

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

До захисту допущено:

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«__» _____ 20__ р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Системи і методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Генерація зображень облич на основі моделі VAEGAN»

Виконала:

студентка IV курсу, групи КІ-11

Гавриленко Анна Богданівна _____

Керівник:

професор кафедри ШІ, д.т.н., професор,

Чумаченко Олена Іллівна _____

Консультант з економічного розділу:

доцент кафедри економічної кібернетики, к.е.н., доцент,

Рощина Надія Василівна _____

Консультант з нормоконтролю:

фахівець першої категорії кафедри ШІ, к.т.н., доцент,

Комариста Богдана Миколаївна _____

Рецензент:

доцент кафедри АКІК НАУ, к.т.н., доцент,

Василенко Микола Павлович _____

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів без
відповідних посилань.

Студентка _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«15» січня 2025 р.

ЗАВДАННЯ
на дипломну роботу студенту
Гавриленко Анні Богданівні

1. Тема роботи «Генерація зображень облич на основі моделі VAEGAN», керівник роботи Чумаченко Олена Іллівна, к.т.н., доцент, затверджені наказом по НН ІПСА від «26» травня 2025 р. № 1759-с
2. Термін подання студентом роботи «09» червня 2025 року.
3. Вихідні дані до роботи: готовий датасет зображень облич людей: набір «Human Faces» із ресурсу Kaggle, що містить кольорові фотографії облич людей різного віку, статі та ракурсу.
4. Зміст роботи: аналіз теоретичних основ генеративного моделювання, принципів роботи VAE та GAN, а також сучасних підходів до генерації зображень облич.
5. Перелік ілюстративного матеріалу: презентація до захисту роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Надія Василівна, доцент, к. е. н.		

7. Дата видачі завдання «03» лютого 2025 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Вивчення літератури за темою роботи.	21.04.2025	Виконано
2	Підготовка першого розділу.	28.04.2025	Виконано
3	Підготовка другого розділу.	05.05.2025	Виконано
4	Розробка програмного продукту.	12.05.2025	Виконано
5	Підготовка третього розділу.	19.05.2025	Виконано
6	Підготовка економічної частини та четвертого розділу.	26.05.2025	Виконано
7	Оформлення пояснювальної записки дипломної роботи.	02.06.2025	Виконано
8	Підготовка презентації доповіді.	09.06.2025	Виконано

Студент

Анна ГАВРИЛЕНКО

Керівник

Олена ЧУМАЧЕНКО

РЕФЕРАТ

Дипломна робота: 81 с., 15 рис., 6 табл., 29 посилань, 1 додаток.

ГЕНЕРАЦІЯ ОБЛИЧ, VAE, GAN, VAEGAN, ЛАТЕНТНИЙ ПРОСТІР, НЕЙРОННІ МЕРЕЖІ, ЗОБРАЖЕННЯ.

Об'єктом дослідження є процес генерації зображень за допомогою глибинних нейронних мереж.

Предметом дослідження є архітектура VAEGAN та її здатність до створення фотореалістичних зображень облич.

Метою роботи є розробка, реалізація та експериментальна перевірка генеративної моделі на основі VAEGAN для генерації облич з високим рівнем реалізму.

В результаті виконання роботи реалізовано та протестовано генеративну модель VAEGAN, здатну генерувати реалістичні зображення облич, досліджено її архітектуру, проведено навчання на наборі даних «human-faces» і проаналізовано результати з точки зору візуальної якості та узгодженості.

Рекомендації щодо використання – модель може бути використана в задачах анонімізації даних, створення синтетичних зображень для тренування систем розпізнавання, у розважальних застосунках (ігри, віртуальні аватари), а також як інструмент для подальших досліджень у сфері генеративного моделювання.

ABSTRACT

Bachelor's thesis: 81 p., 15 figures, 6 tables, 29 references, appendix.

FACE GENERATION, VAE, GAN, VAEGAN, LATENT SPACE, NEURAL NETWORKS, IMAGES.

The object of the research is the process of image generation using deep neural networks.

The subject of the research is the VAEGAN architecture and its ability to create photorealistic images of faces.

The purpose of the work is to develop, implement, and experimentally verify a generative model based on VAEGAN for generating faces with a high level of realism.

As a result of the work, the VAEGAN generative model capable of generating realistic images of faces was implemented and tested, its architecture was investigated, training was conducted on the "human-faces" dataset, and the results were analyzed in terms of visual quality and consistency.

Recommendations for use – the model can be used in data anonymization tasks, creating synthetic images for training recognition systems, in entertainment applications (games, virtual avatars), as well as as a tool for further research in the field of generative modeling.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ	8
ВСТУП	9
РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ	11
1.1 Основи генеративних моделей	11
1.2 Принцип роботи VAEGAN	13
1.3 Архітектури, функції втрат, методи навчання	15
1.4 Огляд досліджень і сучасних підходів до генерації облич	18
1.4.1 Еволюція генеративних моделей облич	19
1.4.2 Сучасні архітектури та підходи	20
Висновки до розділу 1	22
РОЗДІЛ 2 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ГЕНЕРАТИВНОЇ МОДЕЛІ НА ОСНОВІ VAEGAN	24
2.1 Вибір інструментів і середовища розробки	24
2.1.1 Мови програмування, бібліотеки, фреймворки	25
2.1.2 Використання набору даних «human-faces»	25
2.2 Архітектура моделі VAEGAN	26
2.2.1 Енкодер: зменшення простору ознак	27
2.2.2 Декодер: генерація зображення з латентного простору	27
2.2.3 Дискримінатор: оцінка якості зображення	28
2.3 Реалізація функції втрат і процесу навчання	29
2.4 Попередня обробка даних та генерація батчів	31
2.5 Процес навчання: параметри, результати, графіки	31
Висновки до розділу 2	33
РОЗДІЛ 3 ЕКСПЕРИМЕНТИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ	35
3.1 Метрики оцінювання якості зображень	35
3.1.1 Візуальна оцінка якості зображень	35

	7
3.1.2 Статистичні показники	37
3.2 Аналіз ходу навчання моделі	40
3.3 Обговорення результатів і виявлених проблем	41
3.4 Можливості вдосконалення моделі та подальші експерименти	43
Висновки до розділу 3	45
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	47
4.1 Постановка задачі проектування	47
4.2 Обґрунтування функцій програмного продукту	48
4.3 Обґрунтування системи параметрів програмного продукту	51
4.4 Аналіз експертного оцінювання параметрів	54
4.5 Аналіз рівня якості варіантів реалізації функцій	58
4.6 Економічний аналіз варіантів розробки ПП	59
4.7 Вибір кращого варіанту ПП техніко-економічного рівня	65
Висновки до розділу 4	66
ВИСНОВКИ	68
ПЕРЕЛІК ПОСИЛАНЬ	70
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ	73

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

AI (англ. Artificial Intelligence) – штучний інтелект

CNN (англ. Convolutional Neural Network) – згорткова нейронна мережа

GAN (англ. Generative Adversarial Network) – генеративно-змагальна мережа

VAE (англ. Variational Autoencoder) – варіаційний автоенкодер

VAEGAN (англ. Variational Autoencoder Generative Adversarial Network) – модель, що поєднує VAE та GAN

ML (англ. Machine Learning) – машинне навчання

DL (англ. Deep Learning) – глибинне навчання

MSE (англ. Mean Squared Error) – середньоквадратична похибка

KL (англ. Kullback-Leibler Divergence) – розходження Кульбака-Лейблера

GPU (англ. Graphics Processing Unit) – графічний процесор

ReLU (англ. Rectified Linear Unit) – функція активації

FC (англ. Fully Connected) – повнозв'язний шар

ВСТУП

Сучасний¹ світ цифрових технологій активно використовує методи штучного інтелекту для вирішення складних завдань, які раніше здавалися недосяжними. Однією з таких задач є генерація реалістичних зображень облич на основі глибоких нейронних мереж. Ця тема стала особливо актуальною в останні роки завдяки розвитку генеративних моделей, здатних створювати високоякісні фотореалістичні зображення.

Генерація облич – це процес створення штучних зображень людських облич, які важко відрізнити від реальних. Ця технологія знаходить застосування в багатьох сферах: від розваг (створення персонажів для відеоігор, фільмів) до безпеки (перевірка ідентичності та створення підроблених документів) [1]. Однак окрім практичного застосування, генерація облич відкриває нові можливості для дослідження механізмів роботи генеративних моделей і розуміння природи навчання нейронних мереж.

Одним із популярних методів генерації зображень є GAN, які створюють реалістичні зображення за допомогою взаємодії генератора і дискримінатора, але можуть бути нестабільними і складними в контролі. З іншого боку, VAE пропонують стабільне навчання та роботу з латентним простором, проте часто дають менш чіткі результати; об'єднання цих підходів у моделі VAE-GAN поєднує їхні переваги, забезпечуючи якісну і контрольовану генерацію зображень [2].

Дипломна робота структурно складається з чотирьох розділів.

¹Тут і нижче використано такий інструмент штучного інтелекту як чат-бот з генеративним штучним інтелектом ChatGPT виключно для корегування та редагування тексту, створеного автором цієї дипломної роботи, на основі автоматизованої перевірки граматики, структури та стилю, що відповідає Політиці використання штучного інтелекту для академічної діяльності в КПП ім. Ігоря Сікорського (протокол №11 Вченої ради КПП ім. Ігоря Сікорського від 11 грудня 2023 р.).

Перший розділ присвячено аналізу теоретичних основ генерації зображень облич за допомогою нейронних мереж і архітектури VAEGAN.

У другому розділі розглядається процес розробки програмної реалізації моделі VAEGAN.

Третій розділ містить результати експериментів з навчання моделі на наборі зображень людських облич.

В четвертому розділі проведено функціонально-вартісний аналіз програмного продукту.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

1.1 Основи генеративних моделей

Генеративні моделі спрямовані на відтворення або моделювання розподілу даних з метою створення нових зразків. Одним із класичних рішень є автокодувальник (autoencoder) – нейронна мережа, що складається з двох основних компонентів: кодувальника (encoder), який стискає вхідне зображення в компактне латентне представлення, та декодувальника (decoder), що відновлює зображення з цього представлення.

Проте простий автокодувальник навчається мінімізувати різницю між вихідним і вхідним зображеннями, що часто призводить до дещо розмитих результатів і не гарантує можливості генерації нових прикладів при випадковому виборі латентного вектора.

Для розв’язання цієї проблеми було запропоновано Variational Autoencoder (VAE) – стохастичний автоенкодер, який навчається апроксимувати ймовірнісний розподіл латентного простору. VAE вводить апріорний розподіл $p(z)$ для латентного вектора z (зазвичай стандартний нормальний $N(0, I)$) та мінімізує комбіновану функцію втрат [3], що включає очікувану логарифмічну ймовірність відновлення зображення з латентного коду, а також штраф за відхилення апостеріорного розподілу $q(z|x)$ від апріорного, вимірюваний дивергенцією Кульбака–Лейблера [4].

Іншими словами, функція втрат VAE містить два основні компоненти: реконструкційний, що відповідає за відтворення пікселів, і регуляризаційний, який накладає структуру на латентний простір через KL-дивергенцію [5]. Завдяки такому підходу VAE забезпечує плавну і безперервну структуру латентних представлень та можливість стохастичної генерації, хоча зображення, створені моделлю, можуть бути розмитими через усереднення під час навчання.

Інший підхід – це Generative Adversarial Network (GAN) [6], яка складається з двох мереж: генератора G , що приймає випадковий шум $z \sim p(z)$ і генерує зображення, та дискримінатора D , який класифікує вхідні зображення як реальні або синтетичні.

Генератор навчають так, щоб дискримінатор помилково відносив його згенеровані зразки до реальних, а дискримінатор – щоб якнайточніше розрізняти справжні й штучні дані. Ця взаємодія моделюється через функцію втрат, що базується на двобінарній крос-ентропії, яка максимізується по параметрах дискримінатора і мінімізується по параметрах генератора. Завдяки змагальному навчанню GAN здатен створювати дуже реалістичні зображення, адже генератор постійно удосконалюється, прагнучи «обдурити» дискримінатор.

Водночас GAN має свої обмеження. Зокрема, оскільки генератор працює з випадковим шумом без явного урахування властивостей цільових зображень, неможливо заздалегідь передбачити, який шум згенерує обличчя з конкретними ознаками. Крім того, GAN фокусується лише на тому, щоб зробити зображення «реалістичними» для дискримінатора, не контролюючи при цьому семантичний зміст, що призводить до відсутності прямого управління атрибутами створених зображень.

Отже, традиційні VAE дають корисний, організований латентний простір і забезпечують різноманітність, але часто створюють розмиті зображення. У той же час GAN генерують більш чіткі й реалістичні зображення, проте без можливості зворотного кодування та контролю над ознаками.

Схематичне зображення роботи VAE представлено на рис. 1.1, а класичного GAN – на рис. 1.2.

