НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

На правах рукопису
УДК 004.852

До захисту допущено
В.о. завідувача кафедри ШІ
_____О.І. Чумаченко
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 122 «Комп'ютерні науки»

на тему: «Оптимізація та побудова системи генерування зображень на основі генеративно-змагальних мереж»

Виконав:
студент ІІ курсу, групи КІ-з11мп
Дишкант Владислав Олегович _____

Керівник:
Старший викладач кафедри ШІ,
к.т.н., Шаповал Н. В. _____

Рецензент:
доцент кафедри системного проектування
КПІ ім. Ігоря Сікорського, к.ф.-м.н., Шубенкова І. А. _____

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

Київ – 2022 рік

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Рівень вищої освіти – другий (магістерський)

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ О.І. Чумаченко
«___» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Дишканту Владиславу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Оптимізація та побудова системи генерування зображень на основі генеративно-змагальних мереж» науковий керівник роботи Шаповал Наталія Віталіївна, старший викладач кафедри ММСА, к.т.н., затверджено наказом по університету від «02» листопада 2022 р. № 4040-с.

2. Термін подання студентом дисертації 12.12.2022р.

3. Об'єкт дослідження: програмно-апаратні засоби для вирішення задачі генерації зображень на основі генеративно-змагальних мереж.

4. Предмет дослідження: структура та алгоритми навчання нейронних мереж, що вирішують задачу генерації зображень.

5. Перелік завдань, які потрібно зробити:

1) здійснити огляд технічної літератури за темою роботи;

2) дослідити актуальність обраної теми;

3) ознайомитись із існуючими методами генерації зображень;

4) розробити та реалізувати систему генерації зображень;

5) провести експеримент, що засвідчує працездатність запропонованої моделі, виконати аналіз результатів;
6) провести аналіз ринкових можливостей запуску стартап проекту;
7) розробити концептуальні висновки;
8) підготувати ілюстративний матеріал;
9) оформити пояснювальну записку.

6. Перелік ілюстративного матеріалу.

7. Дата видачі завдання 1 вересня 2022 р.

Календарний план

№ з/п	азва етапів виконання дипломного проєкту	Термін виконання етапів дипломного проєкту	Примітка
1.	Вивчення літератури за темою роботи.	02.09.2022 – 14.09.2022	Виконано
2.	Підготовка першого розділу.	15.09.2022 – 29.09.2022	Виконано
3.	Підготовка другого розділу.	30.09.2022 – 18.10.2022	Виконано
4.	Розробка програмного продукту.	20.10.2022 – 05.11.2022	Виконано
5.	Підготовка третього розділу	06.11.2022 – 18.11.2022	Виконано
6.	Підготовка частини стартап-проєкту	19.11.2022 – 22.11.2022	Виконано
7.	Концептуальні висновки. Перспективи розвитку отриманих рішень	23.11.2022 – 25.11.2022	Виконано
8.	Оформлення пояснювальної записки	26.11.2022 – 03.12.2022	Виконано
9.	Вивчення літератури за темою роботи.	02.09.2022 – 14.09.2022	Виконано

Студент Владислав ДИШКАНТ

Керівник Наталія ШАПОВАЛ

АНОТАЦІЯ

Магістерська дисертація: 83 с., 23 табл., 13 рис., 6 джерел, 1 додаток.

Дишкант В.О. «Оптимізація та побудова системи генерування зображень на основі нейронних мереж». НТУУ «КПІ ім. Ігоря Сікорського», Київ, 2022.

Метою виконання магістерської дисертації є: покращення чинних методів розв'язання задачі генерації зображень методом алгоритму генеративно-змагальним нейронним мережам. Об’єктом дослідження є способи генерації зображень нейронною мережею та розгляд найкращого способу генерації зображень.

В магістерській дисертації було виконано наступні задачі: розглянуто існуючі науково-технічну літературу на тему нейронних мереж та теперішні аналоги генерації зображень, алгоритми генерації зображень та визначення найкращих параметрів, які дозволять розв'язувати задачу покращення результатів генерації зображень, створено програмну систему генерації зображень за допомогою нейронних мереж із застосуванням різноманітних методів та алгоритмів.

В результаті виконання дипломної роботи створено та оптимізовано систему генерації зображень за допомогою нейронних мереж.

Перелік ключових слів: нейронні мережі, генеративно-змагальна мережа,генератор, дискримінатор.

ABSTRACT

Master's thesis: 83 pp., 23 tables, 13 figures, 6 sources, 1 appendix.

Dyshkant V.O. "Optimization and construction of image generation system based on neural networks." NTUU "KPI them. Igor Sikorsky », Kyiv, 2022.

The purpose of the master's dissertation is: to improve the current methods of solving the problem of generating images by the method of algorithm generative-competitive neural networks. The object of the research is the methods of generating images by a neural network and the consideration of the best way of generating images.

The following tasks were performed in the master's dissertation: the existing scientific and technical literature on neural networks and current analogues of image generation, algorithms for image generation and determining the best parameters to solve the problem of improving image generation, created a software system for image generation using neural networks using various methods and algorithms.

As a result of the thesis, an image generation system using neural networks was created and optimized.

List of key words: neural networks, generative-competitive network, generator, discriminator.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

NN – нейронна мережа

GAN – генеративно-змагальна мережа

CNN – згорткова нейронна мережа

RNN – рекурентна нейронна мережа

BPNN – нейронна мережа зворотнього поширення

FPNN – нейронна мережа прямого поширення

WGAN – генеративно-змагальна мережа Вассерштейна

CGAN – умовна генеративно-змагальна мережа

ProGAN – прогресивна генеративно-змагальна мережа

ВСТУП

Використання нейронних мереж є незамінимою частиною інформаційних технологій. За допомогою їх ми можемо передбачити курс акцій за допомогою статичної інформації, допомагає у виявленні хвороби по зображенню рентгенівського знімку, прогнозуванні властивостей хімічних сполук, що дає змогу в створенні медикаментів, здійсненні керуванням рухомих об’єктів (наприклад, автомобілів).

Більш складною галуззю машинного навчання є глибоке навчання, яке полягає у відкритті багатих ієрархічних моделей, які представляють ймовірнісні розподіли за типами даних, які зустрічаються в програмах штучного інтелекту, наприклад, природних зображень, звукових сигналів, що містять мовлення і символи природної мови.

Однією із нових тем для нейронних мереж є генерація зображень. Алгоритми та методи для генерації з’явилися недавно. Наприклад, генеративно-змагальна нейронна мережа з’явилася в 2014 році, а перші результати, які на виході давали непоганий результат в 2016 році. На даний момент повноцінної програми у відкритому доступі не має, але відомо загальний принцип, по якій працюють ці алгоритми та математичні формули.

Генерація зображень має перспективу як технології, яка стане незамінна у майбутньому. Створення контенту та даних, наприклад, картинок для різних потреб, що згенеровані автоматично. Завдяки роботі генеративних моделей виникає синтез даних, за допомогою яких потім навчатимуться інші системи.

Автоматичне редагування також є задачею, яка може використовуватися у сучасних електронних пристроях та деяких програмах. Це дозволяє змінювати на фотографії вираз обличчя, прибрати зморшки та змінити зачіску, змінити фон на фото, змінити свій вік на зображенні.

РОЗДІЛ 1. ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ

1.1. Поняття глибоких нейронних мереж (DL)

Глибоке навчання це частина машинного навчання, яке по суті є нейронною мережею з трьома або більше шарами. Ці нейронні мережі намагаються
імітувати поведінку людського мозку — хоча й далеко не відповідають його здібностям — дозволяючи йому «вчитися» на великих обсягах даних. У той час як нейронна мережа з одним шаром все ще може робити приблизні прогнози, додаткові приховані шари можуть допомогти оптимізувати та уточнити для точності.

Глибоке навчання керує багатьма програмами та сервісами штучного інтелекту (AI), які покращують автоматизацію, виконуючи аналітичні та фізичні завдання без участі людини. Технологія глибокого навчання лежить в основі повсякденних продуктів і послуг (наприклад, цифрових помічників, голосових пультів від телевізора та виявлення шахрайства з кредитними картками), а також новітніх технологій (наприклад, самокерованих автомобілів).

1.2. Принцип роботи DL

Мережі глибокого навчання навчаються, відкриваючи складні структури в даних, які вони відчувають. Створюючи обчислювальні моделі, які складаються з кількох шарів обробки, мережі можуть створити кілька рівнів абстракції для представлення даних.

Наприклад, модель глибокого навчання, відому як згортка нейронна мережа, може бути навчена за допомогою великої кількості (як у мільйонах) зображень, наприклад зображень, що містять кішок. Цей тип нейронної мережі зазвичай навчається на пікселях, які містяться в отриманих зображеннях. Він може класифікувати групи пікселів, які представляють риси кішки, з такими групами ознак, як кігті, вуха та очі, які вказують на присутність кішки на зображенні.

Глибоке навчання принципово відрізняється від звичайного машинного навчання. Наприклад, експерту в області знадобиться витратити значний час на розробку звичайної системи машинного навчання, щоб виявити особливості, які представляють кішку. Завдяки глибокому навчанню все, що потрібно, це надати системі дуже велику кількість зображень кішок, і система може автономно вивчати особливості, які представляють кішку.

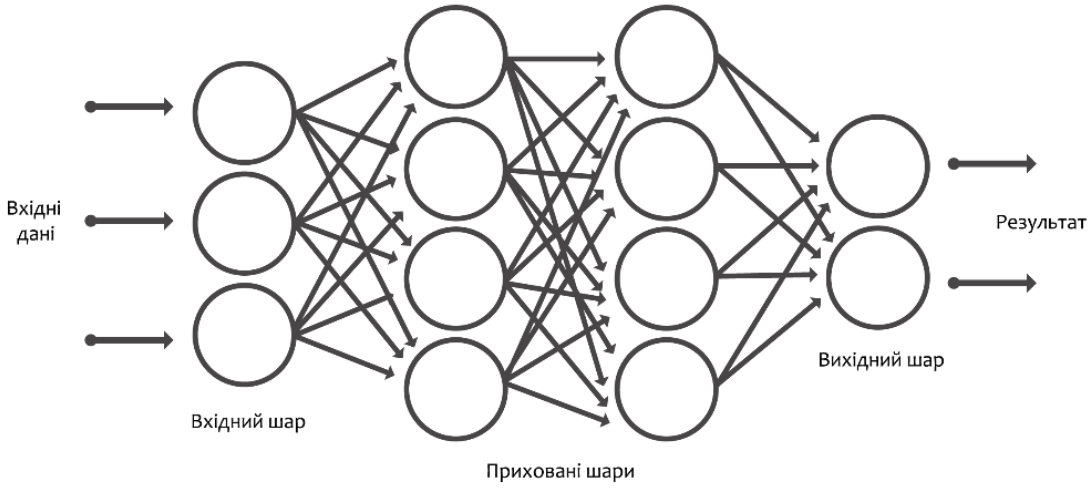

рис. 1.1. Базова модель глибокої нейронної мережі

Для багатьох завдань, таких як комп'ютерний зір, розпізнавання мовлення (також відоме як обробка природної мови), машинний переклад і робототехніка, продуктивність систем глибокого навчання набагато перевищує продуктивність звичайних систем машинного навчання. Це не означає, що побудова систем глибокого навчання відносно проста в порівнянні зі звичайними системами машинного навчання. Хоча розпізнавання функцій є автономним у глибокому навчанні, тисячі гіперпараметрів (регуляторів) потрібно налаштувати, щоб модель глибокого навчання стала ефективною.

1.3. Способи використання DL

Глибоке навчання ідеально підходить для прогнозування результатів, коли у вас є багато даних, з яких можна вчитися – «багато» — це величезний набір даних із сотнями тисяч або навіть більше мільйонів точок даних. Там, де у вас є величезний обсяг даних, як цей, система має те, що їй потрібно для

самотренування.

Це також найкраще застосовувати до складних проблем і речей, вирішення яких за допомогою людського рішення було б дуже дорогим. Обробка зображень є чудовим прикладом цього. Замість того, щоб платити людям, щоб вони перебирали мільйони відео та позначали на них теги для допомоги пошуку користувачам, набагато доцільніше застосувати глибоке навчання. Те жсаме з перекладом і розпізнаванням мовлення.

1.4. Різниця між машинним і глибоким навчанням

Глибоке навчання — це спеціалізована форма машинного навчання. Робочий процес машинного навчання починається з вилучення відповідних функцій із зображень вручну. Потім ці функції використовуються для створення моделі, яка класифікує об’єкти на зображенні. Завдяки робочому процесу глибокого навчання відповідні функції автоматично витягуються із зображень. Крім того, глибоке навчання виконує «наскрізне навчання» – коли мережі надаються необроблені дані та завдання для виконання, наприклад класифікація, і вона вчиться робити це автоматично.

