Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Математичні моделі та методи штучного інтелекту для креативного дизайну у галузі моди
(тема)

Виконав:
студент _2_ курсу, групи _САУм-22-1
Титечко П.В.
(прізвище, ініціали)
Спеціальність
124 Системний аналіз
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Системний аналіз і управління
(повна назва освітньої програми)
Керівник доц. Єсілевський В.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Сидоров М.В.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _124 Системний аналіз
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Системний аналіз і управління
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
"_06_" _листопада_ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _Титечку Павлу Вячеславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи _Математичні моделі та методи штучного інтелекту для креативного дизайну у галузі моди.

затверджена наказом по університету від 2 листопада 2023 р. № 1277 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 січня 2024 р.

3. Вихідні дані до роботи _результати отримані під час проходження професійної практики; тренувальний набір даних; DCGAN модель для генерації зображень.

4. Перелік питань, що потрібно опрацювати в роботі
1. Системний аналіз предметної області
2. Вибір і обґрунтування методу розв'язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Системний аналіз предметної області
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	6 – 12 листопада 2023 р.	виконано
2	Вибір та обґрунтування методу	13 – 26 листопада 2023 р.	виконано
3	Розробка алгоритму і програми	27 листопада – 10 грудня 2023 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	11 грудня – 24 грудня 2023 р.	виконано
5	Робота над текстом пояснювальної записки	25 грудня 2023 р. – 9 січня 2024 р.	виконано
6	Представлення роботи на рецензію в ЕК	10 січня 2024 р.	виконано

Дата видачі завдання _6_ листопада 2023 р.

Студент _____ _____
(підпис)

Керівник роботи _____ доц. Єсілевський В.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 56 с., 1 табл., 18 рис., 2 дод., 15 джерел.

ГЕНЕРАТИВНО-ЗМАГАЛЬНА НЕЙРОННА МЕРЕЖА, МОДА, ОДЯГ, ПАТЕРН, DCGAN, STYLEGAN.

Об’єкт дослідження – процес утворення нових зображень на основі використання штучного інтелекту.

Метою роботи є застосування методів машинного навчання для створення креативного дизайну у галузі моди. Завданням є створення креативних зразків моди, які можна використовувати в різних застосуваннях, таких як проектування нових малюнків або матеріалів для тканин. Починаючи з існуючих патернів, нам потрібно вивчити, як генерувати нові, які виглядають подібно на ті, що є в початковому (навчальному) наборі даних, але з деякими варіаціями, які роблять ці нові малюнки унікальними. Окрім унікальності, ми шукаємо можливість створювати креативний контент, той, який є естетично привабливим і обмежує безмежні можливості в креативному просторі.

Методи дослідження базуються на основі DCGAN та StyleGAN нейронних мереж.

Наукова новизна отриманих результатів полягає у тому, що досліджується збіжність тренувального процесу за умов використання обмеженого, не нормалізованого вхідного набору даних, а також досліджується вплив латентних векторів на процес генерації зображень.

Практична значимість отриманих результатів полягає в тому, що запропонований підхід може бути використаний як акселератор у створені нового дизайну через інтеграцію запропонованого підходу у дизайнерський виробничий процес.

ABSTRACT

Introductory note: 56 pages, 1 table, 18 figures, 2 appendixes, 15 sources.

CLOTH, DCGAN, FASHION, GENERATIVE ADVERSARIAL NETWORK, PATTERN, STYLEGAN.

The object of research is the process of creating new images based on the use of artificial intelligence.

The purpose of the work is to use machine learning methods to produce creative design in the field of fashion. The challenge is to develop creative fashion designs that can be used in various applications, such as designing new patterns or materials for fabrics. Starting with existing patterns, we need to learn how to generate new ones that look like those in the original (training) dataset, but with some variations that make these new patterns unique. In addition to uniqueness, we are looking for an opportunity to spawn creative content, one that is aesthetically pleasing and limits the endless possibilities in the creative space.

Research methods are based on DCGAN and StyleGAN neural networks.

The scientific novelty of the obtained results lies in the fact that the convergence of the training process is investigated under the conditions of using a limited, non-normalized input data set, and the influence of latent vectors on the image generation process is also investigated.

The practical significance of the obtained results is that the proposed approach can be used as an accelerator in the creation of a new design through the integration of the proposed approach into the design production process.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

AdaIN – адаптивна нормалізація екземпляра;

DCGAN – глибока згорткова генеративно-змагальна нейронна мережа;

GAN – генеративно-змагальна нейронна мережа;

LeakyReLU – ректифікований лінійний оператор (пом’якшений);

StyleGAN – генеративно-змагальна нейронна мережа з контролем стилю;

VAE – варіаційний автокодувальник.

ВСТУП

Актуальність роботи зумовлена використанням генеративно-змагальних нейронних мереж (GANs). Це клас моделей глибокого навчання, які набули актуальності протягом останніх 10 років завдяки їхній здатності створювати реалістичні синтетичні дані. Загалом GANs мають широкий спектр застосувань у різних сферах, сприяючи створенню даних, медіа та контенту, які не лише реалістичні, але і корисні для вирішення реальних завдань та підвищення творчості. Їх актуальність продовжує зростати, оскільки досліджуються нові варіації та застосування GANs в різних галузях.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є дослідження можливості застосування методів машинного навчання для створення креативного дизайну у галузі моди. Завданням є створення креативних зразків, які можна використовувати в різних застосуваннях, таких як проектування нових малюнків або матеріалів для тканин. Починаючи з існуючих патернів, нам потрібно вивчити, як генерувати нові, які виглядають подібно на ті, що є в початковому (навчальному) наборі даних, але з деякими варіаціями, які роблять ці нові зображення унікальними. Окрім унікальності, досліджена можливість створювати креативний контент, той, який є естетично привабливим і обмежує безмежні можливості в креативному просторі. Для досягнення поставленої мети необхідно виконати наступні завдання:

– проаналізувати сучасний стан задачі генерації зображень за допомого нейронних мереж;

– вибрати архітектуру для вирішення поставленої задачі;

– вибрати альтернативну архітектуру;

– провести пошук та підготовку даних для тренування;

– запрограмувати архітектури нейронних мереж;

– вибрати та підготувати середовище для навчання нейронних мереж;

– провести тренування моделі на основі підготовленого набору даних;

– оптимізувати моделі;

– порівняти результати генерації обох моделей;

– зробити висновки.

Об’єктом дослідження є процес утворення нових зображень на основі використання штучного інтелекту.

Предметом дослідження є математичні моделі та методи штучного інтелекту для креативного дизайну у галузі моди.

Методи дослідження. У кваліфікаційній роботі використовуються методи дослідження комп’ютерної моделі нейронної мережі на основі DCGAN та StyleGAN. Розглянуті підходи для обчислення функції активації, функції втрат та оптимізаторів.

Публікації. Результати, отримані у роботі, було представлено на XXVII Міжнародному молодіжному форумі «Радіоелектроніка та молодь у XXI столітті» (м. Харків, 10-12 травня 2023 р.) [1].

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Системний аналіз задачі створення креативного дизайну за допомогою штучного інтелекту

Ціллю даною роботи є дослідження побудови креативного дизайну у сфері моди за допомогою штучного інтелекту на основі наявного контенту. Першим кроком був збір і підготовка даних, що включало зображення, концепції дизайну, колірні палітри та інші ресурси, які мали подаватися на вхід для системи штучного інтелекту. Самі дані вимагали додаткової обробки (очистки) для забезпечення їх придатності для аналізу штучним інтелектом. Далі розглядалися підходи штучного інтелекту: тобто методи, техніки і інструменти штучного інтелекту для вирішення завдання. Наступним кроком стала розробка та навчання моделі штучного інтелекту з використанням підготовлених даних. Це також включало в себе проектування та налаштування архітектури моделі, встановлення гіперпараметрів та сам процес навчання моделі на даних. Розглядалися декілька альтернативних підходів до побудови моделей. Вихідні дані оцінювалися враховуючи метрики, такі як якість зображень та креативність. Перші результати були переглянуті дизайнерами для того, щоб переконатися, що створені зображення відповідають креативним цілям. Було проаналізовано можливу інтеграцію в процес дизайну: тобто як систему штучного інтелекту можна інтегрувати в процес креативного дизайну (взаємодію між людиною та штучним інтелектом в процесі дизайну). Було розглянуто можливість оновлення моделі через навчання на нових даних для адаптації до змінних модних тенденцій. Слід зауважити, що не було досліджено етичні аспекти пов'язані з використанням штучного інтелекту в креативному дизайні, включаючи питання авторства та прав інтелектуальної власності.

1.2 Аналіз сценаріїв вирішення задачі створення креативного дизайну за допомогою штучного інтелекту

Генеративний штучний інтелект використовує моделі штучного інтелекту для створення зображень. Ці моделі мають здатність генерувати нові зображення або змінювати існуючі.

Генеративні змагальні мережі (GANs) складаються з двох нейронних мереж, генератора і дискримінатора, які навчаються одночасно. Генератор намагається створити зображення, яке неможливо розрізнити від реальних зображень, в той час як дискримінатор намагається розрізнити реальні та створені зображення. Цей змагальний процес приводить до створення високоякісних, реалістичних зображень. GANs використовуються в різних завданнях генерації зображень, включаючи мистецтво, фотографію та технологію діпфейк (deepfake).

Варіаційні автоенкодери (VAEs) це генеративні моделі, які акцентують на вивченні латентних представлень зображень. Їх можна використовувати для генерації нових зображень шляхом вибору з вивченого латентного простору. VAEs часто використовуються в застосунках, де потрібно створювати нові, схожі зображення.

Глибокі згорткові GANs (DCGANs) це тип GANs, який використовує згорткові нейронні мережі (CNNs) для генерації зображень.

