«INVESTIGAN»

ГЕНЕРАТИВНО-ЗМАГАЛЬНІ МЕРЕЖІ У СФЕРІ СТВОРЕННЯ КОНТЕНТУ

ЗМІСТ

АНОТАЦІЯ

Актуальність – сьогодні GANs застосовуються у мистецтві, рекламі, відеоіграх, науці та інших сферах. Вони розв'язують задачі підвищення чіткості зображень, відновлення втрачених даних, генерації відео й аудіо, реставрації фото чи космічних знімків, створення VR-об'єктів. Добре розуміння того, як працює такий потужний інструмент глибокого навчання знайде широке застосування в інформаційній діяльності людини, а постійний розвиток GANs може призвести до відкриття нових застосувань, де синтез реалістичних даних є ключовим елементом.

Мета роботи є дослідження внутрішніх процесів генеративно-змагальних мереж, вивчення механізмів та чинників, що визначають характер та якість їх роботи.

Поставлені **завдання** стосуються наступних питань: пошук та перевірка способів моніторингу навчання GAN; розкриття сутності, способів взаємодії та властивостей прихованого простору; аналіз внутрішніх процесів на рівні окремих шарів нейронних мереж; визначення різниці в принципах роботи класифікатора та дискримінатора; структурування отриманих результатів.

Використана методика дослідження представляє собою сукупність нейронних мереж з варіативним набором глобальних параметрів при вирішені однакових задач, з застосування різних способів моніторингу навчання, та кінцеве написання власного програмного забезпечення для подальшого візуального вивчення властивостей отриманих генераторів.

Дослідження викладено на 30 сторінках пояснювальної записки, структура якої охоплює три основні розділи. У роботі подано 35 рисунків та 4 таблиці. Список використаних джерел включає 13 найменувань, 3 особисті публікації та 3 електронні ресурси з глибоким розкриттям роботи.

Ключові слова: генеративно-магальні мережі, нейронні мережі, Tensorflow 2, дискримінатор, генератор, генерування образів, координати точки зображення в просторі.

ВСТУП

Генеративні змагальні мережі (GANs) – тип моделі глибокого навчання, що складається з двох частин: генератора (G) і дискримінатора (D).

GAN в даний час представляють собою потужний інструмент глибокого навчання і використовуються у різноманітних сферах, такі як мода, мистецтво та реклама, виробництво, відеоігри, шкідливі програми та глибокі підробки. За допомогою добре навчених GAN можна створювати образи, картини, відео і аудіо, 3D об’єкти або їх прототипи, тощо. Моделі здатні на основі статистичних розподілів передбачати відсутні фрагменти щоб підвищити якість зображень, зроблених телескопами, мікроскопами. Такий постійний розвиток GANs може призвести до відкриття нових застосувань, де синтез реалістичних даних є ключовим елементом.

Загальною задачею даної роботи є генерування контенту за допомогою GAN, знайти архітектури та рекомендації, які дозволять підвищити якість отриманих даних, дослідити процеси навчання моделей і синтезу образів, розглянути внутрішню поведінку створених мереж.

Під час виконання роботи потрібно:

– знайти та перевірити різні способи моніторингу навчання GAN;

– дослідити сутність латентного (прихованого) простору, його основні властивості та способи взаємодії з ним;

– проаналізувати внутрішні процеси, що відбуваються у нейронних мережах на рівні окремих шарів;

– визначити відмінності між класифікатором та дискримінатором, охарактеризувати різницю в їх принципах роботи;

– зробити аналіз отриманих результатів експериментів.

1 ЕКСПЛІКАЦІЯ (EXPLICATIO)

1.1 Опис предметної області роботи

Генеративні змагальні мережі (GAN) – тип моделі глибокого навчання, розроблений Яном Гудфеллоу (Ian Goodfellow) та його колегами у 2014 році [4]. За словами Яна Лекуна, директора з досліджень ШІ у Facebook, GAN є «найцікавіша ідея за останні 10 років у сфері машинного навчання». В архітектурі GAN є дві нейронні мережі (генератор і дискримінатор), які змагаються між собою у грі «з нульовою сумою». Перший отримує навчальну вибірку і вчиться створювати нові зразки зі схожими характеристиками. Дискримінатор намагається з'ясувати, чи є згенеровані дані автентичними, чи сфабрикованими. Одночасне навчання змушує їх постійно розвиватися, покращуючи свої індивідуальні результати. Самі ж моделі з'єднані між собою прихованим простором, де відбувається вся «магія». Слід зазначити, що безпосередньо з простором взаємодіє тільки генератор, а дискримінатор «дивиться» на нього через призму попереднього.

1.2 Опис структурних і функціональних особливостей

Структурні і функціональні особливості GAN:

– архітектура – зазвичай використовується CNN (convolutional neural network) або DNN (Deep neural network), їх комбінації, та шари з абсолютно різними функціями активації;

– вхідні дані – дискримінатор приймає на вхід зображення або інший тип даних, генератори зазвичай приймають випадковий шум;

– функція втрат – дискримінатор частіше за все використовує бінарну кросс-ентропійну (binary crossentropy) функцію втрат для своєї (або альтернативи MSE (Mean Squared Error), MAE (Mean Absolute Error) та інші);

– вихід – значення від дискримінатора може бути одиничним числом (0,54) або вектором ймовірностей ([0,2; 0,8]) належності класу;

– оптимізатор – для навчання використовуються алгоритми оптимізації, такі як стохастичний градієнтний спуск SGD (Stochastic gradient descent), або Adam, nAdam.

1.3 Проведення аналізу проблеми, що вирішується

Навчання складних GAN вимагає високих технічних характеристик системи, щоб одночасно розвивати дві важкі моделі. Генерація зображень додає проблему розмірності, де більша якість потребує більше пам’яті.

Гіперпараметр batch може дещо вирішити це питання. Його малі розміри потребують менше пам’яті графічного процесору (GPU), але збільшують кількість оновлень вагових коефіцієнтів (додаючи шанс проблеми сильної варіації із сповільненням збіжності). Велике значення параметру може пришвидшити навчання, дати стабільні градієнти, але вихід за межі погіршить узагальнюючу здібність моделі.

Значення похибок генератора і дискримінатора можуть почати сильно коливатися (Oscillating Loss), замість стабільних змін впродовж часу [3].

Генератор може знайти невелику кількість прикладів, які «обманюють» дискримінатор, переставши розвиватися, створюючи проблему Mode Collapse.

Проблема моніторингу навчання займає окреме місце. Похибки цих глибоких моделей оцінюються лише у кожен поточний момент, тому не можна порівнювати функцію втрат, отримані в різні проміжки процесу навчання.

Крім того, слід враховувати параметри такі, як batch normalization, dropout, learning rate, активаційні шари, згорткові фільтри, розмір ядра, вимір латентного простору. GAN дуже чутливі навіть до незначних змін, тому пошук такого набору часто є випадком навчання методом проб і помилок, а не дотримання встановленого набору інструкцій.

Ось чому важливо розуміти внутрішню роботу GAN і знати як інтерпретувати функцію втрат, щоб можна було б визначити розумні коригування гіперпараметрів, які можуть покращити стабільність моделі.

2 ФОРМАЛІЗАЦІЯ (FORMALIS)

2.1 Інформаційне забезпечення

Для дослідження того, як працюють мережі GAN, краще не використовувати складних даних, що мають «важку» форму. Основна увага приділена відомому набору даних MNIST з бібліотеки Keras [6], який містить 60 000 зображень (плюс 10 000 тестових) із10 цифр у відтінках сірого розміром 28 х 28. Пікселі мають значення від 0-255, що нормовані у діапазон [-1; 1].

Щодо середовища розробки, то будь-яке навчання нейронних мереж не є простою задачею. Тому є два варіанти: використання власного ПК, або онлайн сервісу, наприклад, Kaggle [8] з безкоштовним доступом до GPU.

Для збору і систематизування інформації буде зручним використання сервісу Figma, що підтримує створення діаграм, презентацій, відображення фото, gif, відео, посилання на інтернет ресурси, і публічним доступом.

2.2 Математичне забезпечення

Одними із параметрів, якими можна регулювати та експериментувати під час навчання глибоких мереж, є функції активації (ФА). Найчастіше рекомендують ReLU [9], що трохи швидше в обчисленні, а завдяки ефекту «не насичуватися» великими вхідними значеннями, градієнтний спуск не затримується на плато. До мінусів відносять «згасаючі елементи ReLU» – фактичне відмирання нейронів, що починають видавати 0 і близькі до нього значення коли оновлюються ваги, де зважена сума входів нейронів стає від’ємною. Повернення нейрона «до життя» малоймовірне, тому що градієнт ReLU рівний 0, коли його вхід від’ємний [10].

$$f(x) = max\_value \text{ if } x >= max\_value,$$

$$f(x) = x \text{ if } threshold <= x < max\_value,$$

$$f(x) = negative\_slope * (x - threshold) \text{ otherwise.}$$

LeakyReLU або ReLU з витоком вирішує проблему відмирання:

$$f(x) = \alpha * x \text{ if } x < 0,$$

$$f(x) = x \text{ if } x >= 0.$$

Інша варіація, RreLU, де α вибирається випадково в заданому діапазоні під час навчання та фіксується на середньому значенні під час випробувань. Параметричний PreLU, де α можна вчити під час навчання. ELU, яка в експериментах дослідників перевершила всі інші різновиди ReLU.

$$f(x) = \alpha * (exp(x) - 1.) \text{ for } x < 0,$$

$$f(x) = x \text{ for } x >= 0.$$

Логістична функція (Sigmoid) – нелінійна функція, що широко використовується в глибокому навчанні як активація останнього шару, вихід якої в діапазоні [0; 1] [11].

$$f(x) = \frac{1}{1 + e^{x}}.$$

Гіперболічний тангенс (Tanh) – інша популярна нелінійна функція, має вихідні значення в діапазоні [-1, 1].

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = 2 * sigmoid(2x) - 1.$$

Що Sigmoid, що Tanh мають проблему насичуватися великими вхідними значеннями, наприклад, коли остання насичується при значеннях 1.

Нижче, на рисунку 2.1, наведено графіки всіх цих функцій.