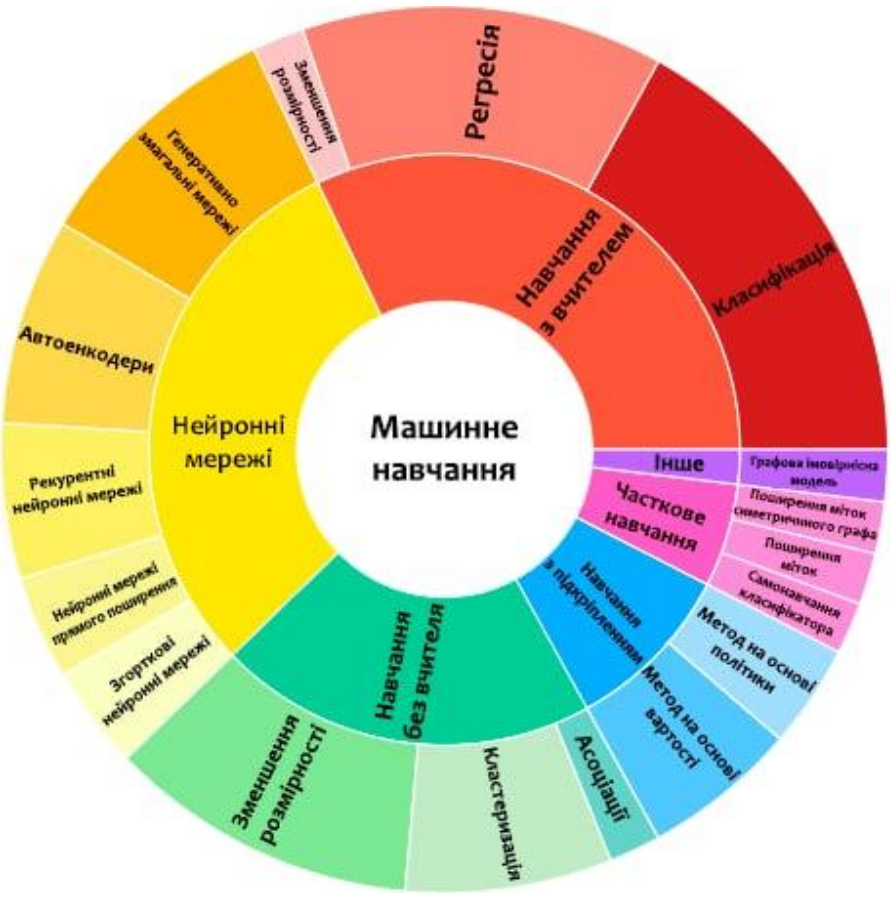

рис. 1.2. Складові машинного навчання

Ще одна ключова відмінність полягає в тому, що алгоритми глибокого навчання масштабуються з даними, тоді як поверхневе навчання збігається. Поверхневе навчання відноситься до методів машинного навчання, які досягають плато на певному рівні продуктивності, коли ви додаєте більше прикладів і навчальних даних у мережу.

Ключова перевага мереж глибокого навчання полягає в тому, що вони часто продовжують вдосконалюватися зі збільшенням розміру ваших даних.

У машинному навчанні ви вручну вибираєте функції та класифікатор для сортування зображень. Завдяки глибокому навчанню етапи вилучення функцій і моделювання виконуються автоматично.

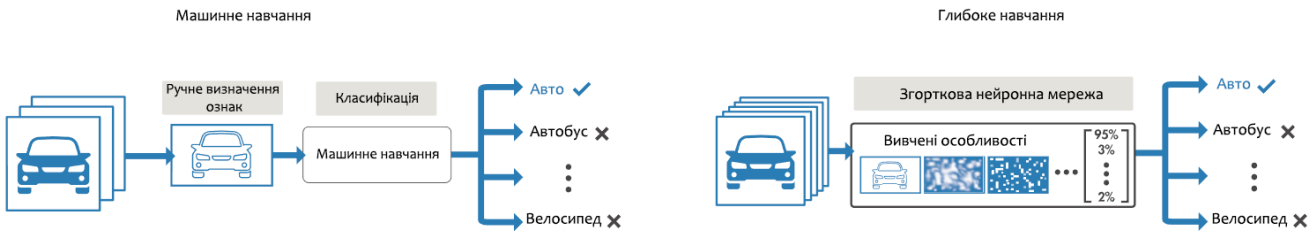

рис. 1.3. Різниця між машинним та глибоким навчанням

Машинне навчання пропонує різноманітні методи та моделі, які можна вибрати залежно від вашої програми, розміру даних, які ви обробляєте, і типу проблеми, яку ви хочете вирішити. Для успішної програми глибокого навчання потрібен дуже великий обсяг даних (тисячі зображень) для навчання моделі, а також графічні процесори або графічні процесори для швидкої обробки ваших даних.

1.5. Висновки

Отже, вибираючи між машинним навчанням і глибоким навчанням, подумайте, чи є у вас високопродуктивний графічний процесор і чи є у вас багато позначених даних. Якщо у вас немає жодної з цих речей, можливо, буде доцільніше використовувати машинне навчання замість глибокого навчання. Глибоке навчання, як правило, складніше, тому вам знадобиться принаймні кілька тисяч зображень, щоб отримати надійні результати. Наявність високопродуктивного графічного процесора означає, що модель потребуватиме менше часу для аналізу всіх цих зображень

І останнє, але не менш важливе, глибоке навчання доцільно лише в тому випадку, якщо у вас є високоякісні обчислювальні потужності, щоб змусити його працювати, або якщо ви співпрацюєте з постачальником аналітики, який має інфраструктуру та навички, яких може не вистачати в компанії.

Якщо ваші обставини відповідають цим критеріям, то глибоке навчання може стати ідеальним рішенням вашої бізнес-проблеми. З іншого боку, меншпередові аналітичні рішення можуть бути кращим шляхом до успіху.

РОЗДІЛ 2. АЛГОРИТМИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

2.1. Генеративно-змагальні мережі (GAN)

GAN це генеративно-змагальна нейромережа, яка одиним із алгоритмів класичного машинного навчання, навчання без вчителя. Суть ідеї у комбінації двох нейромереж, при якій одночасно працює два алгоритми “генератор” та “дискримінатор”. Завдання генератора – генерувати образи заданої категорії. Завдання дискримінатора – намагатися розпізнати створений образ.

Таким чином, генератор генерує певні образи. Наприклад, картинки, схожі на обличчя, а дискримінатор намагається визначити чи це обличчя булоні. І згодом мережа навчається настільки, що генератор генерує дуже реалістичні особи.[1]

2.2. Принцип роботи GAN

2.2.1. Дискримінатор

Для розпізнавання використовуються згорткові нейронні мережі (CNN). Принципи роботи та застосування CNN описані в окремій статті. Для розуміння “на пальцях”: CNN може розпізнавати образи на картинках, наприклад, виділяти зі всього зображення обличчя, цифри тощо. Для того, щоб нейронна мережа навчилася щось розпізнавати, їй потрібно обробити велику кількість зображень, де містяться образи.[2]

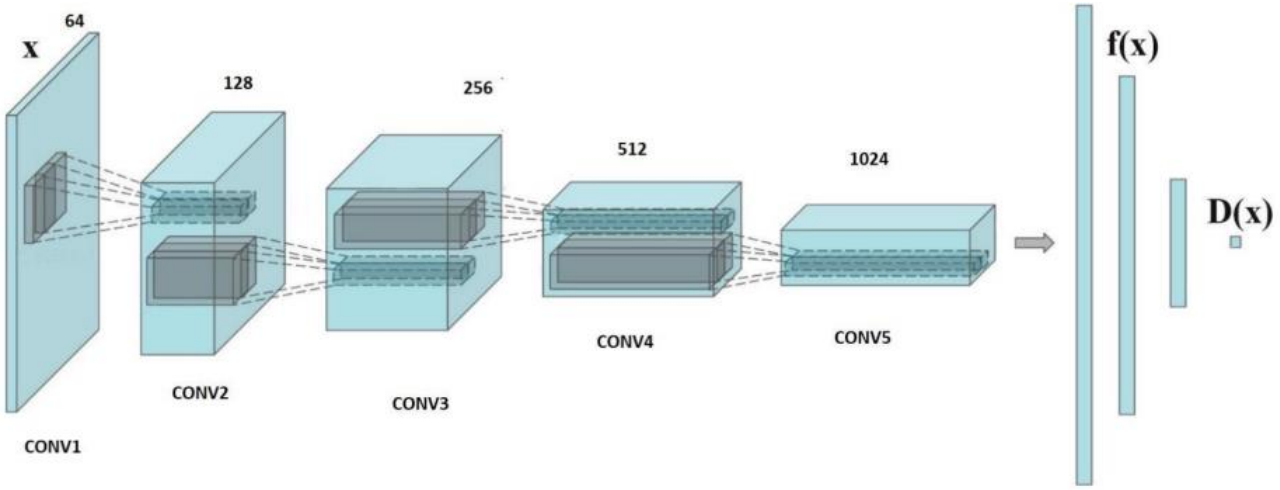

рис. 2.1. Модель дискримінатора

Простою мовою: ви даєте нейронної мережі велику кількість картинок котиків, на яких відзначаєте в якій частині зображення знаходяться котики (це називається "розмітка"), і потім нейромережа вже сама здатна сказати чи є на картинці котики і де вони знаходяться.[3]

Функція втрат дискримінатора може бути отримана з формули бінарної перехресної ентропії:

$$L(\hat{y}, y) = y \cdot ln\,\hat{y} + (1 - y) \cdot \ln(1 - \hat{y})$$

Під час навчання дискримінатора на зображеннях, створених генератором:

$$L(\hat{y}, y) = L(D(G(z)), 0) = \ln\left(1 - D\left(G(z)\right)\right)$$

Мета дискримінатора – правильно класифікувати справжній та підроблений набір даних, для цього функції мають бути максимізовані, а остаточна функція втрат буде виглядати так:

$$L^{(D)} = \max\left[\ln D(x) + \ln\left(1 - D\left(G(z)\right)\right)\right]$$

2.2.2. Генератор

Формування зображень починається з генерації довільного шуму, у якому поступово починають проступати фрагменти шуканого зображення. Уявіть собі, що ви трусите тарілку з піском, поки не вдається "натрісти" щось, що нагадує цифру. А потім продовжуєте трясти доки контури цифри не стануть явнішими. Нейромережа запам'ятовує як саме ви трясли тарілку, щоб досягти такого результату, і наступного разу відтворює ваші дії.[4]

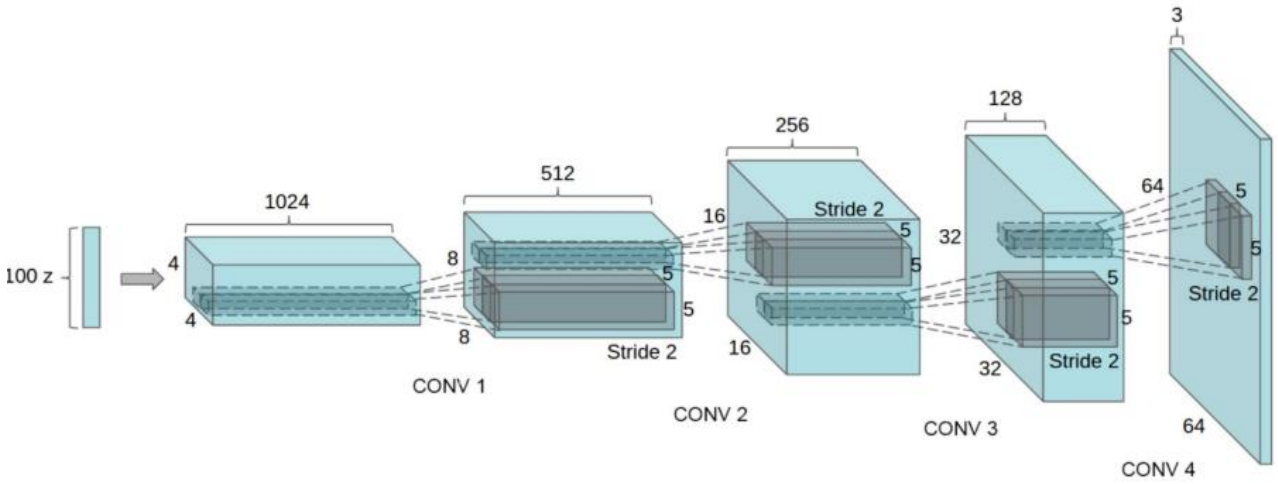

рис. 2.2. Модель генератора

Генератор постійно бореться із дискримінатором і намагається мінімізувати рівняння:

$$L^{(G)} = \min\left[\ln D(x) + \ln\left(1 - D\big(G(z)\big)\right)\right]$$

2.3. Генеративно-змагальні мережі Вассерштейна (WGAN)

Незважаючи на те, що GAN з їх конкуруючими моделями генераторів і дискримінаторів здатні досягти величезного успіху є кілька випадків відмови цих мереж.

Дві з найпоширеніших причин пов’язані або з невдачею конвергенції, або згортанням режиму. При невдачі конвергенції модель не змогла дати оптимальні або якісні результати. У разі згортання режиму модель не змогла створити унікальні зображення, що повторюють подібний шаблон або якість. Отже, щоб вирішити деякі з цих проблем або боротися з численними проблемами, поступово було розроблено багато варіацій і версій для GAN. Мережа WGAN створена для боротьби з такими проблемами. WGAN забезпечує вищу стабільність моделі навчання порівняно з простими архітектурами GAN. Функція втрат, яка використовується в WGAN, також дає нам критерій завершення для оцінки моделі. Хоча інколи тренування може зайняти трохи більше часу, це один із кращих варіантів досягнення ефективніших результатів.

Ідея роботи генеративних змагальних мереж (GAN) полягає у використанні двох первинних розподілів ймовірностей. Однією з основних сутностей є розподіл ймовірностей генератора (Pg), який відноситься до розподілу з виходу моделі генератора. Іншою істотною сутністю є розподіл ймовірностей з реальних зображень (Pr). Мета Generative Adversarial Networks полягає в тому, щоб обидва ці розподіли ймовірностей були близькими один до одного, щоб отримані результати були дуже реалістичними та високоякісними.