StyleGAN і StyleGAN2 призначені для керування стилем і виглядом створених зображень. StyleGAN дозволяє маніпулювати різними аспектами зображень, такими як вік, стать та мистецький стиль, зберігаючи високу якість результатів. StyleGAN2 є вдосконаленою версією оригінальної архітектури StyleGAN. Вона включає ряд покращень та оптимізацій для підвищення якості генерації зображень та стабільності процесу навчання.

StyleGAN-T є моделлю генерації зображень з тексту на основі GAN архітектури.

GANs для підвищення роздільної здатності (Super-Resolution GANs) були спеціально розроблені для генерації високо роздільних зображень з низько

роздільного вхідного зображення. Вони мають застосування для підвищення якості зображень, покращення фотографій та відео.

CycleGANs – це тип GANs, який може виконувати трансформацію зображень-в-зображення, що дозволяє трансформувати зображення з одного домену в інший. Наприклад, він може перетворювати фотографії в картини або змінювати денні сцени на нічні.

1.3 Формальна та змістовна постановка задачі

Компанія, яка працює у сфері моди, має необхідність у покращені процесу створення нових патернів тканин на основі існуючого набору зображень який був розроблений нею до тепер, а також з врахуванням новітніх зразків які постійно появляються на ринку. Тобто, потрібно на основі даного набору згенерувати зображення які наслідують його характеристики, а також є унікальними зображеннями які мають естетичний вигляд.

Тепер побудуємо формальну постановку даної задачі. На вхід нам дано певний набір зображень $data$ які семантично пов’язані і належать простору X.

Нехай Z це шумовий простір. Розглянемо вектор z з даного простору тобто $z \in Z$, значення якого вибираються з розподілу шуму $p_z(z)$ і який підпорядковується нормальному розподілу:

$$p_z(z) \sim N(0,1).$$

Позначимо через Y простір який формує генеративну модель на базі реальних даних (зображень).

Отже, нашу задачу можна визначити як знаходження функції перетворення з вектору шумових даних та простору Y у простір X:

$$(Z \times Y) \rightarrow X.$$

Іншими словами, ми шукаємо функцію перетворення вхідного шуму та простору наближення Y у простір справжніх зображень X.

1.4 Постановка задач дослідження

Завданням кваліфікаційної роботи є генерація нових зразків моди, а саме нових малюнків чи патернів для тканин. Для вирішення поставленої задачі необхідно виконати наступні завдання:

– проаналізувати сучасний стан задачі генерації зображень за допомого нейронних мереж;

– вибрати архітектуру для вирішення поставленої задачі;

– вибрати альтернативну архітектуру;

– провести пошук та підготовку даних для тренування;

– запрограмувати архітектури нейронних мереж;

– вибрати та підготувати середовище для навчання нейронних мереж;

– провести тренування моделі на основі підготовленого набору даних;

– оптимізувати моделі;

– порівняти результати генерації обох моделей;

– зробити висновки.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ’ЯЗАННЯ

2.1 Генеративні змагальні мережі (GAN)

Генеративний штучний інтелект (Generative AI), це галузь штучного інтелекту, яка спеціалізується на розробці систем, що здатні створювати нові дані, зображення, текст або інші об'єкти, що виглядають подібними до реальних. Ці системи використовують різні алгоритми та моделі, такі як глибокі нейронні мережі, генеративні змагальні мережі (GANs), рекурентні нейронні мережі (RNNs) і багато інших, для створення нового контенту. Розглянемо деякі ключові галузі, де застосовується GANs.

Генерація зображень. GANs широко використовуються для створення високоякісних та реалістичних зображень. Зокрема використовуються у мистецтві, дизайні, для створення реалістичних зображень, перенесення стилів та перетворення зображень.

Генерація даних. GANs можна використовувати для розширення наборів даних, особливо в ситуаціях, коли збір більшої кількості даних є складним. Шляхом генерації синтетичних даних, які схожі на реальні дані, GANs можуть покращувати продуктивність моделей машинного навчання.

Виявлення аномалій. Навчаючи GANs на правдивих даних та ідентифікація аномалії як дані, які GANs не можуть відтворити. Застосовується в кібербезпеці.

Синтез зображень за текстом. GANs можуть генерувати зображення на основі текстових описів, що має застосування в створенні графічного контенту.

Генерація голосу. GANs можуть створювати реалістичний голос та аудіо, що важливо в застосуваннях, таких як синтез мовлення з тексту та створення музики.

Безпека даних. GANs використовуються в методах збереження приватної інформації, таких як створення синтетичних даних, які зберігають статистичні властивості вихідних даних, але не розкривають приватну інформацію.

Мода та дизайн. GANs використовуються в індустрії моди для створення нових дизайнів одягу та передбачення модних тенденцій.

Автономні транспортні засоби. GANs можна використовувати в симуляторах для створення реалістичних сценаріїв водіння для навчання автономних транспортних засобів.

GAN був вперше запропонований в 2014 році Яном Гудфеллоу [2]. Архітектура GAN зображена на рис. 2.1.

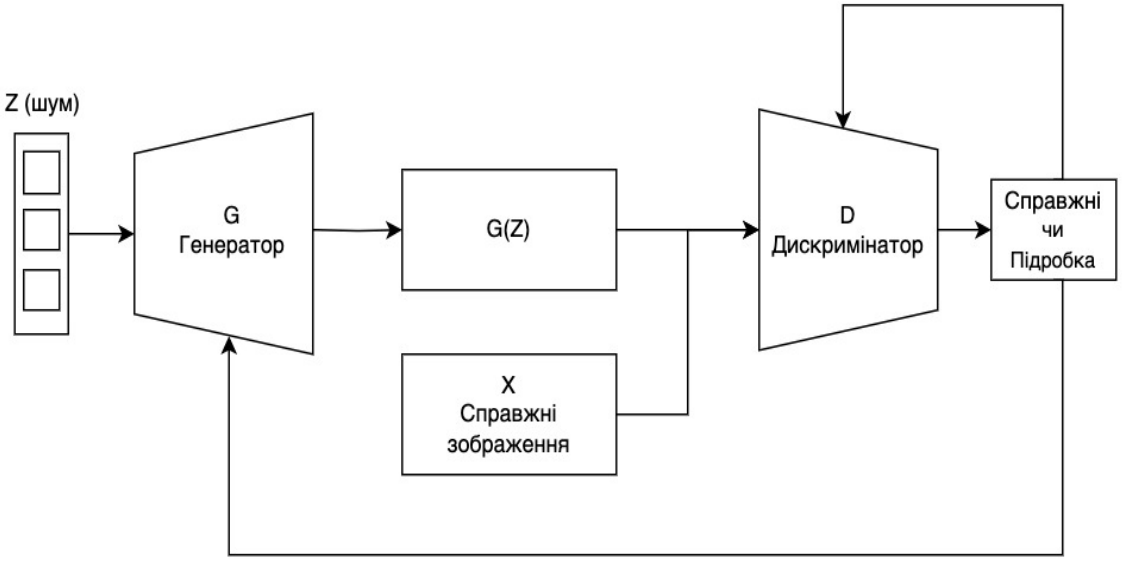

Рисунок 2.1 – Архітектура GAN

GAN складається з двох основних компонентів: генератора та дискримінатора. Генератор відповідає за створення нових зображень, даних чи інших об'єктів. Він приймає на вхід випадковий шум і намагається створити дані, які б найкраще імітували реальні дані, з якими він був навчений. Дискримінатор відповідає за оцінку, наскільки створені генератором дані схожі на реальні дані. Він приймає на вхід як справжні дані, так і дані, згенеровані генератором, і намагається відрізнити одні від інших. Принцип роботи GAN полягає в тому, що генератор і дискримінатор конкурують один з одним. Генератор намагається створити дані, які дуже схожі на реальні, тоді як дискримінатор намагається виявити різницю між реальними і згенерованими

даними. Цей процес триває до того моменту, коли генератор може створити дані, які майже не відрізняються від реальних.

2.2 Обґрунтування використання генеративно-змагальних мереж

Очевидно, що задача описана у розділі 1.3 не є тривіальна. Одним з ефективних методів її розв'язання є генеративні мережі. Застосуємо GAN для вирішення поставленої задачі.

Розглянемо зображення x з реального набору. Введемо функцію D яку назвемо дискримінатором, результат опрацювання дискримінатора даного зображення x є ймовірність того, що дане зображення походить з вхідних (справжніх) даних, а не згенероване функцією G яку назвем генератором. Результат $D(x)$ повинен бути високим, коли зображення є з тренувальних даних, і низьким, коли воно походить від генератора. Будемо розглядати дискримінатор як традиційний бінарний класифікатор.

Розглянемо вектор z з латентного простору, який вибирається зі стандартного нормального розподілу. Функція генератора трансформує латентний вектор у певний простір даних $data$.

Проводячи аналогію з поставленою задачею у розділі 1.3 нам потрібно знайти розподіл, з якого походять реальні дані, щоб змогти генерувати фальшиві зразки з даного розподілу.

Таким чином, результат опрацювання дискримінатора згенерованого зображення є ймовірність, що вихід генератора є реальним зображенням. Використаємо за основу цільову функцію яку запропонував Гудфеллоу [2], Дана функція визначається як сума сподівань від логарифмічних функцій (2.1).

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

де $\mathbb{E}_x$ та $\mathbb{E}_z$ – очікувана ймовірність як для реальних так і згенерованих даних.