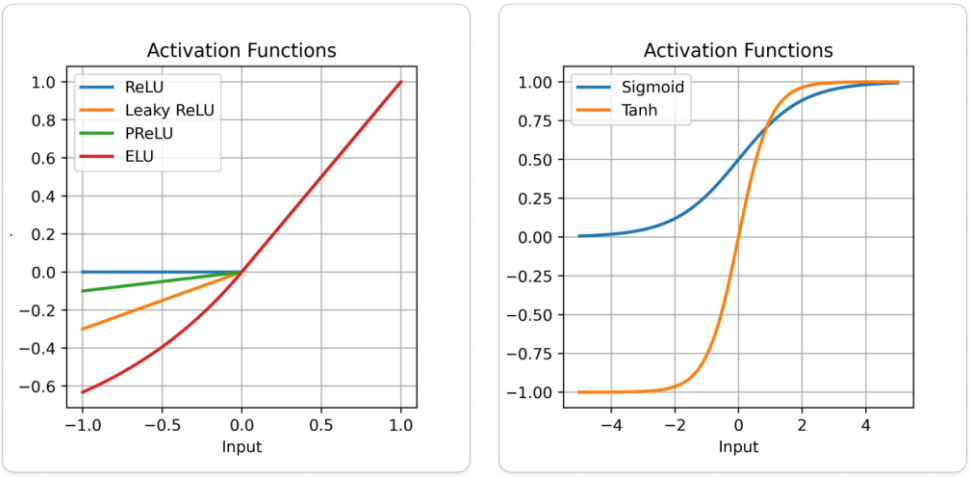

Рисунок 2.1 – Графік розглянутих функцій активації для шарів мереж

MSE (Mean Squared Error):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2.$$

MAE (Mean Absolute Error):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - p_i|.$$

Якщо вирішується проблема класифікації, де кожне спостереження належить лише до одного класу, то використовують Categorical cross-entropy:

$$\text{Categorical cross} - \text{entropy} = -\sum_{i=1}^{m} y_i \log(p_i).$$

Binary crossentropy – обчислює перехресну втрату ентропії між справжніми та прогнозованими мітками в задачі двійкової:

$$\text{Binary Crossentropy} = -\frac{1}{n}\sum_{i=1}^{n}(y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i)),$$

де p_i – ймовірність одиниці; $1 - p_i$ – ймовірність нуля.

Оптимізатор – це алгоритм, який використовуватиметься для оновлення вагових коефіцієнтів у нейронній мережі на основі градієнта функції втрат. Одним із найбільш часто використовуваних і стабільних є Adam [3, 13].

$$w_{ij}{}^{(t+1)} = w_{ij}{}^{(t)} - \frac{\eta \widehat{m}_{ij}^{(t)}}{\sqrt{\widehat{v}_{ij}^{(t)} + \varepsilon}},$$

де $\widehat{m}_{ij}^{(t)}$ – швидкість спаду середнього градієнтів, $\widehat{v}_{ij}^{(t)}$ – середній квадрат градієнтів, w_{ij} – ваги, η – постійний фактор швидкості навчання, ϵ – епсилон, дуже маленьке додатне число, яке вводиться в знаменник, щоб уникнути ділення на нуль.

Втрати генератора (G) та дискримінатора (D) походять від однієї міри відстані між розподілами ймовірностей. Однак генератор може впливати лише на один член вимірювання відстані: член, який відображає розподіл

фальшивих даних. Тому під час навчання генератора відкидають іншу частину, яка відображає розподіл реальних даних.

Дискримінатор намагатиметься класифікувати правильно як фальшиві або синтетично згенеровані зразки, так і справжні. Іншими словами, він намагатиметься максимізувати функцію корисності:

$$U(D,G) = E_{x\sim P_x(x)}[logD(x)] + E_{z\sim P_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right].$$

Формула походить від перехресної ентропії між дійсним і згенерованим розподілами, де x – позначає вибірки реальних даних, взяті з розподілу ймовірностей $P_x(x)$, а z – шум, взятий з попереднього розподілу шуму $P_z(z)$. Тоді $E_{x\sim P_x(x)}$ є об'єкт взятий із вибірки всіх реальних об'єктів, а $E_{z\sim P_z(z)}$ – об'єкт взятий із вибірки всіх генерованих.

Генератор, як говорилося вище, буде використовувати тільки цю частину:

$$E_{z\sim P_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right].$$

На рисунку 2.2 вказано те, як формулу похибки використовують мережі дискримінатор і генератор окремо один від одного.

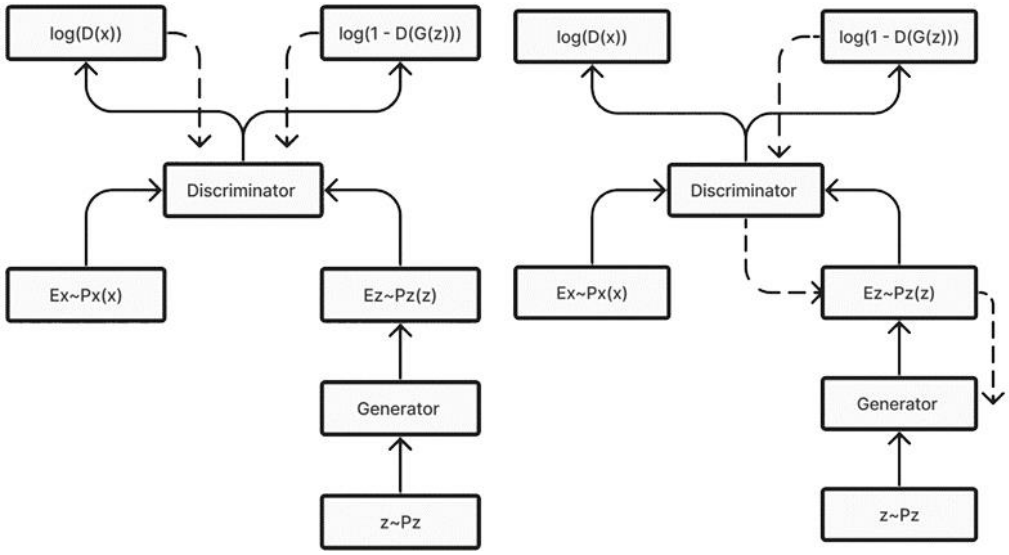

Рисунок 2.2 – Використання похибок під час навчання

3 ІНВЕСТИГАЦІЯ (INVESTIGATIO)

3.1 Процес навчання нейронних мереж

3.1.1 Прихований простір та архітектури мереж

На вхід до мереж GAN подається масив випадкових даних. Рисунок 3.1 відображає те, як «точка» з прихованого простору подається у G, зображення з якого потім перевіряється D. В результаті, перший запам’ятовує свою відповідь на цю точку, щоб потім покращити вихід, тобто записав дуже стиснені дані про створений образ у прихований простір. За попереднім вивченням, дві точки поруч матимуть майже однаковий вид генерації, а певний набір від точки «A» до точки «B» матиме ефект плавного перетікання одного образу в інший. Різні форми вихідних даних можуть дати різні ефекти.

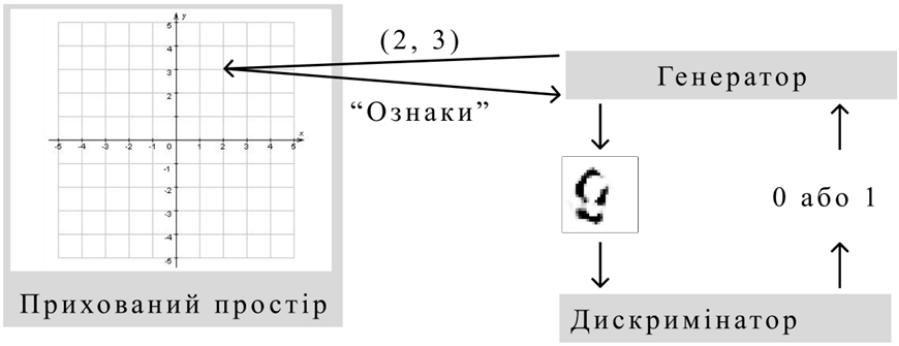

Рисунок 3.1 – Схема процесу запису ознак у прихований простір

Побудова архітектури моделі є початковим етапом. В літературі пропонують безліч рішень, але часто вони недостатньо обґрунтовані авторами. Так, в подальших експериментах методом «спроб і помилок» використана наступна архітектура генератора мережі (рис. 3.2). Після блоку «dense_input» йде напис ?x2 – двійка вказує розмірність вхідних, тому генератори 3D, 4D матимуть значення «?x3», «?x4». Розмір зображення MNIST складає 28 x 28 x 1, тому вихід G та вхід D відповідатимуть цьому значенню. Архітектура останнього зображена на рисунку 3.3 і не змінюватиметься для 3D та 4D.

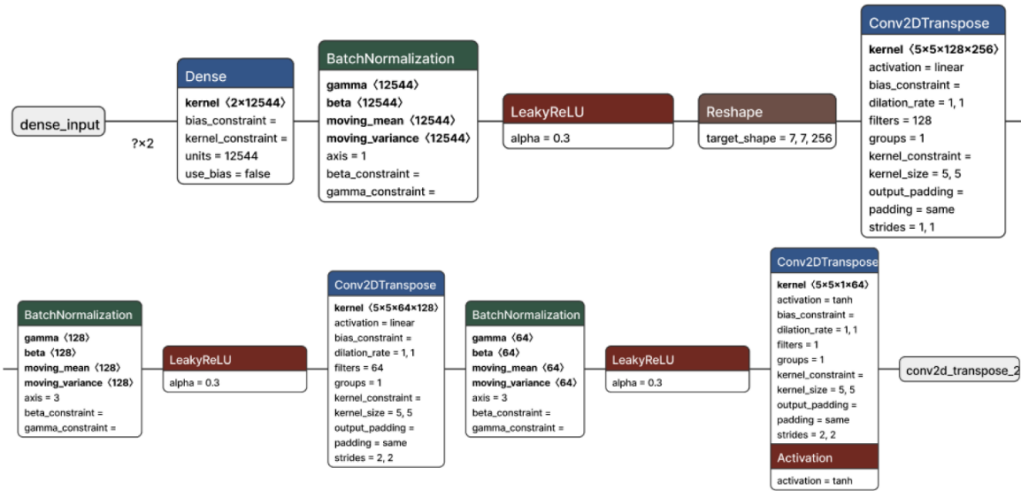

Рисунок 3.2 – Архітектура 2D генератора

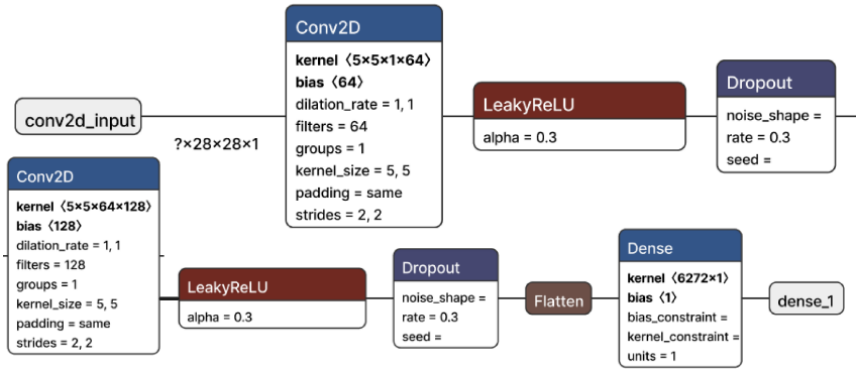

Рисунок 3.3 – Архітектура 2D дискримінатора

Таким чином сформовано загальні архітектури для експериментів.