Для обчислення відстані цих розподілів ймовірностей математична статистика в машинному навчанні пропонує три основні методи, а саме розбіжність Кульбака–Лейблера, розбіжність Єнсена–Шеннона та відстань Вассерштейна. Розбіжність Дженсена–Шеннона (також типова втрата GAN) спочатку є найбільш використовуваним механізмом у простих мережах GAN. Однак цей метод має проблеми під час роботи з градієнтами, які можуть призвести до нестабільного навчання. Тому ми використовуємо відстань Вассерштейна для вирішення таких повторюваних проблем.

WGAN використовує підхід градієнтного штрафу для ефективного вирішення попередніх проблем цієї мережі. Лямбда визначає штрафний коефіцієнт градієнта, тоді як n-критика відноситься до кількості критичних ітерацій на ітерацію генератора. Альфа- та бета-значення відносяться до обмежень оптимізатора Адама. Цей підхід передбачає використання інтерполяційного зображення поряд зі згенерованим зображенням перед додаванням функції втрат із градієнтним штрафом, оскільки це допомагає задовольнити обмеження Ліпшица. Алгоритм виконується, доки ми не зможемо досягти задовільної збіжності необхідних даних.

2.4. Умовна генеративна змагальна мережа (CGAN)

CGAN генерує певний тип даних і має достатній контроль над виходами GAN.У безумовній генеративній моделі немає контролю над режимами даних, що генеруються.

В умовному GAN (CGAN) генератор вчиться генерувати підроблений зразок із певною умовою або характеристиками (наприклад, мітка, пов’язана із зображенням або більш детальним тегом), а не загальний зразок із невідомого розподілу шуму. Просто подача даних у і кондиціонування як генератора, так і дискримінатора.

Наприклад, припустімо, що ви використовували широкий спектр зображень квітів, щоб навчити GAN, здатний створювати підроблені зображення квітів. Хоча ви можете використовувати свою модель для створення зображення випадкової квітки, ви не можете дати їй команду створити зображення, скажімо, тюльпана чи соняшника.

Умовний GAN (cGAN) дозволяє нам обумовлювати мережу додатковою інформацією, такою як мітки класу. Це означає, що під час навчання ми передаємо в мережу зображення з їх справжніми мітками (троянда, тюльпан, соняшник тощо), щоб вона дізналася різницю між ними. Таким чином ми отримуємо можливість попросити нашу модель створити зображення конкретних квітів.

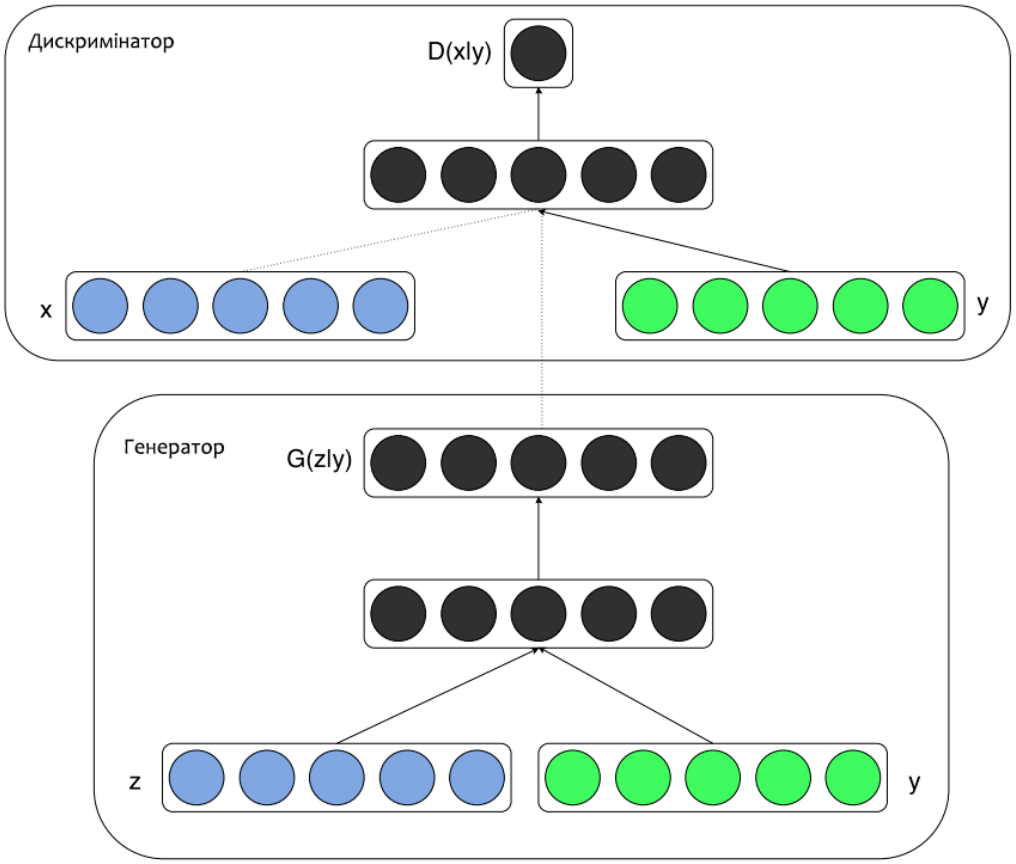

рис. 2.3. Модель умовного GAN

Змагальна втрата навчання CGAN:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

2.5. Прогресивно зростаюча генеративно-змагальна мережа (ProGAN)

Основна концепція полягає в GAN який поступово зростає під час навчання для створення зображень із великою роздільною здатністю, наприклад, 1024x1024.[5]

Великі зображення, наприклад квадратні зображення 1024x1024 пікселів, вимагають більше пам’яті, яка обмежена на сучасному апаратному забезпеченні GPU порівняно з основною пам’яттю. Крім того, великі розміри вносять нестабільність у навчальний процес GAN. Крім того, попередня архітектура GAN легко створювала зображення низької роздільної здатності, але зображення з вищою роздільною здатністю цими моделями було важко

створити.

Рішення полягає в поступовому збільшенні кількості рівнів під час процесу навчання що призводить до прогресивного GAN, або ProGAN. Він дозволяє генерувати великі високоякісні зображення, наприклад реалістичні обличчя розміром 1024 × 1024 пікселів шляхом ефективного навчання моделі генератора.

Щоб це запрацювало, автори представляють концепції прогресивного збільшення рівня як у генераторі, так і в дискримінаторі зі збільшенням роздільної здатності , вирівняної швидкості навчання , рівня нормалізації пікселів і стандартного відхилення mini-batch. ProGAN передбачає додавання блоків шарів і внесення коригувань у додавання блоків шарів, а не додавання їх безпосередньо. Крім того, усі шари залишаються доступними для навчання під час процесу навчання, включаючи існуючі шари, коли додаються нові шари.

GAN з прогресивним зростанням починається з дуже маленьких зображень і використовує модель генератора та дискримінатора з однаковою загальною структурою, наприклад 4×4 пікселя. Пізніше, під час навчання, додаються нові блоки згорткових шарів як на генераторі, так і на дискримінаторі, що необхідне для виведення зображень високої роздільної здатності 1024x1024. Це поступове розширення мереж генератора і дискримінатора дозволяє моделям спочатку ефективно вивчати деталі високого рівня, а потім зосереджуватися на розумінні тонких особливостей піксельних зображень високої роздільної здатності. Це підвищує стабільність моделі та знижує ймовірність «колапсу режиму», тобто низької варіативності генерованих зображень.[6]

Крім того, вони використовують методи згладжування, які називаються mini-batch стандартним відхиленням і піксельною нормалізацією замість пакетної норми для виведення нових шарів, щоб уникнути раптових поштовхів на вже добре навченому шарі з меншою роздільною здатністю під час додавання нових блоків шарів.

2.6. Висновок

В даному розділі було розглянуто основні типи GAN. Він має різноманітні застосування в різних сферах: створення веселих мультяшних персонажів, зразків для наборів даних зображень, створення людських облич, високоякісні реалістичні фотографії, переклад тексту в зображення, людських поз, старіння обличчя, генерація 3D об'єктів, розфарбовування зображення та багато іншого. GAN все ще розвивається, і є багато різних різновидів із більш складними моделями GAN, які можливо досліджувати. Ми можемо побудувати свою модель генерації зображень або, якщо у вас невелика кількість даних зображення, можемо згенерувати більше за допомогою GAN, що є досить хорошими результатами.

РОЗДІЛ 3. ПОБУДОВА СИСТЕМИ ГЕНЕРУВАННЯ ЗОБРАЖЕНЬ

Генеративно-змагальні мережі (GAN) можуть створювати неймовірно реалістичні синтетичні зображення. Під час навчання GAN протиставляє генератор, який створює зображення, та дискримінатор, який намагається відрізнити реальне зображення від синтетичного. «Гонка озброєнь» між ними може дати дуже переконливий генератор.

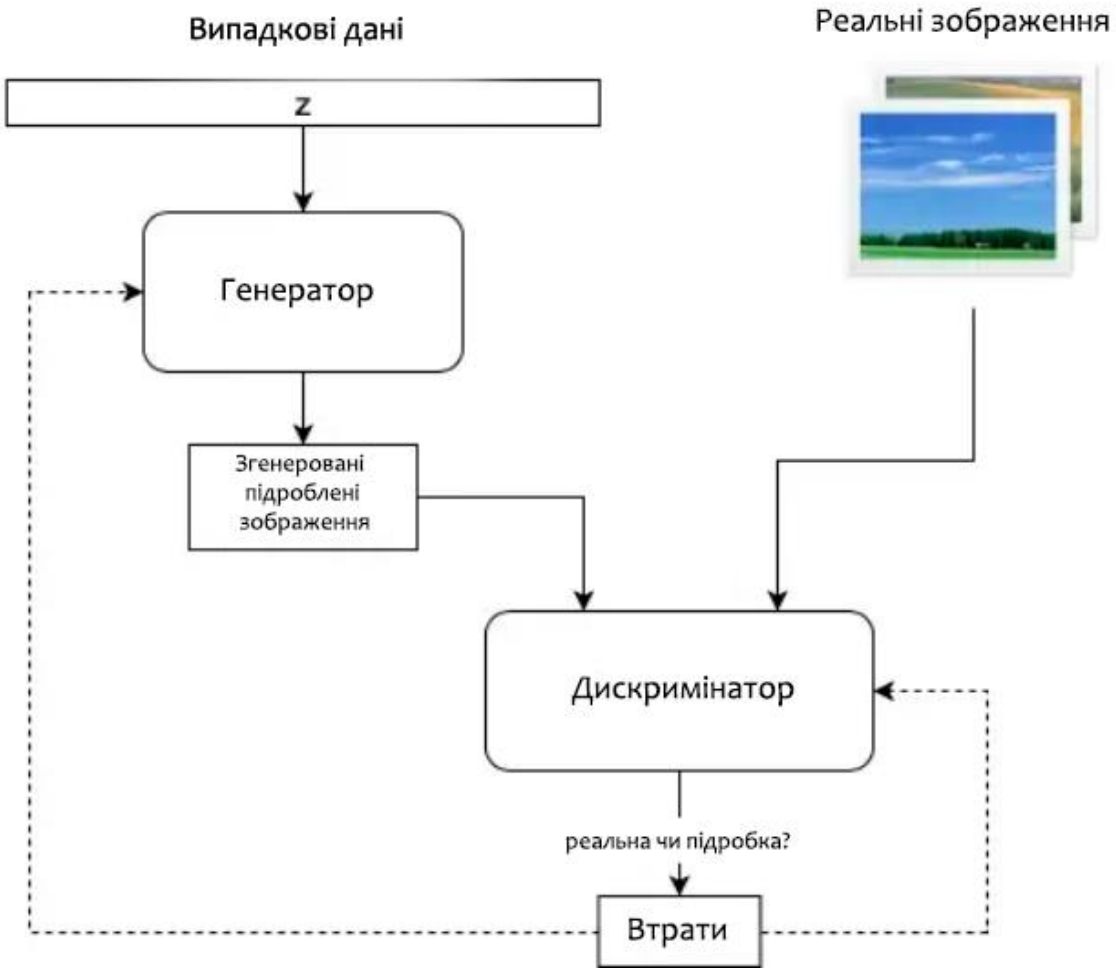

рис. 3.1.

Для створення чітких і різноманітних зображень високої роздільної здатності потрібні великі мережі. Однак, якщо мережа занадто велика, змагальне навчання може не об’єднатися з хорошим генератором. Дослідники вирішують цю проблему, починаючи з невеликого генератора та відповідно

невеликого дискримінатора та поступово додаючи все більше й більше рівнів нейронної мережі до обох, гарантуючи, що генератор підтримує базовий рівень продуктивності в міру зростання його складності.

В даному проєкту буде реалізовано прогресивно зростаючі моделі GAN. Буде побудовано генерування зображень випадкових осіб з розширенням 64x64 із-за необхідності великої кількості графічних процесорів для навчання і довгого часу, необхідної для навчання генерування зображень з розширенням 1024x1024.

3.1. Вибір та підготовка даних

В якості набору даних будемо використовувати CelebFaces Attributes Dataset або скорочено CelebA, що являє собою набір даних зображень, який визначає атрибути обличчя знаменитостей. Він містить 202 599 зображень облич у п’яти визначних місцях із 40 анотаціями бінарних атрибутів для кожного зображення. Наразі він містить дані про понад 10 000 знаменитостей. CelebA містить багато різноманітних зображень, що охоплюють різні пози та варіації фону. Він надає багаті анотації для зображень, корисних для навчання моделям машинного навчання та комп’ютерного зору.

