МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

В. о. завідувача кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 20___ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

ГЕНЕРАЦІЯ СИНТЕТИЧНИХ РЕНТГЕНІВСЬКИХ ЗНІМКІВ НА ОСНОВІ ТЕКСТОВИХ ОПИСІВ ІЗ ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Інтелектуальні інформаційні системи»

Здобувач _____ Костянтин ШЕРЕМЕТ

«___»_____ 20__ р.

Керівник канд. техн. наук, доцент _____ Євген СІДЕНКО

«___»_____ 20__ р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	122 Комп'ютерні науки
Освітня програма	Інтелектуальні інформаційні системи

ЗАТВЕРДЖУЮ

В. о. завідувача кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 20___ р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____Шеремет Костянтин Олександрович_____
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: _Генерація синтетичних рентгенівських знімків на основі текстових описів із використанням нейронних мереж.

Керівник роботи: _Сіденко Євген Вікторович, в. о. завідувача кафедри інтелектуальних інформаційних систем, кандидат технічних наук, доцент.
(прізвище, ім'я, по батькові, посада, науковий ступінь, вчене звання)

Затверджена наказом ЧНУ ім. Петра Могили від «07» липня 2025 р. № 184.

2. Строк представлення кваліфікаційної роботи «16» грудня 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: _Набір даних: Відкритий набір медичних зображень Indiana University Chest X-rays, що містить рентгенівські знімки грудної клітки у прямій та боковій проекціях, а також відповідні їм радіологічні текстові описи. Базова архітектура: Попередньо навчена генеративна модель глибокого навчання Stable Diffusion XL (SDXL) Base 1.0. Інструментарій: Бібліотеки машинного навчання (PyTorch, Diffusers, Accelerate) та середовище розробки з підтримкою GPU-прискорення. Очікуваний результат:

інтелектуальна система для генерації синтетичних рентгенівських знімків на основі текстових описів.

4. Перелік питань, що підлягають розробці: Аналіз сучасного стану генерації синтетичних рентгенівських знімків та дослідження предметної області. Огляд та порівняння нейромережевих моделей та методів. Визначення оптимальної архітектури генерації та критеріїв якості. Розробка алгоритмів очищення текстових описів та підготовки зображень для формування навчальної вибірки. Налаштування та адаптація моделі SDXL за допомогою методу Low-Rank Adaptation (LoRA). Проведення навчання моделі, оцінка якості згенерованих зображень (анатомічна коректність, відповідність тексту) та аналіз ефективності запропонованого підходу.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Костянтин ШЕРЕМЕТ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «28» червня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Генерація синтетичних рентгенівських знімків на основі текстових описів із використанням нейронних мереж

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	27.06.2025	30.06.2025	Виконано
2	Аналіз предметної області та постановка задачі	01.07.2025	12.08.2025	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, аналіз публікацій та існуючих підходів до генерації синтетичних медичних даних для підвищення якості діагностичних систем	13.08.2025	02.10.2025	Виконано
4	Обґрунтування вибору дифузійних моделей та методу LoRA для синтезу рентгенографічних знімків.	03.10.2025	20.10.2025	Виконано
5	Реалізація обраних технологій з аналізом отриманих результатів	21.10.2025	24.11.2025	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	25.11.2025	26.11.2025	Виконано
7	Корегування роботи за результатами попереднього захисту	27.11.2025	05.12.2025	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	09.12.2025	09.12.2025	Виконано
9	Доробка та остаточне оформлення КР	10.12.2025	12.12.2025	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.12.2025	16.12.2025	Виконано

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Костянтин ШЕРЕМЕТ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«08» липня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 601м ЧНУ ім. Петра Могили

Шеремета Костянтина Олександровича

на тему: **«ГЕНЕРАЦІЯ СИНТЕТИЧНИХ РЕНТГЕНІВСЬКИХ ЗНІМКІВ НА ОСНОВІ ТЕКСТОВИХ ОПИСІВ ІЗ ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ»**

Актуальність роботи зумовлена зростаючим використанням методів штучного інтелекту в медичній візуалізації та водночас обмеженою доступністю якісних і збалансованих наборів рентгенівських знімків. Етичні, правові та організаційні обмеження ускладнюють збір і поширення клінічних даних, що негативно впливає на навчання та узагальнювальну здатність діагностичних моделей. У цьому контексті генерація синтетичних рентгенівських зображень на основі текстових описів розглядається як перспективний підхід до розширення медичних датасетів, збереження приватності пацієнтів і моделювання рідкісних патологічних станів.

Об'єкт – процес генерації синтетичних рентгенівських зображень грудної клітки на основі текстових описів.

Предмет – методи та засоби адаптації глибоких генеративних моделей (зокрема архітектури SDXL) для синтезу анатомічно коректних медичних зображень.

Метою роботи є підвищення якості генерації синтетичних медичних даних шляхом розробки та програмної реалізації інтелектуальної системи на базі донавчання моделі Stable Diffusion XL з використанням адаптерів LoRA та попередньої обробки текстових описів.