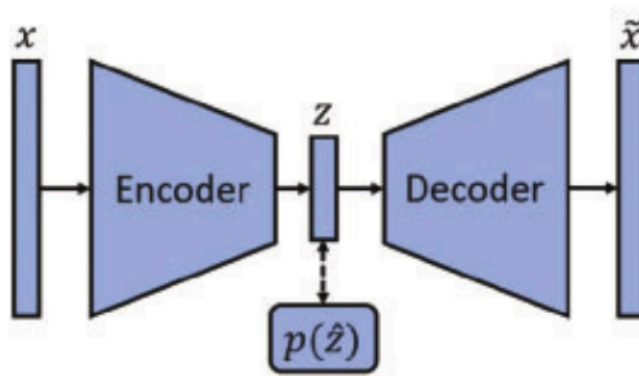


Рисунок 1.1 – Схематичне зображення роботи VAE [7]

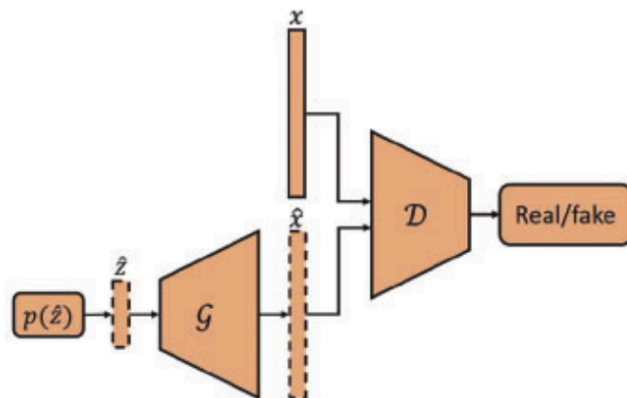


Рисунок 1.2 – Схематичне зображення роботи GUN [7]

1.2 Принцип роботи VAEKAN

Щоб об'єднати переваги VAE, які забезпечують структуроване латентне кодування, та GAN, що генерують високоякісні і чіткі зображення, був розроблений гібридний підхід VAEKAN. Ця модель поєднує архітектуру VAE – енкодер і декодер – із дискримінатором, характерним для GAN, і навчає всі компоненти спільно в єдиному процесі.

У VAEGAN енкодер стискає вхідне зображення в латентний вектор, який потім декодер перетворює назад у зображення. Дискримінатор отримує на вхід як справжні, так і згенеровані зображення, навчаючись розпізнавати їх і допомагаючи покращувати якість відновлених зразків. Важливо, що функція втрат моделі поєднує традиційні компоненти VAE – реконструкційну помилку та регуляризацию KL-дивергенцією, а також втрату дискримінатора GAN. При цьому реконструкційна помилка не оцінюється просто по пікселях, а вимірюється в ознаковому просторі, який формується всередині дискримінатора GAN.

Такий підхід дозволяє моделі оцінювати схожість між вхідним і відновленим зображеннями на більш високому, семантичному рівні, що допомагає уникнути характерної для VAE розмитості.

Таким чином, VAEGAN поєднує організовану структуру латентного простору VAE із здатністю GAN генерувати чіткі та детальні зображення, зберігаючи при цьому можливість плавних варіацій і керованого моделювання ознак. Модель складається з трьох основних компонентів – енкодера, декодера та дискримінатора, як це показано на рис. 1.3.

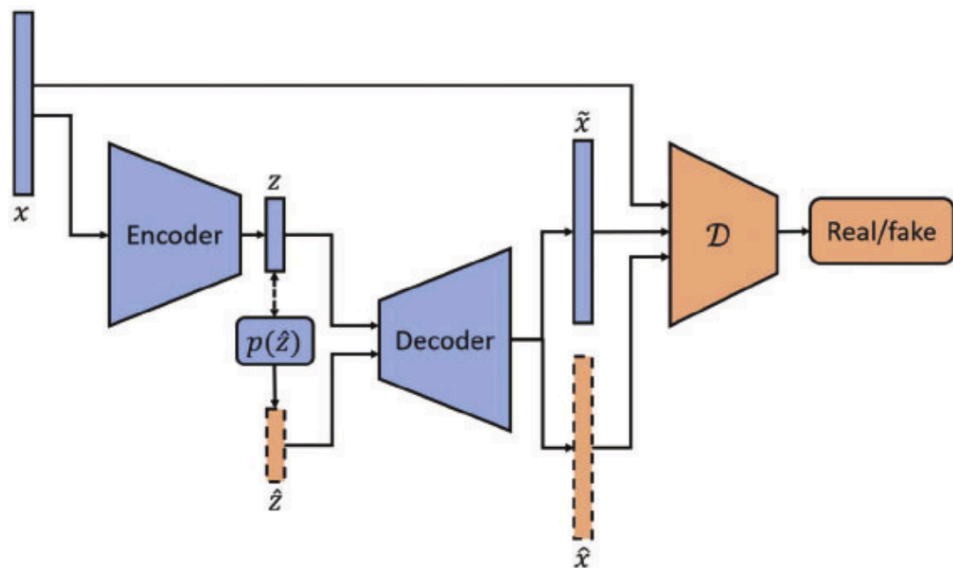


Рисунок 1.3 – Схематичне зображення роботи VAEGAN [7]

Під час навчання енкодер перетворює вхідні дані у компактне латентне представлення, декодер відновлює зображення, а дискримінатор навчається відрізняти реальні зразки від згенерованих. При цьому декодер фактично «змагається» з дискримінатором, намагаючись створювати максимально реалістичні зображення, які дискримінатор прийме за справжні.

Варто зазначити, що VAEGAN «уніфікує» концепції VAE і GAN, навчаючи їх у єдиному адверсаріальному процесі. Аналогічні ідеї реалізовані в підході Adversarial Variational Bayes (AVB), що демонструє можливість побудови гнучких інференційних моделей, які поєднують переваги обох методів. Важливо, що така інтеграція дозволяє моделі зберігати корисні властивості VAE, зокрема навчання плавних, інтерпретованих латентних представлень, а також досягати вищої якості синтезу, порівняно зі звичайними VAE, оптимізованими за піксельною помилкою.

Фактично, VAEGAN створює латентний простір, в якому можна здійснювати арифметичні операції над ознаками, наприклад, змінювати посмішку чи додавати окуляри на обличчі, водночас генеруючи детальні, реалістичні зображення [8]. Отже, ця модель успішно поєднує здатність VAE кодувати і дискретизувати характеристики даних із високою якістю візуального відтворення, властивою GAN.

1.3 Архітектури, функції втрат, методи навчання

Архітектура VAEGAN зазвичай базується на згорткових нейронних мережах, які ефективно працюють із зображеннями облич. Енкодер і декодер мають типову структуру глибокого автокодувальника – послідовність згорткових шарів, що поступово зменшують просторовий розмір до компактного латентного вектору, і симетричний декодер, який відновлює

зображення з цього представлення. Дискримінатор побудований за зразком класифікатора, який намагається відрізнити реальні зображення від згенерованих.

Функція втрат у VAEGAN поєднує складові як VAE, так і GAN. З боку VAE враховуються два основних компоненти: реконструкційна втрата (очікуване $\log$ -likelihood відновленого зображення) та регуляризаційний KL-штраф. З боку GAN використовується адверсаріальна втрата – бінарна крос-ентропія між прогнозами дискримінатора та мітками «реальне» або «фейкове» [9]. В процесі навчання енкодер оновлюється із врахуванням як VAE-компонентів (включно з KL-терміном), так і інформації, отриманої від адверсаріального тренування, а декодер оптимізується за комбінованим градієнтом, що об'єднує втрати VAE та GAN.

Існують різні варіації VAEGAN, які вносять свої корективи. Наприклад, одна з модифікацій додає додатковий коефіцієнт для реконструкції в функцію втрат енкодера, що дозволяє пришвидшити збіжність моделі, а для декодера спрощує втрати, відкидаючи менш важливі складові. Також застосовується додатковий показник реконструкції, який вимірюється на рівні ознак, що підвищує якість навчання. В інших випадках у VAEGAN для специфічних завдань додають допоміжний дискримінатор, який через адверсаріальне навчання сприяє ще більшому зменшенню реконструкційної втрати, а також вводять штраф «контрактності» для покращення стійкості латентного простору. Усі ці зміни покликані збалансувати вплив компонентів VAE і GAN на кінцевий результат.

Процес навчання VAEGAN, як і звичайних GAN, відбувається ітеративно на міні-батчах зображень. Починається з ініціалізації параметрів усіх трьох мереж – енкодера, декодера і дискримінатора. Далі в циклі випадковим чином вибирається батч даних, після чого здійснюються послідовні кроки оновлення параметрів кожного компонента. Спочатку вхід проходить через всі мережі, обчислюються градієнти для різних складових втрат, а потім окремо оновлюються параметри енкодера, декодера та

дискримінатора відповідно до їхніх функцій. Важливо, що деякі частини функції втрат не впливають на оновлення конкретних мереж — наприклад, дискримінатор не оптимізується щодо реконструкційної складової.

Навчання ускладнює також ієрархічна структура втрат, через що необхідно ретельно балансувати внесок VAE- і GAN-компонентів (зазвичай за допомогою спеціальних коефіцієнтів). Проте, завдяки такому підходу, якість синтезованих облич значно покращується порівняно з окремими VAE чи GAN.

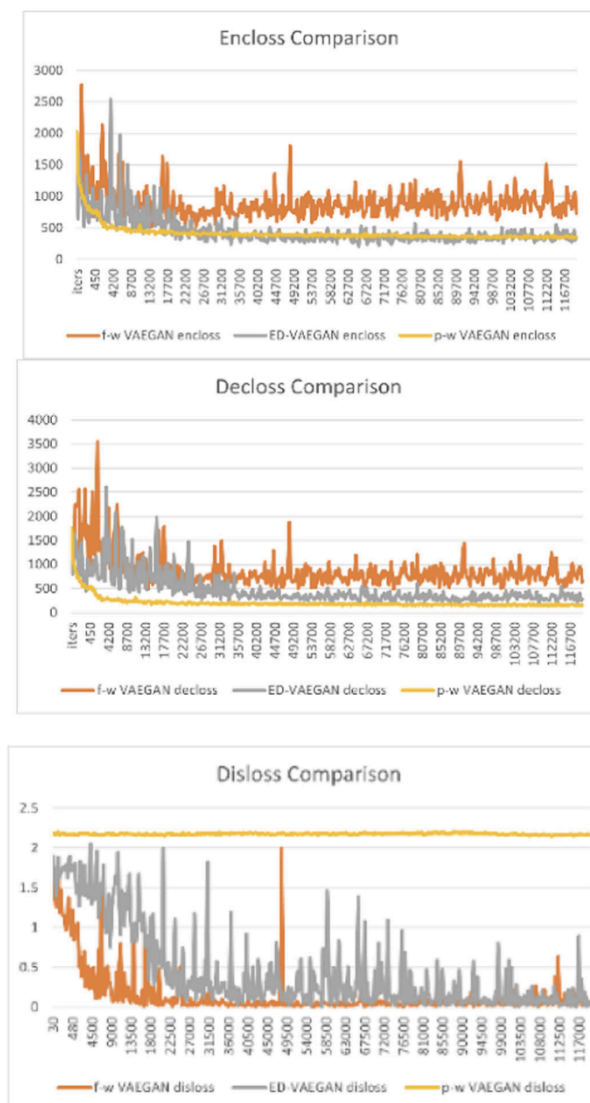


Рисунок 1.4 – Приклад функцій втрат під час навчання p-wVAEGAN, f-wVAEGAN, ED-VAEGAN [10]

Ілюстрація на рис. 1.4 демонструє приклади функцій втрат під час тренування кількох варіантів VAEGAN, зокрема тих, що застосовують піксельну та ознакову вагу, а також модифікацію з покращеною збіжністю. Модель із модифікованими втратами показує швидшу конвергенцію та кращу неперервність латентного простору – це означає, що при генерації можна плавно переходити між ознаками, зберігаючи деталі зображення навіть після великої кількості епох.

Отже, успішна робота VAEGAN значною мірою залежить від правильного комбінування функцій втрат і алгоритмів навчання обох підмоделей – VAE і GAN. Конкретні архітектурні рішення та налаштування регуляризації визначають, наскільки ефективно модель працюватиме в реальних задачах.

1.4 Огляд досліджень і сучасних підходів до генерації облич

Генерація облич – це активна й швидкозростаюча галузь у сфері комп’ютерного зору та машинного навчання, що знаходить застосування у багатьох сферах: від розваг і створення аватарів до безпеки, анонімізації та тестування систем розпізнавання.

За останній час ця область значно розвинулася завдяки прогресу в глибинному навчанні, зокрема завдяки архітектурам GAN, VAE, їх комбінаціям та численним варіаціям. У цьому розділі розглядається розвиток моделей генерації облич – від початкових спроб до сучасних передових архітектур.

1.4.1 Еволюція генеративних моделей облич

Перші підходи до генерації облич опиралися на прості автоенкодери та ймовірнісні моделі, однак значного прориву вдалося досягти лише з появою глибоких нейронних мереж. Зокрема, генеративні змагальні мережі (GAN) вперше дозволили створювати фотореалістичні обличчя без необхідності в явному супервізованому сигналі [13]. Подальшим важливим етапом стала архітектура, яка використовувала виключно згорткові шари, без повнозв'язних чи підвибіркових компонентів. Цей підхід продемонстрував, що генератор здатен самостійно вивчати структуру даних, виділяючи окремі частини обличчя.