3.1.2 Процес навчання 2D GAN

Для всіх GANs навчання відбувається за однаковим підходом, адаптуються тільки функції візуалізації. Загальною буде ще логіка збереження стану моделі (для можливості зробити паузу в навчанні і продовжити згодом).

До методів відстеження проміжних результатів можна віднести:

– збереження генерації фото з кожної епохи використовуючи однаковий шум (з метою формування гіф, відео, для перегляду процесу їх змін);

– збереження похибок моделей та тестування сильним незалежним класифікатором (НК) в кінці навчання, та слабким НК під час – чим гірше відбувається поділ на справжні і генеровані, тим краще навчений GAN;

– мапа областей класів з кожної епохи – підхід, що потребує генерації зображень на певному діапазоні точок прихованого простору. Отримані фото класифікуються окремою моделлю (з похибкою loss 1,76% та val loss 1,58%), де потім класи індивідуальним кольором фарбуються на площині. Відстеження сили змін областей стане одним із сигналів завершення навчання.

Числові результати навчання занесені у таблицю 3.3. Візуальні наступні:

– зображення з першої/останньої ітерації (рис. 3.4а-б відповідно);

– мапа областей з першої/останньої ітерації (рис. 3.5а-б відповідно) отримані шляхом класифікації 10 000 згенерованих зображень за комбінацією ста точок в діапазоні від -1 до 1 по вісі абсцис та вісі ординат;

– loss сильного НК на тисячній епосі склав 0,49%, val loss 0,55%, на п’ятитисячній епосі значення склали 0,73% та 1,54% відповідно. Графік навчання сильного НК наведено на рисунку 3.6, для слабкого – на рисунку 3.7;

– історія похибок генератора і дискримінатора міститься на рисунку 3.8;

– результат генерації навченої моделі представлено на рисунку 3.9.

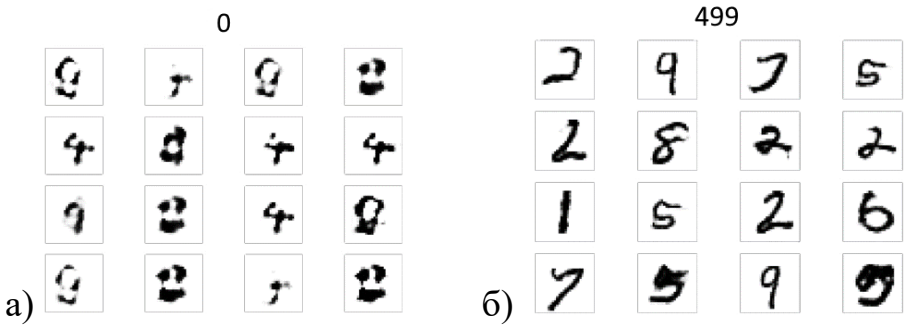

Рисунок 3.4 – Згенеровані зображення на першій та останній ітерації

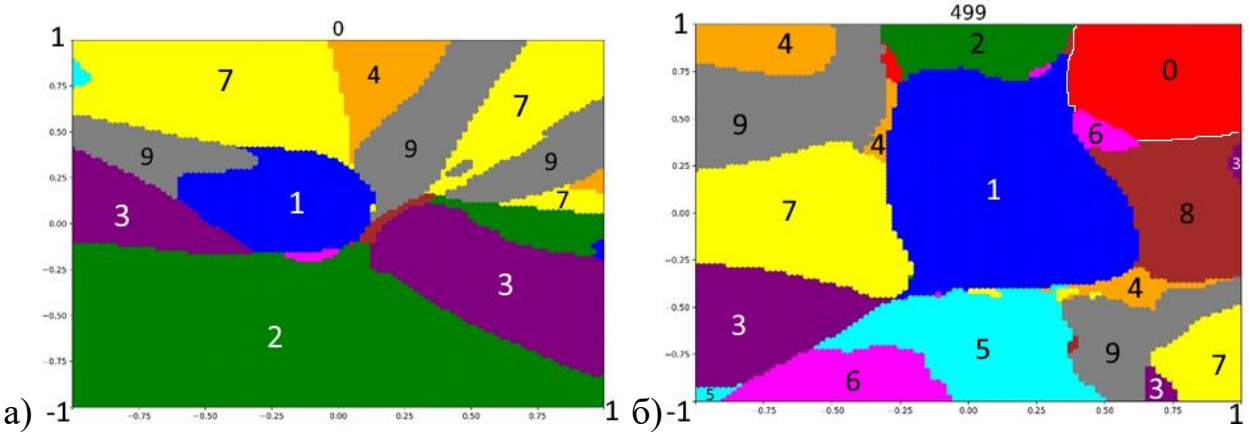

Рисунок 3.5 – Мапи областей класів на першій та останній ітерації

Рисунок 3.4 демонструє, як якість цифр стала кращою з часом. Генератор епоху за епохою покращував свій вихід, поки не досяг стабільного варіанту, хоч деякі зображення цифр і виглядають неякісно. Аналіз мапи областей прихованого простору 3.5 відкриває цікавий факт: між перезапуском навчання моделі, в центрі часто опинялася цифра «1». Її форма має найбільший зв’язок з іншими зображеннями цифр та легко перетворюється у більшість. Як висновок – центр найчастіше займає образ, характеристики якого притаманні більшості іншим об’єктам.

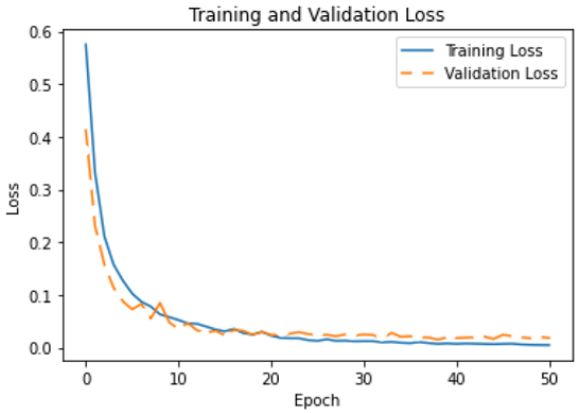

Рисунок 3.6 – Процес сильного навчання незалежного класифікатора

Графік показує, що класифікатор швидко вчиться розпізнавати де реальні зображення. Таку поведінку мав би D, якби оновлення його ваг відбувалося не плавно або з великим кроком, що і є відмінністю між ними.

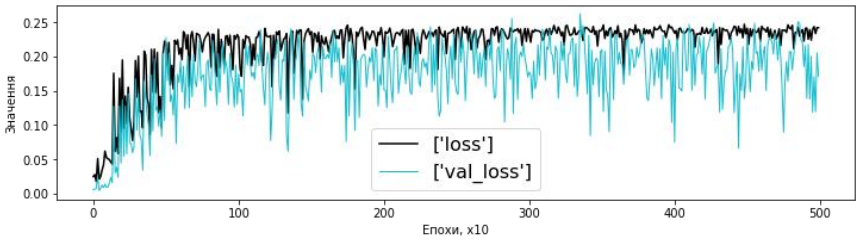

Рисунок 3.7 – Точність незалежних класифікаторів кожні 10 епох

Графік демонструє, що слабкі НК після сотої епохи перестали швидко підніматися у значеннях. Можна було б припустити, що навчання завершене, але це оманливе враження. Ідеальний варіант значення похибки у такій задачі вище 50%, щоб означало «вгадування» класифікатором класів.

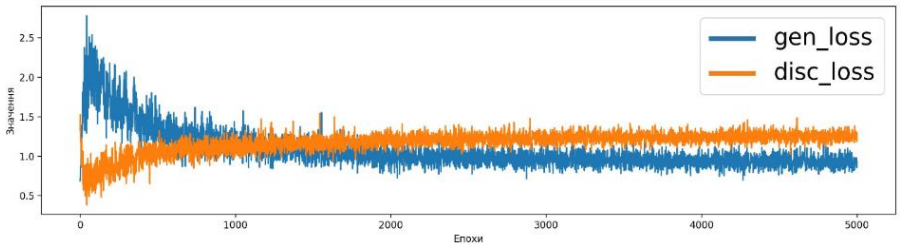

Рисунок 3.8 – Історія похибок генератора і дискримінатора

Рисунок 3.9 – Приклад генерації цифр моделлю 2D GAN

Підсумок отриманих результатів: на рисунку 3.8 помітно, що на протязі 2000 епох похибки плавно віддаляються одна від одної, що є сигналом стабільності навчання; рисунок 3.7 демонструє, що слабкий НК поступово втрачав спроможність розділяти справжні зображенні від генерованих, отже, GAN навчався; тестування сильним НК показало val loss 0,55% на 1000 епохі та 1,54% на 5000, що свідчить про покращення генерації. Зупинка навчання відбулася через зменшення інтенсивності візуальних змін генерацій на останніх епохах. Код блокноту 2D GAN розміщений у окремому проєкті [17].

3.1.3 Процес навчання 3D GAN

Зручна прямокутна мапа областей класів у 2D просторі замінюється на 3D куб. Тому вводиться поняття перерізів: по всім осям куба проводиться 16 «зрізів» (16 точок від -1 до 1 з відповідним кроком), формуючи проекції. Зібравши їх на кожній епосі можна відобразити зміни мапи областей класів.

Числові результати наведені у таблиці 3.3, візуальні наступні:

– генерації зображень на початку та в кінці наведені на рисунку 3.10;

– зміни в мапі областей класів з перерізами по осям на рисунку 3.11;

– динаміка точності слабких НК вказана на рисунку 3.12. Тестування моделі сильним НК показало, що на тисячній епосі loss становить 1,19%, val loss 2,69%. На двохтисячній епосі результати склали відповідно 2,02% та 3,2%;

– значення похибок G і D протягом всіх епох наведено на рисунку 3.13;

– генерації остаточної моделі представлено на рисунку 3.14.