У ході виконання роботи було проведено дослідження сучасних методів генерації медичних зображень і реалізовано інтелектуальну систему для синтезу рентгенівських знімків грудної клітки, яка забезпечує формування текстових запитів, керування параметрами генерації та створення синтетичних зображень із вбудованими метаданими. Особливу увагу приділено аналізу та очищенню текстових описів, що використовуються для навчання моделі, а також структуризації клінічних ознак на рівні окремих знімків. Реалізовано підхід до донавчання дифузійної моделі Stable Diffusion XL із використанням адаптерів

LoRA, що дозволило зосередити модель на відтворенні вузькоспеціалізованих патологічних ознак.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків та додатків.

У першому розділі проаналізовано проблему нестачі медичних даних у рентгенодіагностиці та розглянуто сучасні підходи до генерації синтетичних зображень. Особливу увагу приділено дифузійним моделям і огляду актуальних наукових публікацій та існуючих систем генерації рентгенівських знімків.

У другому розділі розроблено загальну структуру інтелектуальної системи генерації синтетичних рентгенівських зображень і визначено її основні функціональні модулі. Також наведено аналіз архітектури дифузійної моделі Stable Diffusion та обґрунтовано вибір програмних інструментів і середовища реалізації.

У третьому розділі виконано аналіз і комплексну попередню обробку набору даних Chest X-rays (Indiana University), включно з очищенням, фільтрацією та структуризацією текстових описів. Додатково проведено збір статистики поширеності патологій на рівні окремих знімків і сформовано фінальний датасет для донавчання моделі.

У четвертому розділі реалізовано донавчання дифузійної моделі Stable Diffusion XL із використанням адаптерів LoRA та проведено якісний аналіз результатів генерації. Окремо представлено інтелектуальну систему «SynTheX-Ray» з користувацьким інтерфейсом для формування текстових запитів, керування параметрами генерації та збереження синтетичних рентгенівських знімків.

Кваліфікаційна робота містить 126 сторінок, 50 рисунків, 3 таблиці, 30 використаних джерел та 6 додатків.

Ключові слова: *генерація зображень, дифузійні моделі, рентгенографія, Chest X-ray, синтетичні медичні дані, Stable Diffusion, донавчання моделі, LoRA.*

ABSTRACT

to the qualification work
by the student of the group 601m of Petro Mohyla Black Sea National University

Sheremet Kostiantyn

"GENERATION OF SYNTHETIC X-RAY IMAGES FROM TEXTUAL DESCRIPTIONS USING NEURAL NETWORKS"

Relevance of the study is determined by the growing use of artificial intelligence methods in medical imaging and, at the same time, by the limited availability of high-quality and balanced datasets of chest X-ray images. Ethical, legal, and organizational constraints complicate the collection and dissemination of clinical data, which negatively affects the training and generalization capabilities of diagnostic models. In this context, the generation of synthetic chest X-ray images based on textual descriptions is considered a promising approach for expanding medical datasets, preserving patient privacy, and modeling rare pathological conditions.

Object – the process of generating synthetic chest X-ray images based on textual descriptions.

Subject – methods and tools for adapting deep generative models (in particular, the SDXL architecture) for the synthesis of anatomically correct medical images.

The purpose of the study is to improve the quality of synthetic medical data generation through the development and software implementation of an intelligent system based on fine-tuning the Stable Diffusion XL model using LoRA adapters and preprocessing of textual descriptions.

During the study, modern methods for medical image generation were investigated, and an intelligent system for synthesizing chest X-ray images was developed. The system provides functionality for forming textual prompts, controlling generation parameters, and producing synthetic X-ray images with embedded metadata. Special attention was paid to the analysis and cleaning of textual descriptions used for model training, as well as to the structuring of clinical features at the level of individual images. A fine-tuning approach for the Stable Diffusion XL diffusion model using LoRA adapters was implemented, enabling the model to focus on reproducing narrowly specialized pathological features.

The qualification work consists of an introduction, four chapters, conclusions, and appendices.

The first chapter analyzes the problem of medical data scarcity in chest radiography and reviews modern approaches to synthetic image generation. Particular attention is given to diffusion models, as well as to an overview of relevant scientific publications and existing systems for chest X-ray image generation.

The second chapter presents the overall structure of the intelligent system for synthetic chest X-ray generation and defines its main functional modules. In addition, the architecture of the Stable Diffusion diffusion model is analyzed, and the choice of software tools and implementation environment is justified.

The third chapter is devoted to the analysis and comprehensive preprocessing of the Chest X-rays (Indiana University) dataset, including data cleaning, filtering, and structuring of textual descriptions. Additionally, statistics on the prevalence of pathological findings at the level of individual images were collected, and a final dataset for model fine-tuning was formed.

The fourth chapter describes the fine-tuning of the Stable Diffusion XL diffusion model using LoRA adapters and presents a qualitative analysis of the generated results. The intelligent system “SynTheX-Ray” is also introduced, featuring a user interface for forming textual prompts, controlling generation parameters, and saving synthetic chest X-ray images.

The qualification work comprises 126 pages, 50 figures, 3 tables, 30 references, and 6 appendices.

