні 24 категорії генерації даних.

Для реалізації було розроблено дві моделі, а саме модель дискримінатора, що приймає два входи – категорію та зображення (реальне або згенероване) та намагається визначити у відсотковому співвідношенні наскільки дане зображення реальне чи згенероване, видаючи на вихід значення у проміжку від 0 до 1. Генератор, у свою чергу приймає на вхід ту ж саму категорію та випадковий гаусівський шум та намагається його перетворити до зображення у форматі RGB з розширенням 96x96.

Особливістю обох моделей, що є відмінністю від стандартних алгоритмів є побудова зображення або його аналіз шляхом змінення розширення, при якому на кожному шарі нейронної мережі воно змінюється удвічі. Це надає змогу починати генерацію з невеликих розширень та поступово змінювати його, покращуючи всій результат з кожним наступним шаром, а отже знижує кількість коефіцієнтів на кожному з них.

Стосовно отриманих результатів, було отримано дві навчені моделі, що пройшли близько 35 тисяч епох навчання та в результаті надають зображення, близькі до реальних. При цьому, недоліком отриманої моделі є те, що вона не може створювати якісні зображення без аномалій у 100% випадків, через що необхідно мануально тестувати та перевіряти кожне окреме зображення на можливість використання на певних ресурсах. Моделі дискримінатору та генератору отримали 5,757,040 та 4,452,737 параметрів відповідно.

З іншого ж боку, модель іноді додає аномалію градієнту волосся, що навпаки, не погіршило результати деяких генерацій, а значно покращило їх та надало змогу отримати набагато більш унікальні зображення, аніж вони були на початку.

Також, були використані коефіцієнти IS, FID та KID для оцінки створеної моделі. Хоч вони, зазвичай і не надають повної картини якості моделі, але отримані показники виявилися значно кращими за звичайні моделі GAN або CGAN без модифікацій, покращивши дані коефіцієнти від 2 до 6 разів.

В цілому, поставлена задача для роботи була виконана, оскільки отримана модель має кращу швидкість навчання та потребує у два рази менше епох для свого навчання аніж звичайна модель GAN, містить категорії для генерації зображень за запитом користувача та показує результати близькі до реальних зображень у більшості випадках.

Результати даної роботи можуть бути використані у двох різних сферах людського життя. По-перше, як і було поставлено у цілі роботи, вона може бути використана для генерації аватарів на веб ресурсах та, таким чином ідентифікації користувачів без додавання власних фото або завантаження певних стандартних зображень. Це може дозволити змінити підхід до створення особистих кабінетів на веб-ресурсах та дозволить ідентифікуватися та показувати свою унікальність користувачам.

По-друге, дана робота може бути продовжена та розширена та далі розвинута для створення більш якісних моделей GAN, що зможуть генерувати не лише мальовані обличчя аватарів, а і більш реальні зображення, що можуть бути використані користувачами.

Подальше розширення даної роботи може включати у себе вибір більш наповненого та складного набору даних, збільшення розширення створених зображень та подальшої модифікації моделі та її коефіцієнтів для отримання більш якісних зображень у більшій кількості випадків.

ПЕРЕЛІК ПОСИЛАНЬ

1. ODSC - Open Data Science. 6 Unique GANs Use Cases [Електронний ресурс] / ODSC - Open Data Science. – 2019. – Режим доступу до ресурсу: https://odsc.medium.com/6-unique-gans-use-cases-24cab2aa924d;
2. 15 Generative Adversarial Networks (GAN) Based Project Ideas [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.projectpro.io/article/generative-adversarial-networks-gan-based-projects-to-work-on/530;
3. Brownlee J. A Gentle Introduction to Generative Adversarial Networks (GANs) [Електронний ресурс] / Jason Brownlee. – 2019. – Режим доступу до ресурсу: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/;
4. Generative adversarial networks / [I. Goodfellow, J. Pouget-Abadie, M. Mirza та ін.]. // Communications of the ACM. – 2020. – №63. – С. 139–144;
5. Radford A. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks / A. Radford, L. Metz, S. Chintala. // arXiv. – 2015. – №1511. 06434v2 – С. 1–16;
6. GAN and its Applications: Everything you need to know [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://datenwissen.com/blog/gan-applications/;
7. Barla N. Generative Adversarial Networks and Some of GAN Applications: Everything You Need to Know [Електронний ресурс] / Nilesh Barla. – 2023. – Режим доступу до ресурсу: https://neptune.ai/blog/generative-adversarial-networks-gan-applications;
8. Image Generation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://paperswithcode.com/task/image-generation;
9. StyleSwin: Transformer-based GAN for High-resolution Image Generation / B.Zhang, S. Gu, B. Zhang, J. Bao. // Conference: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2022;