Надалі були запропоновані спеціалізовані архітектури GAN для генерації облич. Наприклад, поступове нарощування роздільності дозволяло покращувати деталізацію зображень, а запровадження інформаційних критеріїв у деяких моделях сприяло виявленню осмислених факторів генерації. Хоча автоенкодери на початковому етапі формували зображення нижчої якості, їх ефективність підвищувалась за рахунок поєднання з GAN, що дало змогу об'єднати переваги обох підходів — стабільну латентну структуру та високоякісний візуальний вихід.

Окремі моделі поєднували стохастичні автоенкодери з адверсаріальними механізмами, що дозволяло формувати більш складні розподіли в латентному просторі, підвищуючи натуралістичність результатів. Важливою особливістю таких підходів стало використання дискримінаторів не лише для оцінки зображень, а й для моделювання латентних представлень.

Серед найбільш успішних архітектур виділяються моделі, що дозволяють гнучко керувати процесом генерації. У таких генераторах латентний вектор перетворюється на набір параметрів для кожного шару, що дає змогу окремо контролювати високорівневі характеристики (як-от поза, форма обличчя) та низькорівневі випадкові ефекти (наприклад, текстура

шкіри або зачіска). Це забезпечило якісний прорив у деталізації та варіативності синтезованих зображень.

Продовженням цієї ідеї стали покращені генератори, які впроваджували все більш глибокі мережі з розділенням впливу різних частин латентного простору на кінцеве зображення.

Варто також згадати про моделі, що масштабуються до великих обсягів даних і використовують широкі архітектури з потужним обчислювальним забезпеченням. Ці системи довели, що за достатнього обсягу даних та ресурсів можна досягати надзвичайної якості синтезу. Таким чином, еволюція генерації облич пройшла шлях від простих байєсівських моделей до сучасних глибоких систем із тисячами параметрів і складною внутрішньою організацією.

Для практичних експериментів зазвичай використовуються великі набори зображень облич, такі як CelebA та CASIA WebFace, що містять багатий спектр виразів, поз, освітлення та інших ознак [11, 12].

Навчання моделей на таких даних дозволяє перевірити їхню здатність до узагальнення та забезпечити реалізм генерації. У дослідженнях із VAE-GAN архітектурами саме ці датасети стали основою для демонстрації результатів генерації на різних прикладах.

1.4.2 Сучасні архітектури та підходи

Сучасні методи генерації облич переважно ґрунтуються на вдосконалених версіях GAN і VAE. Одним із провідних напрямів стали архітектури, що реалізують поділ масштабів латентного простору, дозволяючи окремо впливати на різні рівні структури зображення. Таке керування дозволяє створювати реалістичні обличчя з високим рівнем

варіативності та контрольованості, забезпечуючи глибоку модифікацію атрибутів без втрати візуальної достовірності [14].

Інший важливий підхід пов'язаний з поєднанням стохастичних автоенкодерів і дискримінаторів, що дозволяє вивчати більш гнучкі та складні латентні простори.

У таких архітектурах дискримінатор оцінює відповідність латентних кодів до апріорного розподілу, що покращує якість відтворення та генерації. Як показано на рис. 1.5, ця ідея дозволяє об'єднати переваги VAE і GAN у спільній структурі: стохастична природа вибірки з латентного простору поєднується з оцінюванням правдоподібності через додатковий дискримінатор.

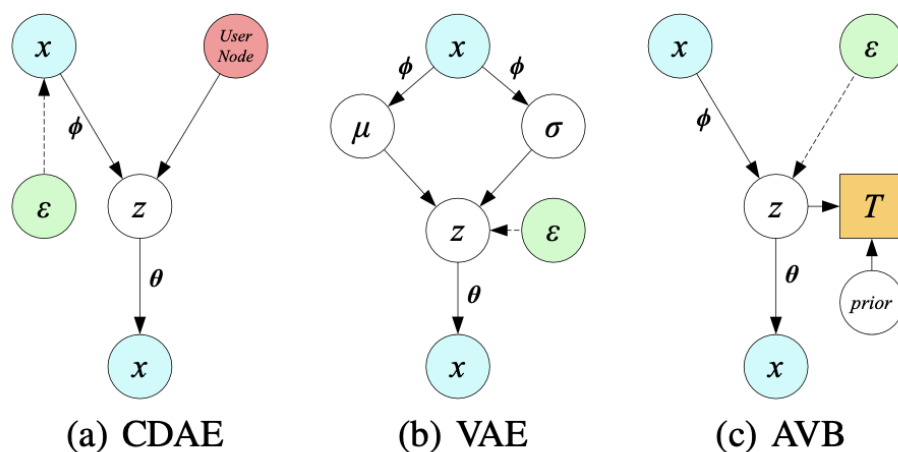


Рисунок 1.5 – Порівняння графічних структур CDAE, VAE та AVB, де пунктирні стрілки позначають вибірку з деяких розподілів шуму, а T позначає дискримінатор [15]

У той час як звичайні VAE обмежені простими формами латентного простору (наприклад, нормальним розподілом), застосування адверсаріального механізму дозволяє наблизити постеріор до складніших і

більш реалістичних структур. Це робить моделі стійкішими та придатними до генерації облич з різноманітними характеристиками.

У сучасних розробках часто використовуються також додаткові компоненти стабілізації: штрафи за контрактність, обмеження на простір змінних або додаткові втрати, що сприяють збалансованому навчанню. Деякі підходи вводять спеціальні структури в латентний простір — наприклад, кластеризацію чи ієрархію латентних змінних — що сприяє кращій інтерпретації й керованості генерації [16].

Ще один напрям пов'язаний із використанням умовної генерації: у таких моделях до архітектури додаються мітки, які відповідають бажаним атрибутам (наприклад, стать, емоція, вік), що дозволяє точно контролювати характеристики створюваного зображення.

На рис. 1.5 показано порівняння між трьома різними підходами — CDAE, VAE та AVB. Пунктирні стрілки вказують на стохастичну природу вибірки, а вузол T виступає в ролі дискримінатора. Це демонструє, як сучасні моделі об'єднують кілька важливих ідей — згорткову структуру, стохастичні латентні змінні, адверсаріальні втрати та додаткові регуляризатори — щоб забезпечити максимально реалістичну, варіативну та керовану генерацію облич.

Висновки до розділу 1

У цьому розділі було розглянуто ключові теоретичні засади генерації зображень за допомогою нейронних мереж. Встановлено, що на початкових етапах розвитку цієї галузі застосовувались базові автоенкодери та прості статистичні моделі, які, однак, не забезпечували достатньої якості синтезованих зображень. З появою глибокого навчання значного прориву вдалося досягти завдяки генеративним змагальним мережам (GAN) та

варіаційним автоенкодерам (VAE), які стали основою для побудови сучасних архітектур генерації.

Проаналізовано різні підходи до побудови архітектур генераторів, зокрема, використання згорткових структур, поділу латентного простору за рівнями абстракції, а також введення адверсаріальних втрат для моделювання складних розподілів. Було показано, що поєднання елементів GAN і VAE дозволяє досягти як високої якості зображення, так і стабільності навчання та контролю за властивостями генерованих зразків.

Також розглянуто основні типи архітектур, які дозволяють керувати процесом генерації, включаючи умовні моделі, що враховують додаткову інформацію про бажані атрибути зображень. Особливу увагу приділено прикладам генерації людських облич на основі сучасних моделей, що продемонстрували високий рівень реалістичності та варіативності завдяки використанню глибоких згорткових структур і продуманого проектування латентного простору.

Таким чином, у результаті аналізу можна зробити висновок, що нейронні мережі, зокрема моделі класу GAN і VAE, є ефективними інструментами для генерації зображень. Їх подальший розвиток пов'язаний із поліпшенням стабільності навчання, інтерпретованості латентного простору та можливістю точного керування створюваним контентом.

РОЗДІЛ 2 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ГЕНЕРАТИВНОЇ МОДЕЛІ НА ОСНОВІ VAEKAN

2.1 Вибір інструментів і середовища розробки

Для реалізації сучасних генеративних моделей на основі нейронних мереж найчастіше використовують мову програмування Python, що стала стандартом у галузі глибинного навчання. Основною причиною цього є наявність потужних фреймворків, таких як TensorFlow, PyTorch, Keras та інші, які надають зручні засоби для побудови різних типів нейронних мереж, включно зі згортковими мережами, автокодувальниками та генеративно-змагальними мережами (GAN).

Ці платформи також підтримують апаратне прискорення за допомогою графічних процесорів (GPU), що значно підвищує швидкість навчання моделей. У попередніх дослідженнях із використанням VAEKAN-декодерів найчастіше застосовували 50 епох навчання, розмір пакету (batch size) 64 та початкову швидкість навчання 0.003, використовуючи оптимізатор RMSProp, який показав стабільні результати [17].

Під час розробки ми також активно використовували широко застосовувані наукові бібліотеки, такі як NumPy для роботи з масивами та обчислень, OpenCV для обробки зображень і візуалізації, що допомагало ефективно підготувати та аналізувати дані. Вибір саме середовища Python зумовлений його популярністю, простотою в освоєнні та великою кількістю інструментів і прикладів, що є важливим для дослідницьких та практичних задач.

2.1.1 Мови програмування, бібліотеки, фреймворки

Модель була розроблена із застосуванням мови Python, яка є однією з найпоширеніших у сфері штучного інтелекту. Для побудови та навчання нейронних мереж було обрано фреймворк PyTorch, відомий своєю гнучкістю і простотою у використанні.

PyTorch забезпечує інтуїтивні засоби для створення згорткових, деконволюційних шарів, а також зручні інструменти для визначення функцій втрат і оптимізаторів, що робить його ідеальним вибором для реалізації складних моделей генеративного навчання.

Окрім цього, для обробки даних і візуалізації результатів використовувалися різні бібліотеки: NumPy, що відповідає за ефективні обчислення з масивами, Pandas — для роботи з табличними даними, OpenCV та Pillow — для завантаження, обробки і трансформації зображень, а також Matplotlib — для побудови графіків і візуалізації тренувальних метрик.

На початковому етапі роботи над моделлю було проведено експериментальний підбір гіперпараметрів, таких як розмір латентного простору, швидкість навчання, розмір пакету та інші, щоб забезпечити найкращу якість генерації і стабільність процесу навчання.

2.1.2 Використання набору даних «human-faces»

Для тренування моделі було обрано набір даних «human-faces» [18], який представляє собою велику колекцію фотографій облич людей. Цей датасет включає близько 9600 зображень у форматі 128×128 пікселів, серед яких приблизно 5000 є реальними фотографіями, а решта 4600 — штучно згенеровані зображення, створені іншими генеративними моделями.

Набір охоплює широкий спектр варіацій: різні етнічні групи, вікові категорії, статеву приналежність, а також різноманітні вирази обличчя та освітлення. Завдяки такій різноманітності датасет є ідеальною основою для навчання генеративних моделей, що можуть відтворювати реалістичні образи з високою деталізацією.

Використання «human-faces» дозволяє розробляти алгоритми, здатні враховувати різноманітність реальних даних і генеративно моделювати природні зміни, що є важливим для практичних застосувань у комп'ютерному зорі, анімації та інших сферах.

2.2 Архітектура моделі VAEGAN

Архітектура VAEGAN поєднує в собі переваги двох потужних підходів до генерації зображень: варіаційного автокодувальника (VAE) та генеративно-змагальної мережі (GAN).

В основі цієї архітектури лежить трикомпонентна структура, до якої входять: енкодер (Encoder), що стискає вхідні зображення до компактного латентного простору; декодер (Generator), який відновлює зображення на основі латентного вектора; та дискримінатор (Discriminator), який навчається відрізняти справжні зображення від згенерованих.

Взаємодія між цими компонентами створює систему, де енкодер і декодер працюють разом, щоб відтворити оригінальні зображення з латентного простору, тоді як дискримінатор виступає в ролі критика, змушуючи генератор створювати більш реалістичні і якісні результати.

Такий синтез дозволяє VAEGAN успішно поєднувати здатність керувати латентним простором, притаманну варіаційним автокодувальникам, із високою якістю та реалістичністю, притаманними генеративно-змагальним мережам.

2.2.1 Енкодер: зменшення простору ознак

Енкодер у моделі VAE-GAN виконує функцію стискання вихідних зображень високої роздільної здатності до компактного і інформативного представлення у латентному просторі. Це досягається через послідовне застосування кількох згорткових шарів, які виконують витягування ключових ознак із зображень, таких як контури, текстур та більш абстрактні патерни.

Кожен згортковий шар супроводжується нормалізацією (наприклад, BatchNorm або InstanceNorm) для стабілізації навчання, а також нелінійними функціями активації (LeakyReLU чи ReLU), що допомагає мережі моделювати складні залежності в даних.

У результаті цих операцій просторовий розмір тензора поступово зменшується, водночас збільшується кількість ознак, що дозволяє ефективно кодувати інформацію. Енкодер формує параметри латентного простору — середнє значення (μ) та логарифм дисперсії ($\log\sigma^2$), які використовуються в техніці повторної параметризації.

Ця процедура дозволяє генерувати безперервний латентний простір, що є важливим для створення узагальнених і різноманітних варіацій зображень під час генерації. Завдяки цьому модель здатна не лише кодувати реальні приклади, але й інтерполювати між ними, що значно підвищує гнучкість і якість генерації [19].