Згідно даної функції дискримінатор та генератор “грають” у мінімаксну гру, в якій дискримінатор намагається максимізувати ймовірність правильної класифікації реальних і фейкових зображень, а генератор намагається мінімізувати ймовірність того, що дискримінатор визначить його результати як фейки (несправжні). Цільова функція побудована таким чином, що дискримінатор намагається максимізувати вираз представлений формулою (2.2).

$$\log(D(x)) + \log(1 - D(G(z))). \tag{2.2}$$

У свою чергу генератор намагається мінімізувати вираз представлений формулою (2.3).

$$\log(1 - D(G(z))). \tag{2.3}$$

Проте, оскільки існують практичні проблеми з значеннями градієнта, введемо інший вираз (2.4), який генератор намагається максимізувати. Тобто результат, який видає генератор, має сприйматися дискримінатором як справжні зображення.

$$\log(D(G(z))). \tag{2.4}$$

У даній мінімаксній грі буде рішення, коли дискримінатор починає випадковим чином визначати(вгадувати), чи вхідні зображення є реальними або ж фейковими. Тобто, коли генератор визначає розподіл з якого походять вхідні зображення, а саме $p_{data} = p_g$.

2.3 Модифікація DCGAN

DCGAN – це безпосереднє розширення описаного вище GAN. Вперше воно було описано у статті Редфорда [3].

DCGAN (Deep Convolutional Generative Adversarial Network) – це конкретний тип GAN з певними архітектурними модифікаціями, які добре підходять для генерації зображень. Розглянемо деякі ключові відмінності між DCGAN і GAN:

а) архітектура;

1) DCGAN використовує глибоку архітектуру зі згортковими нейронними мережами як для генератора, так і дискримінатора, в генераторі використовуються леєри для поступового підняття рівня деталізації зображення з вхідного шуму;

2) GAN не визначає певної архітектури нейронної мережі, його можна реалізувати з різними архітектурними варіаціями, включаючи повністю з'єднані шари(леєри) або інші типи леєрів;

б) згорткові шари (леєри) [3] показані на рис. 2.2;

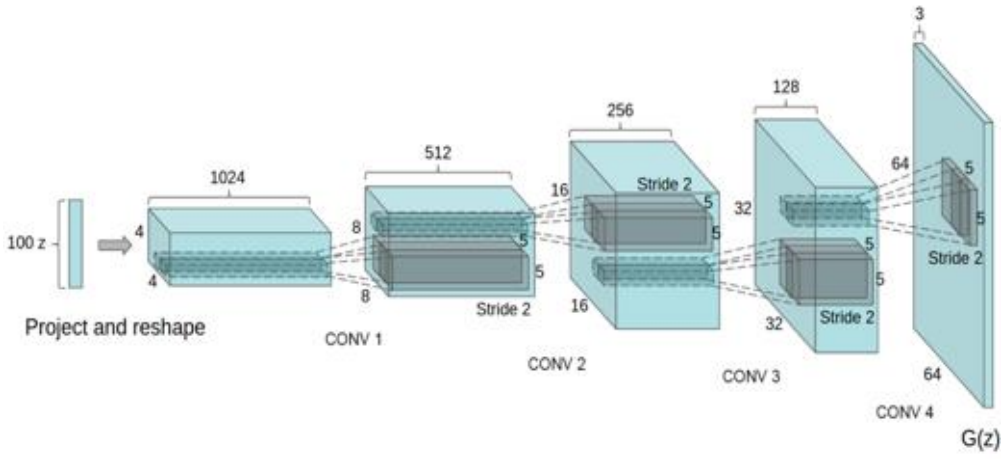

Рисунок 2.2 – Архітектура генератора

1) DCGAN головним чином використовує згорткові і транспоновані згорткові шари в своїй архітектурі, ці шари добре підходять для обробки зображень;

2) GAN може використовувати комбінацію різних типів шарів (леєрів), включаючи повністю з'єднані шари. Вибір шарів залежить від конкретного завдання і архітектури;

в) випадковий шум;

1) DCGAN приймає випадковий шум як вхід для генератора. Цей випадковий шум зазвичай відбирається з простого розподілу (гауссовий чи рівномірний розподіл);

2) GAN також може приймати випадковий шум на вході, але він більш гнучкий і може приймати інші типи вхідних даних, такі як текст чи інші формати даних, залежно від застосування;

г) нормалізація груп;

1) для DCGAN часто використовується нормалізація наборів даних для стабілізації і прискорення навчання;

2) для GAN, використання нормалізації наборів даних не є специфічним для GAN і використовується, залежно від конкретного варіанту архітектури;

ґ) функція втрати;

1) DCGAN зазвичай використовує ту саму функцію втрати змагальної мережі, іншими словами, це мінімаксна гра між генератором і дискримінатором;

2) GAN охоплює різноманітні функції втрат, включаючи метод Васерштейна (WGANs), найменших квадратів (LSGANs), які відрізняються від оригінальної змагальної втрати, яка застосовується в DCGAN;

д) сфери застосування;

1) DCGAN особливо підходить для завдань генерації зображень, створення нових зображень [4];

2) GAN є більш широким класом моделей, які використовуються для різних завдань генеративного моделювання, включаючи генерацію тексту, перенос стилю, аугментацію даних та інше, і не обмежуються лише зображеннями.

Отже, DCGAN є спеціалізованою варіацією GAN, призначеною для генерації високоякісних зображень [5], і вона включає в себе глибокі згорткові шари. В той час як GAN може бути застосованим до широкого спектру завдань у галузі генеративного моделювання, DCGAN розроблено спеціально для завдань генерації зображень та вже стало фундаментальною моделлю у галузі комп'ютерного зору та генеративного моделювання.

2.4 Модифікація StyleGAN

Оскільки результати отримані під час застосування DCGAN на існуючому наборі даних були недостатньо якісні, було вирішено спробувати альтернативні варіанти з GAN простору. На мою думку однією з причин, чому результати генерації є незадовільні, з DCGAN, є замалий набір вхідних зображень (~5000), що і спричиняє проблему перенавчання (overfitting). Чимало вдосконалень архітектури GAN було досягнуто завдяки вдосконаленню моделі дискримінатора. Ці зміни мотивовані ідеєю, що краща модель дискримінатора, у свою чергу, призведе до створення більш реалістичних синтетичних зображень. Таким чином, генератор був дещо занедбаний і залишався чорним ящиком. Наприклад, джерело випадковості, що використовується при створенні синтетичних зображень, недостатньо зрозуміле, включаючи як кількість випадковості в точках вибірки, так і структуру прихованого простору.

Це обмежене розуміння генератора добре ілюструється відсутністю контролю над згенерованими зображеннями. Існує кілька інструментів для керування властивостями згенерованих зображень, наприклад стиль. Це включає функції високого рівня, такі як фон і передній план, і дрібні деталі, такі як характеристики синтезованих об’єктів або суб’єктів. Для цього потрібно роз’єднати властивості в зображеннях, і додати елементи керування для цих властивостей до моделі генератора.

Style Generative Adversarial Network є розширенням архітектури GAN, яке пропонує значні зміни в моделі генератора, включаючи використання мережі відображення для відображення точок у латентному просторі в проміжний латентний простір, використання проміжного прихованого простору для керування стилем у кожній точці моделі генератора та введення шуму як джерела варіацій у кожній точці моделі генератора [6].

Style Generative Adversarial Network, або скорочено StyleGAN, є розширенням архітектури GAN, щоб надати контроль над властивостями стилів створених зображень.

StyleGAN є розширенням прогресивного (progressive) GAN, який є підходом до навчання моделей генераторів, здатних синтезувати високоякісні зображення за допомогою поступового розширення моделей дискримінатора та генератора від малих до великих зображень під час процесу навчання. З поступовим зростанням моделей під час навчання, StyleGAN суттєво змінює архітектуру класичного генератора.

Генератор StyleGAN не приймає точки з латентного простору як вхідні дані; натомість є два нових джерела випадковості, які використовуються для створення синтетичного зображення: автономна мережа набору ознак (features) та шари шуму.

Вихідні дані мережі відображення – це вектор, який визначає стилі, інтегрований у кожну точку моделі генератора через новий рівень, який називається адаптивною нормалізацією екземпляра. Використання даного вектору стилю дає змогу керувати стилем створеного зображення. Стохастична зміна вводиться через шум, доданий у кожній точці моделі генератора. Шум додається до цілого набору функцій, що дозволяє моделі інтерпретувати стиль у дрібнодисперсній манері для кожного пікселя. Таке поблочне включення вектору стилю та шуму дозволяє кожному блоці локалізувати як інтерпретацію стилю, так і стохастичну варіацію до заданого рівня деталізації.

StyleGAN описується як прогресивна архітектура GAN, що розвивається, із п’ятьма модифікаціями, кожна з яких була додана та оцінена поступово під час дослідження. Розглянемо зміни які були зроблені на рівні генератора [7]:

– базова архітектура прогресивного (progressive) GAN;

– білінійне підвищення дискретизації;

– додано мережу відображення та AdaIN (стилі);

– видалене приховане векторне введення шуму в генератор;

– додавання шуму до кожного блоку;

– додавання змішаної регуляризації.

На рисунку 2.2 представлено архітектуру генератора у StyleGAN.