10. Image Generators with Conditionally-Independent Pixel Synthesis / [I. Anokhin, K. Demochkin, Taras Khakhulin та ін.]. // 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2022;
11. LR-GAN: Layered Recursive Generative Adversarial Networks for Image Generation / J.Yang, A. Kannan, D. Batra, D. Parikh. // ArXiv. – 2016. – №1703.01560 – С. 1–9;
12. Durgadevi M. Generative adversarial network (gan): a general review on different variants of gan and applications / Durgadevi. // 2021 6th International Conference on Communication and Electronics Systems (ICCES). – 2021. – С. 1–8;
13. Photo-realistic single image super-resolution using a generative adversarial network / [C. Ledig, L. Theis, F. Huszár та ін.]. // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2017. – №2017. – С. 4681–4690;
14. Group consistent similarity learning via deep crf for person re-identification / [D. Chen, D. Xu, H. Li та ін.]. // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2018. – С. 8649–8658;
15. Li C. Precomputed real-time texture synthesis with markovian generative adversarial networks / C. Li, M. Wand. // European Conference on Computer Vision. – 2016. – С. 702–716;
16. License Plate Image Generation using Generative Adversarial Networks for End-To-End License Plate Character Recognition from a Small Set of Real Images / B.Han, J. Taek Lee, K. Lim, D. Choi. // Applied Sciences. – 2020. – №10(8):2780;
17. High-content image generation for drug discovery using generative adversarial networks / S.Hussain, A. Anees, A. Das, B. Nguyen. // Neural networks: the official journal of the International Neural Network Society. – 2020. – №132(12). – С. 353–363;
18. Alqahtani H. Applications of generative adversarial networks / H. Alqahtani, M. Kavakli-Thorne, G. Kumar. // Arch Comput Methods. – 2021. – №28(2). – С. 525–552;

19. Enhanced super-resolution generative adversarial networks / [X. Wang, K. Yu, S. Wu та ін.]. // Proceedings of the European Conference on Computer Vision (ECCV) Workshops. – 2018;
20. Dewi C. Wasserstein Generative Adversarial Networks for Realistic Traffic Sign Image Generation / C. Dewi, R. Chen, Y. Liu // Intelligent Information and Database Systems / C. Dewi, R. Chen, Y. Liu.. – С. 479–493;
21. Low-Illumination Image Enhancement in the Space Environment Based on the DC-WGAN Algorithm / M.Zhang, Y. Zhang, Z. Jiang, X. Lv. // Sensors. – 2021. – №21. – С. 286;
22. Minaee S. Iris-GAN: Learning to Generate Realistic Iris Images Using Convolutional GAN / S. Minaee, A. Abdolrashidi. // ArXiv. – 2018. – №1812.04822;
23. Constrained Generative Adversarial Networks for Interactive Image Generation [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://openaccess.thecvf.com/content_CVPR_2019/html/Heim_Constrained_Generative_Adversarial_Networks_for_Interactive_Image_Generation_CVPR_2019_paper.html;
24. Goodfellow I. NIPS 2016 Tutorial: Generative Adversarial Networks / Ian Goodfellow. // arXiv. – 2016. – №1701.00160;
25. Doersch C. Tutorial on Variational Autoencoders / Doersch. // arXiv. – 2021. – №1606.05908;
26. Auerbach D. Do Androids Dream of Electric Bananas? [Електронний ресурс] / D. Auerbach. – 2015. – Режим доступу до ресурсу: https://slate.com/technology/2015/07/google-deepdream-its-dazzling-creepy-and-tells-us-a-lot-about-the-future-of-a-i.html;
27. Gharavi-Alkhansari M. A Generalized Method for Image Coding Using Fractal-Based Techniques / M. Gharavi-Alkhansari, T. S. Huang. // Journal of Visual Communication and Image Representation. – 1997. – №8. – С. 208–225;
28. Deep learning for cellular image analysis / E.Moen, D. Bannon, T. Kudo, W. Graf. // Nature Methods. – 2019. – №16;