Keywords: *image generation, diffusion models, radiography, chest X-ray, synthetic medical data, Stable Diffusion, model fine-tuning, LoRA.*

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ШІ – штучний інтелект

МН – машинне навчання

NN – Neural Network

CNN – Convolutional Neural Network

GenAI – Generative AI

CXR – Chest X-Ray

AP / PA – Anteroposterior / Posteroanterior

SDXL – Stable Diffusion XL

LoRA – Low-Rank Adaptation

VAE – Variational Autoencoder

CLIP – Contrastive Language-Image Pre-training

GPU – Graphics Processing Unit

VRAM – Video Random Access Memory

FP16 – Floating Point 16

CSV – Comma-Separated Values

UI – User Interface

API – Application Programming Interface

ВСТУП

У сучасних умовах розвитку медичної діагностики дедалі більшого значення набуває використання методів штучного інтелекту для аналізу, обробки та синтезу медичних зображень. Рентгенографія залишається одним із найпоширеніших та найінформативніших методів діагностики патологій органів грудної клітки, що зумовлює високу потребу в якісних рентгенівських знімках для навчання та валідації автоматизованих систем підтримки прийняття медичних рішень.

Водночас реальні медичні дані часто є обмеженими за обсягом і різноманіттям, що пов’язано з вимогами конфіденційності, складністю збору клінічних матеріалів, а також нерівномірним представленням патологічних випадків. У цьому контексті генерація синтетичних рентгенівських зображень розглядається як перспективний підхід до розширення наявних датасетів, підвищення ефективності навчання моделей комп’ютерного зору та збереження приватності пацієнтів.

Актуальність даної роботи зумовлена необхідністю розробки ефективних і надійних методів генерації синтетичних медичних зображень, які б відповідали анатомічним та клінічним вимогам. Одним із найперспективніших напрямів у цій сфері є використання глибоких нейронних мереж для генерації зображень, зокрема моделей типу Generative Adversarial Networks (GANs), Diffusion Models та Transformer-based Models. Перші забезпечують високий рівень фотореалістичності зображень завдяки змагальному навчанню двох нейронних мереж, другі – використовують поступове усунення шуму для відтворення структури зображення, а моделі на основі трансформерів вирізняються здатністю ефективно поєднувати текстові та візуальні ознаки.

Серед дифузійних моделей особливу увагу привертає Stable Diffusion, яка поєднує гнучкість, відкриту архітектуру та здатність до донавчання на спеціалізованих наборах даних, що робить її ефективним інструментом для задач медичної візуалізації. Разом із цим актуальною науково-практичною проблемою

залишається забезпечення відповідності між текстовими описами та згенерованими зображеннями, що потребує ретельної попередньої обробки й очищення текстових даних. Тому дослідження впливу якості текстових описів і спеціалізованого донавчання моделі на результати генерації є важливим і своєчасним.

Метою роботи є підвищення якості генерації синтетичних медичних даних шляхом розробки та програмної реалізації інтелектуальної системи на базі донавчання моделі Stable Diffusion XL з використанням адаптерів LoRA та попередньої обробки текстових описів.

Для досягнення поставленої мети в роботі необхідно було виконати такі **завдання**:

– проаналізувати сучасні підходи до генерації медичних зображень із використанням глибоких нейронних мереж;

– дослідити принципи роботи дифузійних моделей та особливості архітектури Stable Diffusion XL;

– виконати аналіз і попередню обробку набору рентгенівських знімків грудної клітки та відповідних текстових описів;

– розробити методику очищення, фільтрації та уніфікації текстових описів, що використовуються для навчання генеративної моделі;

– здійснити донавчання моделі Stable Diffusion XL із використанням підходу LoRA для медичного домену;

– виконати якісний аналіз результатів генерації та оцінити ефективність донавчання моделі;

– реалізувати інтелектуальну систему генерації синтетичних рентгенівських знімків із користувацьким інтерфейсом.

Об’єкт роботи – процес генерації синтетичних рентгенівських зображень грудної клітки на основі текстових описів. **Предмет** – методи та засоби адаптації глибоких генеративних моделей (зокрема архітектури SDXL) для синтезу анатомічно коректних медичних зображень.

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ГЕНЕРАЦІЇ СИНТЕТИЧНИХ ДАНИХ У МЕДИЧНІЙ ВІЗУАЛІЗАЦІЇ

1.1 Аналіз особливостей предметної області та проблеми нестачі даних у рентгенодіагностиці

У сучасному світі штучний інтелект стрімко розвивається, відкриваючи нові можливості для автоматизації, моделювання та творчості. Одним із найпомітніших напрямів останніх років є генерація зображень на основі текстових описів (Text-to-Image generation) – технологія, що поєднує досягнення у галузях комп'ютерного зору та обробки природної мови. Її поява стала можливою завдяки розвитку потужних нейронних архітектур, великих обсягів даних і зростанню обчислювальних ресурсів [1].

Сучасні системи здатні перетворювати короткий текстовий опис у складні, фотореалістичні зображення, які часто важко відрізнити від реальних фотографій. Це стало можливим завдяки еволюції генеративних моделей – від класичних автокодерів до складних багаторівневих нейронних мереж, таких як дифузійні моделі.

Генерація зображень має широкий спектр практичних застосувань. Вона використовується у дизайні, кінематографі, рекламі, освіті, моделюванні віртуальних середовищ і навіть у наукових дослідженнях. Завдяки таким моделям художники, розробники та дослідники отримують можливість швидко візуалізувати ідеї без потреби у великих виробничих ресурсах.