Зображення з набору даних є цінним ресурсом для тестування та навчання для різних випадків використання. Ви можете використовувати CelebA для навчання моделей розпізнавання облич і атрибутів облич. Набір даних також корисний для локалізації орієнтирів, редагування зображень обличчя та задач синтезу зображень.

3.2. Побудова Progressive Growing of GANs

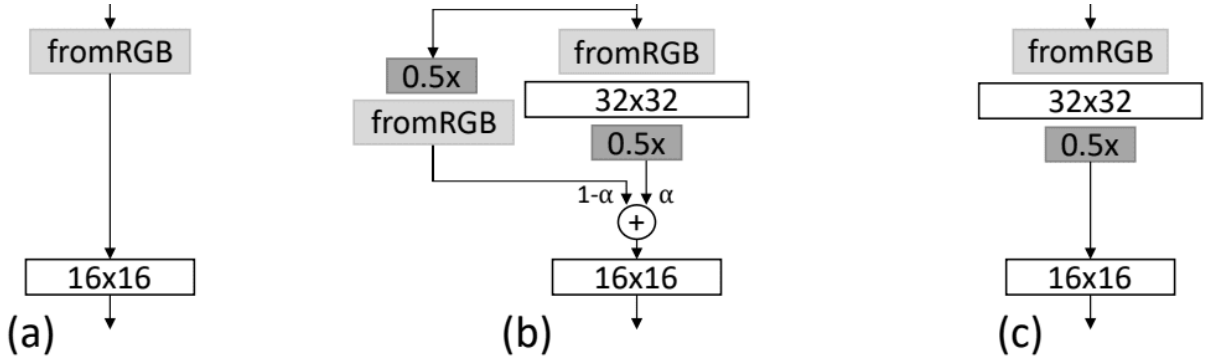

рис. 3.2. Зростання моделі дискримінатора до (а) під час (б) і після (в) поступового впровадження високої роздільної здатності.

Основні переваги порівняно із звичайним GAN:

- більш реалістичні зображення;
- зображення у високій якості;
- велика варіативність зображень;
- легше та швидше навчається;

Для якісної оцінки зображень, створених генератором, можна використовувати такі характеристики:

- подібність із зображеннями навчальної вибірки;
- відсутність дублікатів із навчальної вибірки;
- різноманітність зображень;
- відсутність артефактів;

Прийоми для стабільного навчання

- нормалізація вхідних даних (зображення в діапазоні [-1, 1])
- tanh як функція активації генератора;
- нормальний розподіл замість рівномірного шуму;
- роздільні пакети для справжніх та фейкових зображень;
- використання батч-нормалізації;
- використання LeakyReLU замість ReLU;
- додавання шуму в мітки ([0, 0.3] та [0.7, 1.2] замість 0 та 1)
- Використання Adam зі значенням моменту 0.5
- використання згорткових шарів із кроком > 1 замість макспулінгу;

3.3. Особливості архітектури

3.3.1. Попіксельная нормалізація

Для більшості даних зображень значення пікселів є цілими числами зі значеннями від 0 до 255.

Нейронні мережі обробляють вхідні дані, використовуючи невеликі значення ваги, а вхідні дані з великими цілими значеннями можуть порушити або уповільнити процес навчання. Таким чином, добре нормалізувати значення пікселів, щоб кожне значення пікселя мало значення від 0 до 1.

Для зображень допустимо мати значення пікселів у діапазоні 0-1, і зображення можна переглядати нормально.

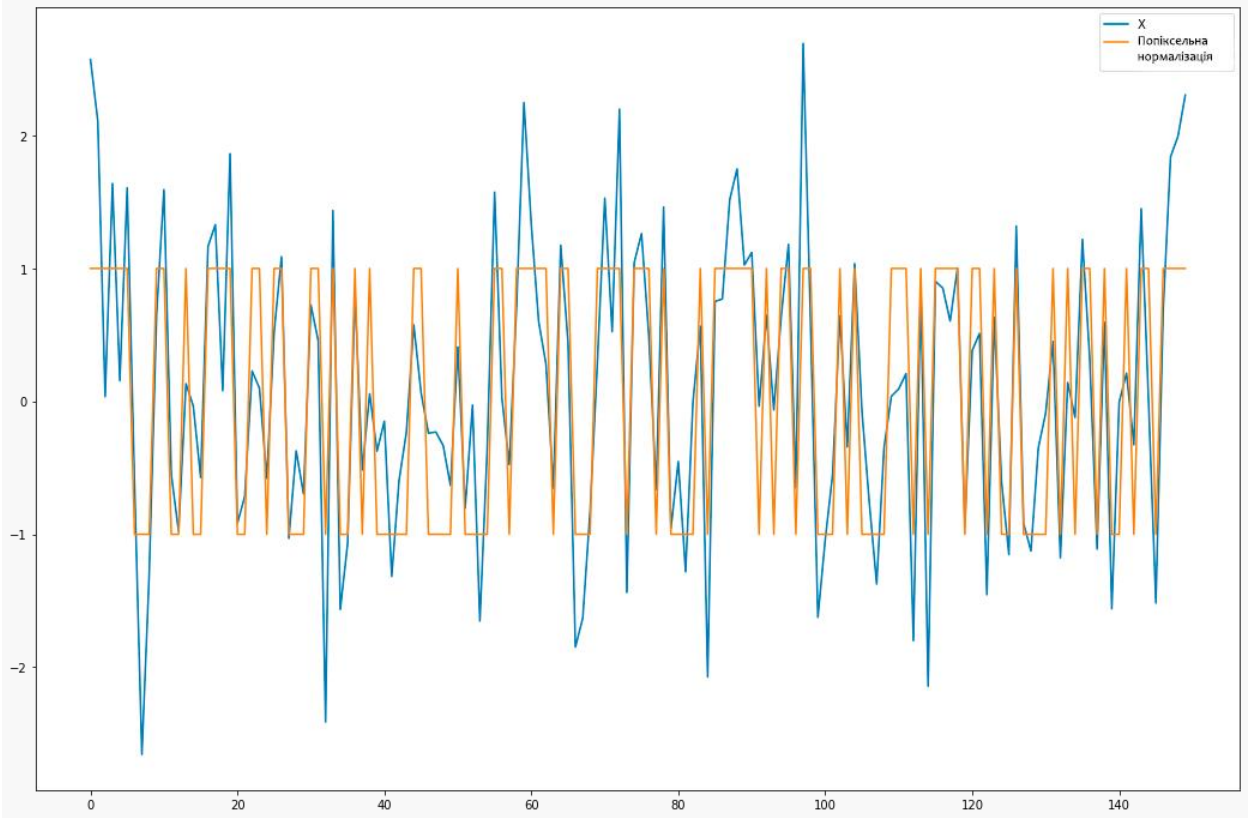

рис. 3.3.

Формула векторної піксельної нормалізації функції у генераторі:

$$b_{x,y} = a_{x,y} / \sqrt{\frac{1}{N} \sum_{j=0}^{N-1} \left(a_{x,y}^{j}\right)^2 + \epsilon}$$

Цього можна досягти, поділивши всі значення пікселів на найбільше

значення пікселів; тобто 255. Це виконується для всіх каналів, незалежно від фактичного діапазону значень пікселів, присутніх на зображенні.

Дані включають фотографії з поганою контрастністю, наприклад, із-за бліків. Нормалізація іноді називається растягненням контраста або растягненням гістограми. У більш загальних обробках даних, таких як цифрова обробка сигналів, це називається розширенням динамічного діапазону.

Метою розширення динамічного діапазону в різних програмах зазвичай є переведення зображення або іншого типу сигналу в діапазон, більш звичний або нормальний для почуттів, звідси термін нормалізація. Часто мотивація полягає в досягненні узгодженості динамічного діапазону для набору даних, сигналів або зображень, щоб уникнути розумового відволікання або втоми. Наприклад, газета намагатиметься зробити так, щоб усі зображення у номері мали однаковий діапазон відтінків сірого .

Наприклад, якщо діапазон інтенсивності зображення становить від 50 до 180, а бажаний діапазон – від 0 до 255, процес передбачає віднімання 50 від інтенсивності кожного пікселя, утворюючи діапазон від 0 до 130. Тоді інтенсивність кожного пікселя множиться на 255/130 , становлячи діапазон від 0 до 255.

3.3.2. Ініціалізація та нормалізація ваг

Припустимо, що проста мережа з двома рівнями (L-1) і (L) має однакову функцію активації з двома наборами ваг w1 і w2. Якщо вагові коефіцієнти w1 і w2 мають однакове початкове значення, то це ясно, що різні два вузли z(L)_(1) і z(L)_(2) мають однакове значення.

Це означає що вузли z(L)_(1) і z(L)_(2) представляють ту саму межу рішення в просторі гіпотез. Оскільки z(L)_(1) і z(L)_(2) мають однакові вхідні дані z(L-1) , ваги (w1=w2) і функцію активації, вага w1 буде оновлено в тому самому як w2 оновлюється. Це, зрештою, призводить до втрати цілі, тому ми розміщуємо кілька прихованих одиниць (вузлів) у шарі. Тому, оскільки ми хочемо, щоб кожен вузол представляв іншу функцію (межу рішення), ми випадково ініціалізуємо ваги, щоб порушити симетрію між вузлами. Можна

отримати сильніший ефект порушення симетрії при ініціалізації ваг з великими значеннями, що також допомагає уникнути надлишкових одиниць. Крім того, оскільки ваги великі, також корисно запобігти зникненню сигналу як у прямому, так і у зворотному напрямку. Однак якщо ваги занадто великі, це може спричинити рознесення значень під час передачі прямого/зворотного слова. Крім того, великі ваги мають більший шанс призвести до екстремальних значень, які спричиняють насиченість функції активації та втрату градієнтів.

В даному проєкті використаємо формулу для вирівнення швидкості навчання:

$$\hat{w}_i = w_i / c$$
$$c = \sqrt{2/N}$$

Ідея вирівняної швидкості навчання полягає в тому, щоб масштабувати вагові коефіцієнти на кожному рівні з константою так, щоб оновлена вага w' масштабувалася до w' = w /c, де c є константою на кожному рівні. Це робиться під час тренування, щоб зберегти вагу в мережі в однаковому масштабі під час тренування.

3.3.3. Нарощування шарів

Прогресивна зростаюча генеративна змагальна мережа це підхід до навчання моделі глибокої згорткової нейронної мережі для створення синтетичних зображень.

Це розширення більш традиційної архітектури GAN, яка передбачає поступове збільшення розміру створеного зображення під час навчання, починаючи з дуже маленького зображення, наприклад 4×4 пікселя. Це забезпечує стабільне навчання та зростання моделей GAN, здатних генерувати дуже великі високоякісні зображення, наприклад зображення синтетичних облич знаменитостей розміром 1024 × 1024 пікселів.

3.3.4. Шар пропуску альфа-переходу

Модель дискримінатора отримує зображення як вхідні дані, і вона повинна класифікувати їх як справжні (з набору даних) або підроблені (згенеровані).

Під час процесу навчання дискримінатор повинен зростати, щоб підтримувати зображення постійно зростаючого розміру, починаючи з кольорових зображень 4×4 пікселів і подвоюючи до 8×8, 16×16, 32×32 тощо.
Це досягається шляхом вставки нового вхідного шару для підтримки більшого вхідного зображення, а потім нового блоку шарів. Вихід цього нового блоку потім зменшується. Крім того, нове зображення також безпосередньо зменшується та проходить через старий вхідний рівень обробки, перш ніж об’єднати його з виходом нового блоку. Вихідні дані нового блоку зі зниженою дискретизацією та вихідні дані старого рівня обробки вхідних даних об’єднуються за допомогою середньозваженого значення, де зважування контролюється новим гіперпараметром під назвою «альфа».

Спочатку зважування повністю зміщено до старого рівня обробки вхідних даних (*alpha=0*) і лінійно збільшується протягом ітерацій навчання, щоб новому блоку надавалося більше ваги, доки, зрештою, результат не буде повністю продуктом нового блоку (*alpha= 1*). У цей час старий блок можна видалити.

3.3.5. Дискримінатор mini-Batch стандартного відхилення

Генеративні змагальні мережі мають тенденцію фіксувати лише невеликі варіації даних навчання. Іноді всі вхідні вектори шуму генерують подібні зображення. Ця проблема також відома як «згортання режиму». Щоб додати невеликі варіації до згенерованих зображень, використаємо стандартне mini-batch відхилення.

Відбувається обчислення стандартного відхилення кожної функції в карті активації, а потім усереднюються для mini-batch. За допомогою цієї нової активації потім створюються карти. Ці нові карти додаються в кінці мережі дискримінатора.

3.3.6. Зменшення дискретизації

Набір даних класифікації з перекошеними пропорціями класу називається незбалансованим набором даних. Під час роботи над проблемою класифікації ви коли-небудь стикалися з набором даних зміщення, який містить більшість зразків певного класу. Отже, щоб перетворити набір даних таким чином, щоб він містив однакову кількість класів у цільовому значенні, ми можемо зменшити вибірку набору даних. Зниження дискретизації означає зменшення кількості вибірок, які мають клас зміщення.