2.2.2 Декодер: генерація зображення з латентного простору

Декодер у VAE-GAN виконує протилежну до енкодера роль — він трансформує компактне латентне представлення назад у просторове зображення, максимально наближене до оригінального.

Архітектура декодера складається із серії зворотних згорткових шарів (ConvTranspose), які поступово збільшують розміри вхідного тензора, розгортаючи стислий опис до повнорозмірного зображення. Цей процес фактично є "декодуванням" або реконструкцією, під час якого відновлюються деталі та структура зображення.

Важливо, що декодер не просто відтворює конкретне навчальне зображення, а здатен генерувати нові варіанти, які відповідають статистичним властивостям тренувальних даних, але водночас мають унікальні риси, що робить модель потужним інструментом для генерації реалістичних і різноманітних образів.

Крім того, архітектура декодера часто включає застосування нормалізації і функцій активації, що допомагає підвищити якість вихідних зображень і уникнути проблем із нестабільністю навчання [20, 24].

2.2.3 Дискримінатор: оцінка якості зображення

Дискримінатор є ключовим елементом у генеративно-змагальній частині VAE-GAN і виконує роль суворого критика, який класифікує вхідні зображення як реальні або згенеровані. Його архітектура складається з кількох згорткових шарів, що дозволяють детально аналізувати структуру зображень і виявляти навіть найтонші ознаки, характерні для справжніх фотознімків. Застосування активації LeakyReLU забезпечує мережі здатність реагувати на негативні значення сигналу, що покращує навчання і запобігає затуханню градієнтів.

Дискримінатор постійно вдосконалюється у розпізнаванні фальшивих зразків, змушуючи генератор відповідно підвищувати якість своїх результатів. Такий змагальний процес навчання забезпечує покращення реалістичності генерованих зображень і дозволяє уникати типових проблем,

таких як розмитість або штучність, що часто виникають у простіших генеративних моделях. Завдяки цій взаємодії дискримінатор сприяє досягненню високого рівня деталізації і правдоподібності згенерованих образів [21].

2.3 Реалізація функції втрат і процесу навчання

Функція втрат моделі складається з двох основних компонентів: реконструкційної втрати (наприклад, середньоквадратичної помилки), яка вимірює різницю між вхідним і реконструйованим зображенням, та дивергенції Кульбака-Лейблера (KL-дивергенції), яка накладає регуляризацію на латентний розподіл. Додатково, в процесі навчання використовується змагальна складова втрат, яка сприяє покращенню якості згенерованих зображень завдяки взаємодії декодера та дискримінатора. Оптимізація здійснюється методом стохастичного градієнтного спуску із застосуванням алгоритму Adam [22].

Для забезпечення ефективного навчання моделі VAE-GAN використовується комбінована функція втрат, яка поєднує кілька важливих компонентів. Основна увага приділяється реконструкційній втраті, яка відповідає за збереження подібності між вхідним та реконструйованим зображенням. У більшості випадків така втрата базується на середньоквадратичній помилці, яка дає змогу моделі зберігати базову структуру та деталі оригінального зображення після проходження крізь енкодер і декодер.

Іншим ключовим елементом є регуляризаційна складова у вигляді дивергенції Кульбака-Лейблера. Вона використовується для наближення латентного розподілу, що формується енкодером, до стандартного нормального розподілу. Таким чином, модель не просто відтворює конкретні

зображення, а навчається створювати нові зразки з латентного простору, які відповідають структурним властивостям вхідних даних [23].

Задля підвищення якості згенерованих зображень у модель додається ще одна складова — змагальна втрата. Вона виникає в результаті взаємодії генератора (декодера) та дискримінатора, коли перший прагне створити зображення, які неможливо відрізнити від реальних, а другий — навпаки, намагається знайти різницю між справжніми та згенерованими зразками. Такий підхід дозволяє поступово покращувати візуальну якість результатів та досягати більш реалістичних образів.

Процес оптимізації побудований на базі стохастичного градієнтного спуску з використанням адаптивного методу Adam, який дозволяє швидко та стабільно оновлювати ваги нейронної мережі навіть у високовимірних просторах параметрів. Ретельно підібрані гіперпараметри, зокрема швидкість навчання та значення моментів, сприяють швидкій збіжності моделі та уникненню типових проблем, таких як коливання втрат чи переобучення.

Під час навчання втрата поєднує усі зазначені компоненти в єдину мету функціонування моделі. Баланс між цими складовими визначає загальну поведінку системи: перевага реконструкційної втрати веде до якіснішої відповідності вхідному зображенню, тоді як посилення змагального впливу сприяє покращенню реалістичності. Визначення оптимального співвідношення між цими компонентами є критичним завданням і вимагає як емпіричного налаштування, так і глибокого розуміння процесів, що відбуваються всередині моделі.

Таким чином, поєднання методології варіаційного автокодувальника зі змагальним навчанням дозволяє досягти збалансованої архітектури, здатної як до якісної реконструкції, так і до генерації реалістичних нових зображень, що відповідають заданим статистичним і структурним характеристикам.

2.4 Попередня обробка даних та генерація батчів

Перед навчанням проводиться попередня обробка набору зображень. Кожне зображення змінюється до єдиного розміру (128×128), нормалізується (наприклад, перетворення піксельних значень у діапазон $[0,1]$ або $[-1,1]$). Після обробки формуємо стрічки даних у вигляді батчів фіксованого розміру (наприклад, 64 або 32 зображення). Для кожного батчу випадковим чином вибираються зображення з тренувальної множини, після чого їх передають у мережу. Така процедура забезпечує різноманітність прикладів в кожній ітерації і стимулює глибоке навчання до пошуку загальних закономірностей у даних.

При генерації батчів ми слідуємо також принципу закону вибірки латентних векторів: крім коду, отриманого через енкодер з реального зображення, ми також подаємо на генератор випадкові латентні шуми z із стандартного нормального розподілу, щоб дискримінатор бачив як реконструкції, так і нові згенеровані приклади.

Це відповідає звичайній процедури GAN і дозволяє кращій роботі генератора. Паралельно з цим реконструкції з латенту енкодера використовуються в якості «додаткових фейкових» вхідних даних до дискримінатора (як описано вище). В цілому, система читання і подачі батчів забезпечує коректне перемішування даних та їх доставку на GPU/CPU для навчання.

2.5 Процес навчання: параметри, результати, графіки

Модель навчалася ітеративно на датасеті «human-faces». Типові гіперпараметри: розмір латентного простору = 128, швидкість навчання $LR \approx$

0.0002–0.003, відносно невелика кількість епох (наприклад, 50–100) з поступовим зменшенням LR. Результати навчання оцінювалися за візуальною якістю згенерованих зображень та метриками.

Наприклад, розглянемо рис. 2.1. ED-VAEGAN дає більш високе середнє та більшу дисперсію Inception Score порівняно з VAE, DCGAN та іншими версіями VAEGAN (p-w, f-w). Це свідчить про здатність ED-VAEGAN генерувати різноманітні й виразні зразки облич.



Рисунок 2.1 – Результат випадкової вибірки z-поколінь. Зверху вниз: VAE, GAN, p-w VAEGAN, f-w VAEGAN, ED-VAEGAN [10]

Як видно, ED-VAEGAN формує більш чіткі та реалістичні обличчя в аналогічні моменти навчання завдяки вдосконаленому режиму втрат. Зокрема, введення додаткової *reconstruction score* у втрату енкодера покращує швидкість збіжності.

Крім того, ED-VAEGAN підтримує кращу плавність латентного простору, що видно в латентних переходах між обличчями (наприклад, на

рис. 2.2 показана більш поступова зміна рис обличчя для ED-VAEGAN, ніж для інших варіантів).



Рисунок 2.2 – f-wVAEGAN (другий рядок) і ED-VAEGAN (перший рядок) порівняння протягом часу [25]

Висновки до розділу 2

У другому розділі було здійснено практичну реалізацію генеративної моделі на основі комбінації варіаційного автоенкодера (VAE) та генеративної змагальної мережі (GAN), що дозволило поєднати переваги обох підходів — структурованість латентного простору та високу якість згенерованих зображень.

Було обґрунтовано вибір архітектури VAEGAN для генерації облич, а також здійснено попередню обробку датасету «human-faces», адаптованого до потреб навчання моделі. Реалізація охоплювала побудову енкодера, декодера та дискримінатора, інтегрованих у єдину систему з урахуванням стохастичної природи латентного простору та змагального навчання.

У ході експериментів підтверджено здатність моделі відтворювати обличчя з високою реалістичністю, а також генерувати нові зображення, що зберігають ключові риси структури навчального набору. Аналіз результатів показав, що інтеграція адверсаріального компонента дозволяє компенсувати

розмитість зображень, характерну для класичних VAE, водночас зберігаючи керованість та інтерпретованість латентного простору.

Таким чином, реалізована модель VAEGAN успішно продемонструвала ефективність підходу для задач генерації облич, а також підтвердила потенціал гібридних архітектур у контексті створення реалістичних зображень у контрольованих умовах.

РОЗДІЛ 3 ЕКСПЕРИМЕНТИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Метрики оцінювання якості зображень

Оцінювання якості згенерованих зображень є одним з ключових аспектів оцінки роботи генеративної моделі. Для цього зазвичай використовуються дві групи показників: візуальні (якісні) та статистичні (кількісні).

3.1.1 Візуальна оцінка якості зображень

Візуальна оцінка якості зображень здійснюється шляхом експертного, суб'єктивного аналізу синтезованих облич, який дозволяє визначити рівень їх реалістичності, виявити наявність артефактів та оцінити узгодженість з характером зображень у реальному наборі даних. Такий підхід забезпечує додаткову гнучкість у порівнянні з формальними кількісними метриками, оскільки дозволяє враховувати візуальні особливості, які не завжди охоплюються обчислювальними критеріями.

У процесі оцінювання першочергову увагу приділяли ступеню подібності синтезованих облич до реальних. Зокрема, аналізувалися риси обличчя, пропорції, деталізація текстур шкіри, чіткість контурів таких елементів, як очі, ніс чи губи. На завершальному етапі навчання моделі були отримані результати, які свідчать про те, що загальна структура та ключові ознаки облич успішно відтворюються, хоча іноді спостерігалися окремі локальні дефекти, зокрема нечіткість у зоні очей або невиразні текстури в окремих фрагментах.

Крім того, важливим аспектом є відсутність характерних артефактів, що часто виникають у процесі генерації зображень. Йдеться про розмиття,

шуми або неприродні переходи між ділянками обличчя, які можуть вказувати на нестабільність або недостатнє узгодження між компонентами моделі. Візуальний огляд дозволив виявити, що в більшості випадків модель справляється з відтворенням плавних переходів та природного освітлення, хоча в окремих зразках все ще фіксувалися незначні спотворення чи неприродні тіні.

Ще однією важливою характеристикою є рівень різноманітності синтезованих облич. Під час аналізу зверталася увага на те, наскільки результати генерації демонструють варіативність у розміщенні та формі ключових рис обличчя, зокрема очей, носа та рота. Виявлено, що модель, у цілому, здатна продукувати обличчя з різними морфологічними ознаками, проте у випадках недостатньої насиченості латентного простору можливі прояви схожості між окремими зображеннями, що може вказувати на ефект колапсу моди.

У підсумку, якісна візуальна перевірка продемонструвала, що синтезовані обличчя загалом мають високу ступінь реалістичності, зберігають базові ознаки, властиві реальним зображенням, а також демонструють помітну варіативність. Разом з тим, незначні локальні похибки, пов'язані з деталізацією або фоном, залишаються потенційними напрямками для вдосконалення. Типові результати генерації облич, які ілюструють описану вище оцінку, наведені на рис. 3.1.