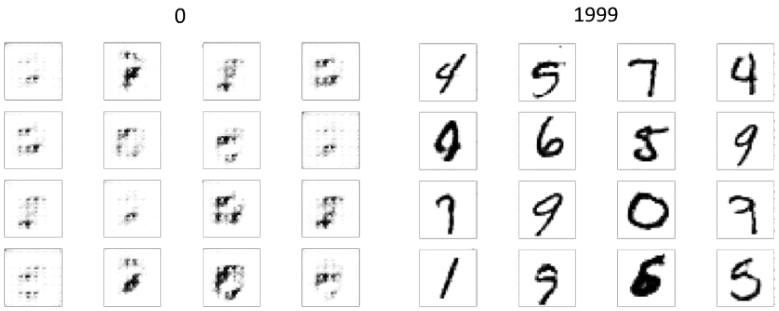

Рисунок 3.10 – Генерація на першій і останній епохах навчання

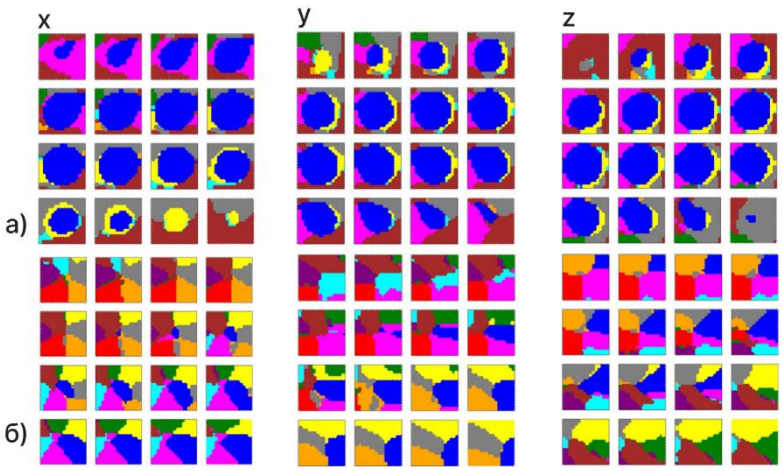

Рисунок 3.11 – Мапа областей класів на початку (а) і в кінці навчання (б)

Візуальний аналіз проекцій знову показує, що в центрі (координати 0), яким відповідає останнє зображення другого ряду та перше зображення третьої проекції «ах» рис. 3.11) знаходиться цифра «1», як і в 2D випадку.

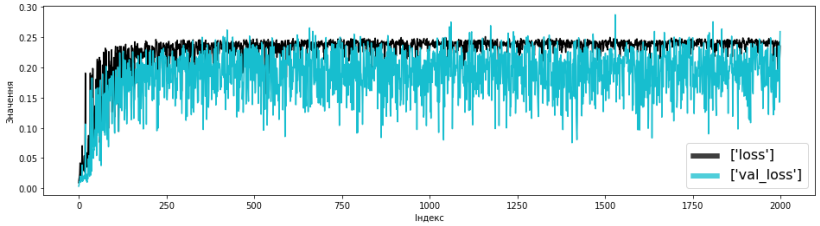

Рисунок 3.12 – Зміни похибок слабких незалежних класифікаторів

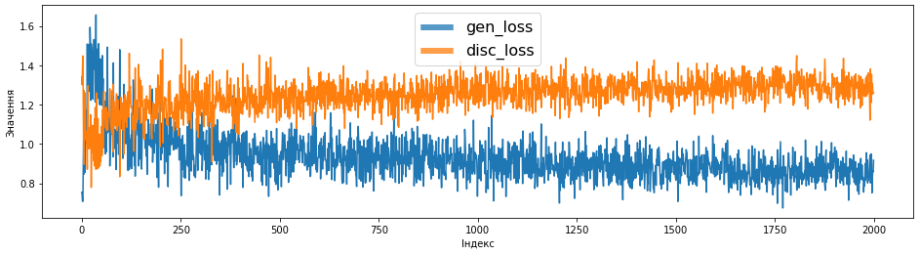

Рисунок 3.13 – Зміни похибок генератора і дискримінатора

Рисунок 3.14 – Генеровані цифри остаточним станом мережі

Аналіз результатів: навчання швидше ніж в 2D, якість генерованих зображень вище (порівнюючи рис. 3.4, 3.10 та 3.9, 3.14). Тест сильного НК складає 3,2%. Динаміка похибок моделей G і D стабільна. Зупинка навчання відбулася з тих самих причин: на останніх епохах візуальні зміни ставали менш суттєвими. Таким чином, збільшення виміру шуму має позитивний вплив.

3.1.4 Процес навчання 4D GAN

4D має проблеми з візуалізацією, адже в ньому безліч 3D кубів, тому слідкувати за змінами без сильних часових втрат не вийде. Кінцевий ж стан можна розглянути наступним методом: розрахувати всі потрібні точки по трьом осям із фіксуванням четвертої, щоб відобразити їх у 3D просторі. Наприклад, візуалізація градієнтного спуску нейронної мережі (апроксимація функції), що складається з 3х ваг. Достатньо зберегти ваги та loss під час навчання, отримавши 4D простір. Прорахувати всі можливі стани моделі (похибку l) для ваг *w1*, *w2*, та *w3* коли рівний значенню *w3* з кожної епохи навчання, можна у вигляді точки на площі функції похибки отримати

початковий стан мережі (рис. 3.15-а), а продовження навчання змінює вигляд площини і положення точки, поки не буде досягнений мінімум (рис. 3.15-б).

Цей принцип можна застосувати до перегляду кінцевого стану 4D GAN. На рисунку 3.16-а наведено положення точки в кубі при значені четвертої координати -1, а потім при +1 (рис. 3.16-б).

Числові результати наведені у таблиці 3.3, візуальні наступні: генерації на першій/останній епохах (рис. 3.17); тест сильним НК – loss 3,56%, val 8,29%.

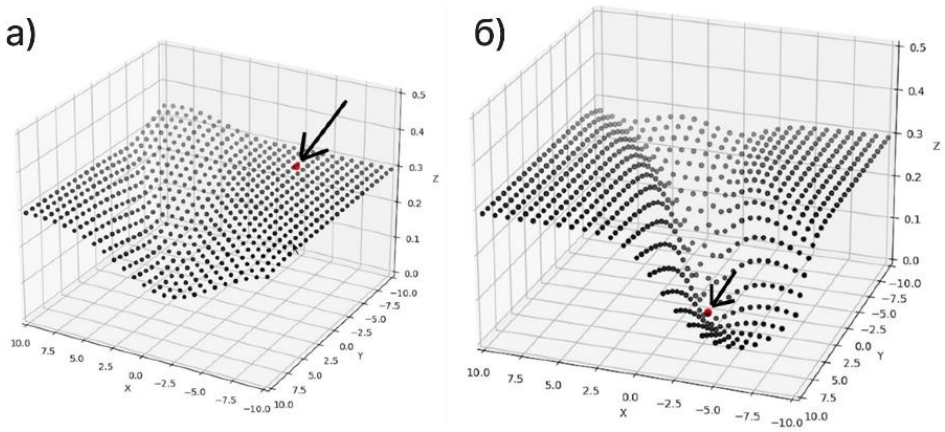

Рисунок 3.15 – Початковий (а) та кінцевий (б) стан навчання

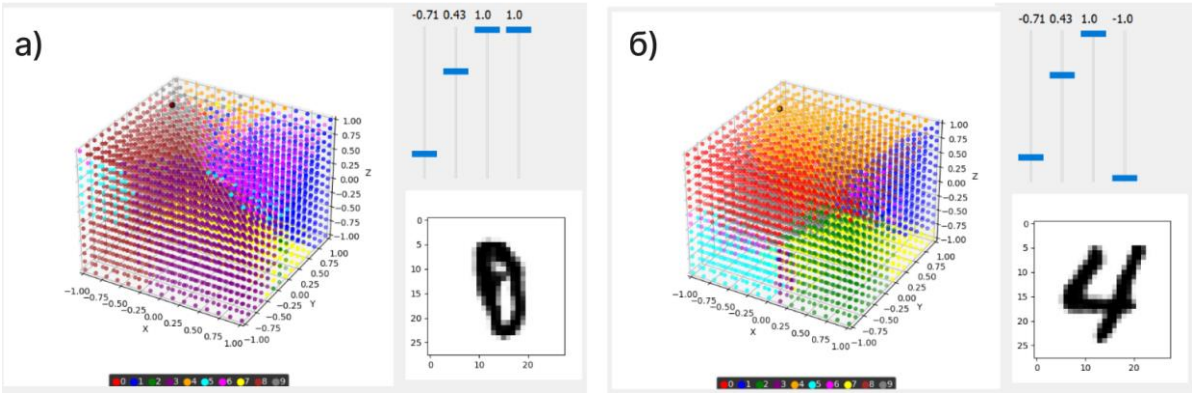

Рисунок 3.16 – Стан системи коли четверта координата рівна «-1» та «1»

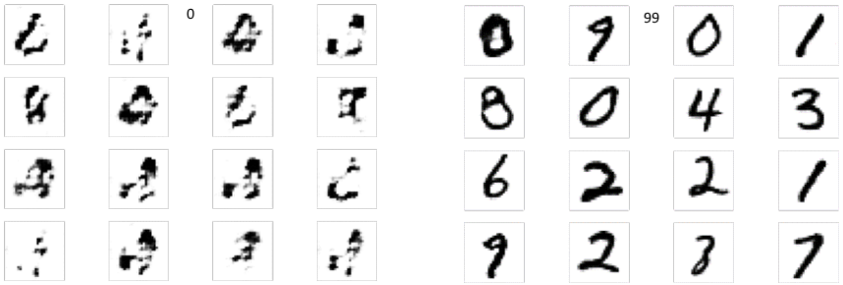

Рисунок 3.17 – Генерація зображень на першій і останній ітерації

Отже, правильна величина шуму значно впливає на якість навчання.

3.2 Дослідження властивостей отриманих мереж

3.2.1 Двовимірний простір

До отриманої мапи областей класів 2D GAN можна додати точку, рух якої дозволить досліджувати прихований простір (рис. 3.18). Інтерфейс програми має декілька областей: згенерований образ «Generated image»; проміжні результати генерації від першого до останнього нейронного шару «Activations for each layer (average values)»; мапа областей класів «Map of class areas»; повзунки «x» та «y» для зміни координат точки [17, 18, 19].