Для цього спочатку треба вибирати рядки, де цільові значення дорівнюють 0 і 1 у двох різних об’єктах. Перевіривши вихідні дані ми побачимо, що кількість зразків, які мають цільові значення як 1, набагато більші за 0. Отже, під час зменшення дискретизації ми випадковим чином виберемо кількість рядків із цільовими значеннями як 1 і зробимо її рівною кількості рядків із цільовими значеннями 0.

3.4. Аналіз та порівняння результатів

Очікуваний час оцінки та результати для з використанням одного графічних процесорів Tesla V100 складає:

GPUs	**1024×1024**	**512×512**	**256×256**
1	~41 дні	~25 днів	~15 днів
2	~22 дні	~13 днів	~9 днів
4	~11 днів	~7 днів	~5 дні

Як ми бачимо з таблиці навчання генератора займає деякий час навіть з викристання сучасних графічних процесорів що показують себе краще для навчання нейронних мереж і вираховується не в годинах, а зазвичай в тижнях, для великих проєктів в місяцях.

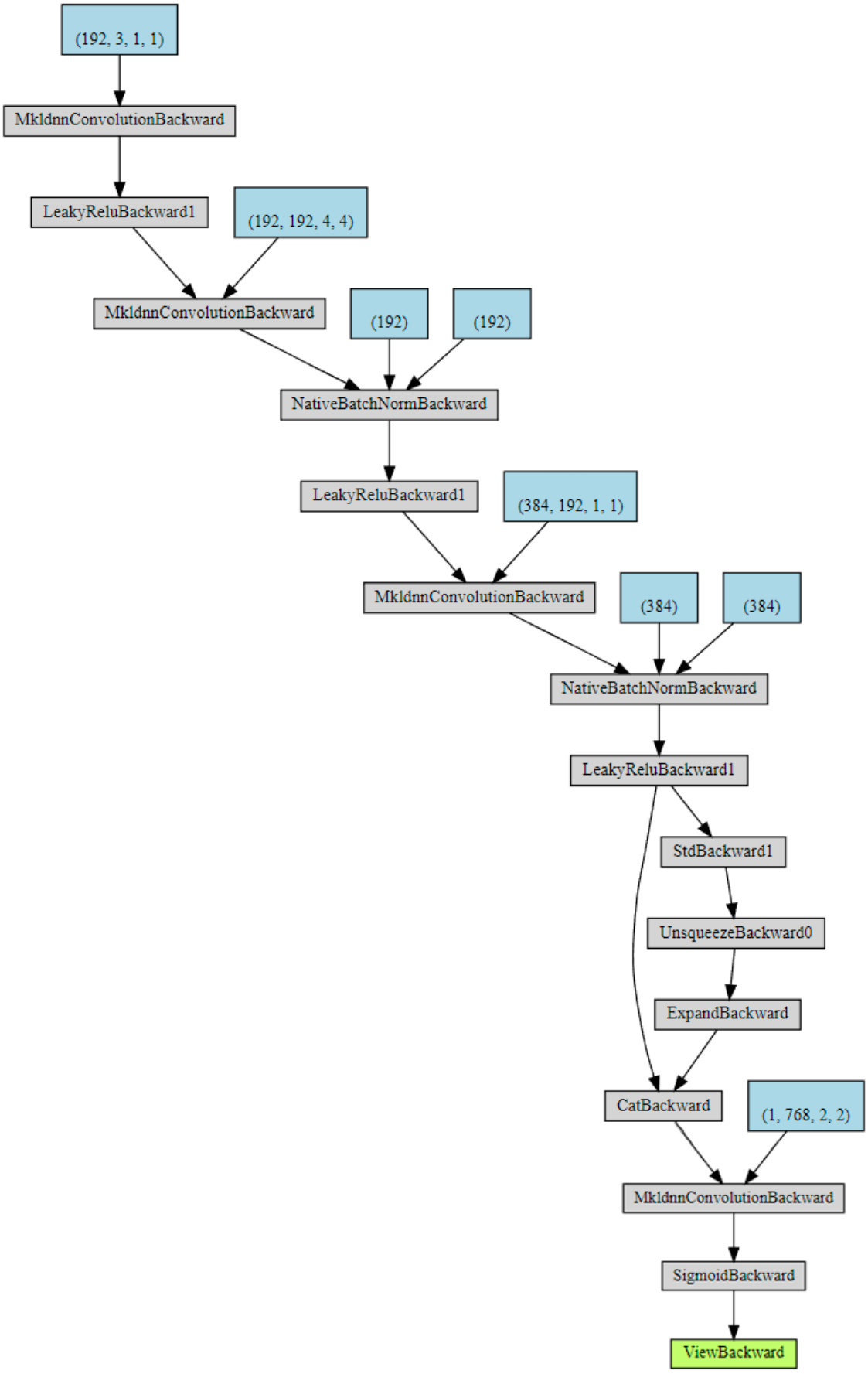

рис. 3.4.Візуалізація створеного дискримінатора

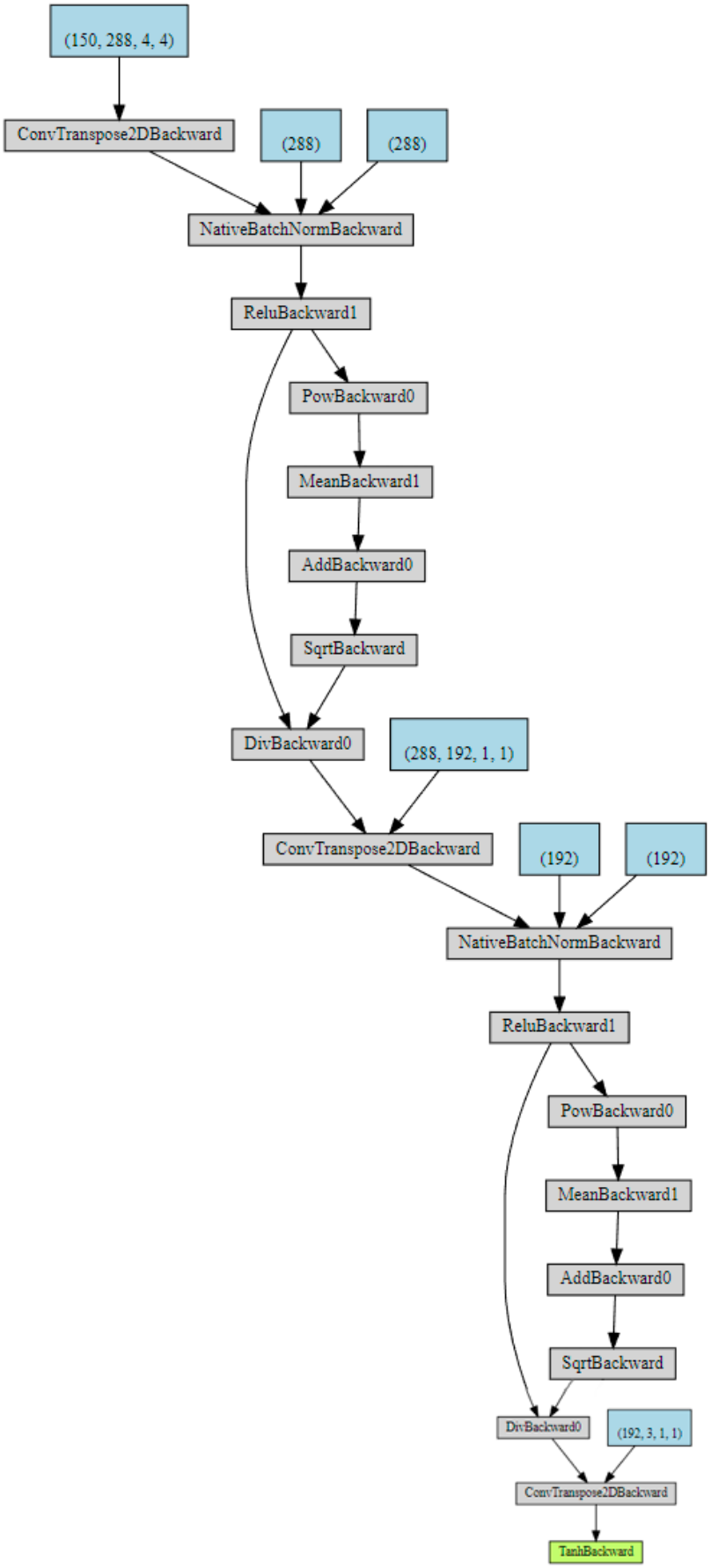

рис. 3.5. Візуалізація створеного генератора

рис. 3.6. Візуалізація випадкових зображень набору даних з навчального набору даних

Із проблем які виникли при навчання генератора це:

- загасаючі градієнти, якщо дискримінатор дуже гарний, то навчання генератора може не працювати. Фактично, оптимальний дискримінатор не надає достатньо інформації для роботи генератора;
- нездатність зійтися в прийнятну точку, тому що генератор складно навчати і тому багато GAN моделей зовсім не здатні зійтися в прийнятну точку;
- колапс моди - режим роботи GAN, при якому генератор видає лише дуже обмежену кількість зображень. Для вирішення цього використовується інтерполяція в прихованому просторі.

Генеративні моделі здатні вивчати низькорозмірний розподіл ймовірностей для зразків із різних класів. Такий розподіл ймовірностей може використовуватися для контрольованого навчання та для генерації синтетичних зразків. Генеративні моделі здатні виконувати інтерполяцію для реальних вибірок вздовж будь-якої осі для генерації неіснуючих керованих вибірок. Наприклад, глибокі генеративні моделі можуть маніпулювати зображеннями людських осіб по осях, таким як вік, стать, колір волосся тощо. Інтерполяція

працює шляхом виконання простої лінійної алгебри у прихованому просторі.

В модульному тестуванні ми отримали наступні результати:

Verified Size 8 x 8 @ Level 1.00009
Verified Size 8 x 8 @ Level 1.10009
Verified Size 8 x 8 @ Level 1.20009
Verified Size 8 x 8 @ Level 1.3000900000000002
Verified Size 8 x 8 @ Level 1.4000900000000003
Verified Size 8 x 8 @ Level 1.5000900000000004
Verified Size 8 x 8 @ Level 1.6000900000000005
Verified Size 8 x 8 @ Level 1.7000900000000005
Verified Size 8 x 8 @ Level 1.8000900000000006
Verified Size 8 x 8 @ Level 1.9000900000000007
Verified Size 16 x 16 @ Level 2.000090000000001
Verified Size 16 x 16 @ Level 2.1000900000000007
Verified Size 16 x 16 @ Level 2.200090000000001
Verified Size 16 x 16 @ Level 2.300090000000001
Verified Size 16 x 16 @ Level 2.4000900000000014
Verified Size 16 x 16 @ Level 2.500090000000001
Verified Size 16 x 16 @ Level 2.6000900000000016
Verified Size 16 x 16 @ Level 2.700090000000001
Verified Size 16 x 16 @ Level 2.8000900000000017
Verified Size 16 x 16 @ Level 2.9000900000000014

Результат тренування генератора по епохам:

E:0, L:4.00009, G Loss:63.12200164794922, D Loss:0.0
E:10, L:4.00009, G Loss:63.3973274230957, D Loss:0.0
E:20, L:4.00009, G Loss:63.29313278198242, D Loss:0.0
E:30, L:4.00009, G Loss:63.2193603515625, D Loss:0.0
E:40, L:4.00009, G Loss:62.89963912963867, D Loss:0.0
E:50, L:4.00009, G Loss:63.07286071777344, D Loss:0.0

E:100, L:4.00009, G Loss:62.993221282958984, D Loss:0.0

E:150, L:4.00009, G Loss:63.12217712402344, D Loss:0.0

E:250, L:4.00009, G Loss:63.12873840332031, D Loss:0.0

E:500, L:4.00009, G Loss:63.261966705322266, D Loss:0.0

E:1000, L:4.00009, G Loss:62.84462356567383, D Loss:0.0

E:1500, L:4.00009, G Loss:62.834388732910156, D Loss:0.0

E:2000, L:4.706452616418172, G Loss:62.77192687988281, D Loss:0.0

E:2550, L:6.000062965300964, G Loss:74.92013549804688, D Loss:0.0

При генерації зображень 64x64 отримали наступний результат:

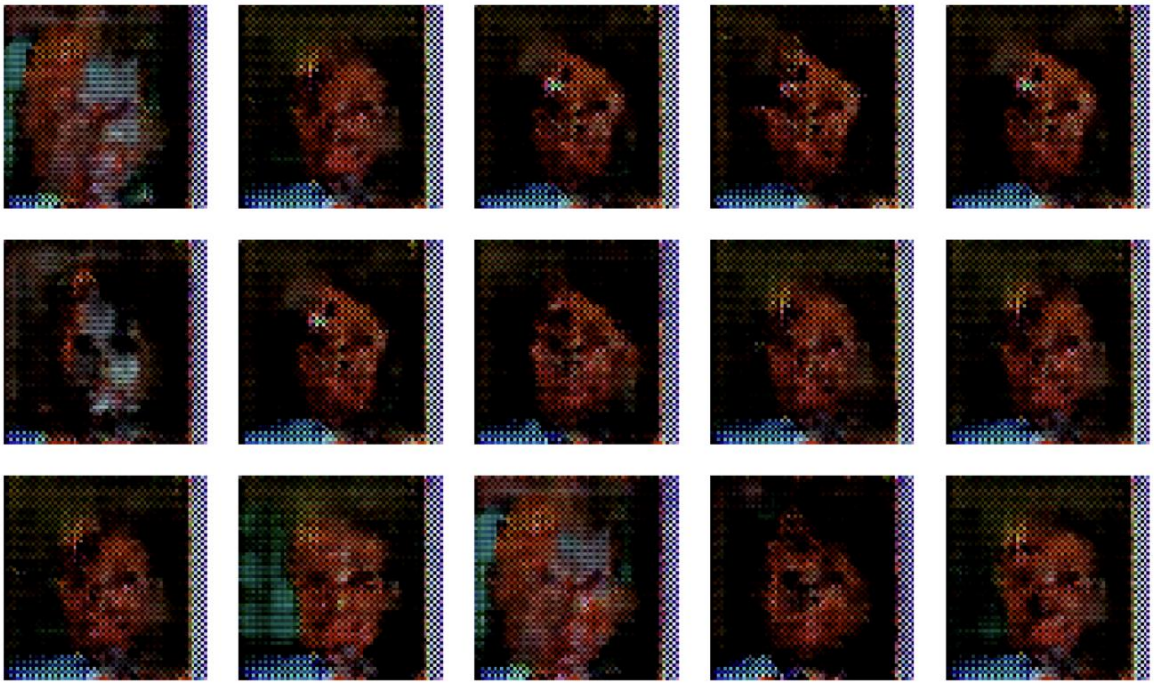

рис 3.7.