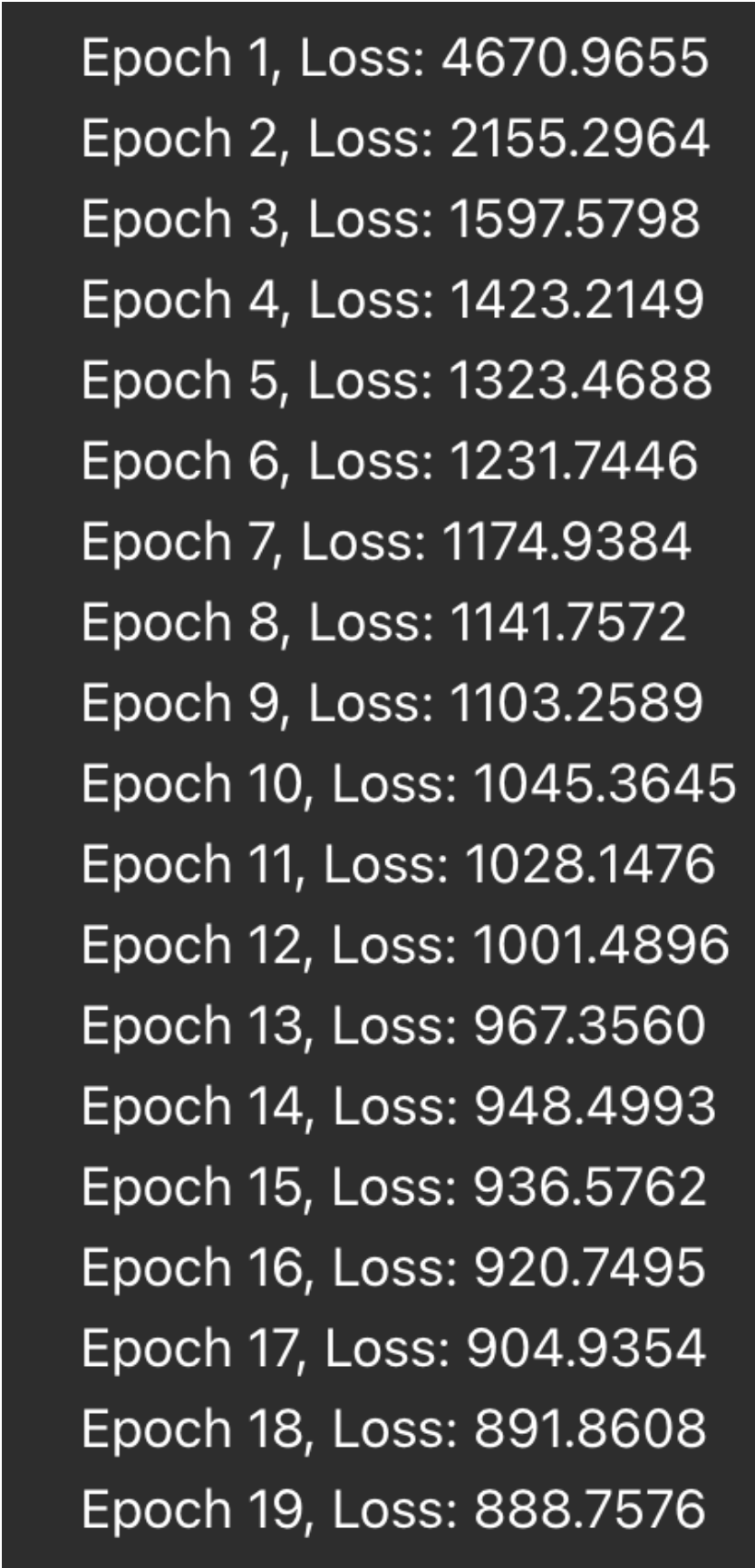
Рисунок 3.1 – Результати генерації облич

3.1.2 Статистичні показники

На відміну від візуальної оцінки, статистичні показники дають змогу здійснити об'єктивне і кількісне порівняння якості згенерованих зображень з референтними даними. У ході навчання моделі ключовим показником є значення функції втрат, яке реєструється окремо для різних складових архітектури. Зокрема, відстежується динаміка втрат, пов'язаних із варіаційним автокодувальником, які охоплюють як відповідність між вхідними та реконструйованими зображеннями, так і регуляризацію латентного простору, а також окремо аналізуються втрати, пов'язані зі змагальною частиною моделі, що стосуються взаємодії генератора і дискримінатора.

У процесі навчання було зафіксовано, що на початкових етапах функція втрат зазвичай демонструє стрімке зниження, що свідчить про активну адаптацію моделі до структури вхідних даних (рис. 3.2). Подальше уповільнення цього процесу є ознакою наближення до стабільного стану. На завершальних етапах, при досягненні оптимального балансу між генератором і дискримінатором, значення втрат стабілізуються, що може свідчити про досягнення рівноваги між частинами моделі.

Оцінка пропускної здатності навчання здійснювалася через аналіз змін у значеннях втрат для кожного компонента. Зокрема, у випадках, коли падіння втрат одного з них супроводжувалося підвищенням іншого, це розглядалося як можливий прояв нестабільності у процесі оптимізації. Тоді як збереження одночасного поступового зниження обох типів втрат розцінювалося як ознака збалансованого навчання.



```
Epoch 1, Loss: 4670.9655  
Epoch 2, Loss: 2155.2964  
Epoch 3, Loss: 1597.5798  
Epoch 4, Loss: 1423.2149  
Epoch 5, Loss: 1323.4688  
Epoch 6, Loss: 1231.7446  
Epoch 7, Loss: 1174.9384  
Epoch 8, Loss: 1141.7572  
Epoch 9, Loss: 1103.2589  
Epoch 10, Loss: 1045.3645  
Epoch 11, Loss: 1028.1476  
Epoch 12, Loss: 1001.4896  
Epoch 13, Loss: 967.3560  
Epoch 14, Loss: 948.4993  
Epoch 15, Loss: 936.5762  
Epoch 16, Loss: 920.7495  
Epoch 17, Loss: 904.9354  
Epoch 18, Loss: 891.8608  
Epoch 19, Loss: 888.7576
```

Рисунок 3.2 – Результат роботи програми

Загалом, аналіз статистичних показників підтвердив ефективність реалізованої архітектури. Значення втрат демонстрували поступове зменшення з кожною епохою, що відображає успішність тренувального процесу. На рис. 3.3 наведено графік, який відображає зміну функції втрат протягом епох навчання.

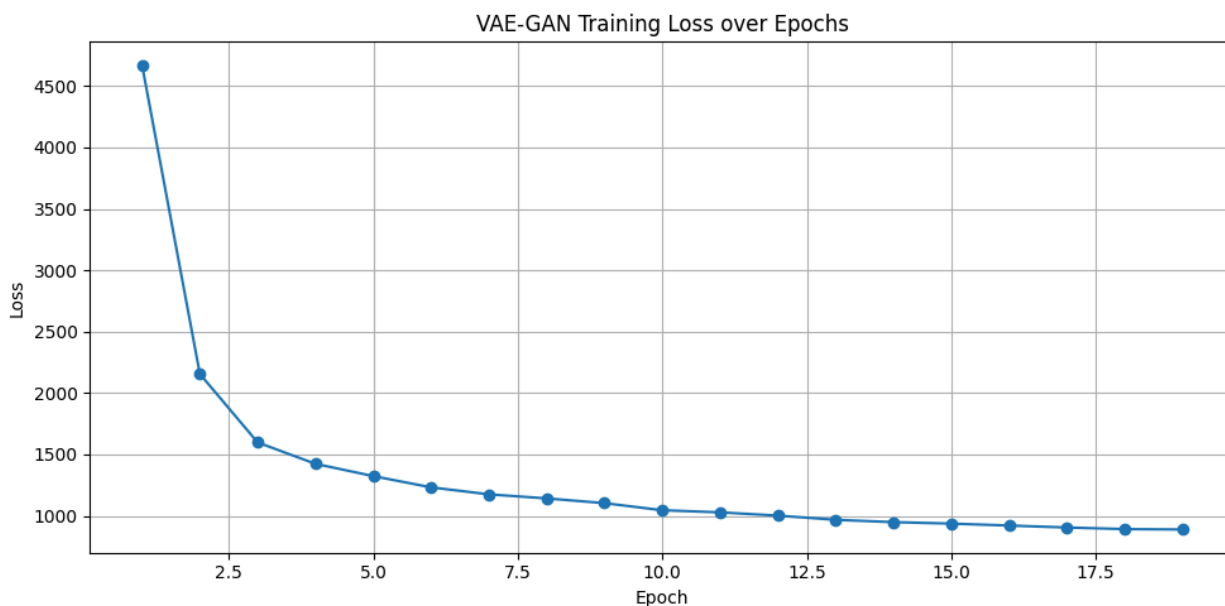


Рисунок 3.3 – Графік, який відображає зміну функції втрат протягом епох навчання

Цей тип кількісного аналізу дозволив здійснити об'єктивну оцінку загальної динаміки вдосконалення моделі, а також виявити особливості її збіжності без потреби у суб'єктивному тлумаченні результатів. Таким чином, поєднання статистичного аналізу та візуальної перевірки дає змогу отримати повну картину ефективності генеративної моделі.

3.2 Аналіз ходу навчання моделі

Процес навчання моделі VAEGAN здійснювався відповідно до встановлених гіперпараметрів, серед яких ключову роль відігравали швидкість навчання, розмір батчу та архітектура мережі. Під час тренування одночасно оновлювалися ваги автокодувальника й дискримінатора, що відповідало класичному підходу до навчання адверсаріальних моделей. Цикл навчання передбачав послідовну подачу партій зображень облич із вхідного датасету, обчислення втрат реконструкції та адверсаріальних втрат, після чого відбувалося коригування ваг методом градієнтного спуску. Для відстеження перебігу навчання було організовано регулярний вивід статистики, зокрема номеру епохи та значень функцій втрат. Приклад такого виводу показано на рис. 3.2.

З аналізу тренувального лог-файлу випливає, що на початку тренування спостерігалось різке зниження значень функції втрат, що є свідченням швидкої адаптації моделі до початкових характеристик даних. З подальшим просуванням епох темп зменшення втрат сповільнювався, що є типовим наближенням до оптимуму в процесі навчання. Ця динаміка добре відображена на рис. 3.3, де показано поступове зниження сумарної втрати впродовж усього тренування. У пізніших етапах втрати змінювалися незначно, що вказує на досягнення стабільної фази навчання.

Ще одним позитивним індикатором стало збалансоване співвідношення втрат генератора та дискримінатора. Вони не мали різких коливань, і не спостерігалось жодного з критичних проявів нестабільності, як-от колапс моди або перевага однієї зі сторін моделі. Це дає підстави вважати, що обрані гіперпараметри — зокрема швидкість навчання — були підібрані вдало, що забезпечило рівновагу у процесі навчання.

Особливу увагу привертає перебіг нормування латентного простору. У контексті VAEGAN критично важливо, щоб латентний розподіл

стабілізувався завдяки варіаційному компоненту. У нашому випадку було зафіксовано поступове зменшення регуляризаційного члена у функції втрат, зокрема KL-дивергенції, що свідчить про впорядкування простору латентних векторів. Це позитивно впливає на генеративну здатність моделі, дозволяючи їй створювати різноманітні та реалістичні зображення на основі різних вхідних шумів.

Побудований графік залежності функції втрат від номера епохи, наведений на рис. 3.3, став наочним підтвердженням досягнення конвергенції моделі. Крива втрат різко знижувалась на початкових етапах тренування, тоді як у другій половині процесу навчання вона виходила на плато. Це вказує на те, що модель вичерпала основні ресурси для навчання в межах заданої архітектури й обсягу даних, і подальше покращення вимагало б додаткових змін — зокрема, розширення датасету або ускладнення мережі. Загалом аналіз ходу тренування підтверджує поступове та стабільне покращення параметрів моделі, що є очікуваним результатом для грамотно організованого процесу навчання VAE-GAN. Важливо, що вся статистика втрат зберігалася для подальшої оцінки якості моделі та своєчасного виявлення можливих аномалій у разі відсутності подальшого зниження втрат.

3.3 Обговорення результатів і виявлених проблем

Після завершення навчання моделі та отримання результатів генерації зображень було проведено ґрунтовне обговорення її роботи, досягнень та проблем, що виникли під час експериментів. Загалом модель VAE-GAN показала обнадійливі результати: вона здатна ефективно реконструювати вхідні зображення та генерувати нові приклади, які візуально наближені до реальних зображень з навчального набору. Це свідчить про те, що модель навчилася вловлювати високорівневі особливості зображень облич та

успішно переносити їх у компактне латентне представлення. У багатьох випадках відновлені зображення мали чітко визначені риси обличчя, добре збережену симетрію та коректні пропорції.

Проте, не обійшлося й без проблем. Однією з найпомітніших є поява артефактів у згенерованих зображеннях — іноді можна спостерігати неузгоджені переходи кольору, змащені області або спотворення рис обличчя, особливо у випадках, коли вхідні дані є нетиповими або містять шум. Це свідчить про те, що модель має певну чутливість до складних прикладів та, можливо, недостатньо узагальнила особливості даних. Ще однією проблемою є помітна перевага в генерації певного типу облич — зокрема, модель частіше генерує обличчя з типовими рисами, які домінували у навчальному наборі. Це вказує на можливе зміщення у даних і на те, що модель може неохоче досліджувати рідкісні варіанти, що негативно впливає на різноманіття результатів.

Також було помічено, що в окремих випадках модель має тенденцію до розмитості зображень, що може бути пов'язано із втратами в автоенкодері або недостатньою потужністю дискримінатора. Ще одна проблема — це відносно повільне збіження при навчанні, яке вимагало ретельного добору гіперпараметрів, таких як коефіцієнт ваги KL-дивергенції та архітектура латентного простору. Без належного налаштування модель або втрачала здатність до генерації якісних зображень, або переставала узагальнювати дані. Це свідчить про необхідність додаткової стабілізації процесу навчання, можливо, через зміну архітектури або використання більш сучасних методів нормалізації та регуляризації.

У цілому, результати експериментів вказують на високий потенціал моделі, однак наявні проблеми відкривають простір для подальших досліджень і вдосконалень, які дозволили б підвищити стабільність, якість та універсальність генерації.

3.4 Можливості вдосконалення моделі та подальші експерименти

Аналізуючи отримані результати та виявлені проблеми, можна виділити декілька ключових напрямів для подальшого вдосконалення моделі VAEGAN і проведення експериментів, які можуть суттєво покращити її продуктивність та стабільність.