29. Brock A. Large Scale GAN Training for High Fidelity Natural Image Synthesis / A. Brock, J. Donahue, K. Simonyan. // arXiv. – 2018. – №1809.11096;
30. ImageNet [Електронний ресурс] – Режим доступу до ресурсу: https://www.image-net.org/;
31. Barratt S. A Note on the Inception Score / S. Barratt, R. Sharma. // arXiv. – 2018. – №1801.01973;
32. Brownlee J. How to Implement the Inception Score (IS) for Evaluating GANs [Електронний ресурс] / J. Brownlee. – 2019. – Режим доступу до ресурсу: https://machinelearningmastery.com/how-to-implement-the-inception-score-from-scratch-for-evaluating-generated-images/;
33. J. Nunn E. Compound Frechet Inception Distance for Quality Assessment of GAN Created Images / E. J. Nunn, P. Khadivi, S. Samavi. // arXiv. – 2021. – №2106.08575;
34. Conditional Frechet Inception Distance / E.Morin, M. Soloveitchik, A. Wiesel, T. Diskin. // arXiv. – 2022. – №2103.11521;
35. Hvilshøj F. Backpropagating through Fréchet Inception Distance / F. Hvilshøj, A. Mathiasen. // arXiv. – 2021. – №2009.14075;
36. Onyshchenko V. Image generations techniques using Generative adversarial networks / V. Onyshchenko, A. Ivanov. // Adaptive systems of automatic control. – 2023. – №1(42);
37. Learning with a Wasserstein Loss / [C. Frogner, C. Zhang, H. Mobahi та ін.]. // arXiv. – 2015. – №1506.05439;
38. UC Irvine Machine Learning Repository [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://archive.ics.uci.edu/ml/index.php;
39. Kaggle datasets [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.kaggle.com/datasets;
40. Snap Inc. Kaggle datasets | Bitmoji faces [Електронний ресурс] / Snap Inc.. – 2021. – Режим доступу до ресурсу: https://www.kaggle.com/datasets/romaingraux/bitmojis;

41. PRASOON KOTTARATHIL. Kaggle datasets | Anime GAN Lite [Електронний ресурс] / PRASOON KOTTARATHIL. – 2021. – Режим доступу до ресурсу: https://www.kaggle.com/datasets/prasoonkottarathil/gananime-lite;
42. BARTLEY. Kaggle datasets | Cartoon Faces (Google's Cartoon Set) [Електронний ресурс] / BARTLEY. – 2022. – Режим доступу до ресурсу: https://www.kaggle.com/datasets/brendanartley/cartoon-faces-googles-cartoon-set;
43. R. Parker K. Criteria for the selection of a programming language for introductory courses / K. R. Parker, T. A. Ottaway, J. T. Chao. // International Journal of Knowledge and Learning. – 2006. – №2. – С. 119–139;
44. Koo K. Image recognition performance enhancements using image normalization / K. Koo, E. Cha. // Human-centric Computing and Information Sciences. – 2017. – №7(1);
45. Intrinsic Multi-scale Evaluation of Generative Models / [A. Tsitsulin, M. Munkhoeva, D. Mottin та ін.]. // arXiv. – 2019. – №1031.