Як бачимо навіть при малому розширенні видно контури обличчя людини.

Використання системи генерації зображень дуже різноманітнє:

- ігрова промисловість

У деяких іграх є багато людей і їх доводиться малювати, як і весь навколишній світ. Якщо гра досить масштабна, то там можуть бути тисячі людей і до кожного потрібно придумати свою зовнішність. Генерувавння обличчя сильно полегшить роботу ігрових дизайнерів;

- анонімність у мережі

Використовуючи згенеровану особу, ви можете бути спокійні за авторське право і не зобов'язані будете виплачувати фото моделі дивіденди до кінця життя;

- арт-проєкти

У UI дизайні цей сервіс можна використовувати як аватар для проєкту;

- Комерційний сегмент

Можливість бачити як би людина виглядала якщо була б старша або молодша буде мати попит у редагуванні своїх фото або просто щоб побачити як змінюватиметься його зовнішність у віком;

- наукові дослідження

Використання генерації осіб допоможе в науці, щоб за зовнішністю батьків можна було показати, як приблизно можуть виглядати їхні діти. Якщо є зовнішність всієї родини та родичів, навіть далекий, то це тільки збільшить точність з якою генеруватиметься обличчя. Після штучний інтелект також може довчитися порівнювати згенеровані зображення з фактичним результатом;

- мистецтво

Генерування портретів людей має також художню цінність.

- криміналістика

Згенерований фоторобот допоможе знайти потрібну людину по опису свідка, що полегшить роботу правоохоронних органів.

3.5. Висновки до розділу

Люди, як правило, не замислюються про вплив, який нейронні мережі надають на наше життя, тому що зазвичай ми бачимо результат їхньої роботи, а не "обличчя" нейронної мережі. Можливо, саме тому генератор підроблених фотографій став головною темою обговорення останніх років. Не всі могли здогадатися, що за кілька секунд комп’ютер може згенерувати реалістичну особу неіснуючої людини.

Майже неможливо розпізнати зображення фальшивої людини. Гарно навчана на великій вибірці даних нейронна мережа, над якими працювало

багато людей настільки розвинений, що 90% підробок не розпізнаються звичайною людиною, а 50% не розпізнаються досвідченим фотографом. Послуг із розпізнавання не існує. Іноді нейронна мережа припускається помилок, через що з'являються: неправильно вигнутий візерунок, дивний колір волосся і так далі. Єдине, що вам потрібно зробити, це придивитися уважніше: системи візуальної обробки у людей набагато сильніші, ніж у комп'ютерів, тому підробку можна розпізнати по виявленню.

Але існують і неточності у генерації осіб. Одна з найпоширеніших – симетричні проблеми, зокрема окуляри та сережки. Такі ж нерівні проблеми із зубами теж досить поширені. Шукайте дивні характеристики, такі як пікселі та різці, що повторюються. Накладне волосся, загалом, може здаватися з деяким світінням навколо нього або виглядати занадто прямим і з прожилками, знову ж таки, з видимою асиметрією.

РОЗДІЛ 4. РОЗРОБКА СТАРТАП-ПРОЄКТУ

За останні роки довели, що стартапи сильніші, ніж більшість думала. Гнучкий та інноваційний підхід був ключовим, коли світ зіштовхнувся з такими проблемами, як віддалена робота, масштабні зміни в галузі та ринку, а також абсолютно нова реальність.

Можливівсть стартапів швидко змінюватися та адаптуватися і є ключовою властивістю даного виду бізнесу, не кажучи вже що багато стартапів продовжували рости та розширювати свої команди.

Стартап це бізнес-структура, що працює на основі інновацій, створена для вирішення проблеми шляхом надання нової пропозиції в умовах надзвичайної невизначеності.

Власне, стартап являється бізнесом, що:

– швидко росте;
– порушує ринок або галузь;
– займається вирішенням проблеми;
– працює в умовах надзвичайної невизначеності.

Багато підприємців та відомих бізнес-магнатів визначають стартап як культуру та менталітет для побудови бізнесу на основі інноваційної ідеї та вирішення критичних проблем.

Одне, те що відрізняє стартапи від інших компаній так це зв’язок між їхнім продуктом та його попитом. Різні стартапи мають продукти, орієнтовані на невикористаний ринок. Новички підприємці знають ідеальну стратегію, щоб створити такий продукт, що хоче ринок, а також охопити і обслуговувати їх усіх. Це є причиною швидке зростання.

4.1. Опис ідеї проєкту

В межах підпункту проаналізовано та подано у вигляді таблиць:

– зміст ідеї;
– можливі напрямки застосування;
– основні вигоди, який може отримати користувач товару;
– те, чим відрізняється від існуючих аналогів та замінників.

Перші три пункти подані як таблиця (таблиця 4.1) та дає цілісне уявлення про зміст ідеї і можливі базові потенційні ринки, в межах яких необхідно шукати групи потенційних клієнтів.

Таблиця 4.1 — Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Дана система дозволяє розвязати проблему генерування зображень осіб.	Видача рекомендацій користувачу	Збільшення прибутку шляхом покращення якості генерування
	Генерування динамічних вкладень графу взаємодій	Збільшення об’єму клієнської бази

Аналіз потенційних технічних та економічних переваг ідеї порівняно із пропозиціями конкурентів передбачає:

– визначення переліку техніко-економічних властивостей та характеристик ідеї;

– визначення попереднього кола конкурентів або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проєкту та проєктів-конкурентів відповідно до визначеного вище переліку;

– проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають:

– гірші значення (W, слабкі);

– аналогічні (N, нейтральні) значення;

– кращі значення (S, сильні) (табл. 4.2).

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Технікоекономічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	BoredHumans	Fotor			
1.	Форма виконання	Надавання послуг	Надавання послуг	Надавання послуг		+	
2.	Собівартість	Низька	Висока	Висока			+
3.	Функціонал	Широкий	Широкий	Широкий	+		

Перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2. Технологічний аудит ідеї проєкту

В межах даного підрозділу було проведено аудит технології, за допомогою якої можна реалізувати ідею проєкту (технології створення товару). Визначення технологічної здійсненності ідеї проєкту передбачає аналіз таких складових (таблиця 4.3):

1. Якою технологією виготовлено товар згідно ідеї проєкту?
2. Існують такі технології, чи їх потрібно розробити/доробити?
3. Доступні такі технології авторам проєкту?

Таблиця 4.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Створення системи генерування рекомендацій користувачу за даними покупок користувачів	Використання мови програмування Python	Наявна	Доступна
		Використання мови програмування Java	Наявна	Доступна
		Використання мови C++	Наявна	Доступна
Обрана технологія реалізації ідеї проєкту: мова програмування Python.				

За результатами аналізу таблиці робимо висновок щодо можливості технологічної реалізації проєкту. За технологічним шляхом реалізації проєкту обрано такі технології, як Python через те, що вона доступна та має найкращі бібліотеки для розробки нейронних мереж.

Визначення ринкових можливостей, що можна використати під час ринкового впровадження проєкту та ринкових загроз, що можуть перешкодити реалізації проєкту, дозволяє спланувати в якому напрямку розвитку проєкту із урахуванням стану ринкового середовища та потреб потенційних клієнтів та пропозицій проєктів-конкурентів.

В першу чергу було проведено аналіз попиту: наявність попиту, обсяг та динаміка розвитку ринку (таблиця 4.4).

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	50
2	Загальний обсяг продаж, грн/ум.од	10000
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі, %	10%

За результатами аналізу таблиці 4.4 зроблено висновок, що ринок є привабливим для входження.

Далі були визначені потенційні групи клієнтів та їх характеристики та сформовано орієнтовний перелік щодо вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проєкту

Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару

Точна та швидка генерація рекомендацій цільовій аудиторії	Власник бізнесу	Велика кількість даних	Простота використання, висока точність
Підбір рекомендацій	Кінцевий користувач	Цікавить простота у використанні, низька ціна підтримки системи	Швидкість створення, низька ціна

Після визначення потенційних груп клієнтів проведено аналіз ринкового середовища: складено таблиці факторів, що сприяють ринковому впровадженню проєкту, та факторів, що йому перешкоджають (табл. 4.6, 4.7).

Таблиця 4.6 – Фактори загроз

№ п/ п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Вихід на ринок продуктів з кращими характеристиками	Передбачити додаткові переваги власного програмного продукту (ПП) для того, щоб повідомити про них саме після виходу на ринок конкурентів. Вдосконалення технічних моментів власного продукту. Обрати нову цільову аудиторію і зосередитися на ній: зниження цін.
2	Зміна потреб користу	Користувачам необхідний сервіс з більшим/новим	Розроблення гнучкої архітектури програмного забезпечення для легшого впровадження нового функціоналу

	вачів	функціоналом.	

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Гнучкі ціни	Зменшення ціни товару задля збільшення попиту	Введення власних гнучких цін
2	Поява нових методів квантування зображення	З’являться нові методи, що будуть швидше, та більш точно квантувати зображення	Покращити ПП додаванням нового функціоналу, розширення можливостей

Далі було проведено аналіз пропозиції: визначити загальні риси конкуренції на ринку (таблиця 4.8).

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції-чиста	Існує величезна кількість конкурентів на ринку.	Якісно провести рекламу.

2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти з інших країн	Створити основу ПП таким чином, щоб можна було легко переробити даний ПП для використання у галузях інших країн.
3. За галузевою ознакою - міжгалузева	Продукт може використовуватись для різних галузей	Постійне вдосконалення продукту, що не має прив'язки до сфери
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між видами ПП, їх особливостями.	Створити ПП, враховуючи недозображення конкурентів
5. За характером конкурентних переваг - неціновa	Вдосконалення технології створення ПП, щоб собівартість була нижчою	Удосконалення моделі. Використання більш дешевих технологій для розробки, ніж використовують конкуренти, але тільки якщо ці технології відповідають необхідним вимогам якості.
6. За інтенсивністю - не марочна	Бренд присутній, але його роль незначна	Реклама, участь у конференціях, семінарах.

Проведено аналіз конкуренції у галузі за моделлю М. Портера (табл. 4.9).

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товаризамінники

	Навести перелік прямих конкурентів	Визначити бар'єри входження в ринок	Визна чити фактори сили постахальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
	Fotor	Наявність вже існуючих рішень	-	Контроль якості продукту	Наявність більш широкого функціоналу, зручнішого інтерфейсу та авторитет
Висновки:	Досить інтенсивна конкурентна боротьба з іншими гравцями	Є можливості виходу на ринок, але є і конкуренти.	-	Клієнти диктують умови роботи на ринку: зручний інтерфейс	Необхідно випускати ПЗ не гірше, ніж у конкурентів та розширяти функціонал.

За результатами аналізу було зроблено висновок про можливість роботи на ринку з огляду на конкурентну ситуацію.

Цей висновок врахований при формулюванні переліку факторів конкурентоспроможності у наступному пункті. На основі аналізу конкуренції, проведеного в таблиці, а також із урахуванням характеристик ідеї проєкту (табл. 4.2), вимог споживачів до товару (табл. 4.5) та факторів маркетингового середовища (табл. 4.6, 4.7) визначили та обґрунтували перелік факторів конкурентоспроможності.

Аналіз оформлено у (табл. 4.10).

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Ціна	Один із факторів для вибору продукту клієнтом.
2	Якість	Один із факторів для вибору продукту клієнтом.
3	Зручність роботи з програмою	Дозволяє користувачу легко працювати з програмою

За визначеними факторами конкурентоспроможності (табл. 4.10) проведено аналіз сильних та слабких сторін проєкту (табл. 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проєкту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			–3	–2	–1	0	+1	+2	+3
1	Ціна	15					*		
2	Якість	10			*				
3	Зручність роботи з програмою	15					*		

Фінальним етапом ринкового аналізу можливостей впровадження проєкту є складання SWOT-аналізу (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 4.11).

Перелік ринкових загроз та ринкових можливостей було складено на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками впливу факторів, та на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 4.12 – SWOT-аналіз стартап-проєкту

Сильні сторони: Якість Простота використання Висока швидкодія	Слабкі сторони: Дуже насичений ринок, мала кількість функціоналу, відсутня кросплатформеність.
Можливості: насичення ринку новим підходом до прогнозування; різноманітна клієнтура, вдосконалення системи	Загрози: Конкуренція

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки (перелік заходів) для виведення проєкту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проєкти конкурентів, що можуть бути виведені на ринок.