Перш за все, важливо звернути увагу на архітектуру моделі. В подальших роботах доцільно розглянути інтеграцію більш глибоких та складних нейронних мереж, зокрема, використання блоків ResNet або архітектур з механізмами уваги (Attention), які довели свою ефективність у задачах генерації зображень. Ці технології допомагають краще захоплювати складні просторові залежності та підвищують якість відновлення та генерації деталей, що особливо важливо для складних структур обличчя. Впровадження таких рішень може призвести до значного підвищення реалістичності та різноманіття генерованих зображень.

Другим важливим напрямом є розвиток та оптимізація латентного простору. В сучасних генеративних моделях якість та організованість латентного простору відіграє ключову роль у контролі над процесом генерації. Для покращення моделей VAEGAN можна розглянути застосування умовних варіантів (Conditional VAEGAN), що дозволить керувати параметрами створюваних зображень через додаткові умови — наприклад, вік, стать, емоційний стан або інші атрибути [26].

Це відкриває можливість створення цілеспрямованих та персоналізованих генерацій, що є особливо корисним у прикладних задачах, таких як медична діагностика, розваги чи дизайн. Крім того, варто експериментувати з різними варіантами розподілу латентних змінних: замість класичного нормального розподілу, можна застосувати більш складні, наприклад, змішані або варіаційно навчені пріори, що підвищують гнучкість і покращують різноманіття результатів.

Третім напрямом є покращення самого процесу навчання. Враховуючи спостереження щодо нестабільності тренування та наявності артефактів, необхідно дослідити нові методи регуляризації та нормалізації. Використання таких технік, як *spectral normalization*, *dropout* у генераторі, а також більш адаптивних методів оптимізації, може суттєво знизити ризик переобучення та покращити узагальнення моделі.

Також перспективним є застосування стратегій *curriculum learning*, де модель починає навчання з простих прикладів, поступово переходячи до складніших, що дозволяє стабілізувати процес тренування та підвищити якість генерації. Окрім цього, слід розглянути варіанти навчання з підкріпленням або використання ансамблів моделей, які можуть покращити здатність моделі до адаптації на різних типах даних.

Ще одним важливим аспектом є робота з навчальним набором даних. Якість та різноманіття тренувальних даних безпосередньо впливають на продуктивність моделі. Для підвищення різноманіття та зменшення зміщень варто розширити датасет, додавши більше прикладів з різними характеристиками облич, а також застосувати різноманітні методи аугментації — наприклад, зміну освітлення, положення або масштабу. Це сприятиме кращому узагальненню та зменшенню ймовірності генерації одноманітних або артефактних зображень.

Крім того, варто експериментувати з балансуванням класів і врахуванням рідкісних категорій, щоб модель була більш чутливою до різних варіантів.

Для подальшої перевірки та оцінки ефективності запропонованих покращень рекомендується провести порівняльний аналіз з іншими сучасними генеративними архітектурами, такими як *StyleGAN*, *StyleGAN2* або *Diffusion Models* [27, 28, 29]. Такий порівняльний експеримент дозволить виявити сильні та слабкі сторони моделі *VAEGAN* і зорієнтувати подальший розвиток у бік найбільш перспективних технологій.

Також корисними будуть додаткові дослідження латентного простору, наприклад, інтерполяції між точками у латентному просторі для оцінки плавності та послідовності генерації, а також аналіз просторової структури латентних представлень. Ці дослідження допоможуть краще зрозуміти, наскільки модель навчилася узагальнювати та зберігати важливі характеристики вхідних даних.

Насамкінець, варто розглянути можливість інтеграції моделей VAE-GAN з іншими підходами — наприклад, об'єднання генеративних мереж із сегментаційними чи класифікаційними модулями, що дозволить створювати більш комплексні системи для автоматичного аналізу зображень. Цей напрямок відкриває нові горизонти застосування технології в практичних задачах, таких як створення систем відеомоніторингу, медичної діагностики або персоналізованих розважальних сервісів.

Таким чином, існує широкий спектр можливостей для покращення моделі та проведення подальших експериментів, які можуть значно підвищити якість, стабільність та функціональність VAE-GAN, а також розширити межі її застосування в реальних умовах.

Висновки до розділу 3

У ході виконання розділу 3 було проведено детальний аналіз роботи моделі VAE-GAN на прикладі генерації зображень облич. Результати показали, що обрана архітектура здатна генерувати якісні та досить реалістичні зображення, що підтверджує ефективність поєднання варіаційного автокодера та генеративно-змагальної мережі. Проте, одночасно було виявлено низку проблем, пов'язаних зі стабільністю навчання та появою артефактів на згенерованих зображеннях, що вказує на необхідність подальшої оптимізації як самої моделі, так і процесу її тренування.

Аналіз отриманих результатів дав змогу окреслити основні напрямки для покращення, серед яких — удосконалення архітектури шляхом впровадження більш сучасних блоків та механізмів уваги, оптимізація латентного простору для кращого контролю над генерацією, а також застосування нових методів регуляризації і адаптивних стратегій навчання. Окрім того, було визначено важливість розширення та балансування навчального набору даних, що може суттєво підвищити загальну якість і різноманітність генерації.

Також проведено обґрунтування необхідності подальших експериментів із різними параметрами моделі, а також порівняльного аналізу з іншими передовими генеративними архітектурами. Це дозволить не лише підвищити продуктивність і стабільність, а й розширити функціональні можливості розробленої системи, відкриваючи нові шляхи її застосування в практичних задачах.

Отже, розділ 3 підкреслив як потенціал, так і виклики, що стоять перед моделлю VAE-GAN, та окреслив перспективи для її подальшого розвитку, що є важливою основою для наступних етапів дослідження.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В цьому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на генерації зображень облич за допомогою технології VAE-GAN. Також в цьому дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення.

Для проведення аналізу використовується економічна, технічна та конструкторська інформація. Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та

реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є такі:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства.

Беручи за основу цю функцію, можна виділити наступні:

- F_1 – генерація нових зображень облич;
- F_2 – візуалізація та аналіз результатів генерації;
- F_3 – збереження та повторне використання моделей.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1 :

- на основі латентного простору моделі VAE-GAN;
- дозволяє створювати синтетичні зображення із заданими або

випадковими ознаками.

Функція F_2 :

- забезпечення оцінки якості зображень – візуальної, статистичної;
- відображення середнього, мінімального, максимального піксельного значення, а також вивід сітки зображень.

Функція F_3 :

- збереження навченої моделі (Encoder, Decoder);
- можливість повторного завантаження для генерації без повторного тренування.

Варіанти реалізації основних функцій представлено у морфологічній карті системи на рис. 4.1.

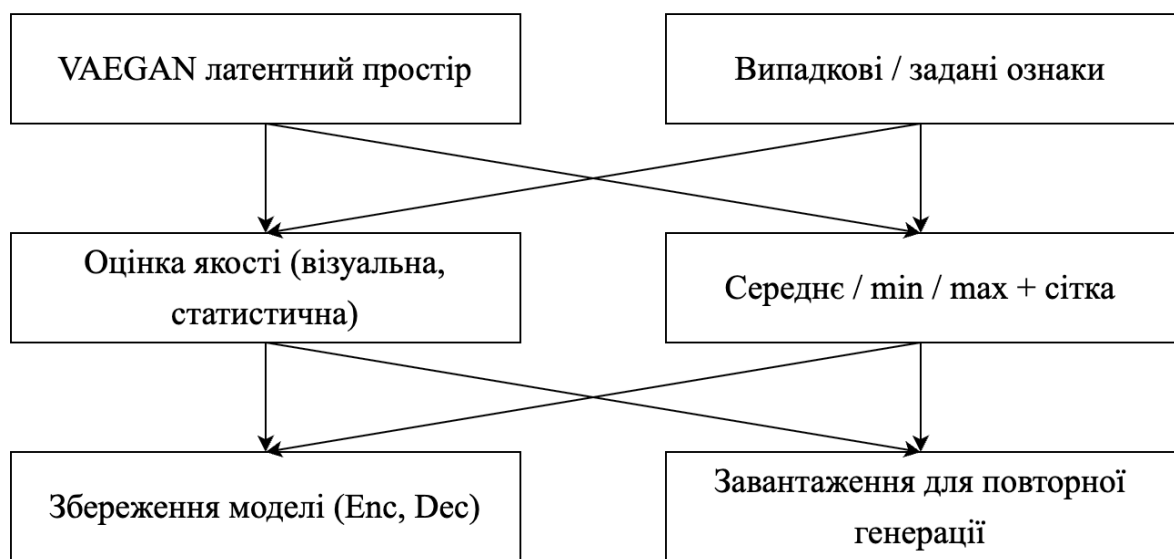


Рисунок 4.1 – Морфологічна карта системи

Морфологічна карта відображає множину всіх можливих варіантів основних функцій.

Позитивно-негативна матриця показана в табл. 4.1.

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	<i>A</i>	Простота реалізації, контрольованість генерації	Обмежена варіативність без додаткових модифікацій
	<i>B</i>	Дає більше варіацій, можливість цільового керування	Необхідне додаткове кодування або ручний підбір параметрів
F_2	<i>A</i>	Дає загальне уявлення про якість, проста реалізація	Суб'єктивність візуальної оцінки
	<i>B</i>	Дає конкретні числові показники, зручно для порівняння	Не враховує семантику чи сприйняття образу
F_3	<i>A</i>	Швидке повторне використання, не потребує повторного навчання	Може потребувати значного місця на диску
	<i>B</i>	Автоматизація, зручність при масштабуванні	Може потребувати сумісності версій моделей/бібліотек

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізацій функцій варто відкинути, тому що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1 : перевагу даємо варіанту *B* – генерація із заданими або випадковими ознаками, оскільки він забезпечує більшу варіативність та контроль генерації. Для спрощення роботи по написанню коду варіант *A* – генерація лише з латентного простору має бути відкинута, оскільки він не передбачає керовану генерацію.

Функція F_2 : перевагу даємо варіанту *B* – вивід середнього,

мінімального, максимального значення пікселів та сітки зображень, оскільки він забезпечує більш об'єктивну та репрезентативну оцінку якості. Для спрощення роботи по написанню коду варіант А – суб'єктивна візуальна/статистична оцінка має бути відкинута.

Функція F_3 : програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$$F_{1A} - F_{2A} - F_{3A'}$$

$$F_{1A} - F_{2A} - F_{3B}$$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати такі параметри:

- X_1 – якість згенерованих зображень;
- X_2 – час генерації одного зображення;
- X_3 – гнучкість та масштабованість використання моделі;
- X_4 – простота інтеграції та використання.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного

продукту, як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Якість згенерованих зображень	X_1	експертна оцінка (експертна шкала 1 - 5)	1	3	5
Час генерації одного зображення	X_2	секунди	> 2 с	1 – 2 с	< 1 с
Гнучкість та масштабованість моделі	X_3	умовні бали (0–10)	0	5	10
Простота використання та інтеграції	X_4	умовні бали (0–10)	0	5	10

За даними табл. 4.2 будуються графічні характеристики параметрів (рис. 4.2 – рис 4.5).

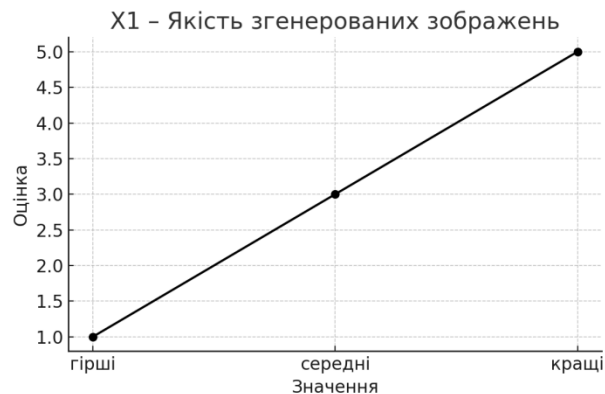


Рисунок 4.2 – Якість згенерованих зображень

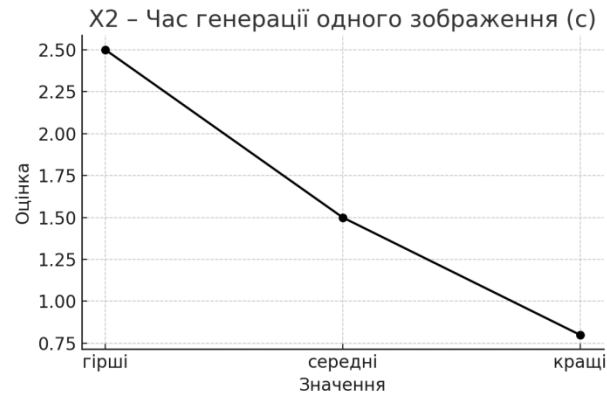


Рисунок 4.3 – Час генерації одного зображення

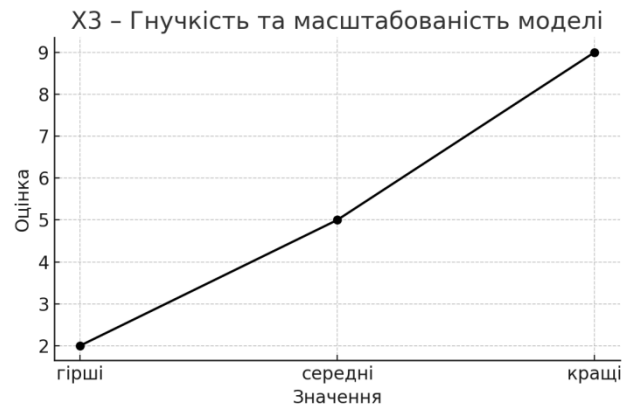


Рисунок 4.4 – Гнучкість та масштабованість моделі

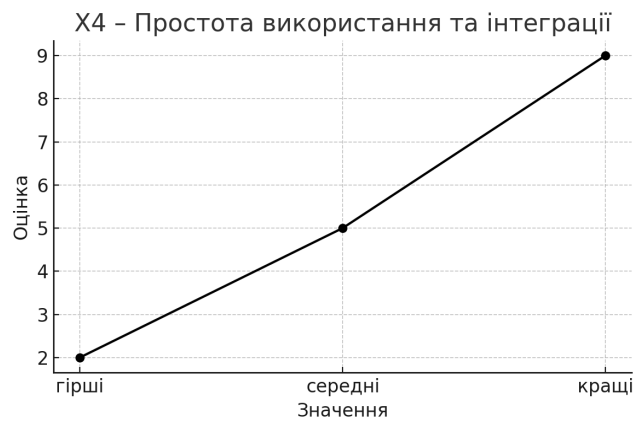


Рисунок 4.5 – Простота використання та інтеграції

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який забезпечує точне, гнучке та ресурсоефективне визначення емоційного контексту зображень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей.