Особливої актуальності цей напрям набуває у медичній сфері, де кількість якісних даних часто є обмеженою. Медичні зображення, зокрема рентгенограми, комп'ютерні та магнітно-резонансні томограми, є важливим джерелом інформації для побудови систем діагностики на основі машинного навчання. Однак отримання великих та збалансованих наборів таких даних ускладнене через етичні, юридичні та технічні обмеження [2]. Саме тому генерація синтетичних медичних знімків, які

зберігають статистичні властивості реальних даних, але не містять конфіденційної інформації, є актуальним завданням сучасних досліджень [3].

Задача генерації зображень на основі текстового опису полягає у відтворенні візуального контенту, який відповідає заданому змісту тексту. На вхід системи подається текстова підказка (prompt), що містить опис бажаного результату, а на виході модель створює зображення, яке максимально узгоджується зі змістом запиту. Ця задача є міждисциплінарною, оскільки поєднує два різні напрямки штучного інтелекту:

- Natural Language Processing (NLP) – для інтерпретації тексту;
- Computer Vision (CV) – для формування зображення.

Завдяки такому поєднанню виникла можливість створювати універсальні системи, здатні не лише розуміти семантичний зміст запиту, а й передавати його через складні візуальні структури, кольори, перспективу та стиль.

Останні роки позначені появою низки генеративних систем, які значно змінили уявлення про можливості нейронних мереж. Моделі DALL·E 3 (OpenAI), Midjourney, Imagen (Google) та Stable Diffusion (Stability AI) стали знаковими прикладами успішної реалізації підходів до генерації зображень на основі тексту. Вони відрізняються архітектурними рішеннями, розміром моделей та обсягом навчальних даних, проте всі вони базуються на ідеї поступового уточнення випадкового шуму до осмисленого зображення, керованого текстовим контекстом.

Завдяки цьому підходу користувачі отримали змогу буквально за лічені секунди створювати складні сцени, які раніше вимагали годин роботи дизайнера або фотографа. Наприклад, сьогодні досить ввести короткий текстовий опис і система здатна візуалізувати навіть абстрактні ідеї або технічні концепції (рис. 1.1). Така доступність генерації відкрила широкі можливості для застосувань – від творчих індустрій до наукової візуалізації та освітніх проєктів.

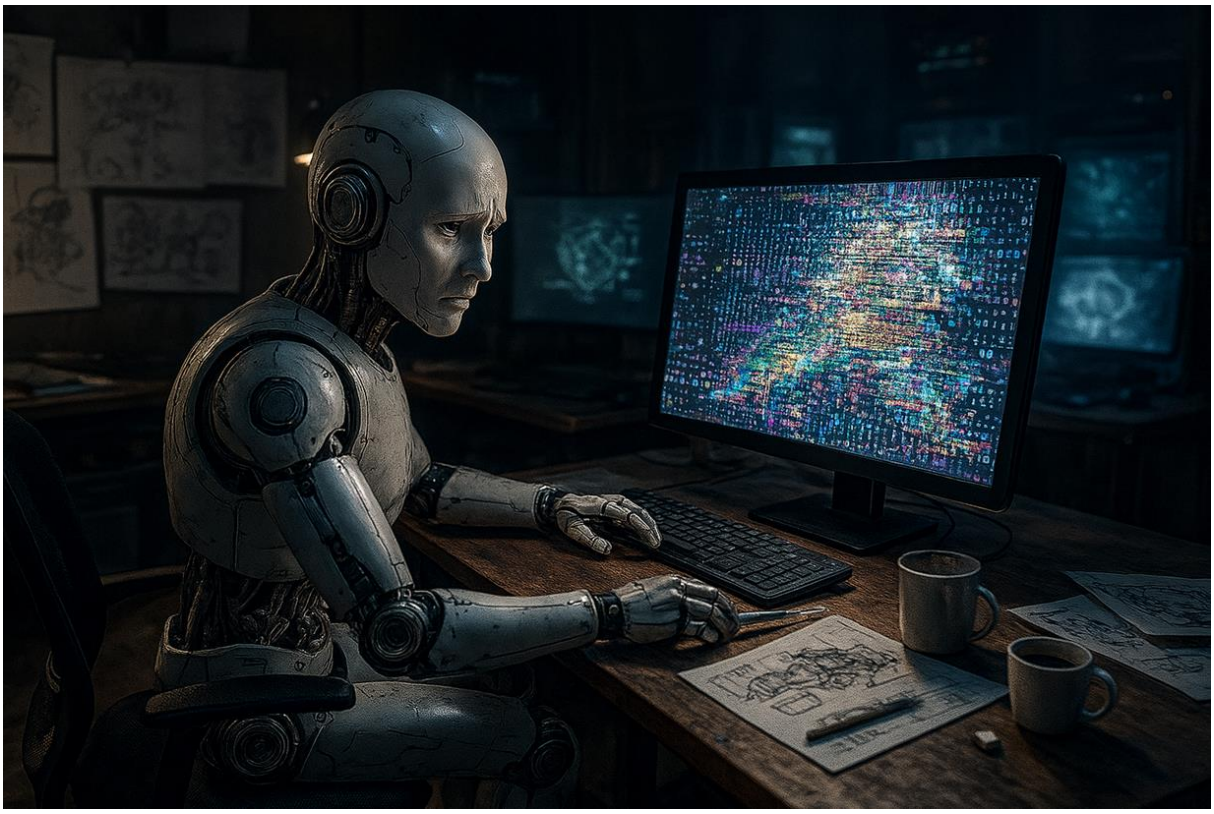

Рисунок 1.1 – Приклад зображення, згенерованого за допомогою нейронної мережі на основі текстового опису [4]