Визначені альтернативи були проаналізовані з точки зору строків та ймовірності отримання ресурсів (таблиця 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	PR, просування бренду	50%	6 місяців
2	Перехід на безкоштовне розповсюдження	75%	3 місяців
3	Партнерство для об’єднання продукції	65%	2 місяці

Після аналізу було обрано альтернативу №2.

4.3. Аналіз ринкової стратегії проєкту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: було проведено опис цільових груп потенційних споживачів (таблиця 4.14).

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Підприємці	Висока	Високий	Сильна	Просто
2	Великі компанії	Середня: велика конкуренція і можливість власних веб-відділів	Високий	Сильна	Складно
3	Маленькі компанії.	Низька	Низький	Слабка	Середня
Які цільові групи обрано:1,2,3					

За результатами аналізу потенційних груп споживачів було обрано цільові групу, для яких буде запропоновано даний товар, та визначено стратегію охоплення ринку - стратегію диференційованого маркетингу, тобто передбачити можливість того, щоб компанія працювала з декількома сегментами.

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Постійне оновлення і покращення продукту	Ринкове позиціонування на індивіуальних користувачів	Швидкодія, якість продукту	Концентрований маркетинг

Наступним кроком обрано стратегію конкурентної поведінки (таблиця 4.16).

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні.	Компанія буде шукати нових споживачів та забирати існуючих у конкурентів	Буде копіювати, удосконалювати та створювати свої унікальні пропозиції	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (табл. 4.5), а також в залежності від обраної базової стратегії розвитку (табл. 4.15) та стратегії конкурентної поведінки (таблиця 4.16) розроблено стратегію позиціонування (таблиця 4.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають

ідентифікувати торгівельну марку/проєкт.

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартаппроєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
1	Легкість розуміння, зручний інтерфейс, надійний, швидкий, точний та достовірний ПП для генерації рекомендацій.	Стратегія диференціації	Позиція на основі порівняння фірми з товарами конкурентів; Відмінні особливості споживача	Економія часу; Зручність застосування; Практичність та точність результату

Результатом виконання підрозділу стала узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап-компанії на ринку.

4.4. Розроблення маркетингової програми стартап-проєкту

Сформовано маркетингову концепцію товару, який отримає споживач. Для цього підсумовано результати попереднього аналізу конкурентоспроможності товару (таблиця 4.18). Концепція товару – письмовий опис фізичних та інших характеристик товару, які сприймаються споживачем, і набору вигод, які він обіцяє певній групі споживачів.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)

1	Швидкість отримання результату	Швидка видача рекомендацій наступної пропозиці	Необхідно покращити швидкість навчання моделі для видачі рекомендацій наступної пропозиці
2	Зручність застосування	Нативна підтримка мови програмування Python	Розробка зручного прикладного програмного інтерфейсу
3	Практичність та точність результату	Користувач отримує точні (з малою похибкою розбіжності) результати рекомендацій.	Користувач на виході роботи ПП отримує модель та прогноз, котрі відповідають необхідним показникам варіативності та точності.

Розроблено трирівневу маркетингову модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 4.19).

1-й рівень. При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов’язаний з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

2-й рівень. Цей рівень являє рішення того, як буде реалізований товар в реальному/ включає в себе якість, властивості, дизайн, упаковку, ціну.

3-й рівень Товар з підкріпленням (супроводом) – додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін).

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Генерація палітри кольорів зображення за допомогою інтелектуальних систем

II. Товар реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	1. Індивідуальний підхід. 2. Низька ціна. 3. Простота у використанні.	1.Нм 2.Нм 3.Нм	1.Технологічна 2.Економічна 3.Технологічна
	Якість: тестування фірмами аудиторами		
	Пакування: відсутнє		
	Марка: ROSO		

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар, бо остаточне визначення ціни відбувається під час фінансово-економічного аналізу проєкту, яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (таблиця 4.20). Аналіз проведено експертним методом.

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	15000$	25000$	У всіх трьох груп високий рівень доходів	6000$--

Наступним кроком є визначення оптимальної системи збуту, в межах якого було прийняте рішення (таблиця 4.21).

Таблиця 4.21 – Формування системи збуту

№	Специфіка	Функції збуту, які	Глибина	Оптимальна

п/п	закупівельної поведінки цільових клієнтів	має виконувати постачальник товару	каналу збуту	система збуту
1	Канал нульового рівня	Продаж	0(напряму)	Власна

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 4.22).

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/ п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Інтеграція API у клієнтській системі	Інтернет	Низька ціна, простота використання, універсальність	Показати переваги рішення над конкурентами, виділити ключові особливості	Створення сайту продукту, розповсюдження інформації про продукт на спеціалізованих ресурсах.

Було визначено, що придбання продукту буде проводитись через мережу Інтернет або при безпосередньому спілкуванні із представниками компанії. Розповсюдження інформації про продукт буде проводитись виключно через Інтернет, адже аудиторія даного продукту активно користується всесвітньою мережею.

Результатом підрозділу стала маркетингова програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей

ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого впроваджено проєкт, відповідну обрану альтернативу ринкової поведінки.

4.5. Висновки

В даному розділі проведено аналіз для впровадження розробленої системи в якості проєкту. Досліджено аналогічні системи, виявлено сильні та слабкі сторони системи в порівнянні з ними. Також досліджено шляхи розповсюдження продукту та його ймовірну аудиторію, очікувавний дохід та ймовірну ціну продукту, що розробляється.

Було проведено аналіз потенційних ризиків і можливостей, а також розраховані основні фінансово-економічні показники проєкту. Отримані результати кажуть про те, що реалізація має сенс. Було визначено сильні сторони проєкту: зручність у використанні, ціна утримання, якість.. Серед слабких варто виділити повільне навчання та необхідність хороших графічних процесорівв для навчання. Варто відмітити можливість реклами продукту на спеціалізованих ресурсах із зазначенням сильних сторін проєкту.

ВИСНОВКИ

У даній магістерській дисертації було створено систему генерування людей за допомогою зростаючої генеративно-змагальної нейронної мережі, а також запропоновано поєднати в собі методи знижувальної дискретизації для навчання на непропорційно низькому підмножині прикладів більшості класів, нормалізації ваг, попіксельної нормалізації щоб мати можливість генерувати зображення високої якості та використати в якості одного із шарів mini-batch стандартного відхилення для збільшення варіативності генерованих зображень. Загалом створена мережа генерує зображення, яку можна донавчити щоб підвищити якість зображення. Основними критеріями успіху є навчання програми на великої кількості даних з різною кількістю даних різних класів та генерація тестових зображень. Запропонована нейронна має навіть при низькій кількості зображень одного класу стабільний результат.

ПЕРЕЛІК ПОСИЛАНЬ

1. A Gentle Introduction to Generative Adversarial Networks[Електронний ресурс] – 2019 – Режим доступу до ресурсу: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

2. What is Generative Adversarial Networks GAN? [Електронний ресурс] – 2018 – Режим доступу до ресурсу: https://jonathan- hui.medium.com/gan-whats-generative-adversarial-networks-and- its-application-f39ed278ef09

3. Sukarna Barua, A Fully Connected and Convolutional Net Architecture for GANs[Електронний ресурс] / Sukarna Barua, Sarah Monazam Erfani, James Bailey – 12 с. – [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1905.02417.pdf

4. Introduction to Generative Adversarial Networks [Електронний ресурс] – 2021 – Режим доступу до ресурсу: https://learnopencv.com/introduction-to-generative-adversarial-networks/

5. ProGAN: How NVIDIA Generated Images of Unprecedented Quality? [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/progan-how-nvidia-generated-images-of-unprecedented-quality-51c98ec2cbd2

6. A Gentle Introduction to the Progressive Growing GAN [Електронний ресурс] – Режим доступу до ресурсу: https://machinelearningmastery.com/introduction-to-progressive-growing-generative-adversarial-networks/

ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
import torch
import torchvision
import numpy as np
from matplotlib import pyplot as plt
from math import ceil
from google.colab import drive
from torchviz import make_dot
from math import floor
from torch.utils.tensorboard import SummaryWriter
import os


%matplotlib inline


#Визначаємо максимальне розширення нашого зображення кратне 2

max_levels = 6  #1:4x4,2:8x8,3:16x16,4:32x32,5:64x64,6:128x128,7:256x256
image_size = 128  # 4,    8 ,   16,     32,     64,      128,     256

#За можливості активуємо функцію паралельних обчислень використовуюючи графічні процесори #Nvidia, щоб мати можливість можливість на свій розсуд організовувати доступ до набору #інструкцій графічного або тензорного прискорювача та керувати його пам'яттю і без яких час #навчання збільшиться
device = torch.device('cpu')

if torch.cuda.is_available():

    device = torch.device('cuda')
```

#Підключаємося до нашого гугл диску на якому знаходяться дані, визначаємо папку з даними
#Встановлюємо розмір даних в одній ітерації та кількість процесорів що будуть використовуватися
#Посилання на дані: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

```
drive.mount('/content/drive')
dataroot = './celeba_dataset'
batch_size = 48
workers = 8

# Стиснення зображень до -1 і +1
class SquashTransform:

    def __call__(self, inputs):
        return 2 * inputs - 1



# Створюємо набір даних
dataset = torchvision.datasets.ImageFolder(
    root=dataroot,
    transform=torchvision.transforms.Compose(
        [
            torchvision.transforms.Resize(
                image_size),
            torchvision.transforms.CenterCrop(
                image_size),
            torchvision.transforms.ToTensor(),
            SquashTransform()
        ]
    )
```

```
)

# Створюємо завантажувач даних
dataloader = torch.utils.data.DataLoader(
    dataset,
    batch_size=batch_size,
    shuffle=True,
    num_workers=workers
)

# Візьмемо кілька навчальних зображень
real_batch = next(iter(dataloader))

plt.figure(figsize=(20, 20))

plt.axis("off")

plt.title("Training Images")

plt.imshow(
    np.transpose(
        torchvision.utils.make_grid(
            real_batch[0].to(device),
            padding=2,
            normalize=True,
            nrow=10
        ).cpu(),
        (1,2,0)
    )
)
```

```
# Візуалазуємо набір даних
plt.show();

# Нормалізація вектора попіксельно в генераторі
class PixelNormLayer(torch.nn.Module):

    def __init__(self):
        super(PixelNormLayer, self).__init__()

    def forward(self, x):
        return x / torch.sqrt(
            torch.mean(x ** 2, dim=1, keepdim=True) + 1e-8
        )

torch.mean(
    torch.tensor(
        torch.randn((5, 2))
    ),
    dim=0
)
# tensor([ 0.4716, -0.1336])

torch.mean(
    torch.tensor(
        torch.randn((5, 2))
    ),
    dim=1
).shape
# torch.Size([5])
```

```
torch.mean(
    torch.tensor(
        torch.randn((5, 2))
    ),
    dim=1,
    keepdim=True
).shape
# torch.Size([5, 1])
sample = torch.tensor(
    torch.randn((150, 1))
)

l = PixelNormLayer()
norm_sample = l(sample)

plt.figure(figsize=(18, 12))
plt.plot(sample, label='X')
plt.plot(norm_sample, label='Pixelwise Norm')
plt.legend()
plt.show();

# Нормалізація ваги
class WScaleLayer(torch.nn.Module):

    def __init__(self, incoming, gain=2):

        super(WScaleLayer, self).__init__()

        self.gain = gain
        self.scale = (self.gain / incoming.weight[0].numel()) ** 0.5
```

```
    def forward(self, input):
        return input * self.scale

# Генератор згорткового блоку
def GConvBlock(
    in_channels,
    out_channels,
    kernel_size,
    stride,
    padding,
    bias
):

    return torch.nn.Sequential(
            torch.nn.ConvTranspose2d(
                in_channels=in_channels,
                out_channels=out_channels,
                kernel_size=kernel_size,
                stride=stride,
                padding=padding,
                bias=bias
            ),
            torch.nn.BatchNorm2d(
                num_features=out_channels
            ),
            torch.nn.ReLU(
                inplace=True
            ),
            PixelNormLayer()
```

```
    )

# Дискримінатор згорткового блоку
def DConvBlock(
    in_channels,
    out_channels,
    kernel_size,
    stride,
    padding,
    bias
):

    return torch.nn.Sequential(
            torch.nn.Conv2d(
                in_channels=in_channels,
                out_channels=out_channels,
                kernel_size=kernel_size,
                stride=stride,
                padding=padding,
                bias=bias
            ),
            torch.nn.BatchNorm2d(
                num_features=out_channels
            ),
            torch.nn.LeakyReLU(
                negative_slope=0.2,
                inplace=True
            )
    )
```

```
# Визначаємо глобальні змінні

# Кількість каналів у навчальних зображеннях.
# Для кольорових зображень це 3
nc = 3

# Розмір прихованого вектора z (тобто розмір входу генератора)
nz = 150

# Розмір карт функцій у генераторі
ngf = 6

# Розмір карт функцій у дискримінаторі
ndf = 6

# Конструкція генератора
class Generator(torch.nn.Module):

    def __init__(self):

        super(Generator, self).__init__()

        self.blocks = torch.nn.ModuleList()
        self.toRGBs = torch.nn.ModuleList()

        # перший згортковий блок
        self.blocks.append(
            torch.nn.Sequential(
                GConvBlock(
                    in_channels=nz,
```