Визначення коефіцієнтів значущості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування представлені у табл. 4.3.

Для перевірки ступеня достовірності експертних оцінок, визначимо такі параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70.$$

де N – число експертів;

n – кількість параметрів;

R_{ij} – ранг i -го параметру проставлений j -им експертом.

Таблиця 4.3 – Результати ранжування параметрів

Назва параметра	Позначення параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
Якість згенерованих зображень	X_1	Бали (1–5)	1	2	1	2	1	1	3	11	-6.5	42.25
Час генерації одного зображення	X_2	Секунди	3	3	4	3	3	2	2	20	2.5	6.25
Гнучкість та масштабованість моделі	X_3	Бали (0–10)	2	1	2	1	2	3	1	12	-5.5	30.25
Простота використання та інтеграції	X_4	Бали (0–10)	4	4	3	4	4	4	4	27	9.5	90.25
Разом			10	10	10	10	10	10	10	70	0	169

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17,5.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T.$$

Сума відхилень по всіх параметрах повинна дорівнювати 0.

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 169.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 169}{7^2(4^3-4)} = 0.6898 > W_k = 0,67.$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у табл. 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 і X_2	<	<	<	<	<	<	>	<	0,5
X_1 і X_3	<	>	<	>	<	<	>	<	0,5
X_1 і X_4	<	<	<	<	<	<	<	<	0,5
X_2 і X_3	>	>	>	>	>	<	>	>	1,5
X_2 і X_4	<	<	>	<	<	<	<	<	0,5
X_3 і X_4	<	<	<	<	<	<	<	<	0,5

Числове значення, що визначає ступінь переваги i -го параметра над j -им, a_{ij} визначається по формулі:

$$a_{ij} = \{1.5 \text{ при } X_i > X_j; 1.0 \text{ при } X_i = X_j; 0.5 \text{ при } X_i < X_j\}.$$

З отриманих числових оцінок переваги складемо матрицю $A = \|a_{ij}\|$.

Для кожного параметра зробимо розрахунок вагомості K_{bi} за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i},$$

$$b_i = \sum_{j=1}^N a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).

На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b'_i}{\sum_{i=1}^n b'_i},$$

$$b'_i = \sum_{j=1}^N a_{ij} b_j.$$

Як видно з табл. 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітерація		Друга ітерація		Третя ітерація	
	X_1	X_2	X_3	X_4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X_1	1	0,5	0,5	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X_2	1,5	1	1,5	0,5	4,5	0,28	16,25	0,27	59,125	0,27
X_3	1,5	0,5	1	0,5	3,5	0,22	12,25	0,21	44,875	0,21
X_4	1,5	1,5	1,5	1	5,5	0,34	21,25	0,36	77,875	0,36
Всього:					16	1	59	1	216	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X_2 (час генерації одного зображення) X_3 (гнучкість та масштабованість моделі) та X_4 (простота використання та інтеграції) відповідають технічним вимогам умов функціонування.

Абсолютне значення параметра X_1 (якість згенерованих зображень) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так:

$$K_K(j) = \sum_{i=1}^n K_{bi,j} B_{i,j},$$

де n – кількість параметрів;

K_{bi} – коефіцієнт вагомості i -го параметра;

B_i – оцінка i -го параметра в балах.

Таблиця 4.6 – Розрахунок вагомості параметрів

Функція	Варіант реалізації функції	Параметр	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	Б	X_1	3	5	0,16	0,8
F_2	Б	X_2	0.8	10	0,28	2,8
F_3	А	X_3	8	8	0,22	1,76
	Б	X_4	8	8	0,34	2,72

За даними з табл. 4.6 визначаємо рівень якості кожного з варіантів за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \dots + K_{\text{ТУ}}[F_{zk}].$$

Отримаємо:

$$K_{K1} = 0,8 + 2,8 + 1,76 = 5,36,$$

$$K_{K2} = 0,8 + 2,8 + 2,72 = 6,32.$$

Як видно з розрахунків, кращим є 2 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

- розробка проекту програмного продукту;
- розробка програмної оболонки.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б.

За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для

кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_0 = T_p \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М'}$$

де T_p – трудомісткість розробки ПП;

K_{Π} – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_p = 30$ людино-днів.

Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_{\Pi} = 1.9$.

Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК} = 1$.

Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0.9$.

Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 30 \cdot 1.9 \cdot 0.9 = 51,3 \text{ людино} - \text{днів.}$$

Для другого завдання (використовується алгоритм третьої групи

складності, ступінь новизни Б) – $T_p = 35$ людино-днів, $K_{II} = 0,8$, $K_{СК} = 1$,
 $K_{СТ} = 0,8$:

$$T_2 = 35 \cdot 0,8 \cdot 0,8 = 22,4 \text{ людино} - \text{днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (51,3 + 22,4 + 4,8 + 22,4) \cdot 8 = 807,2 \text{ людино} - \text{годин,}$$

$$T_{II} = (51,3 + 22,4 + 7,9 + 22,4) \cdot 8 = 832 \text{ людино} - \text{годин.}$$

Найбільш високу трудомісткість має варіант I.

В розробці беруть участь два програмісти з окладом 20000 грн, один аналітик в області даних з окладом 25000 грн.

Визначимо середню зарплату за годину за формулою:

$$СЧ = \frac{M}{T_m \cdot t} \text{ грн,}$$

де M – місячний оклад працівників;

T_m – кількість робочих днів на тиждень;

t – кількість робочих годин в день.

$$СЧ = \frac{20000+20000+25000}{3 \cdot 21 \cdot 8} = 128,97 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$СЗП = C_{\text{ч}} \cdot T_i \cdot K_{\text{д}},$$

де C_q – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

K_d – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$C_{зп I} = 128,97 \cdot 807,2 \cdot 1,2 = 124925,5 \text{ грн},$$

$$C_{зп II} = 128,97 \cdot 832 \cdot 1,2 = 128763,65 \text{ грн}.$$

Відрахування на єдиний соціальний внесок становить 22%:

$$C_{від I} = C_{зп I} \cdot 0,22 = 124925,5 \cdot 0,22 = 27483,61 \text{ грн},$$

$$C_{від II} = C_{зп II} \cdot 0,22 = 128763,65 \cdot 0,22 = 28328 \text{ грн}.$$

Тепер визначимо витрати на оплату однієї машино-години. (C_M).

Так як одна ЕОМ обслуговує одного програміста з окладом 20000 грн, з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{\Gamma} = 12 \cdot M \cdot K_3 = 12 \cdot 20000 \cdot 0,2 = 48000 \text{ грн}.$$

З урахуванням додаткової заробітної плати:

$$C_{зп} = C_{\Gamma} \cdot (1 + K_3) = 48000 \cdot (1 + 0,2) = 57600 \text{ грн}.$$

Відрахування на соціальний внесок:

$$C_{від} = C_{зп} \cdot 0,22 = 57600 \cdot 0,22 = 12672 \text{ грн}.$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 18000 грн.

$$C_A = K_{TM} \cdot K_A \cdot C_{PP} = 1,4 \cdot 0,25 \cdot 18000 = 6300 \text{ грн},$$

де K_{TM} – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

C_{PP} – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{TM} \cdot C_{PP} \cdot K_P = 1,4 \cdot 18000 \cdot 0,08 = 2016 \text{ грн},$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{EF} = (D_K - D_B - D_C - D_P) \cdot t_3 \cdot K_B = 627,2 \text{ год},$$

де D_K – календарна кількість днів у році;

D_B, D_C – відповідно кількість вихідних та святкових днів;

D_P – кількість днів планових ремонтів устаткування;

t_3 – кількість робочих годин в день;

K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{EL} = T_{EF} \cdot N_C \cdot K_3 \cdot C_{EH} = 627,2 \cdot 0,2 \cdot 0,3 \cdot 9,43 = 355 \text{ грн},$$

де N_c – середньо-споживча потужність приладу;

K_3 – коефіцієнтом зайнятості приладу;

Π_{EH} – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_H = \Pi_{PP} \cdot 0,67 = 18000 \cdot 0,67 = 12060 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{EKC} = C_{ЗП} + C_{ВІД} + C_A + C_P + C_{EL} + C_H \text{ грн,}$$

$$\begin{aligned} C_{EKC} &= 57600 + 12672 + 6300 + 2016 + 355 + 12060 = \\ &= 91003 \text{ грн.} \end{aligned}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{M-\Gamma} = C_{EKC} / T_{EF} = 91003 / 627,2 = 145 \text{ грн/год.}$$

Оскільки в цьому випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_M = C_{M-\Gamma} \cdot T,$$

$$C_{MI} = 145 \cdot 807,2 = 117044 \text{ грн,}$$

$$C_{MII} = 145 \cdot 832 = 120640 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_H = C_{3П} \cdot 0,67,$$

$$C_{HI} = 124925,5 \cdot 0,67 = 83700 \text{ грн},$$

$$C_{HII} = 128763,65 \cdot 0,67 = 86272 \text{ грн}.$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{3П} + C_{ВІД} + C_M + C_H,$$

$$C_{ППI} = 124926 + 27484 + 117044 + 83700 = 353154 \text{ грн},$$

$$C_{ППII} = 128764 + 28328 + 120640 + 86272 = 364004 \text{ грн}.$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{Kj} / C_{\Phi j},$$

$$K_{\text{ТЕР}1} = 5,36 / 353154 = 1,52 \cdot 10^{-5},$$

$$K_{\text{ТЕР}2} = 6,32 / 364004 = 1,74 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{\text{ТЕР}2} = 1,74 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розробляється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного

комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $K_{\text{ТЕР2}} = 1,74 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- генерація із заданими або випадковими ознаками;
- вивід середнього, мінімального, максимального значення пікселів та сітки зображень;
- повторне завантаження для генерації.

Цей варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

Висновки до розділу 4

В цьому розділі проведено повний функціонально-вартісний аналіз програмного продукту. Також знайдено оцінку основних функцій програмного продукту.

За допомогою експертного методу було визначено вагомість кожного параметра, що дозволило обґрунтувати вибір пріоритетних функціональних характеристик для реалізації. На основі побудованої морфологічної карти варіантів реалізації функцій було визначено найбільш раціональне поєднання функціональних рішень. Проведений аналіз дозволив не лише оцінити якість кожного варіанту, а й встановити перевагу одного з них за сукупністю критеріїв.

Окремо розраховано очікувану трудомісткість реалізації кожного з варіантів, з урахуванням складності алгоритмів, виду інформації та інших коригувальних коефіцієнтів. У результаті зіставлення отриманих оцінок було

обрано найоптимальніший варіант, який забезпечує найкраще співвідношення між якістю реалізації функцій та витратами на її створення - це поєднання генерації із заданими або випадковими ознаками, виводу середнього, мінімального, максимального значення пікселів та сітки зображень та повторного завантаження для генерації.

Отже, результати аналізу підтверджують доцільність і економічну ефективність впровадження розробленого програмного продукту в обраному варіанті.

ВИСНОВКИ

У цій дипломній роботі було проведено всебічне дослідження проблеми генерації зображень облич за допомогою нейронних мереж, зосереджене на гібридній архітектурі VAEGAN. У першому розділі розглянуто теоретичні основи розвитку генеративних моделей: від базових автоенкодерів до сучасних потужних систем на основі GAN та VAE. Було проаналізовано ключові архітектурні рішення, які дозволили досягти високого рівня фотореалістичності, а також визначено роль комбінованих підходів у подоланні обмежень класичних моделей.

У другому розділі було розроблено та реалізовано генеративну модель на базі VAEGAN. Модель поєднує латентне кодування варіаційного автоенкодера з дискримінатором, притаманним GAN, що дозволяє поєднати переваги обох підходів: стабільність латентного простору та високу якість згенерованих зображень. Проведене навчання на наборі даних «human-faces» продемонструвало здатність моделі відтворювати реалістичні обличчя з варіативними атрибутами, зберігаючи структурну узгодженість та візуальну достовірність.