Разом із тим, попри стрімкий розвиток, більшість доступних платформ залишаються частково обмеженими у використанні. Частина систем є комерційними й вимагає підписки або кредитів для генерації зображень, інші системи мають технічні обмеження щодо якості, кількості запитів чи швидкодії. Навіть найсучасніші моделі часто демонструють труднощі у створенні спеціалізованих типів зображень, зокрема медичних, технічних чи наукових. Це зумовлено тим, що більшість навчальних наборів містить переважно побутові або художні сцени, що не відображають специфіки професійних галузей.

Варто також зазначити, що різні моделі мають власне спрямування, деякі орієнтовані на фотореалістичність і деталізацію, деякі вирізняються художнім стилем і виразною композицією, або гібридний підхід пропонує гнучкість у поєднанні реалістичних і творчих елементів. Така різниця у спрямуванні впливає не лише на візуальний стиль отриманих результатів, а й на ефективність їх

застосування у певних сферах – від мистецтва до медицини чи промислового дизайну.

Саме тому виникає потреба в донавчанні існуючих моделей на спеціалізованих даних, щоб вони могли відтворювати більш точні та достовірні зображення. У сфері медицини це має особливе значення, адже синтетичні знімки можуть стати основою для розширення навчальних вибірок без порушення конфіденційності пацієнтів.

1.2 Аналіз нейромережевих технологій для генерації зображень

Основні параметри нейронної мережі формуються за допомогою запитів, також відомих як промпт (prompt), для генерації зображення на основі текстових описів. У цьому процесі мовна модель використовується для перетворення текстового опису на числове представлення. Далі використовується це представлення для створення зображення.

Сучасні підходи до генерації зображень можна умовно поділити на три основні типи архітектур:

- генеративно-змагальні мережі (Generative Adversarial Networks, GANs);
- дифузійні моделі (Diffusion Models);
- моделі на основі трансформерів (Transformers).

Кожна з цих технологій має власні особливості, переваги та сфери застосування. Наприклад, GAN найкраще проявляють себе у створенні деталізованих, фотореалістичних зображень, що робить їх популярними у сфері синтезу облич, покращення якості зображень та художньої стилізації [6]. Дифузійні моделі, на яких базується більшість сучасних систем, таких як Stable Diffusion чи DALL·E 3, вирізняються високою стабільністю та контрольованістю процесу генерації, дозволяючи поетапно формувати структуру зображення від шуму до чіткого результату [7]. Натомість моделі на основі трансформерів, що поєднують текстові та візуальні представлення (наприклад, CLIP або Imagen), демонструють

глибше розуміння контексту запиту та здатність створювати зображення, які узгоджені зі змістом опису навіть при складних композиціях [8].

Такий підхід забезпечує високу гнучкість у налаштуванні вхідних даних, адаптацію до різних стилів і здатність створювати унікальні, креативні та надзвичайно деталізовані результати. Водночас якість кінцевого зображення значною мірою залежить від точності мовної моделі, ефективності обраної архітектури та обсягу обчислювальних ресурсів.

Генеративно-змагальні мережі (GANs): одними з найбільш поширених технологій у сфері генерації зображень, були вперше запропоновані Яном Гудфеллоу у 2014 році, а пізніше вдосконалені Алеком Редфордом та іншими дослідниками у 2015 році, що сприяло стандартизації їхньої архітектури. GAN складаються з двох основних компонентів: генератора та дискримінатора (рис. 1.2).