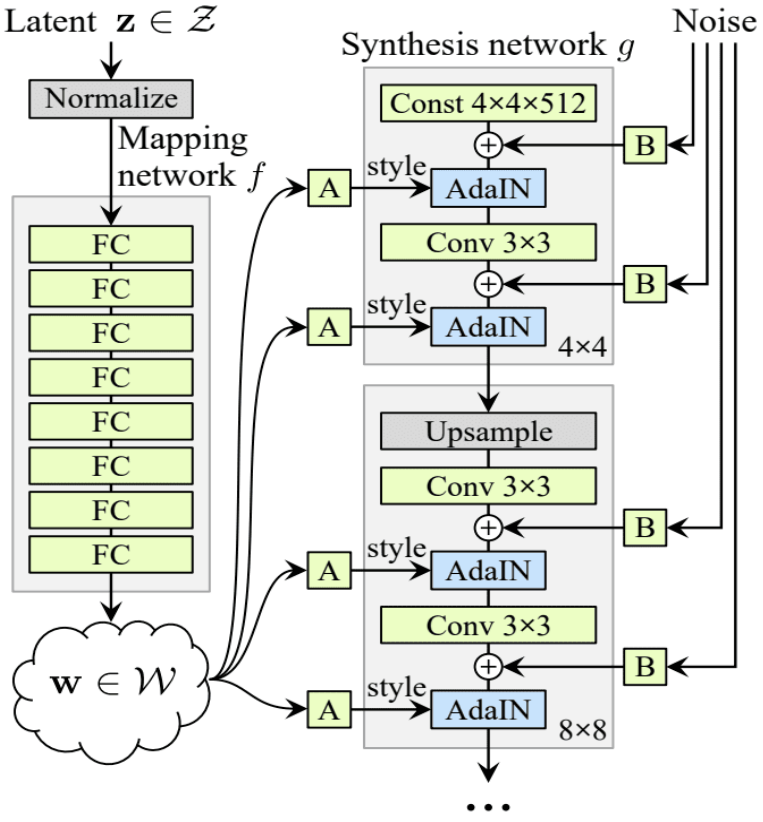

Рисунок 2.2 – Архітектура генератора у StyleGAN

Моделі генератора та дискримінатора StyleGAN навчаються за допомогою методу навчання прогресивного зростання GAN. Це означає, що обидві моделі починаються з невеликих зображень, у даному випадку зображення 4×4. Моделі підганяються, поки вони не стануть стабільними,

потім і дискримінатор, і генератор розширюються до подвоєння ширини та висоти (8×8). До кожної моделі додається новий блок для підтримки більшого розміру зображення, яке повільно зникає під час навчання. Після затухання моделі знову тренуються, поки вони не стануть достатньо стабільними, і процес повторюється з дедалі більшими розмірами зображення, доки не буде досягнуто бажаного цільового розміру зображення. Поступово зростаючий GAN використовує леєри найближчих сусідів для підвищення роздільності замість транспонованих згорткових шарів, які є звичайними в інших моделях генераторів.

Перша відмінність в StyleGAN полягає в тому, що використовуються білінійні леєри для підвищення роздільності (замість найближчих сусідів). Ми замінюємо підвищення/зменшення частоти дискретизації найближчого сусіда в обох мережах білінійною дискретизацією, яку ми реалізуємо шляхом низькочастотної фільтрації активацій за допомогою роздільного біноміального фільтра 2-го порядку після кожного рівня підвищення дискретизації та перед кожним шаром зниження дискретизації. Далі використовується автономна мережа відображення, яка приймає випадкову вибірку точки з латентного простору як вхідні дані та генерує вектор стилю. Мережа ознак складається з восьми повністю пов’язаних рівнів, тобто це стандартна глибока нейронна мережа. Потім вектор стилю перетворюється та включається в кожен блок моделі генератора після згорткових шарів за допомогою операції, яка називається адаптивною нормалізацією екземпляра або AdaIN. Шари AdaIN передбачають спочатку стандартизацію виводу функцій ознак до стандартного розподілу Гауса, а потім додавання вектору стилю як зсув. Додавання нової мережі відображення до архітектури також призводить до перейменування моделі генератора на «мережу синтезу».

Наступна зміна передбачає модифікацію моделі генератора таким чином, щоб вона більше не приймала вектор з латентного простору як вхідні дані. Натомість модель має постійне вхідне значення 4x4x512, щоб почати процес синтезу зображення.

Вихід кожного згорткового шару в мережі синтезу є блоком активації. Шум Гауса додається до кожної з цих блоків активації перед операціями AdaIN. Для кожного блоку генерується окремий зразок шуму, який інтерпретується за допомогою коефіцієнтів масштабування для кожного рівня. Цей шум використовується для введення варіацій на рівні стилю на заданому рівні деталізації.

Регулярізація змішування включає спочатку генерацію двох векторів стилю з мережі відображення (ознак). Вибирається точка розділення в мережі синтезу, і всі операції AdaIN до точки розділення використовують перший вектор стилю, а всі операції AdaIN після точки розділення отримують другий вектор стилю. Це заохочує шари та блоки локалізувати стиль для певних частин моделі та відповідного рівня деталізації створеного зображення.

Навчання генеративно змагальних мереж (GAN) з використанням надто малої кількості даних зазвичай призводить до відхилення дискримінатора, спричиняючи розбіжності у навчанні. Існує адаптивний механізм розширення (adaptive discriminator augmentation) дискримінатора, який значно стабілізує навчання в режимах обмежених даних [8]. Цей підхід не потребує змін у функціях втрати чи архітектурі мереж і застосовний як під час навчання з початку, так і під час налаштування існуючого GAN на іншому наборі даних. Хороші результати тепер можливі, використовуючи лише кілька тисяч навчальних зображень.

Висновки за розділом 2

У даному розділі було розглянуті дві моделі (DCGAN, StyleGAN) машинного навчання які належать до генеративних мереж.

Основні відмінності між цими двома нейронними мережами полягають у наступному:

– StyleGAN використовує базову архітектуру зі згортковими шарами як у генераторі, так і у дискримінаторі;

– StyleGAN має більш складну архітектуру з використанням генераторів на основі стилів;

– DCGAN процес навчання може іноді призводити до згортання режиму (collapse mode), що обмежує різноманітність генерованих зображень;

– DCGAN процес навчання захоплює глобальні структури, а не деталізацію;

– StyleGAN забезпечує кращий контроль над конкретними особливостями згенерованих зображень;

– під час навчання DCGAN можуть виникати проблеми, і результати можуть варіюватися залежно від набору даних та гіперпараметрів;

– StyleGAN має техніки для стабілізації процесу навчання, роблячи його більш надійним і простим у навчанні.

DCGAN є простішим та базовим, StyleGAN вводить інновації в архітектурі, що призводить до поліпшеної стабільності, кращого контролю над особливостями та високоякісної генерації зображень. Незважаючи на те, що підхід (DCGAN) є простіший ніж StyleGAN за часом імплементації та кількістю ресурсів які вимагаються для навчання, він має одну суттєву проблему пов’язану з перенавчанням (overfitting) [9] при малому вхідному наборі зображень. Дана проблема є критична у нашому випадку, оскільки вхідний набір складається з 5000 зображень. Далі ми практично перевіримо результати генерації зображень з використанням обох підходів базуючись на тих самих вхідних даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Побудова DCGAN моделі за допомогою бібліотеки машинного навчання PyTorch

Нагадаємо, що метою роботи є генерація патернів зображень тканин на основі початкового набору зображень [10] з використанням DCGAN підходу. Програмна реалізація виконана на основі PyTorch фреймворку для машинного навчання. PyTorch – це набір інструментів машинного навчання, побудований на бібліотеці Torch. PyTorch є частиною Linux Foundation, хоч був розроблений компанією Meta AI.

Програмна частина базується на розробках описаних у статті [11].

Дискримінатор будуємо зі шарів згорткового перетворення з використанням крокових згорток, шарів нормалізації та активацій LeakyReLU [3]. На вхід подаємо зображення розміром 3x64x64 (колір/висота/ширина), а на виході – ймовірність того, що вхідні дані належать розподілу реальних даних. Як було зазначено, дискримінатор – це мережа бінарної класифікації, яка приймає зображення на вході і видає скалярну ймовірність того, що вхідне зображення є реальним тобто не фальшивим. Тут дискримінатор приймає вхідне зображення розміром 3x64x64, обробляє його через послідовність шарів Conv2d, BatchNorm2d і LeakyReLU, і видає кінцеву ймовірність за допомогою функції активації Sigmoid. Цю архітектуру можна розширити додатковими шарами, якщо це необхідно для задачі, але важливо використовувати стрічкову згорткову операцію, BatchNorm і LeakyReLUs. У статті про DCGAN [3] зазначено, що це є хорошою практикою використовувати стрічкову згорткову операцію для зменшення розміру зображення, оскільки це дає можливість мережі вивчити власну функцію пулінгу (pooling). Крім того, функції пакетної нормалізації та LeakyReLU сприяють ефективному набору градієнтів, що є критичним для процесу навчання як генератора, так і дискримінатора. Клас дискримінатора наведений на рис. 3.1.

```
class FashionDiscriminator(nn.Module):
    def __init__(self, gpus_number):
        super(FashionDiscriminator, self).__init__()
        self. gpus_number = gpus_number
        self.main = nn.Sequential(
            nn.Conv2d(latent_vector_size, feature_map_size,
                      4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(feature_map_size, feature_map_size * 2,
                      4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_size * 2),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(feature_map_size * 2, feature_map_size * 4,
                      4, 2, 1, bias=False),
                      nn.BatchNorm2d(feature_map_size * 4),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(feature_map_size * 4, feature_map_size * 8,
                      4, 2, 1, bias=False),
                      nn.BatchNorm2d(feature_map_size * 8),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(feature_map_size * 8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )

    def forward(self, input):
        return self.main(input)
```

Рисунок 3.1 – Клас дискримінатора

Крім того, функції пакетної нормалізації та LeakyReLU сприяють ефективному набору градієнтів, що є критичним для процесу навчання як генератора, так і дискримінатора.