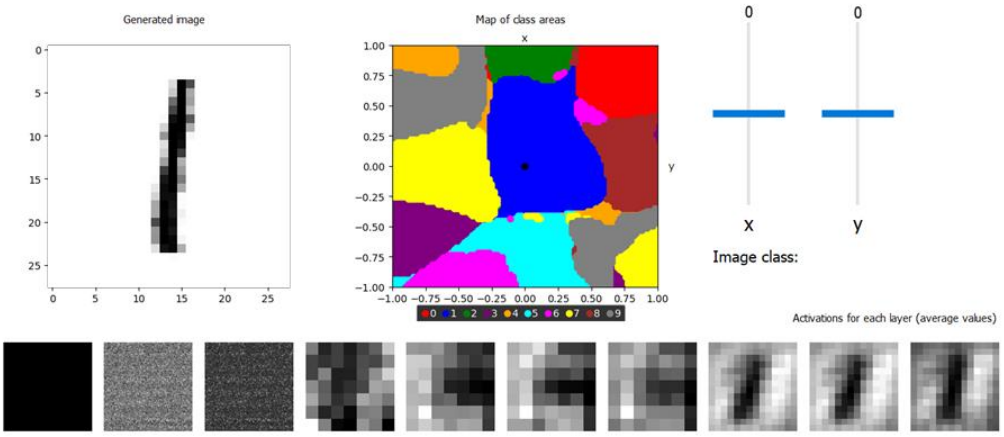

Рисунок 3.18 – Інтерфейс дослідження точкою 2D прихованого простору

Рухаючи повзунки можна змінити стан системи (рис. 3.19): точка з координатами (-0,84; 0,54) зайняла область цифри «9» на мапі; координати (0; 0) відповідають цифрі «1». Інші області, вказано на рисунку 3.20.

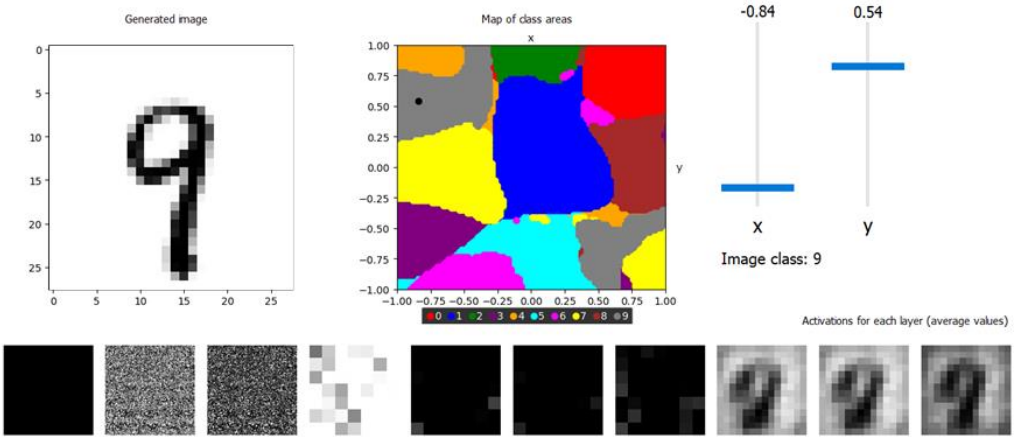

Рисунок 3.19 – Новий стан системи через зміну положення точки

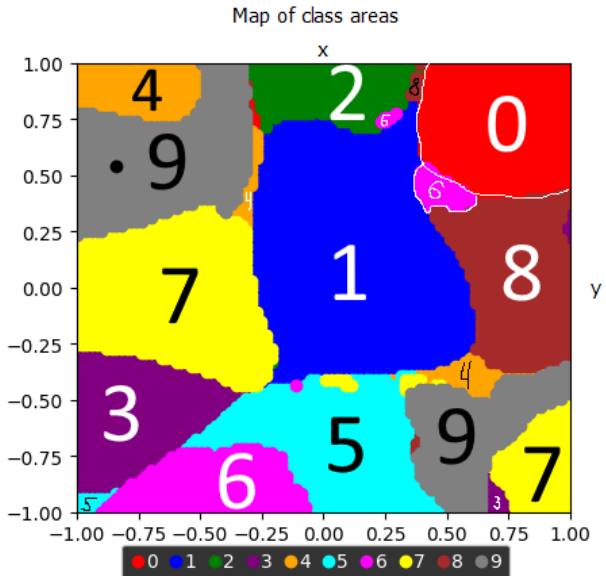

Рисунок 3.20 – Текстове позначення областей класів

Якщо генерувати зображення від точки «A» до «B» с певним кроком, можна отримати плавне перетікання. На рисунку 3.21 зображено такий процес, від точки [-1; 0] до [1; 0], де лінія між ними показує пересічені області – цифра мала вигляд сімки, потім одиниці, в кінці перетворившись у вісім.

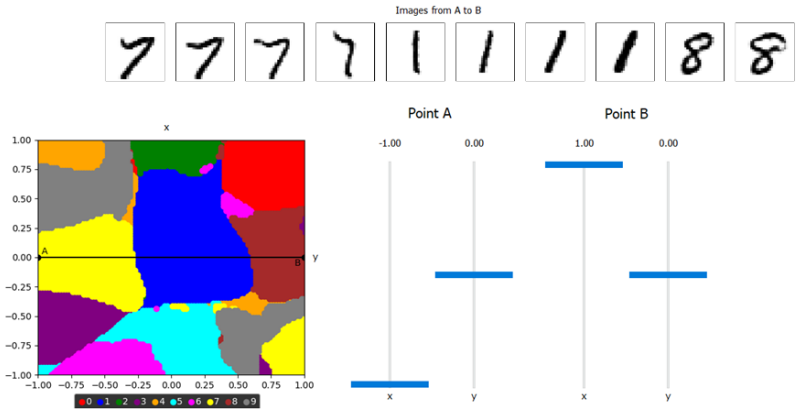

Рисунок 3.21 – Плавне перетікання зображень між двома точками

Тепер можна перенести розглянуту логіку у вищу розмірність.

3.2.2 Тривимірний простір

Інтерфейс, представлений на рисунку 3.23, так само має: область згенерованого зображення; проміжні генерації по шарам мережі; куб мапи

областей класів; повзунки для зміни координат точки, для якої додатково будуються три проекції місця знаходження (рис. 3.22).

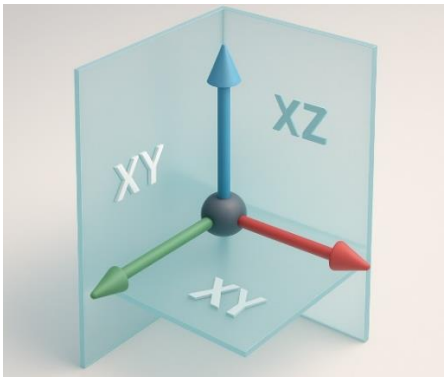

Рисунок 3.22 – Зображення проекцій точки на площині

Рисунок 3.23 демонструє погану видимість точки з нульовими координатами в центрі кубу, але за рахунок проекцій ця проблема компенсується, що дозволяє визначити – точка займає положення цифри один. Зміна координат створить новий стан системи (рис. 3.24), де точка займає кут, що відповідає цифрі вісім.

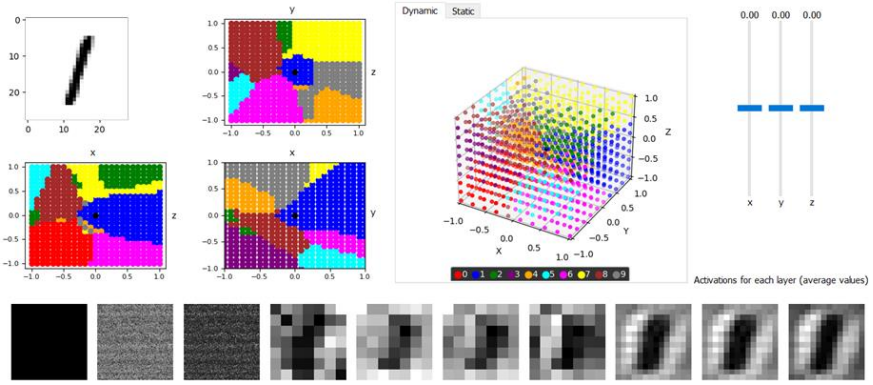

Рисунок 3.23 – Інтерфейс для роботи з однією точкою 3D простору

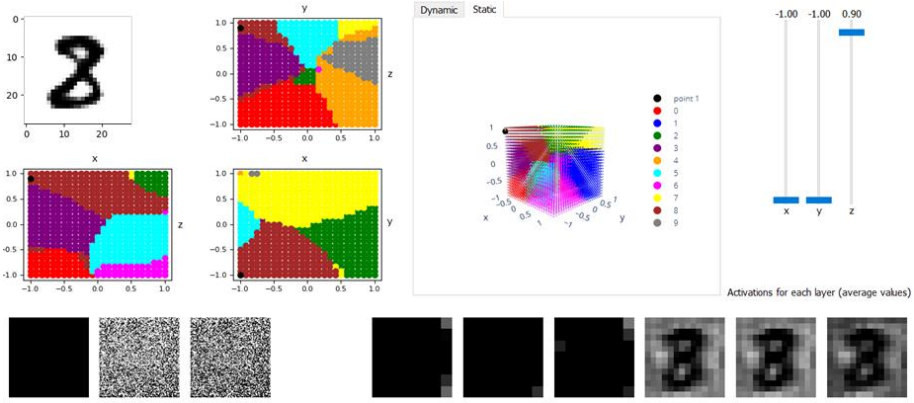

Рисунок 3.24 – Зміна положення точки у просторі

Крім того реалізовано і плавне перетікання між двома точками. На рисунку 3.25 вказаний запропонований інтерфейс для цього. Оскільки точок дві, то є області з проекціями для кожної з неї, та повзунки для зміни їх координат. По серединні розміщений куб мапи областей класів, зверху – перетікання образів. З допомогою інтерактивного графіку можна виділити тільки ті області, через які безпосередньо пройде лінія (рис. 3.26).