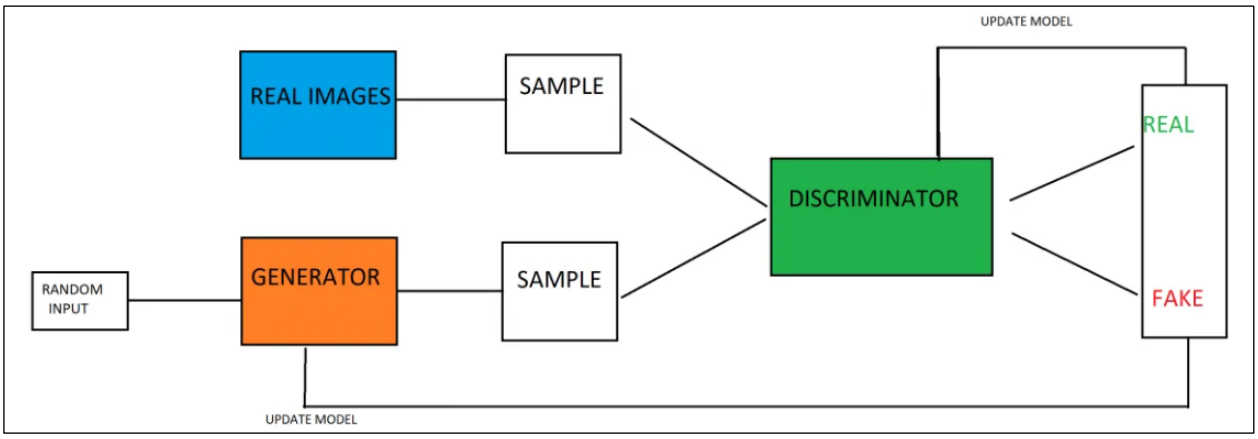

Рисунок 1.2 – Схема роботи генеративно-змагальної мережі (GAN) [6]

Генератор створює нові зображення, використовуючи шаблони, які модель вивчила з навчальних даних. Дискримінатор аналізує як реальні, так і згенеровані зображення, визначаючи їх автентичність. Процес навчання GAN базується на конкуренції між генератором і дискримінатором. Генератор намагається обдурити дискримінатора, створюючи реалістичні зображення, тоді як дискримінатор удосконалюється, розпізнаючи підроблені зображення. Ця взаємодія покращує обидві моделі, дозволяючи генератору створювати якісніші зображення. Після

завершення навчання генератор здатен створювати зображення, які важко відрізнити від справжніх. Він використовує випадковий вектор, перетворюючи його на зображення через латентний простір, що є стислою версією розподілу даних. Таким чином, GAN генерує нові зображення, схожі на ті, що були в навчальних даних.

Дискримінатор навчається на реальних зразках з навчального набору даних та підроблених зображеннях, створених генератором. Його навчання схоже на базову модель бінарної класифікації, де він прогнозує ймовірність того, чи є зображення справжнім чи підробленим.

Генератор перетворює випадкові числові дані на зображення, орієнтуючись на розподіл Гауса. Згенеровані зображення разом із реальними з навчального набору вводяться в дискримінатор, який оцінює їхню автентичність, надаючи ймовірність від 0 (підроблене) до 1 (справжнє).

У процесі навчання діє подвійна петля зворотного зв’язку. Дискримінатор отримує зворотний зв’язок від генератора, покращуючи свої параметри, тоді як генератор адаптується на основі відгуків дискримінатора. Взаємодія між генератором і дискримінатором є грою з нульовою сумою: виграш одного означає втрату іншого. Якщо дискримінатор успішно розпізнає підроблені зображення, його параметри залишаються незмінними, а генератор отримує штраф. У випадку, коли дискримінатор помиляється, його параметри оновлюються, а генератор отримує винагороду. Ідеальною метою є досягнення рівня, коли дискримінатор не здатен розрізнити справжні та підроблені зображення, досягаючи 50% ймовірності для обох категорій. Хоча це рідко трапляється на практиці, GAN залишаються корисними для численних завдань.

Для розпізнавання зображень зазвичай використовуються згорткові нейронні мережі (CNN), які ефективно виділяють особливості, такі як обличчя або цифри. Для їхнього навчання потрібна велика кількість зображень. Нейронні мережі прямого поширення (FFNN) часто застосовуються для генерації зображень, забезпечуючи їх перетворення на основі заданих даних.

Генеративні змагальні мережі мають широке застосування, особливо у створенні зображень та їх компонентів. Вони зазвичай використовуються в ситуаціях, коли відсутні або обмежені дані, забезпечуючи генерацію необхідної кількості даних.

Основні випадки використання GAN наведені нижче.

1. Створення нових прикладів для наборів даних: GAN дозволяють створювати нові приклади для навчальних наборів даних. Це особливо корисно, коли навчальних прикладів небагато. Модель генерує дані під різними орієнтаціями або кутами, що підвищує ефективність класифікації зображень. Найпоширеніша модель для цієї задачі – BigGAN.

2. Створення унікальних людських облич: після належного навчання GAN можуть генерувати високореалістичні зображення людських облич. Такі зображення знаходять застосування у тренуванні систем розпізнавання облич або в інших творчих проєктах (рис. 1.3). Найпоширеніша модель для цієї задачі – StyleGAN.

Рисунок 1.3 - Згенеровані обличчя за допомогою генеративно-змагальних мереж [5]

3. Перетворення зображення в зображення: GAN використовуються для розфарбовування чорно-білих зображень, трансформації ескізів у фотографічні зображення, а також для перетворення денних фотографій у нічні та навпаки. Найпоширеніша модель для цієї задачі – CycleGAN.

4. Перетворення тексту в зображення: GAN можуть створювати зображення на основі текстових описів. Наприклад, якщо подати текст, що описує сцену, модель генерує відповідний образ. Однак ця функція не є найсильнішою стороною GAN через обмеження у складності й деталізації зображень.

5. Редагування та відновлення зображень: GAN допомагають редагувати зображення, наприклад, видаляючи такі елементи, як дощ чи сніг, або відновлюючи старі та пошкоджені фотографії. Найпоширеніша модель для цієї задачі – CycleGAN.