Результати експериментів підтверджують ефективність використаної архітектури для задач генерації зображень. Реалізований підхід дозволяє генерувати нові обличчя з заданими властивостями, а також моделювати розподіл ознак у латентному просторі, що відкриває широкі можливості для керованої генерації.

Отримані результати мають практичне значення для різноманітних застосувань, зокрема у створенні віртуальних персонажів, анонімізації даних або синтетичному розширенні наборів зображень для задач комп'ютерного зору.

Таким чином, дана робота демонструє актуальність і перспективність гібридних генеративних моделей у сучасному штучному інтелекті та створює підґрунтя для подальших досліджень у галузі генерації зображень.

ПЕРЕЛІК ПОСИЛАНЬ

1. Elasri M., Elharrouss O., Al-Maadeed S., Tairi H. Image Generation: A Review // Springer Nature. – 2022. – P. 1–39.
2. Niu Z., Yu K., Wu X. LSTM-Based VAE-GAN for Time-Series Anomaly Detection // Sensors. – 2020. – Vol. 20, No. 13. – Article 3738.
3. Odaibo S.G. Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function // RETINA-AI Health, Inc.; MD Anderson Cancer Center. – 2019. – P. 1–8.
4. Belov D.I., Armstrong R.D. Distributions of the Kullback–Leibler divergence with applications // British Journal of Mathematical and Statistical Psychology. – 2011. – T. 64. – P. 291–309.
5. Hershey J.R., Olsen P.A. Approximating the Kullback–Leibler divergence between Gaussian Mixture Models // IBM T. J. Watson Research Center. – P. 1–4.
6. Gui J., Sun Z., Wen Y., Tao D., Ye J. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications // Journal of LaTeX Class Files. – 2015. – Vol. 14, No. 8. – P. 1–28.
7. VAEGAN Graph. . URL:
https://www.researchgate.net/figure/Ov-ervie-w-of-a-VAE-b-GAN-and-c-VAEGAN-which-integrates-the-encoder-decoder-and_fig2_371570680 (дата звернення 27.04.2025)
8. Luo Y., Wang X., Pourpanah F. Dual VAEGAN: A Generative Model for Generalized Zero-Shot Learning // Applied Soft Computing Journal. – 2021. – Vol. 107. – Article 107352. – P. 1–9.
9. Liu H., Ong Y.-S., Jiang X., Wang X. Deep Latent-Variable Kernel Learning // IEEE. – 2023. – P. 1–13.
10. Li J., Hemachandran K., Data Generation and Latent Space Based Feature Transfer Using ED-VAEGAN, an Improved Encoder and Decoder Loss

- VAEGAN Network // Proceedings of ICAID 2023, AHIS 9. – 2023. – P. 123–135.
11. CelebA Dataset. URL: <https://paperswithcode.com/dataset/celeba> (дата звернення 27.04.2025)
 12. CASIA WebFace Dataset. URL: <https://paperswithcode.com/dataset/casia-webface> (дата звернення 27.04.2025)
 13. Creswell A., White T., Dumoulin V., Arulkumaran K., Sengupta B., Bharath A.A. Generative Adversarial Networks: An Overview // Submitted to IEEE Signal Processing Magazine. – 2017. – P. 1–14.
 14. Doersch C. Tutorial on Variational Autoencoders // Carnegie Mellon / UC Berkeley. – 2016 (with minor revisions in 2021). – P. 1–23.
 15. Yu X., Zhang X., Cao Y., Xia M. VAEGAN: A Collaborative Filtering Framework based on Adversarial Variational Autoencoders // Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19). – 2019. – P. 4206–4212.
 16. Abdal R., Qin Y., Wonka P. Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space? // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). – 2019. – P. 4432–4441.
 17. RMSprop. URL: <https://docs.pytorch.org/docs/stable/generated/torch.optim.RMSprop.html> (дата звернення 15.05.2025).
 18. Human Face Dataset. URL: <https://www.kaggle.com/datasets/ashwingupta3012/human-faces> (дата звернення 15.05.2025).
 19. Bojanowski P., Joulin A., Lopez Paz D., Szlam A. Optimizing the Latent Space of Generative Networks // Proceedings of the 35th International Conference on Machine Learning. – 2018. – P. 1–10.
 20. Notin P., Hernández-Lobato J.M., Gal Y. Improving Black-box Optimization in VAE Latent Space Using Decoder Uncertainty // Proceedings of the 35th

- Conference on Neural Information Processing Systems (NeurIPS). – 2021. – P. 1–13.
21. Miyato T., Koyama M. CGANS with Projection Discriminator // Proceedings of the International Conference on Learning Representations (ICLR). – 2018. – P. 1–21.
 22. Kingma D.P., Ba J.L. ADAM: A Method for Stochastic Optimization // Proceedings of the International Conference on Learning Representations (ICLR). – 2015. – P. 1–15.
 23. Van Erven T., Harremoës P. Renyi Divergence and Kullback-Leibler Divergence // Journal of LaTeX Class Files. – 2007. – Vol. 6(1). – P. 1–24.
 24. Rybkin O., Daniilidis K., Levine S. Simple and Effective VAE Training with Calibrated Decoders // Proceedings of the 38th International Conference on Machine Learning (ICML), PMLR. – 2021. – Vol. 139. – P. 1–11.
 25. Gruber I. Generating Facial Images using VAEGAN // StudentSká 2018 vědecká konference. – 2018. – P. 38–39.
 26. Yang C., Zhuang P., Shi W., Lu A., Li P. Conditional Structure Generation through Graph Variational Generative Adversarial Nets // Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada. – 2019. – P. 1–12.
 27. Bermano A. H., Gal R., Alaluf Y., Mokady R., Nitzan Y., Tov O., Patashnik O., Cohen-Or D. State-of-the-Art in the Architecture, Methods and Applications of StyleGAN // Tel Aviv University. – 2022. – P. 1–19.
 28. Back J., Fine-tuning StyleGAN2 for Cartoon Face Generation [Preprint] // Cornell University. – 2021. – June 24. – P. 1–10.
 29. Croitoru F.-A., Hondru V., Ionescu R.T., Shah M. Diffusion Models in Vision: A Survey // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2022. – Vol. 14(8). – P. 1–25.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```

IMAGE_SIZE = 64
N_HIDDEN = 512
BATCH_SIZE = 32
EPOCHS = 100
LEARNING_RATE = 0.0002
DATA_PATH = "data/human-faces"
MODEL_SAVE_PATH = "models/"

import os
from PIL import Image
import numpy as np

def load_images(path=DATA_PATH, image_size=IMAGE_SIZE,
max_images=None):
    images = []
    for i, file in enumerate(os.listdir(path)):
        if file.endswith(('.jpg', '.png')):
            img = Image.open(os.path.join(path,
file)).convert('RGB')
            img = img.resize((image_size, image_size))
            img = (np.asarray(img) / 127.5 -
1.0).astype(np.float32) # Normalize to [-1, 1]
            images.append(img.transpose(2, 0, 1)) # CHW
            if max_images and i >= max_images:
                break
    return np.stack(images)

import numpy as np

```

```

def minibatches( minibatch_size, X, Y=None, shuffle_f=False
):
    indices = np.arange( len(X) )
    if shuffle_f:
        np.random.shuffle( indices )

    for start in range( 0, len(X), minibatch_size ):
        minibatch_indices = indices[ start :
start+minibatch_size ]
        if Y:
            yield X[minibatch_indices], Y[minibatch_indices]
        else:
            yield X[minibatch_indices]

import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np

class Encoder(nn.Module):
    def __init__(self, latent_dim=512):
        super().__init__()
        self.latent_dim = latent_dim
        self.conv = nn.Sequential(
            nn.Conv2d(3, 64, 5, stride=2, padding=2),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.Conv2d(64, 128, 5, stride=2, padding=2),
            nn.BatchNorm2d(128),
            nn.ReLU(),

```

```

        nn.Conv2d(128, 256, 5, stride=2, padding=2),
        nn.BatchNorm2d(256),
        nn.ReLU(),
    )
    self.fc_mu = nn.Linear(256 * 8 * 8, latent_dim)
    self.fc_logvar = nn.Linear(256 * 8 * 8, latent_dim)

    def forward(self, x):
        x = self.conv(x)
        x = x.view(x.size(0), -1)
        mu = self.fc_mu(x)
        logvar = self.fc_logvar(x)
        return mu, logvar

class Decoder(nn.Module):
    def __init__(self, latent_dim=512):
        super().__init__()
        self.fc = nn.Linear(latent_dim, 256 * 8 * 8)
        self.deconv = nn.Sequential(
            nn.ConvTranspose2d(256, 128, 4, stride=2,
padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64, 4, stride=2,
padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 3, 4, stride=2,
padding=1),
            nn.Tanh(),

```

```
)
```

```
def forward(self, z):
    x = self.fc(z).view(-1, 256, 8, 8)
    return self.deconv(x)
```

```
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3, 64, 4, stride=2, padding=1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(64, 128, 4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2),
            nn.Conv2d(128, 256, 4, stride=2, padding=1),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2),
            nn.Conv2d(256, 1, 4, stride=1),
            nn.Sigmoid()
        )
```

```
def forward(self, x):
    return self.model(x).view(-1, 1)
```

```
class VAEGAN(nn.Module):
    def __init__(self, latent_dim=512):
        super().__init__()
        self.latent_dim = latent_dim
```

```

        self.encoder = Encoder(latent_dim)
        self.decoder = Decoder(latent_dim)
        self.discriminator = Discriminator()

    def reparameterize(self, mu, logvar):
        std = torch.exp(0.5 * logvar)
        eps = torch.randn_like(std)
        return mu + eps * std

    def forward(self, x):
        mu, logvar = self.encoder(x)
        z = self.reparameterize(mu, logvar)
        recon = self.decoder(z)
        return recon, mu, logvar

def loss_function(recon_x, x, mu, logvar):
    BCE = F.mse_loss(recon_x, x, reduction='sum')
    KLD = -0.5 * torch.sum(1 + logvar - mu.pow(2) -
logvar.exp())
    return BCE + KLD

def train(model, data_loader, optimizer, epochs=10,
device='cuda'):
    model.to(device)
    model.train()
    for epoch in range(epochs):
        total_loss = 0
        for batch in data_loader:
            x, _ = batch # Ignore labels
            x = x.to(device)

```

```

optimizer.zero_grad()
recon, mu, logvar = model(x)
loss = loss_function(recon, x, mu, logvar)
loss.backward()
optimizer.step()
total_loss += loss.item()
    print(f"Epoch {epoch+1}, Loss: {total_loss /
len(data_loader.dataset):.4f}")

```

```

import kagglehub
import torch
from torch.utils.data import DataLoader, Subset
from torchvision import datasets, transforms
import torchvision.utils as vutils
import random
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def get_data_loader(batch_size=32):
    path =
kagglehub.dataset_download("ashwingupta3012/human-faces")

transform = transforms.Compose([
    transforms.Resize(64),
    transforms.CenterCrop(64),
    transforms.ToTensor(),
    transforms.Normalize((0.5,) * 3, (0.5,) * 3),
])

```

```

dataset = datasets.ImageFolder(
    root="/kaggle/input/human-faces",
    transform=transform
)

indices = list(range(len(dataset)))
random.seed(42)
random.shuffle(indices)
subset_indices = indices[:2000]
small_dataset = Subset(dataset, subset_indices)

return DataLoader(small_dataset, batch_size=batch_size,
shuffle=True)

def save_images(images, filename='generated.png',
grid_size=4):
    images = (images + 1) * 127.5 # [-1, 1] to [0, 255]
    images = images.astype(np.uint8).transpose(0, 2, 3, 1)

    rows = []
    for i in range(0, grid_size ** 2, grid_size):
        row = np.hstack(images[i:i + grid_size])
        rows.append(row)
    grid = np.vstack(rows)
    Image.fromarray(grid).save(filename)

device = torch.device('cuda' if torch.cuda.is_available()
else 'cpu')
model = VAEGAN().to(device)

data_loader = get_data_loader(batch_size=64)

```

```

optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)

train(model,      data_loader,      optimizer,      epochs=5,
device=device)

import torchvision

# Generate image file and print a few
def generate_images(model, device='cpu'):
    model.eval()
    z = torch.randn(16, model.latent_dim).to(device)
    with torch.no_grad():
        images = model.decoder(z).cpu().numpy()
    save_images(images, 'generated_faces.png')

def print_generated_images(model, device='cpu'):
    model.eval()
    z = torch.randn(4, model.latent_dim).to(device)
    with torch.no_grad():
        images = model.decoder(z).cpu()

    print("\nGenerated Image Stats:")
    for i, img in enumerate(images):
        print(f"Generated Image {i + 1}:
mean={img.mean():.3f}, min={img.min():.3f}, max={img.max():.3f}")

# Optional: visualize
    grid = torchvision.utils.make_grid(images, nrow=4,
normalize=True)

```

```
npimg = grid.permute(1, 2, 0).numpy()
plt.imshow(npimg)
plt.title("Generated Faces")
plt.axis("off")
plt.show()

generate_images(model, device=device)
print_generated_images(model, device=device)
```