Генератор складається зі шарів транспонованого згорткового перетворення, шарів нормалізації та активацій ReLU. На вхід надходить латентний вектор, який вибирається зі стандартного нормального розподілу, а на виході отримується зображення розміром 3x64x64. Власне шари згорткового перетворення дозволяють перетворити латентний вектор в структуру з тим самим розміром, що і зображення. Оскільки наші дані це зображення, перетворення латентного вектору означає в кінцевому підсумку створення RGB

зображення з таким же розміром, як навчальні зображення (тобто 3x64x64). На практиці це досягається за допомогою послідовності двовимірних шарів зворотніх згорткових операцій із стрічками, кожен із яких супроводжується шаром 2D пакетної нормалізації та активацією ReLU. Клас генератора наведений на рис. 3.2:

```
class FashionGenerator(nn.Module):
    def __init__(self, gpus_number):
        super(FashionGenerator, self).__init__()
        self. gpus_number = gpus_number
        self.main = nn.Sequential(
            nn.ConvTranspose2d(latent_vector_size,
                               feature_map_size * 8, 4, 1, 0,
                               bias=False),
            nn.BatchNorm2d(feature_map_size * 8), nn.ReLU(True),
            nn.ConvTranspose2d(feature_map_size * 8,
                               feature_map_size * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_size * 4),
            nn.ReLU(True),
            nn.ConvTranspose2d( feature_map_size * 4,
                               feature_map_size * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_size * 2),
            nn.ReLU(True),
            nn.ConvTranspose2d(feature_map_size * 2,
                               feature_map_size, 4, 2, 1, bias=False),
            nn.BatchNorm2d(feature_map_size),
            nn.ReLU(True),
            nn.ConvTranspose2d(feature_map_size, channel_number,
                               4, 2, 1, bias=False), nn.Tanh() )


    def forward(self, input):
        return self.main(input)
```

Рисунок 3.2 – Клас генератора

Вихід генератора проходить через функцію tanh, щоб повернутися до діапазону вхідних даних [-1, 1]. Важливо відзначити наявність функцій

пакетної нормалізації після шарів зворотніх згорткових операцій, оскільки це є важливою особливістю DCGAN. Ці шари допомагають зі спрямуванням градієнтів під час навчання. Будемо використовувати наступні вхідні дані:

– кількість зображень в одній тренувальній групі (batch): 128;

– розмірність зображень: 3x64x64 (колір/висота/ширина);

– розмір латентного вектору який подається на вхід генератора: 100;

– розмір мапи ознак(feature map) для генератора: 64;

– розмір мапи ознак(feature map) для дискримінатора: 64;

– кількість тренувальних епох: 200;

– швидкість навчання для оптимізаторів: 0.0002;

– у якості тренувального набору даних використовується дані з [10].

Всі ваги моделі початково ініціалізовані випадковими значеннями з нормального розподілу зі середнім значенням 0 та стандартним відхиленням 0,02. Зі встановленими початковими параметрами та підготовленим набором даних, розглянемо реалізацію. З налаштованими дискримінатором та генератором ми можемо вказати, як вони навчаються за допомогою функцій втрат і оптимізаторів. Ми будемо використовувати функцію бінарної перехресної ентропії (BCELoss). Дана функція забезпечує обчислення обох логарифмічних компонентів у функції цілі. Далі ми визначаємо нашу реальну мітку як 1, а фальшиву мітку як 0. Ці мітки використовуватимуться при обчисленні втрат для дискримінатора та генератора. Нарешті, ми налаштовуємо два окремі оптимізатори, один для дискримінатора та один для генератора. Використовуються оптимізатори Adam з швидкістю навчання lr = 0,0002 та beta1 = 0,5. Код ініціалізації показаний на рис. 3.3.

```
optimizerD.Adam(iDiscriminator.parameters(),
                lr=0.0002, betas=(0.5, 0.999))
optimizerG.Adam(iGenerator.parameters(),
                lr=0.0002, betas=(0.5, 0.999))
```

Рисунок 3.3 – Параметризація оптимізаторів

Для відстеження прогресу навчання генератора ми будемо створювати фіксований набір латентних векторів, які взято з гауссового розподілу (шум). Під час навчання ми будемо періодично подавати цей шум в генератор, і протягом ітерацій ми побачимо, як із шуму виникають зображення.

Отже, коли ми маємо всі частини структури GAN визначені, ми можемо почати процес навчання. Важливо пам’ятати, що навчання GAN є до певної міри мистецтвом, оскільки неправильні налаштування гіперпараметрів можуть призвести до згортання тренування (mode collapse), без пояснень того, що пішло не так. Будемо дотримуватися алгоритму зі статті Гудфеллоу [2], а також відомих практик [12][13]. Зокрема, формуємо різні набори для реальних і фальшивих зображень, і також налаштовуватимемо цільову функцію генератора так, щоб максимізувати позитивне рішення дискримінатора щодо згенерованих зображень. Навчання розділяється на дві основні частини: оновлення дискримінатора, та оновлення генератора. Схема ітераційного процесу навчання показана на рис. 3.4.

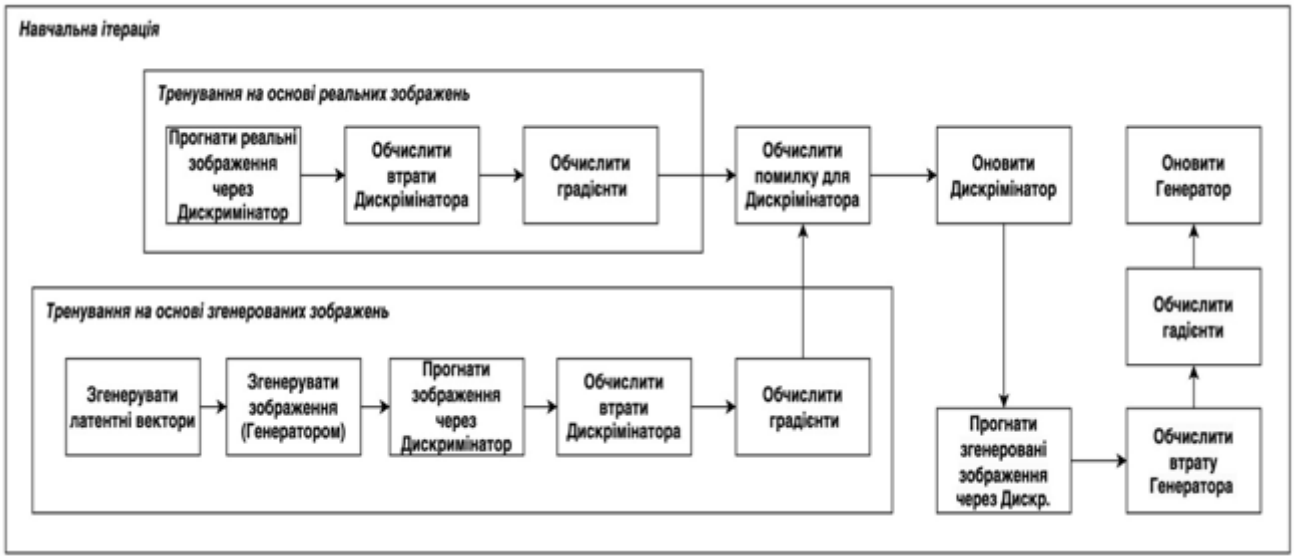

Рисунок 3.4 – Процес навчання

Метою навчання дискримінатора є максимізація ймовірності правильної класифікації заданого входу як реального чи фальшивого. Згідно Гудфеллоу, ми намагаємося оновити дискримінатор, збільшуючи його стохастичний градієнт.

Використовуючи наявні практики проводимо розрахунки у два етапи. По-перше, ми формуємо набір реальних вибірок з навчального набору, пропускаємо через дискримінатор, розраховуємо втрати логарифмічної функції від дискримінатора, а потім розрахуємо градієнти під час зворотного проходу. По-друге, ми формуємо набір фальшивих зразків за допомогою генератора, пропускаємо даний набір через дискримінатор, розрахуємо втрати, і накопичуємо градієнти під час зворотного проходу. Тепер, з градієнтами, накопиченими з обох наборів (з усіма реальними та усіма фальшивими), ми викликаємо крок оптимізатора дискримінатора.

Наступний крок навчити генератор створювати кращі підробки. Як зазначено [2], Гудфеллоу показав, що це не забезпечує достатні градієнти, особливо на початку процесу навчання. Як виправлення, ми хочемо максимізувати логарифмічну функцію від дискримінатора на генерованому наборі даних. Ми досягаємо цього, поділяючи вивід генератора з дискримінатором, розраховуючи втрати генератора, використовуючи реальні мітки, розраховуючи градієнти генератора під час зворотного проходу і, нарешті, оновлюючи параметри генератора за допомогою кроку оптимізатора. В кінці кожної епохи ми пропускаємо фіксований набір через генератор, щоб візуально відстежувати прогрес навчання генератора. Під час навчання на кожній ітерації ми розраховуємо наступні дані:

– втрата дискримінатора, розрахована як сума втрат для усіх реальних і усіх фальшивих наборів зображень;

– втрата генератора (ітераційна), розрахована як середнє значення виходу дискримінатора для усіх реальних зразків. Воно повинно бути близьким до 1, а теоретично збігатися до 0.5, коли генератор покращується;

– середні виходи дискримінатора для усіх фальшивих зразків. Перше число – до оновлення дискримінатора, а друге число – після оновлення дискримінатора. Ці числа повинні починатися близько до 0 і збігатися до 0.5, коли генератор покращується.

Лістинг програми знаходиться у додатку А.

3.2 Побудова StyleGAN моделі за допомогою бібліотеки машинного навчання TensorFlow

TensorFlow – це відкрита платформа для машинного навчання, яка охоплює велику кількість аспектів цього процесу. TensorFlow використовується для управління всіма етапами роботи з системами машинного навчання. TensorFlow API організовані у ієрархічній структурі, з високорівневими API, які у свою чергу базуються на API низького рівня. API низького рівня використовується для створення та дослідження нових алгоритмів машинного навчання. На рис. 3.5 показана ієрархія основних абстракцій та інструментів TensorFlow.