6. Надроздільна здатність: GAN здатні підвищувати якість зображень із низькою роздільною здатністю, додаючи деталі й покращуючи загальну якість.

Хоча генеративні змагальні мережі (GAN) мають багато сильних сторін у сфері генерації зображень, вони не завжди найкраще підходять для завдань, пов'язаних з генерацією зображень на основі текстових описів:

– проблеми зі стабільністю навчання: GAN відомі своєю складністю у навчанні. Дуже важко досягти стабільності між генератором та дискримінатором, особливо коли додається ще одна змінна – текстовий опис;

– необхідність великих обсягів даних: для якісного навчання GAN потрібні великі набори даних пар "текст-зображення". Такий підхід потребує значних ресурсів для збору та підготовки даних;

– складність узгодження тексту та зображення: трансформація текстових описів у зображення є складним завданням, оскільки текст може бути дуже різноманітним і неточним у визначенні деталей, які потрібно передати на зображенні. Це може призвести до низької якості зображень.

Дифузійний підхід: дифузійні моделі займають центральне місце в сучасній екосистемі штучного інтелекту, визначаючи напрямок і темпи технологічного

прогресу. Вони використовують математичні принципи дисперсії, диференціальних рівнянь, генеративних послідовностей і Гауса, що кардинально змінює підхід до генеративних завдань у штучному інтелекті. Дифузійні моделі є центром уваги сучасних технологій від Nvidia, Google, Adobe та OpenAI. Здатність цих моделей перетворювати прості текстові підказки на реальні зображення, що стало популярним серед користувачів Інтернету, продемонстрована прикладами, такими як DALL.E 3, Stable Diffusion та MidJourney.

Згідно з дослідженням «Denoising Diffusion Probabilistic Models», дифузійні моделі є параметризованими ланцюгами Маркова, що навчаються через варіаційний висновок для створення вибірок, які відповідають даним через кінцевий час. Ланцюги Маркова описують систему, що перемикається між станами з певними ймовірностями. Поточний стан системи визначає можливі переходи в інші стани. Навчання через варіаційний висновок включає складні обчислення ймовірностей, спрямовані на оптимізацію параметрів ланцюга Маркова для відображення спостережуваних даних. У результаті модель генерує зразки, що відповідають реальним даним.

Простіше кажучи, дифузійні моделі можуть генерувати дані, подібні до тих, на яких вони були навчені. Наприклад, тренуючись на зображеннях кішок, модель здатна створювати реалістичні зображення кішок.

Дифузійні моделі додають шум (гауссовий шум) до навчальних даних у процесі прямої дифузії, а потім поступово усувають його у процесі зворотної дифузії, відновлюючи дані (рис. 1.4). Навчений алгоритм видалення шуму дозволяє генерувати високоякісні зображення з випадкових шумових зразків.

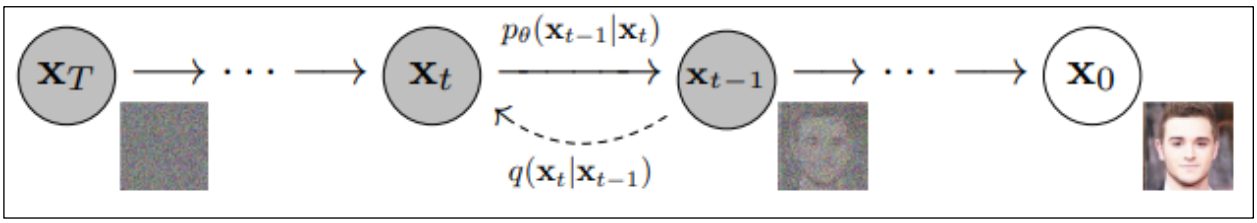

Рисунок 1.4 – Процес зворотної дифузії [7]

Ключові типи дифузійних моделей наведені нижче.

1. Імовірнісні моделі дифузії знешумлення (DDPM): ці моделі видаляють шум із візуальних або звукових даних, досягаючи вражаючих результатів у задачах покращення якості зображень і аудіо. Наприклад, у кіноіндустрії такі моделі використовуються для обробки зображень і відео.

2. Генеративні моделі на основі балів (SGM): ці моделі генерують нові зразки, оцінюючи логарифм щільності цільового розподілу. Вони можуть створювати високоякісні обличчя знаменитостей, демонструючи результати, подібні до GAN.

3. Стохастичні диференціальні рівняння (SDE): використовуються для моделювання випадкових процесів у фізиці та фінансах, описуючи зміни системи щодо часу.

Дифузійні моделі є потужним інструментом у сфері штучного інтелекту, пропонуючи інноваційні підходи до генерації, покращення та аналізу даних. Основні недоліки включають високу складність розробки і значний час навчання. Проте вони можуть бути адаптовані для задач, де необхідна висока деталізація та точність.

Моделі на основі трансформерів: трансформери є інноваційним типом моделей машинного навчання, спеціалізованим на обробці та інтерпретації послідовних даних, що робить їх оптимальними для завдань обробки природної мови (NLP). Щоб зрозуміти трансформатори, необхідно дослідити їх попередників – моделі «послідовність до послідовності», такі як рекурентні нейронні мережі (RNN) і довготривала короткочасна пам'ять (LSTM). RNN і LSTM побудовані на базі кодерів та декодерів, які аналізують дані поетапно. Кодер створює закодоване представлення вхідних даних, проходячи через кожний часовий крок і оновлюючи приховані стани. Наприкінці цього процесу формується «вектор контексту», який передається декодеру. Декодер прогнозує найбільш ймовірні слова вихідної послідовності на основі вектора контексту.