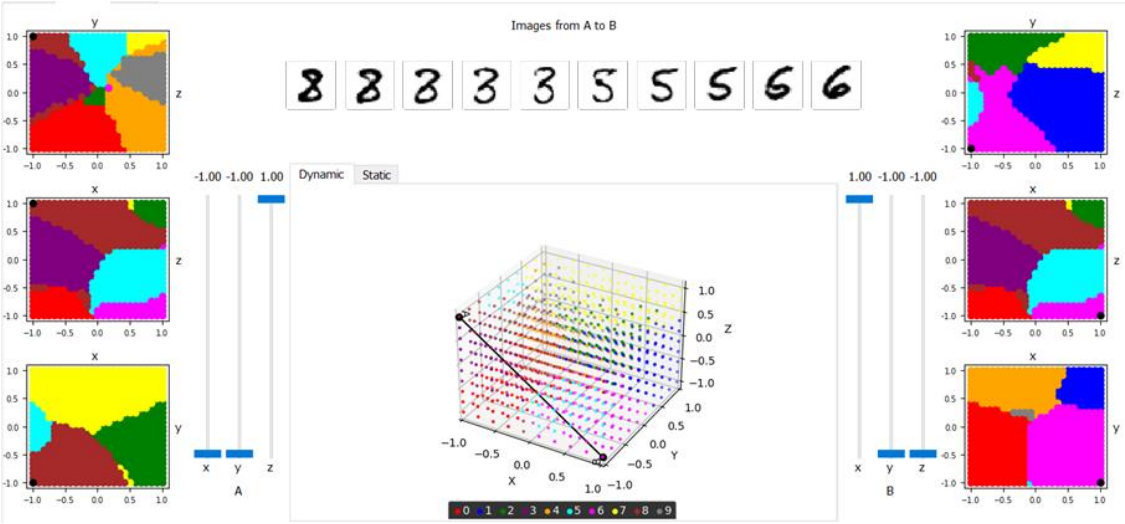

Рисунок 3.25 – Інтерфейс для дослідження тривимірного перетікання образу з точки «A» в точку «B»

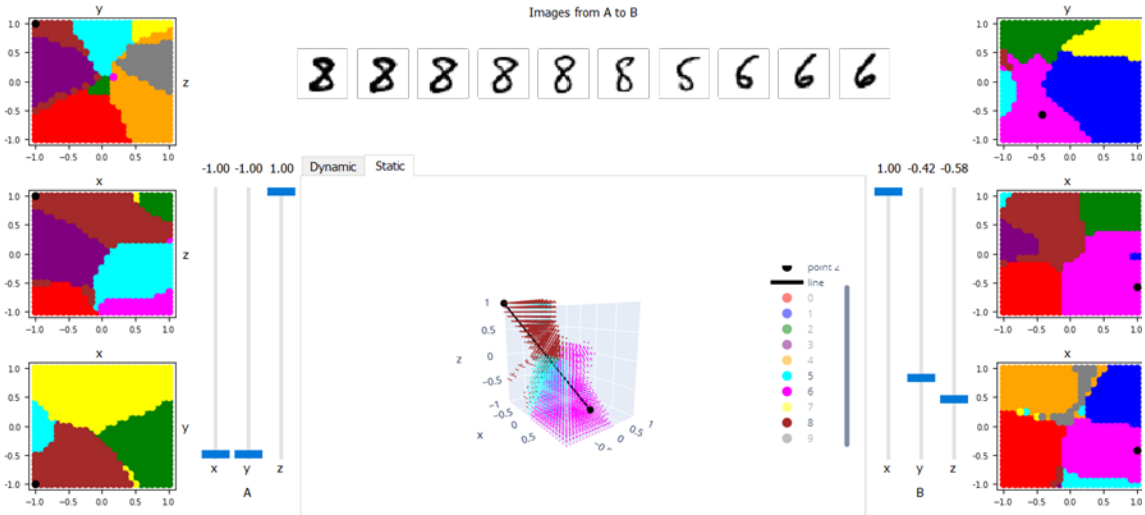

Рисунок 3.26 – Перегляд шляху перетікання за допомогою інтерактивного графіку

Таким чином, логіка розглянута для 2D простору працює і для 3D.

3.3 Генеративно-змагальна мережа 3D CGAN

Conditional Generative Adversarial Network (CGAN) – вид GAN, які під час генерації отримують на вхід шум і умову, тобто мітку класу образу, який потрібно отримати. У ідеальному CGAN кожна точка одночасно відповідає всім класам (рис. 3.27), у реальному модель не завжди видає очікуваний образ.

Розроблено наступний інтерфейс дослідження (рис. 3.28), що має такі елементи: генероване зображення; повзунки координат точки; вибір мітки класу «Change Number»; вибір кубу мапи областей прихованого простору «Change 3D map», який потрібний через те, що CGAN одночасно будує N просторів, де N – кількість класів. В давному випаду цифр десять, тому і кубів теж десять. Візуалізація кубів подібна 3D GAN, але для кожного класу окремо.

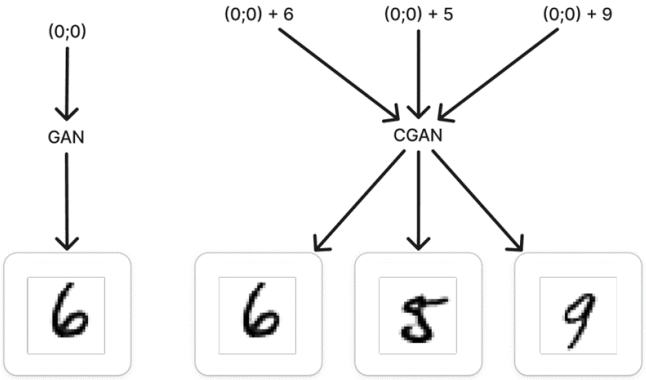

Рисунок 3.27 – Відмінність між GAN та CGAN

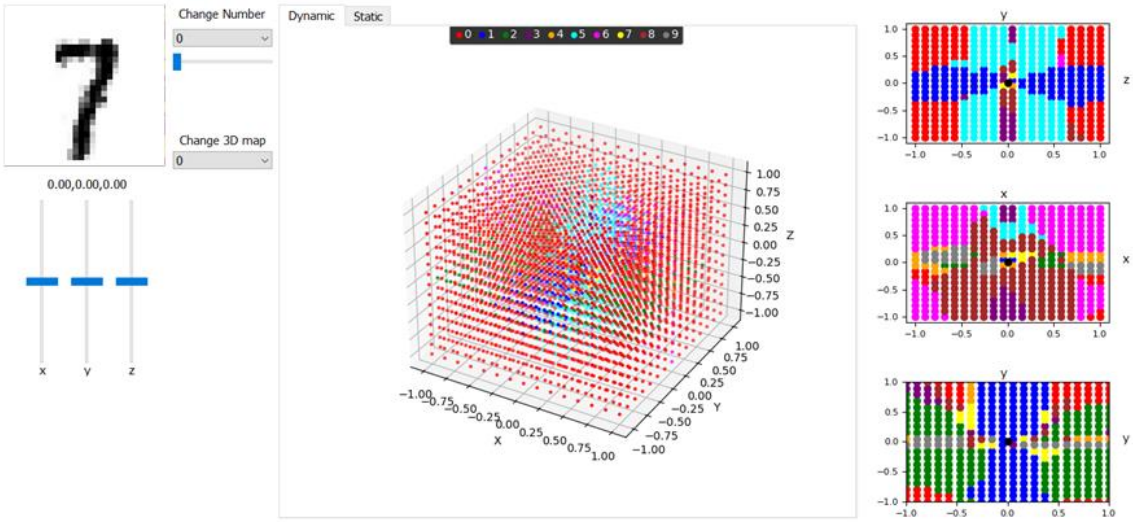

Рисунок 3.28 – Інтерфейс дослідження CGAN

На рисунку 3.28 представлено мапу класів при обраній мітці «0». Оскільки навчання не ідеальне, проекції положення точки демонструють інші кольори (саме зображення відповідає цифрі 7). В ідеалі, всі точки були б одного кольору відповідного класу. Положення точки в області цифри «0» продемонстровано на рисунку 3.29. Можна зазначити, що на мапі знаходяться образи всіх інших цифр. Якщо обрати куб і мітку «3» (рис. 3.30), то мапа має також образи 7 і 8.

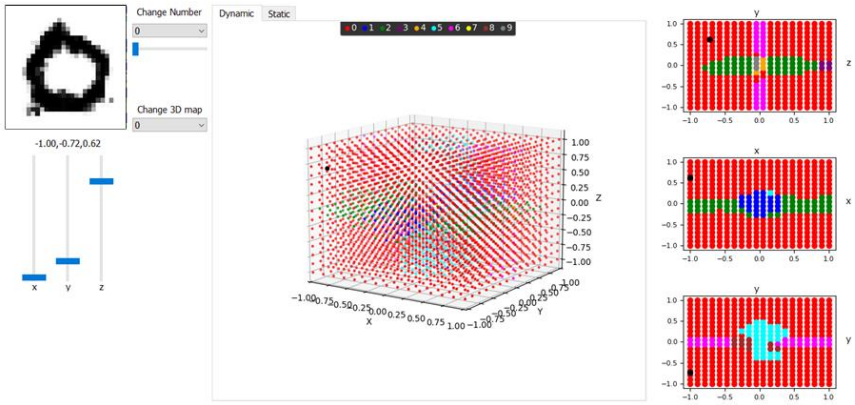

Рисунок 3.29 – Приклад зміни координат точки

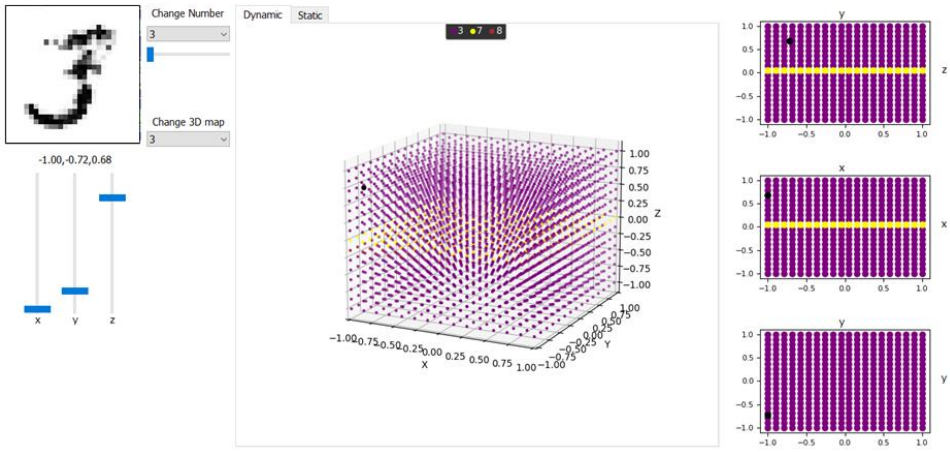

Рисунок 3.30 – Приклад генерації за мітко класу «3»

Таким чином, отримана універсальна модель, яка вміє генерувати більше різних образів, на одиницю шуму. У звичайному GAN кількість точок рівна 10 000, в яких розподілені всі класи, у CGAN по 10 000 на кожен клас. З мінусів: мало з цих образів унікальні або виглядають природньо, але це більше питання про процес та методи навчання, структуру самої моделі.