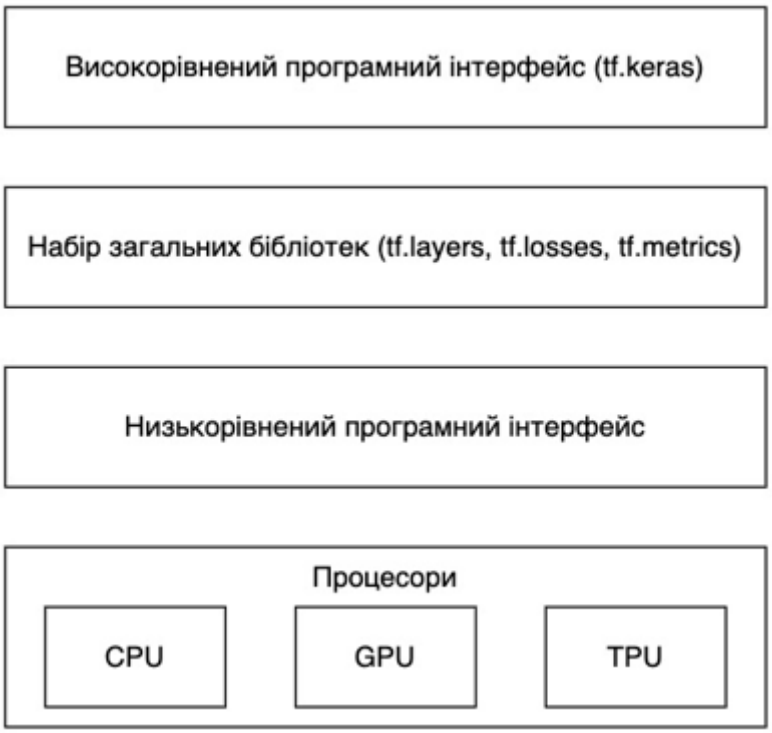

Рисунок 3.5 – Ієрархічна структура TensorFlow

TensorFlow використовується в різноманітних застосунках, починаючи від обробки природної мови (NLP) та розпізнавання зображень і закінчуючи прогнозуванням аналітичних даних та управлінням автономними транспортними засобами. Він може бути використаний для навчання глибоких

нейронних мереж для виявлення та класифікації об'єктів, генерації рекомендацій, класифікації зображень та створення додатків з голосовим управлінням. Крім того, TensorFlow використовується для прогнозування, застосунків на основі тексту, алгоритмічної торгівлі та оптимізації. Його також використовують в різних областях охорони здоров'я, таких як медична діагностика та пошук ліків. Усі ці випадки використання демонструють гнучкість та потужність, які TensorFlow надає для даних вчених та розробників. З його гнучкістю та масштабованістю TensorFlow дозволяє реалізовувати складні завдання машинного навчання в обраний термін порівняно з традиційними методами. У основі TensorFlow знаходиться граф потоку даних, який описує, як дані рухаються через послідовні операції або перетворення. Основна ідея за графом потоку даних полягає в тому, що операції виражаються як вузли, при цьому кожен вузол виконує одну операцію на своїх входах. Входи та виходи операцій передаються через ребра (тензори). Це дозволяє розбити складні обчислення на менші, більш керовані частини. TensorFlow також надає низку інструментів для створення та навчання нейронних мереж. Одним з найпопулярніших інструментів є утиліта tf.keras, яка дозволяє користувачам швидко створювати та навчати моделі глибокого навчання, та не писати код з нуля. Вона також включає потужні інструменти візуалізації, щоб користувачі могли краще розуміти дані та параметри моделі. TensorFlow також є розширюваним і може використовуватися з багатьма мовами програмування, включаючи Python, C++, JavaScript та Go. Він також підтримує використання на графічних процесорах (GPU) для досягнення максимальної продуктивності. Зі своїм надійним набором інструментів TensorFlow є однією з найпопулярніших платформ для глибокого навчання і продовжує розвиватися з появою більш потужних алгоритмів.

Для побудови тренувального процесу використовується підхід описаний у репозиторії [14] на основі вхідних даних отриманих з [10]. Лістинг програми знаходиться у додатку Б.

Висновки за розділом 3

У даному розділі було розглянуто два підходи до побудови нейронних мереж за допомогою конкуруючих платформ машинного навчання PyTorch та TensorFlow. На основі PyTorch фреймворку було побудовано процес навчання DCGAN моделі; а TensorFlow фреймворк використовувався для побудови StyleGAN мережі.

Обидві бібліотеки мають свої переваги та недоліки. Вибір між TensorFlow та PyTorch залежить від особистих вподобань, проектних вимог та від конкретного випадку використання. Обидві бібліотеки продовжують активно розвиватися.

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

4.1 Зображення згенеровані на основі DCGAN мережі

Розглянемо результати виконання тренування. Початковий тренувальний набір даних наведено на рис. 4.1.

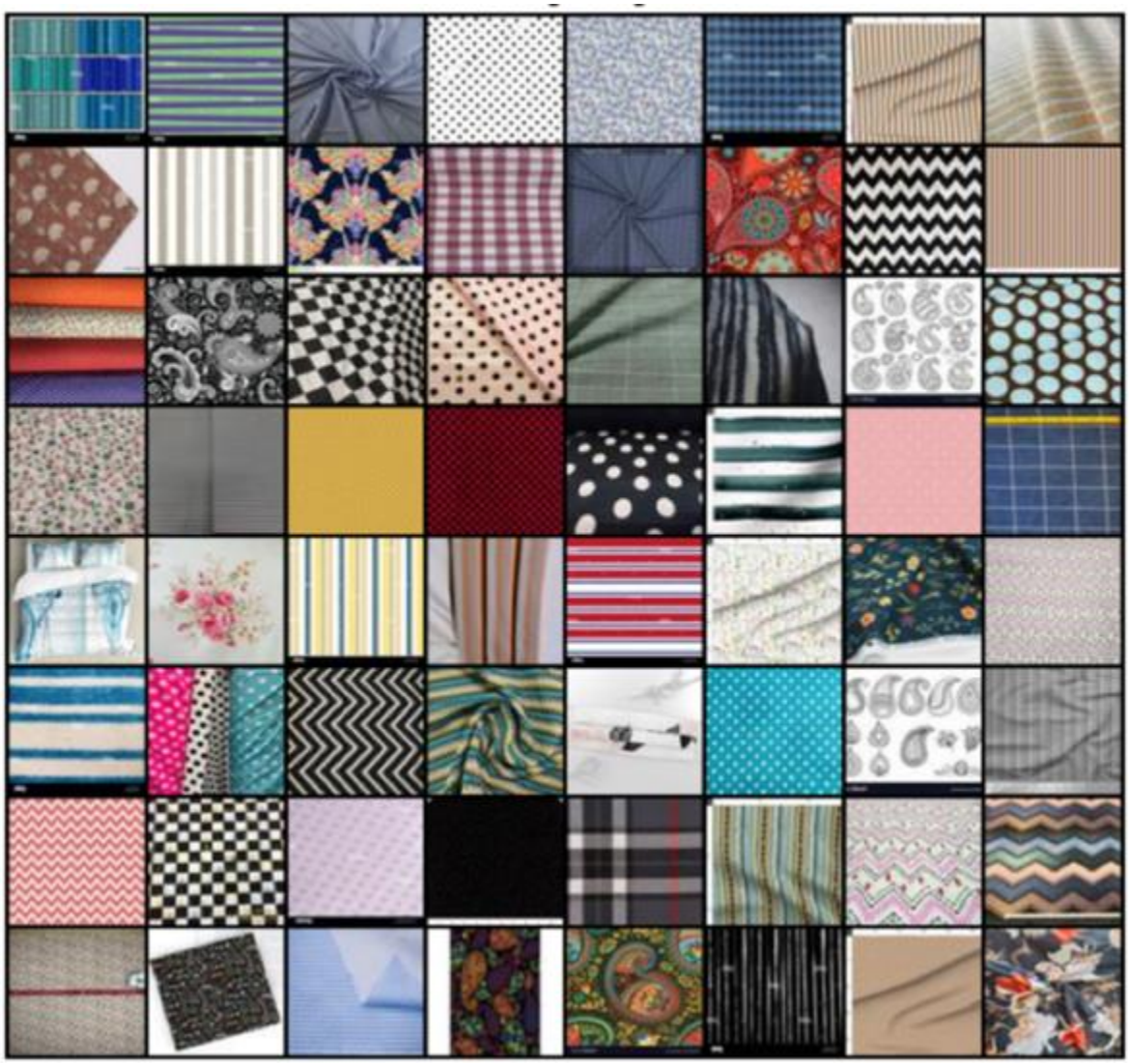

Рисунок 4.1 – Приклад вхідних зображень

На рис. 4.2 наведено графік втрат дискримінатора та генератора відносно ітерацій навчання. Таблиця 4.1 містить основні статистичні дані зібрані під час процесу навчання.

Рисунок 4.2 – Порівняння втрат

Таблиця 1.1 – Статистичні дані процесу навчання

Епоха	Дискримінатор (втрата)	Генератор (втрата)	Дискримінатор (згенеровані дані)
1	3.9251	18.1796	0.1773
2	2.2143	4.8897	0.0274
3	2.0594	10.6951	0.0155
4	3.3205	7.4855	0.0324
5	2.0188	5.1863	0.0103
6	2.9351	5.5848	0.0005
7	2.1948	10.0149	0.0034
8	2.0057	6.0204	0.0137
9	0.9245	5.4971	0.0020
10	1.0016	10.1162	0.0012

Проміжні результати роботи генератора для оригінального (необробленого) набору даних представленні на рис. 4.3.