```
                out_channels=ngf * 48,
                kernel_size=4,
                stride=1,
                padding=0,
                bias=False
            ),
            GConvBlock(
                in_channels=ngf * 48,
                out_channels=ngf * 32,
                kernel_size=1,
                stride=1,
                padding=0,
                bias=False
            )
        )
    )

# збільшення згорткових
# блоків: 4 * 4, 8 * 8, 16 * 16, 32 * 32, 64 * 64
    i = 5
    j = 4

    while j <= image_size:

        in_channels = int(
            ngf * ( 2 ** i )
        )  # 32 => 16 => 8 => 4 => 2
        out_channels = int(
            ngf * ( 2 ** (i - 1) )
        )  # 16 => 8  => 4 => 2 => 1
```

```
self.blocks.append(
    torch.nn.Sequential(
        GConvBlock(
            in_channels=in_channels,
            out_channels=in_channels,
            kernel_size=4,
            stride=2,
            padding=1,
            bias=False
        ),
        GConvBlock(
            in_channels=in_channels,
            out_channels=out_channels,
            kernel_size=1,
            stride=1,
            padding=0,
            bias=False
        )
    )
)

self.toRGBs.append(
    torch.nn.Sequential(
        torch.nn.ConvTranspose2d(
            in_channels=in_channels,
            out_channels=nc,
            kernel_size=1,
            stride=1,
            padding=0,
```

```
                bias=False
            ),
            torch.nn.Tanh()
        )
    )

    i = i - 1
    j = j * 2

# Ініціалізація ваг
for m in self.modules():
    if isinstance(m, torch.nn.Conv2d):
        torch.nn.init.kaiming_normal_(
            m.weight)
        if m.bias is not None:
            torch.nn.init.constant_(
                m.bias, 0)
    elif isinstance(m, torch.nn.BatchNorm2d):
        torch.nn.init.constant_(
            m.weight, 1)
        torch.nn.init.constant_(m.bias, 0)

# Рівень = 1: 4x4, 2: 8x8, 3: 16x16, 4: 32x32, 5:64x64
def forward(self, inputs, level):

    alpha = False  # level - int(level)
    level = min(int(ceil(level)), max_levels)

    feature_map = self.blocks[0](inputs)
```

```
        for i in range(1, level):

            feature_map = self.blocks[i](
                feature_map)

            # альфа-перехід
            if level > 1 and i == level-1 and alpha != 0:

# підвищення дискретизації в 2 рази
# попередній_вихід = torch.nn.functional.upsample(
# карта_об'єктів,
# scale_factor=2
        prev_output = self.toRGBs[i](
                    feature_map)

        output = self.toRGBs[level-1](feature_map)

        if alpha != 0:
            return output * alpha + prev_output * ( 1 - alpha )
        else:
            return output



# Дизайн дискримінатора
class Discriminator(torch.nn.Module):

    def __init__(self):

        super(Discriminator, self).__init__()
```

```
        self.fromRGBs = torch.nn.ModuleList()
        self.blocks = torch.nn.ModuleList()

        # протилежні зростаючі згорткові
    # блоки: 4 * 4, 8 * 8, 16 * 16, 32 * 32, 64 * 64
        i = 0
        j = 4

        while j <= image_size:

            in_channels = int(
                ndf * ( 2 ** i )
            )  # 1 => 2 => 4 => 8  => 16
            out_channels =int(
                ndf * ( 2 ** (i + 1) )
            )  # 2 => 4 => 8 => 16 => 32

            self.fromRGBs.append(
                torch.nn.Sequential(
                    torch.nn.Conv2d(
                        in_channels=nc,
                        out_channels=in_channels,
                        kernel_size=1,
                        stride=1,
                        padding=0,
                        bias=False
                    ),
                    torch.nn.LeakyReLU(
                        negative_slope=0.2,
                        inplace=True
```

```
            )
        )
    )

    self.blocks.append(
        torch.nn.Sequential(
            DConvBlock(
                in_channels=in_channels,
                out_channels=in_channels,
                kernel_size=4,
                stride=2,
                padding=1,
                bias=False
            ),
            DConvBlock(
                in_channels=in_channels,
                out_channels=out_channels,
                kernel_size=1,
                stride=1,
                padding=0,
                bias=False
            )
        )
    )

    i = i + 1
    j = j * 2

# додавання останнього блоку
self.blocks.append(
```

```
            torch.nn.Sequential(
                torch.nn.Conv2d(
                    in_channels=ndf * ( 2 ** (i + 1) ),
                    out_channels=1,
                    kernel_size=2,
                    stride=1,
                    padding=0,
                    bias=False
                ),
                torch.nn.Sigmoid()
            )
        )

        # Ініціалізація ваг
        for m in self.modules():
            if isinstance(m, torch.nn.Conv2d):
                torch.nn.init.kaiming_normal_(m.weight)
                if m.bias is not None:
                    torch.nn.init.constant_(m.bias, 0)
            elif isinstance(m, torch.nn.BatchNorm2d):
                torch.nn.init.constant_(m.weight, 1)
                torch.nn.init.constant_(m.bias, 0)

    # Рівень = 1: 4x4, 2: 8x8, 3: 16x16, 4: 32x32, 5:64x64
    def forward(self, inputs, level):

        alpha = False  # level - int(level)
        level = min(int(ceil(level)), max_levels)

        feature_map = self.fromRGBs[max_levels - level](inputs)
```

```
feature_map = self.blocks[max_levels - level](feature_map)

# альфа-перехід
if level > 1 and alpha != 0:

    # зменшення дискретизації 0.5x
    prev_input = torch.nn.functional.avg_pool2d(
        inputs,
        kernel_size=2,
        stride=2
    )

    prev_feature_map = self.fromRGBs[max_levels - level + 1](prev_input)

    # альфа-злиття
    feature_map = alpha * feature_map + ( 1 - alpha ) * prev_feature_map

for i in range(max_levels - level + 1, max_levels):

    feature_map = self.blocks[i](feature_map)

# стандартне відхилення на міні-партію
stdv = torch.std(feature_map, dim=0)

# об’єднати карту функцій і стандартне відхилення
y = torch.cat(
    (
        feature_map,
        stdv.unsqueeze(0).expand_as(
            feature_map
```

```
            )
        ),
        dim=1
    )

    # проходження через останній шар
    ret = self.blocks[-1](y).view(-1, 1)

    return ret

# Візуалізація мережі
netD = Discriminator()
out = netD(torch.zeros(1, 3, 4, 4), level=1)
make_dot(out)

netG = Generator()
out = netG(torch.zeros(1, nz, 1, 1), level=1)
make_dot(out)

# Модульне тестування
for i in np.arange(1.00009, max_levels+0.2, 0.1):

    rounded_level = min(int(ceil(i)), max_levels)

    out = netG(torch.zeros(batch_size, nz, 1, 1), level=i)

    assert out.shape == (batch_size, 3, 4 * (2 ** (rounded_level-
1)), 4 * (2 ** (rounded_level-1))), "Вихідний розмір
генератора={}".format(out.shape)
```

```
    valid = netD(out, level=i)

    assert valid.shape == (batch_size, 1), "Вихідний розмір дискримінатора ={}".format(valid.shape)

    print(
        'Перевірений розмір {} x {} \t @ Рівень {}'.format(
            4 * ( 2 ** (rounded_level-1)),
            4 * ( 2 ** (rounded_level-1)),
            i
        )
    )

# Перетворення моделі під графічні процесори
netD = netD.to(device)
netG = netG.to(device)

# Оптимізація
# Встановлення Адам оптимізації для генератора і дискримінатора
optimizerD = torch.optim.RMSprop(
    netD.parameters(),
    lr=0.0001
)

optimizerG = torch.optim.RMSprop(
    netG.parameters(),
    lr=0.0001
)

criterion = torch.nn.BCELoss()
```

```
fixed_noise = torch.randn(
    25, nz, 1, 1
).to(device)

real_labels = torch.ones(batch_size, 1).to(device)
fake_labels = torch.zeros(batch_size, 1).to(device)

# Навчання дискримінатора
def trainD(images, level):

    real_images = images.to(device)

    fake_images = netG(
        torch.randn(
            batch_size, nz, 1, 1
        ).to(device),
        level
    )

    optimizerD.zero_grad()

    real_outputs = netD(real_images, level)
    fake_outputs = netD(fake_images, level)

    d_x = criterion(real_outputs, real_labels)
    d_g_z = criterion(fake_outputs, fake_labels)

    d_x.backward()
    d_g_z.backward()
```

```
    optimizerD.step()

    loss = d_x + d_g_z

    return loss

# Навчання генератора
def trainG(level):
    z = torch.randn(
        batch_size, nz, 1, 1
    ).to(device)

    netG.zero_grad()

    outputs = netD(
        netG(z, level),
        level
    )

    loss = criterion(outputs, real_labels)

    loss.backward()

    optimizerG.step()

    return loss

# Контроль прогресивного росту
# стандартизація до діапазону [0 - 1], потім масштабувати до діапазону [1, max_levels]def updateLevel(epoch, step, total_epochs, total_steps, max_levels):
```

```
    total_spectrum = total_epochs * total_steps

    current_point = epoch * total_steps + step

    return min(
        max(
            (current_point / total_spectrum ) * max_levels, 4), max_levels)

ret = []
res = []

for i in range(50):
    for j in range(20):
        level = updateLevel(i, j, 50, 20, max_levels)
        ret.append(level)
        res.append(2 ** (ceil(level + 1.001)))

# Зменшення роздільної здатності пакетного зображення
def downsampleMiniBatch(images, level):

    level = min(int(ceil(level)), max_levels)

    d = 4 * (2 ** (level - 1))

    return torch.nn.functional.adaptive_avg_pool2d(
                images,
                output_size=d
            )
```

```
# Відображення кількох навчальних зображень із роздільною здатністю 4x4
real_batch = next(iter(dataloader))

resized_batch = downsampleMiniBatch(real_batch[0], 1)

plt.figure(figsize=(20, 20))

plt.axis("off")

plt.title("Training Images")

plt.imshow(
    np.transpose(
        torchvision.utils.make_grid(
            resized_batch,
            padding=2,
            normalize=True,
            nrow=10
        ),
        (1,2,0)
    )
)

plt.show();

# Відображення кількох навчальних зображень із роздільною здатністю 16x16
real_batch = next(iter(dataloader))

resized_batch = downsampleMiniBatch(real_batch[0], 3)
```

```
plt.figure(figsize=(20, 20))

plt.axis("off")

plt.title("Training Images")

plt.imshow(
    np.transpose(
        torchvision.utils.make_grid(
            resized_batch,
            padding=2,
            normalize=True,
            nrow=10
        ),
        (1,2,0)
    )
)

plt.show();

 # Навчання мережі
# За замовчуванням запис буде виводитися в каталог ./runs/
tb_writer = SummaryWriter()



# контрольні точки
if os.path.exists('section-5-optim-d.pytorch'):
    optimizerD.load_state_dict(torch.load('section-5-optim-d.pytorch'))

if os.path.exists('section-5-optim-g.pytorch'):
```

```
    optimizerG.load_state_dict(torch.load('section-5-optim-g.pytorch'))

if os.path.exists('section-5-network-d.pytorch'):
    netD.load_state_dict(torch.load('section-5-network-d.pytorch'))

if os.path.exists('section-5-network-g.pytorch'):
    netG.load_state_dict(torch.load('section-5-network-g.pytorch'))



num_epochs = 2550


for epoch in range(num_epochs):

    d_loss = 0
    g_loss = 0

    for i, (images, _) in enumerate(dataloader):

        if i == num_steps:
            break

        level = updateLevel(epoch, i, num_epochs, num_steps, max_levels) + 0.00009

        for k in range(1):

            images = downsampleMiniBatch(images, level)

            d_loss += trainD(
                images,
                level
```

```
        )

    g_loss += trainG(level)


# Логи та збереження контрольних точок тільки кожні X епох
if epoch % 10 == 0:

    # G & D Loss
    print(
        "E:{}, L:{}, G Loss:{}, D Loss:{}".format(
            epoch,
            level,
            g_loss / num_steps,
            d_loss / num_steps
        )
    )

    # контрольні точки
    torch.save(optimizerD.state_dict(), 'section-5-optim-d.pytorch')
    torch.save(optimizerG.state_dict(), 'section-5-optim-g.pytorch')
    torch.save(netD.state_dict(), 'section-5-network-d.pytorch')
    torch.save(netG.state_dict(), 'section-5-network-g.pytorch')

    generated = netG(fixed_noise, ceil(level)).detach().cpu()
```

```
#         .view(
#             -1,
#             3,
#             2 ** (ceil(level + 1.00001)),
#             2 ** (ceil(level + 1.00001))
#         )

        grid = torchvision.utils.make_grid(
            generated,
            nrow=5,
            padding=10,
            pad_value=1,
            normalize=True
        )

        tb_writer.add_image(
            'PGGAN/Output',
            grid,
            epoch
        )

# Генерація випадкових осіб з розширенням 64x64
generated = netG(fixed_noise, level=5).detach().cpu().view(-1, 3, 64, 64)


grid = torchvision.utils.make_grid(
    generated,
    nrow=5,
    padding=10,
    pad_value=1,
    normalize=False,
```

```
#     range=(-1, 1)
)


plt.figure(figsize=(22, 20))
plt.axis("off")
plt.title("Generated Images")

plt.imshow(
    np.transpose(
        grid,
        (1,2,0)
    )
);
```