3.4 Дослідження різних функцій активацій у Digit GAN

В процесі дослідженні постало питання, а як саме функції активації впливають на навчання. У літературі зазначено, що більш прості функції краще справляються з задачами. Навіть описують деяку послідовність «якості» (від звичайної до найкращої): ReLU, LeakyReLU, PReLU, ELU, але про підґрунтя таких висновків нажаль не наведено. Тому вирішено провести серію навчань: 500 епох, датасет MNIST, однакова архітектура але різні ФА у генераторі (останній шар завжди tanh), і однаковим дискримінатором. Числові результати наведено в таблиці 3.1, графічне відображенням на рисунку 3.31.

Таблиця 3.1 – Результати експерименту з функціями активацій

Епоха	ReLU	LeakyReLU	PReLU	ELU
100	0,0064	0,0065	0,0068	0,0186
200	0,0198	0,0538	0,0473	0,0369
300	0,0810	0,0502	0,0670	0,0967
400	0,0631	0,0821	0,0263	0,0706
500	0,1561	0,1428	0,0789	0,0526

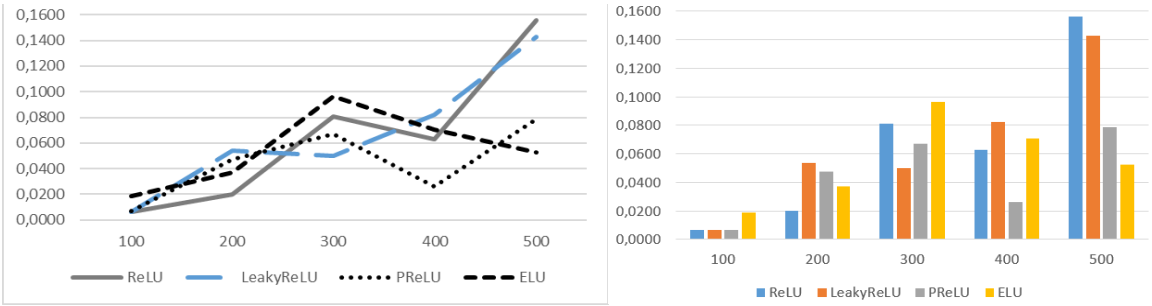

Рисунок 3.31 – Лінійна та стовбчаста діаграма результатів

Дані отримані за допомогою сильного НК. Графіки демонструють темпи навчання моделей, що в цілому мають одночасний приріст у показниках похибок з синхронним погіршенням на чотирьохсотій епосі. В результаті ФА ReLU досягла найкращих результатів (val loss склав 0,1561 – 15,61% похибці). ФА в цілому вишукувалися у оберненому порядку, ніж було зазначено вище, таким чином доводячи необхідність експериментів для кожної окремої задачі.

3.5 Дослідження різних функцій активацій у Landscape GAN

Для глибини розуміння ФА вирішино провести аналогічний експеримент для більш складної задачі – генеруванні зображень пейзажів. Для порівняння була навчена стартова модель з ФА LeakyReLU, динамікою навчання (рис. 3.32) та максимальною похибкою НК приблизно 8%.

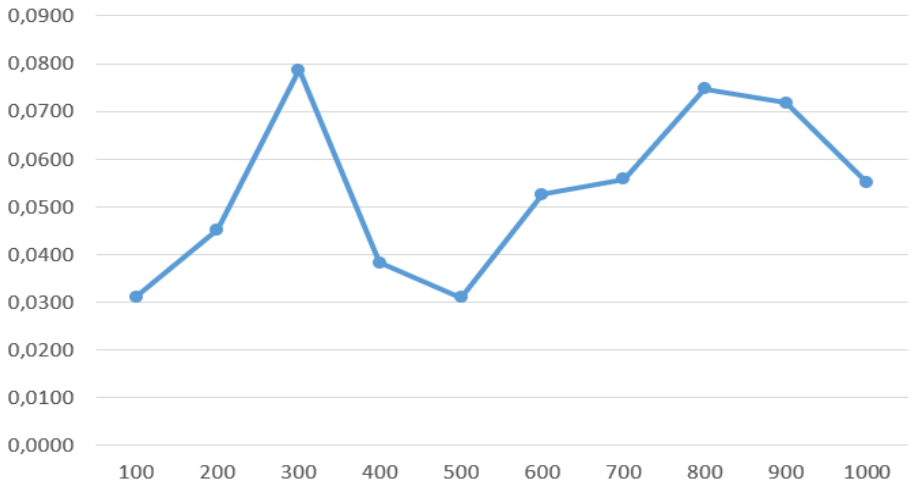

Рисунок 3.32 – Динаміка навчання моделі photo-landscape-64

Перед експериментом архітектура моделі була покращена, кількість фільтрів згортки збільшена. У таблиці 3.2 наведені числові результати, діаграми на рисунку 3.33. Аналіз першої показує, що всі моделі, крім PReLU, перевищили значення минулої версії. Також зауважимо синхронне погіршення у якості після 550 епохи. Це наводить на думки, що однаковий дискримінатор повторює собою площину похибки для кожної версії моделі. Не дивлячись на те, що з плином декількох епох їх навчання повністю рухається по різному. Друга діаграма показує результати в порівнянні між собою, де як висновок, на першому місці ФА ReLU з найкращим результатом у 11,45% похибки НК, що було досягнуто на 550-й епосі. Наступні йдуть ELU з 10,61% на 600-й, LeakyReLU з 9,74% на 700-й, та PReLU з 0,92% на 250-й.

Зауважимо, що кожна модель вчилась приблизно на 20% довше, ніж попередня версія. В цілому на експеримент витрачено загально (при умові паралельності навчання) 480 год.

Таблиця 3.2 – Результати експерименту

Епоха	ReLU	LeakyReLU	PReLU	ELU
10	0,001	0,0036	9,86E-04	0,0019
50	0,0288	0,0189	0,0057	0,0233
100	0,0541	0,0626	0,003	0,052
150	0,0618	0,0605	0,0048	0,0558
200	0,0631	0,0532	0,0062	0,0713
250	0,1081	0,0778	0,0092	0,089
300	0,0689	0,0593	0,0063	0,067
350	0,0907	0,0476	0,0036	0,0766
400	0,0967	0,0725	0,0052	0,0587
450	0,1145	0,0636	0,0062	0,084
500	0,1086	0,0654	0,0035	0,0795
550	0,1139	0,0521	0,0055	0,0818
600	0,0909	0,0890	0,0044	0,1061
650	0,0844	0,0947	0,0045	0,0807
700	0,0438	0,0974	0,0017	0,1032
750	0,0799	0,0874	0,0081	0,0707
800	0,0853	0,0458	0,0036	0,099
850	0,056	0,0798	0,0052	0,0721
900	0,0728	0,0763	0,0024	0,098
950	0,0399	0,0631	0,0057	0,056
1000	0,0397	0,0718	0,0059	0,0825

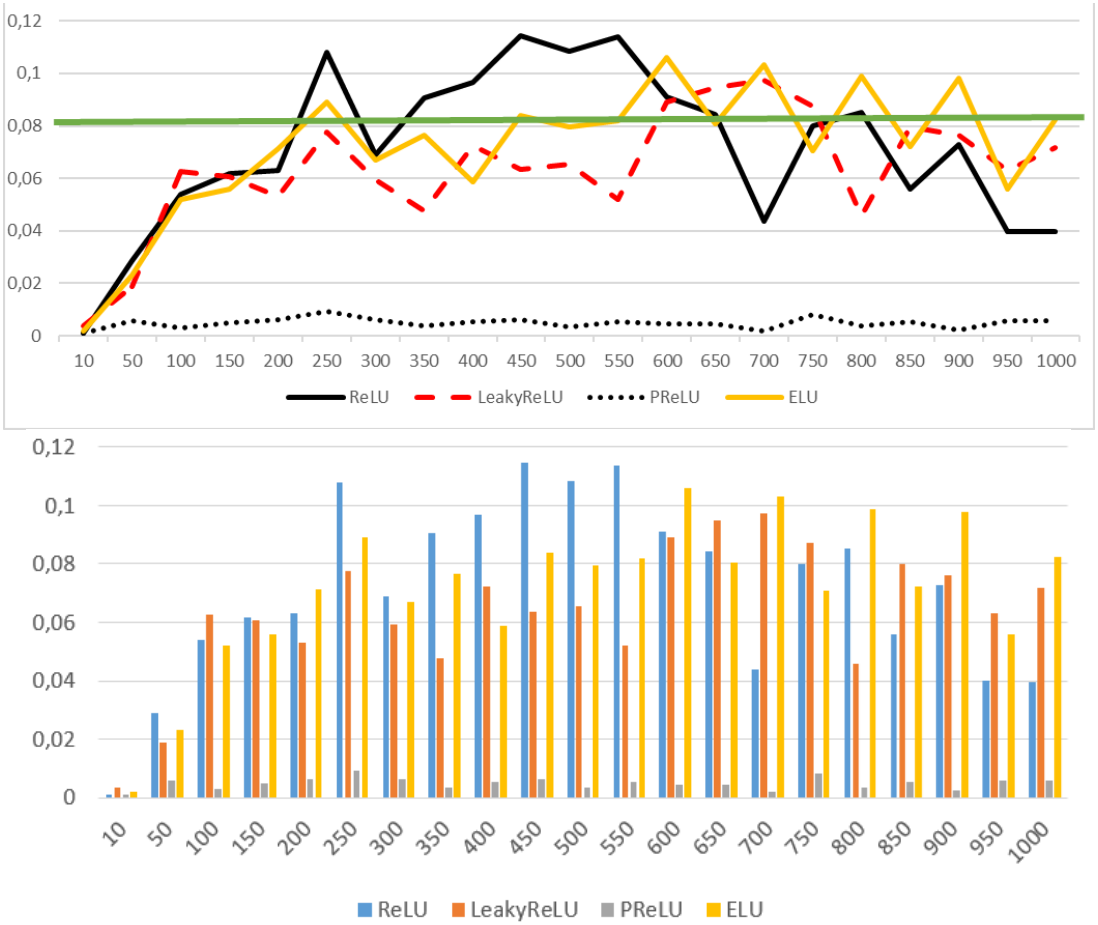

Рисунок 3.33 – Лінійна та стовбчаста діаграми експериментів

3.6 Порівняння активаційних карт отриманих моделей GAN

Після минулих експериментів стає цікавим візуалізація активацій між шарами у найкращих моделях photo-landscape. На рисунку 3.35 вказані активації перших шарів моделі а) ReLU, б) ELU, в) LeakyReLU, г) PReLU.