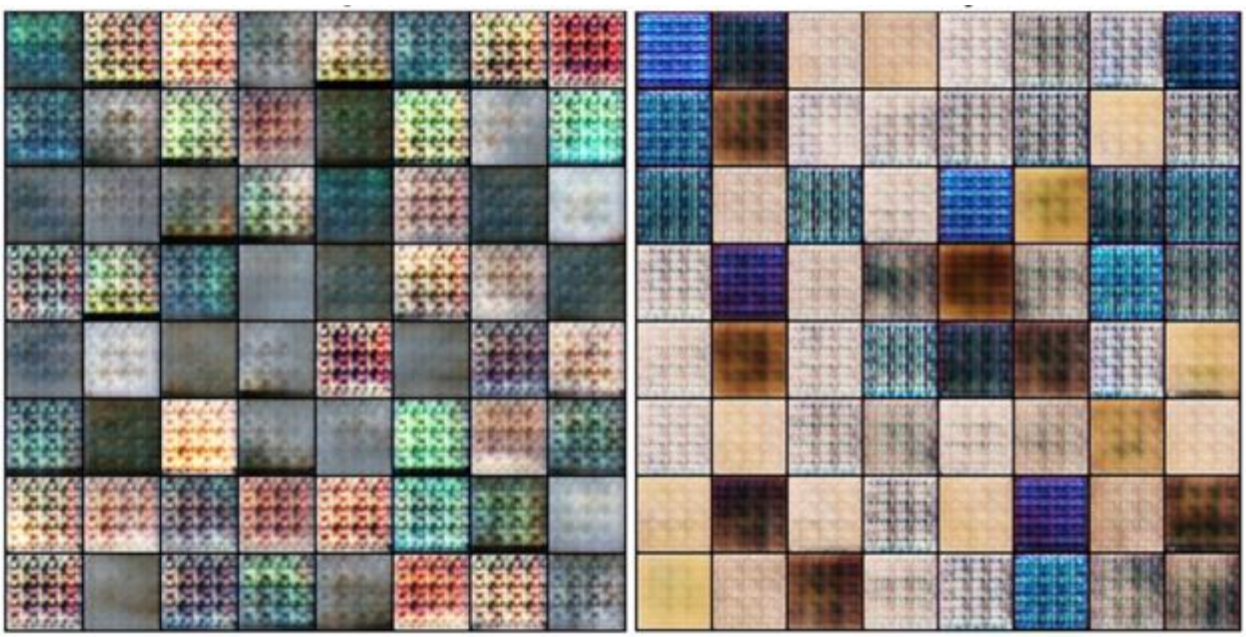

Рисунок 4.3 – Результат генерації на базі необроблених вхідних даних

Після фільтрації вхідних даних, я отримав результати які відображені на рис. 4.4.

Рисунок 4.4 – Результат генерації на базі оброблених вхідних даних

Реальні дані поруч із згенерованими після закінчення тренувального процесу зображені на рис. 4.5.

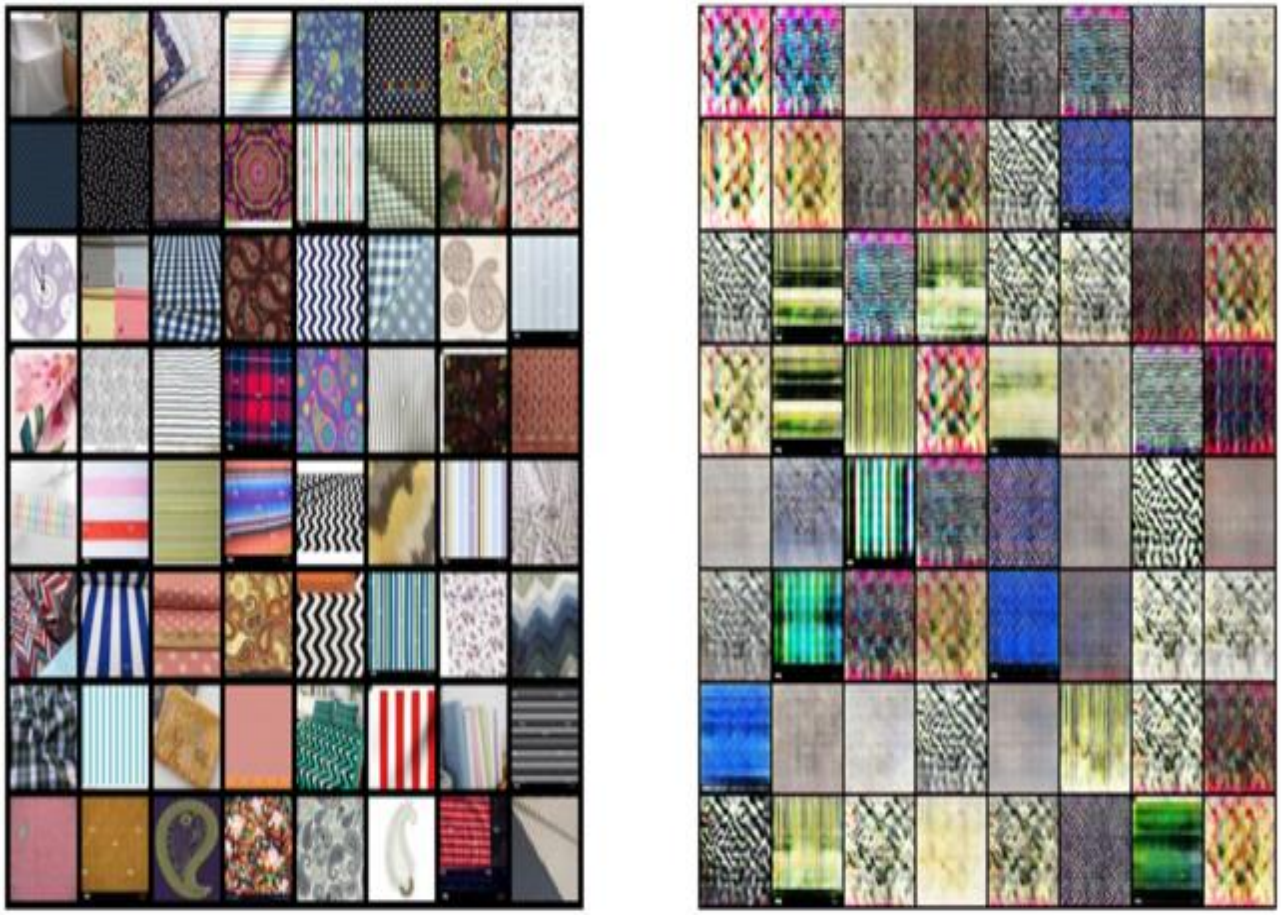

Рисунок 4.5 – Реальні та згенеровані дані

Тренування були проведені для різних комбінацій гіперпараметрів моделей, хоча найкращі результати були отримані для набору який описано вище, а також для фільтрованого набору даних. Максимально було проведено 500 тренувальних епох.

4.2 Згенеровані зображення на основі StyleGAN моделі

Розглянемо результати виконання тренування. Початковий тренувальний набір даних наведено на рис. 4.6.

Рисунок 4.6 – Реальні зображення

Проміжні результати роботи генератора для оригінального набору даних представленні на рис. 4.7 та 4.8.

Рисунок 4.7 – Результат генератора (епоха 3)

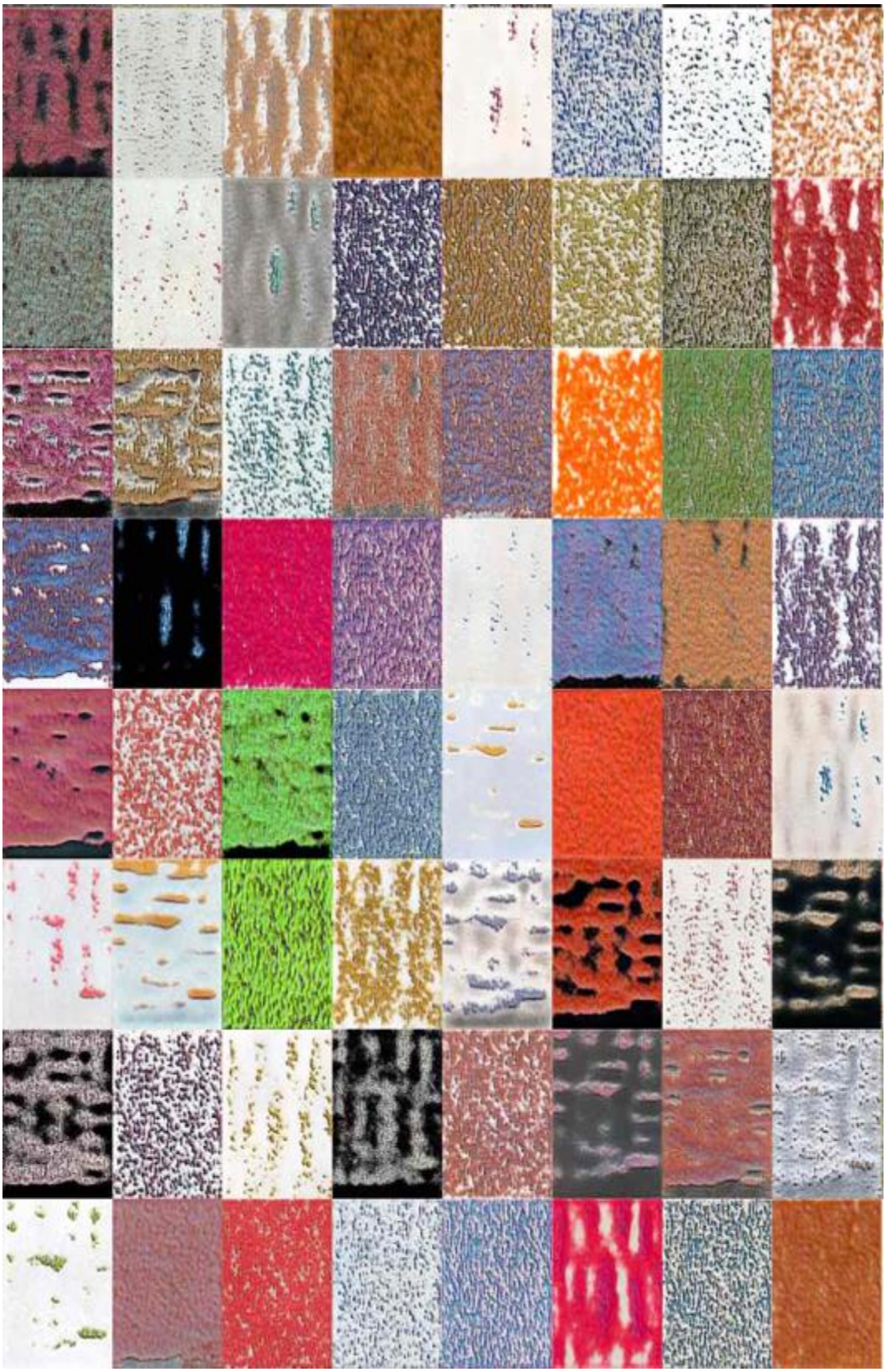

Рисунок 4.8 – Результат генератора (епоха 6)

Результат роботи генератора на кінцевих етапах навчання зображено на рисунках 4.9 та 4.10.

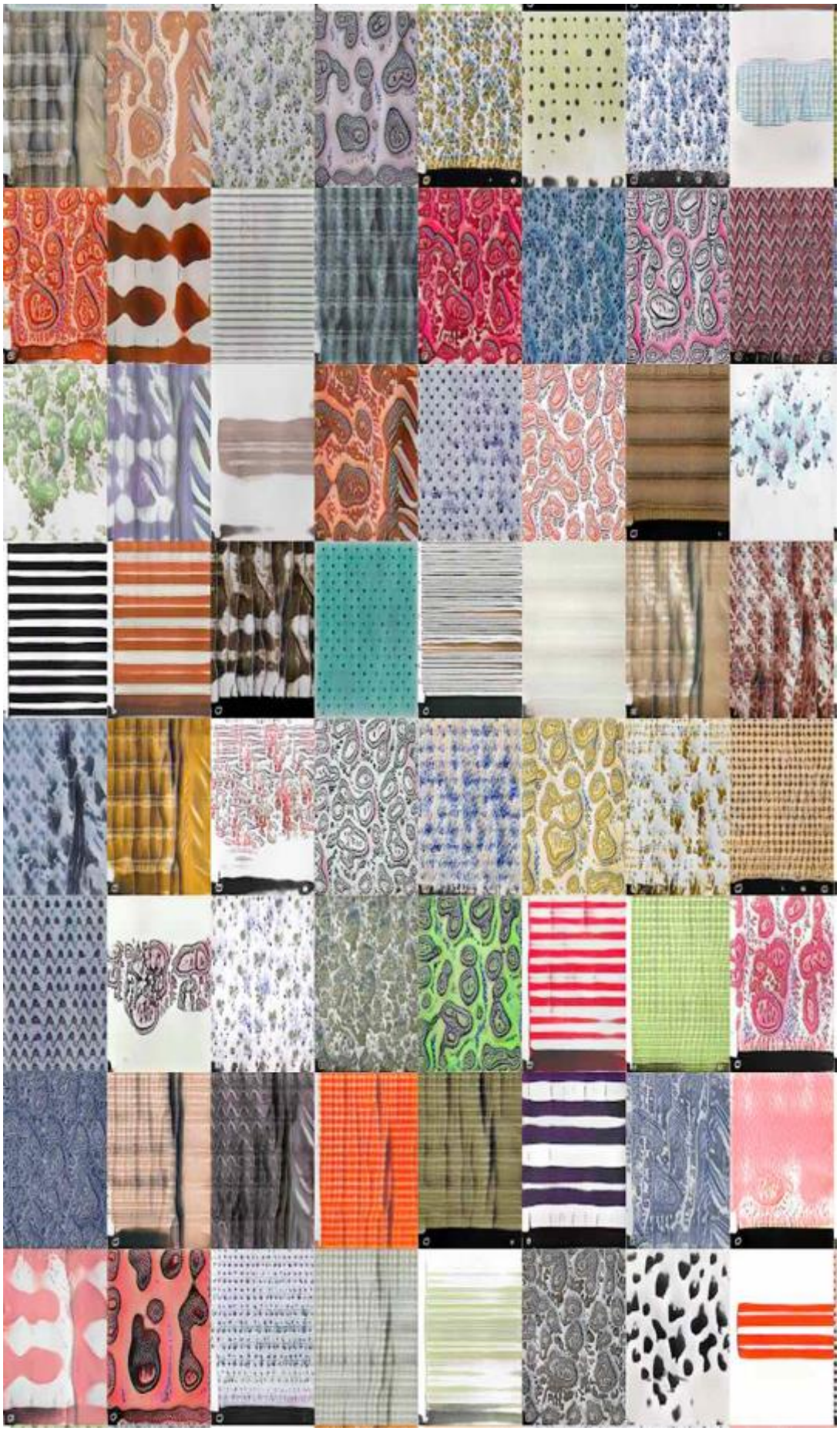

Рисунок 4.9 – Результат генератора (епоха 13)

Рисунок 4.10 – Результат генератора (епоха 15)

Висновки за розділом 4

Отже, з отриманих результатів можна стверджувати, що дієвим способом генерації нових зображень на основі обмеженого набору зображень патернів тканин є спосіб який ґрунтується на StyleGAN мережі яка є стійкою до проблеми перенавчання (overfitting) на обмеженому вхідному наборі даних. З великою ймовірністю можна стверджувати, що підхід DCGAN також може генерувати якісні нові зображення проте потребує значно більшого вхідного набору даних (від 50 000) для навчання.

ВИСНОВКИ

У даній роботі були побудовані моделі які дозволяють генерувати нові зображення у галузі дизайну на основі існуючих даних.

Наукова новизна отриманих результатів полягає у тому, що було досліджено збіжність тренувального процесу за умов використання різних вхідних даних, функцій втрат, а також досліджено вплив латентних векторів на процес генерації зображень. Було опрацьовано два підходи (DCGAN, StyleGAN) які належать до генеративно-змагальних нейронних мереж.

Практична значимість отриманих результатів полягає в тому, що запропонований підхід може бути використаний як акселератор у створені нового дизайну через інтеграцію у виробничий дизайнерський процес.

Запропонований підхід має певну обмеженість яка пов’язана з якістю та кількістю вхідних даних. Тобто, початкові дані та їх кількість мають вагомий вплив на результат тренування та генерацію зображень.

У звіті були перелічені можливості для застосування GAN мереж. Даний підхід продовжує розвиватися, що підтверджується появою різних GAN варіацій, архітектур та підходів. Вони використовуються в різних сферах для вирішення завдань і створення нових можливостей у генерації та обробці даних. Що стосується даної роботи то перспективами подальших досліджень є формування більш якісних зображень, а також пошук оптимальних шляхів збіжності тренувального процесу для DCGAN та подальша інтеграція з StyleGAN3. Крім цього, перспективним подальшим напрямком буде наступна інтеграція з StyleGAN-T (генерація зображення на основі тексту) [15], що наддасть додаткові можливості інженерам у створені нового графічного контенту у галузі моди.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Титечко П. В. Побудова креативних зразків у галузі моди за допомогою GAN мереж. *27-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті»* : зб. матеріалів форуму. Т. 7. Харків : ХНУРЕ, 2023. С. 205–206. URL: https://nure.ua/konferencii-ta-workshops/mizhnarodnij-molodizhnij-forum-radioelektronika-i-molod-u-hhi-stolitti/xxvii-mizhnarodnyj-molodizhnyj-forum-radioelektronika-i-molod-u-khkhi-stolitti (дата звернення: 24.11.2023).

2. Generative adversarial nets / Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. 2014. URL: https://arxiv.org/abs/1406.2661 (дата звернення 25.11.2023).

3. Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. 2015. URL: https://arxiv.org/abs/1511.06434 (дата звернення 24.11.2023).

4. CAN: Creative Adversarial Networks Generating “Art” by Learning About Styles and Deviating from Style Norms / Elgammal A., Liu B., Elhoseiny M., Mazzone M. 2017. URL: https://doi.org/10.48550/arXiv.1706.07068 (дата звернення 24.10.2023).

5. Application of an Improved DCGAN for Image Generation / Liu B., Lv J., Fan X., Luo J. 2022. URL: https://www.researchgate.net/publication/362092988_Application_of_an_Improved_DCGAN_for_Image_Generation (дата звернення 24.11.2023).

6. Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks. 2019. URL: https://arxiv.org/abs/1812.04948 (дата звернення 27.12.2023).

7. BrownLee J. A Gentle Introduction to StyleGAN. 2020. URL: https://machinelearningmastery.com/introduction-to-style-generative-adversarial-network-stylegan/ (дата звернення 27.12.2023).

8. Karras T., Aittala M., Hellsten J. Training Generative Adversarial Networks with Limited Data. 2020. URL: https://arxiv.org/abs/2006.06676 (дата звернення 27.12.2023).

9. Yazici Y., Foo C., Winkler S. Empirical Analysis of Overfitting and Mode Drop in GAN Training. 2020. URL: https://arxiv.org/abs/2006.14265 (дата звернення 27.12.2023).

10. Mehta R. Cloth pattern. 2022. URL: https://www.kaggle.com/datasets /rahilmehtaucoe2784/cloth-pattern (дата звернення 26.10.2023).

11. Inkawhich N. DCGAN Tutorial. URL: https://pytorch.org/tutorials/ beginner/dcgan_faces_tutorial.html (дата звернення 24.11.2023).

12. Kingma D., BA J. Adam: A Method for Stochastic Optimization. 2017. URL: https://arxiv.org/abs/1412.6980 (дата звернення 24.10.2023).

13. Brownlee J. How to Implement GAN Hacks in Keras to Train Stable Models. 2019. URL: https://machinelearningmastery.com/how-to-code-generative-adversarial-network-hacks/ (дата звернення 24.11.2023).

14. Karras T., Aittala M., Hellsten J. Stylegan2-ada-pytorch. 2020. URL: https://github.com/NVlabs/stylegan2-ada-pytorch (дата звернення 27.12.2023).

15. Sauer A., Karras T., Laine S. StyleGAN-T: Unlocking the Power of GANs for Fast Large-Scale Text-to-Image Synthesis. 2023. URL: https://arxiv.org/abs/2301.09515 (дата звернення 27.12.2023).