Аналіз: всі ФА намагаються сильно «анулювати» вхідні значення до нейронів – чорний означає що активації близькі до 0; генероване зображення створюється великою кількістю «паличок точок», які потім останнім шаром групуються у фото (рис. 3.34-а), де на чорних квадратах присутні ледве помітні білі риски. Змінюючи власноруч значення активацій передостаннього шару, можна генерувати окремі ділянки фото (рис. 3.34-б).

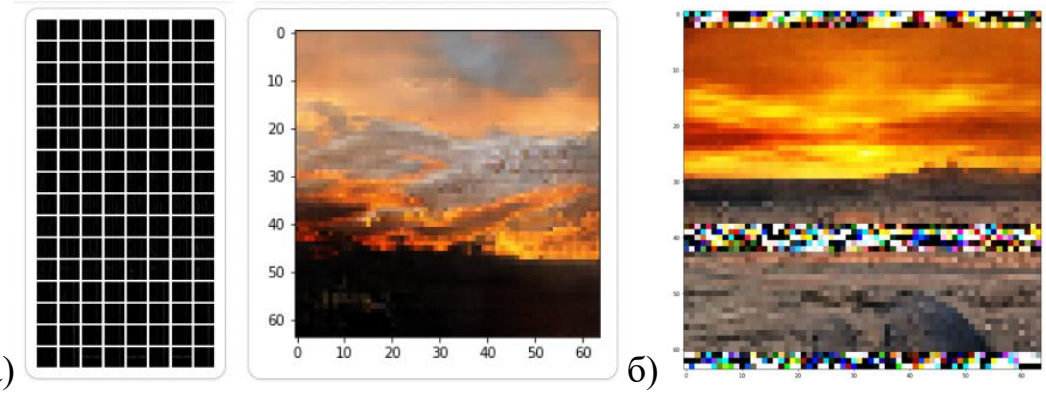

Рисунок 3.34 – а) Активація передостаннього шару і вихідне зображення, б) Результат заповнення нулями рядків активацій передостаннього шару

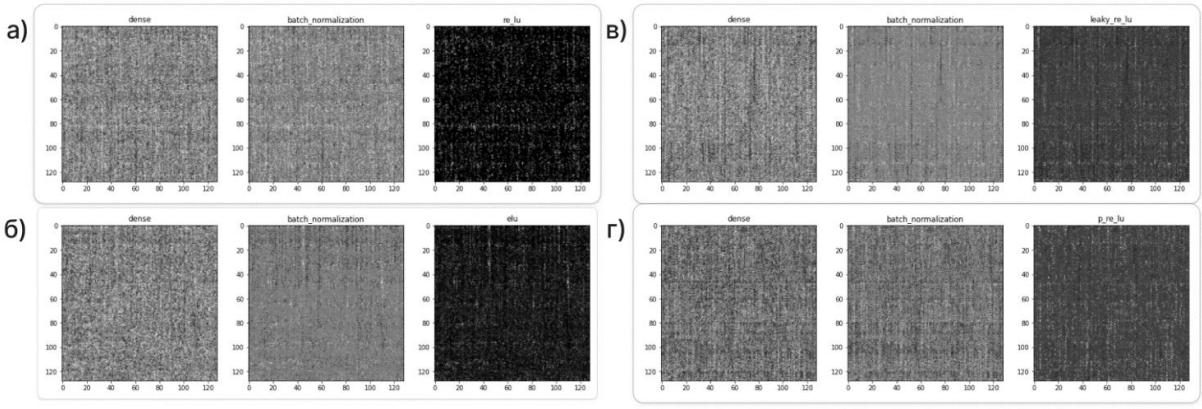

Рисунок 3.35 – Активацій для перших шарів генератора з функцією активації а) ReLU, б) ELU, в) LeakyReLU, г) PreLU

Таким чином, можна завершити дослідження в рамках даної роботи. Отримані результати від навчання генерування цифр зібрано та структуровано у таблицю 3.3, а також опубліковано в матеріалах тез доповіді [14]. Таблиця містить інформацію про час навчання, кількість епох, розмір датасету та зображень, якість генерування за оцінкою НК, кількість параметрів моделей. У таблицю 3.4 зібрано інформацію про тестування з ФА, включаючи пейзажі.

Таблиця 3.3 – Результати експериментів з моделями відносно якості генерування, що були оцінені незалежними класифікаторами

Генератор			Цифри (шум)			
			2D	*3D, CGAN*		*4D*
t навчання, *год*			15	10	5,5	3
$t_{сер}$ на 1 епоху, *с*			10	20	0,125	10
n епох, *тис*			5	2	60	1
Датасет, *тис*, shape			60, 28 x 28 x 1			
Пам’ять, ГБ			0,18			
НК, ε, %			1,5	3,2	0,084	8,29
params	G	Total	1 101 632	1 114 176	1 114 206	1 126 720
		Trainable	1 076 160	1 088 704	1 088 734	1 101 248
		Non-trainable	25 472			
	D	Total	212 865	212 865	220 705	212 865

Таблиця 3.4 – Результати тестування моделей з різними ФА

Цифри				
Функція активації	ReLU	LeakyReLU	PReLU	ELU
Найбільша похибка, %	15,61	14,28	7,89	9,67
Час навчання	9 с – 1 год 15 хв 06 с	10 с – 1 год 22 хв 37 с	10 с – 1 год 15 хв 14 с	9 с – 1 год 19 хв 32
Параметри генератора	1,126,720			
Дискримінатора	212,865			
Пейзажі				
Найбільша похибка, %	11,45	9,74	0,92	10,61
Час навчання 1 епохи, *с*	410	410	430	420
Параметри генератора	10 729 344			
Дискримінатора	2 563 713			

За результатами експерименту з функціями активації можна зробити висновок, що ReLU займає перше місце в обох видах задач генерації. В таблиці присутня інформація про час навчання та кількість параметрів моделей.

ВИСНОВКИ

Кінцевим етапом дослідницької роботи буде підсумовування результатів за всіма порушеними темами та експериментами.

Було проведено детальне розкриття змісту, як історія виникнення GAN, загальна схема їх архітектури, опис функціональних та структурних особливостей, а також аналіз можливих проблем.

Оглянуто інформаційне та математичне забезпечення, у тому числі: функції активації, метрики якості, оптимізатори та функції похибки GAN.

Дослідницький розділ зосереджує увагу на вивчені прихованого простору та особливості його інтерпретації. Представлено шаблонні архітектури генератора та дискримінатора, а також описані процеси навчання мереж 2D, 3D та 4D GAN, з подальшим порівнянням результатів між собою. Для цього використано зображення з кожної епохи, карти станів областей класів та значень похибки незалежного класифікатора. Досліджено поведінку остаточних версій моделей після їх навчання, що подано у вигляді інтерактивних графіків. Окремою підтемою стали генеративно-змагальні мережі з умовою CGAN, їх особливості та приховані простори.

Проведено експерименти з впливом функцій активацій на якість генерації зображень рукописних цифр та фото реальних пейзажів. В результаті тестуванням незалежним класифікатором визначено, що найкраще зарекомендувала себе функція ReLU. Додатково наведено внутрішні карти активацій проміжних шарів, на яких видно, як саме функції активацій трансформують дані.

За результатами досліджень було опубліковано матеріали тез доповіді [14], наукову статтю у фаховому виданні Національної академії наук України «Штучний інтелект» [15] та отримано свідоцтво про реєстрацію авторського права [16]. Повний проєкт роботи розміщено на Google Disk, YouTube, Figma [17, 18, 19].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. A neural network from scratch. Part 4: Better, faster, stronger. URL: https://towardsdatascience.com/part-4-better-faster-stronger-dd6ded07b74f

2. Курс Goggle “Machine Learning”. GAN Variations. URL: https://developers.google.com/machine-learning/gan/applications?hl=ru (Дата звернення 24.09.2025).

3. David Foster. Generative Deep Learning. O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. 2019. 330 с.

4. Orhan Gazi Yalçın. Applied Neural Networks with TensorFlow 2: API Oriented Deep Learning with Python. Springer Science+Business Media New York,1 NY Plazar, New York, NY 10014. 2021. 306 с.

5. Santanu Pattanayak. Pro Deep Learning with TensorFlow 2.0: A Mathematical Approach to Advanced Artificial Intelligence in Python. Apress Media, LLC, 1 New York Plaza, New York, NY 10004. 2023. 667 с.

6. Офіційний сайт API «Keras»: Datasets. URL: https://keras.io/api/datasets/ (Дата звернення: 24.09.2025).

7. Встановлення Tensorflow та допоміжних програм. URL: https://www.tensorflow.org/install/pip (Дата звернення: 24.09.2025).

8. Офіційний форум Kaggle. What is a Kaggle. URL: https://www.kaggle.com/discussions/general/328265 (Дата звернення: 24.09.2025).

9. Функція активації RELU. Офіційна документація Tensorflow. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/ReLU (Дата звернення: 24.09.2025).

10. Aurelien Geron. Hands-On Machine Learning with Scikit-Learn and TensorFlow. O’Reilly Media, Inc. 2017. 684 с.

11. Liangqu Long, Xiangming Zeng - Beginning Deep Learning with TensorFlow – 2022

12. Binary Cross Entropy: Where To Use Log Loss In Model Monitoring. URL: https://arize.com/blog-course/binary-cross-entropy-log-loss/ (Дата звернення: 24.09.2025).

13. Офіційний сайт API Keras. URL: https://keras.io/api/optimizers/ (Дата звернення: 24.09.2025).

14. ...

....

... Міжнародна наукова конференція (... 2024 р., м. Київ).

...

15. ...

... Artificial intelligence. National Academy of Sciences of Ukraine In-stitute of Artificial Intelligence Problems MES of Ukraine and NAS of Ukraine. 2024 ...

16. Свідоцтво про реєстрацію авторського права на твір № ... від ...2024 Україна. Комп’ютерна програма: ...

...

(Україна). – ...2025 р. Бюл. № ...

17. Посилання на програмний проєкт роботи. URL: ...

... (Дата звернення: 24.09.2025).

18. Повний відео-запис презентації роботи. URL: ... (Дата звернення: 24.09.2025).

19. Презентація наукової роботи з детальними слайдами у Figma. URL: ...

... (Дата звернення: 24.09.2025).