Трансформерна нейронна мережа виглядає приблизно так (рис. 1.5):

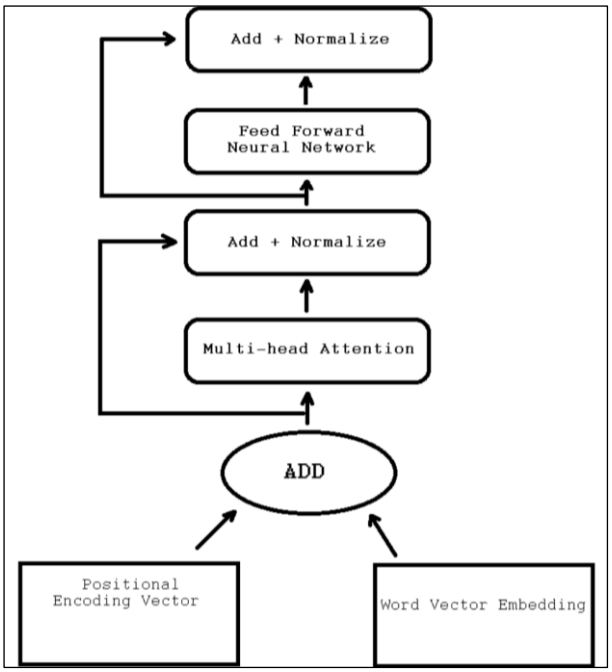

Рисунок 1.5 – Схема роботи трансформерної нейронної мережі [8]

Для підвищення ефективності таких моделей використовується механізм уваги. Він дозволяє зосереджуватися на найбільш релевантних частинах вхідних даних, ігноруючи менш важливу інформацію. Це покращує точність прогнозування та узгодженість результатів. Трансформери вирішують проблему обмежень RNN і LSTM, використовуючи позиційне кодування для врахування порядку слів у реченні. Векторні представлення слів у трансформерах гнучкіші завдяки синусоїдальним функціям, які дозволяють змінювати значення векторів залежно від позиції слова. Це забезпечує збереження інформації про порядок слів навіть при проходженні через шари нейронної мережі (рис. 1.6).

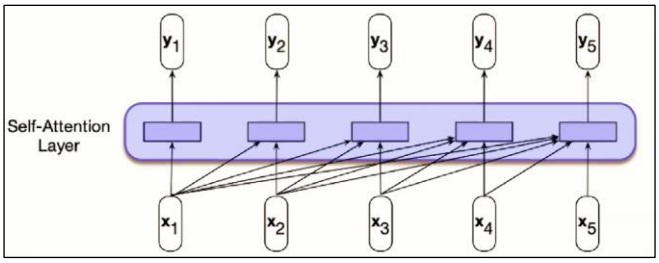

Рисунок 1.6 – Схема самоуваги в трансформерній нейронній мережі [9]

Головна відмінність трансформерів від RNN та LSTM полягає в паралельній обробці даних. Усі вхідні дані подаються до мережі одночасно, що суттєво підвищує швидкість обчислень. Кодери трансформерів перетворюють вхідні дані на представлення для навчання, а декодери – генерують вихід на основі отриманого представлення. І кодер, і декодер оснащені механізмом уваги.

Здатність трансформерів одночасно звертати увагу на кілька слів через механізм «уваги кількох голов» забезпечує глибоке розуміння контексту. Завдяки цьому трансформери мають значну перевагу перед RNN і LSTM у завданнях, які вимагають складної обробки контексту. Механізм уваги є найважливішою частиною трансформаторної мережі. Механізм уваги – це те, що дозволяє моделям трансформаторів вийти за межі уваги типової моделі RNN або LSTM. Традиційні моделі «Послідовність до послідовності» відкидають усі проміжні стани та використовують лише кінцевий вектор стану/контексту під час ініціалізації мережі декодера для створення прогнозів щодо вхідної послідовності. Відкидання всього, крім кінцевого вектора контексту, працює добре, коли вхідні послідовності досить малі. Проте зі збільшенням довжини вхідної послідовності продуктивність моделі погіршуватиметься під час використання цього методу. Це пов’язано з тим, що стає досить важко узагальнити довгу вхідну послідовність як один вектор. Рішення полягає в тому, щоб підвищити «увагу» моделі та використовувати проміжні стани кодера для побудови векторів контексту для декодера.

Механізм звернення уваги визначає, наскільки важливі інші вхідні маркери для моделі під час створення кодувань для будь-якого даного маркера. Наприклад, «воно» – це загальний займенник, який часто використовують для позначення тварин, коли їхня стать невідома. Механізм уваги дозволить трансформерній моделі визначити, що в поточному контексті «воно» відноситься до білки, оскільки вона може перевірити всі відповідні слова у вхідному реченні (рис. 1.7).

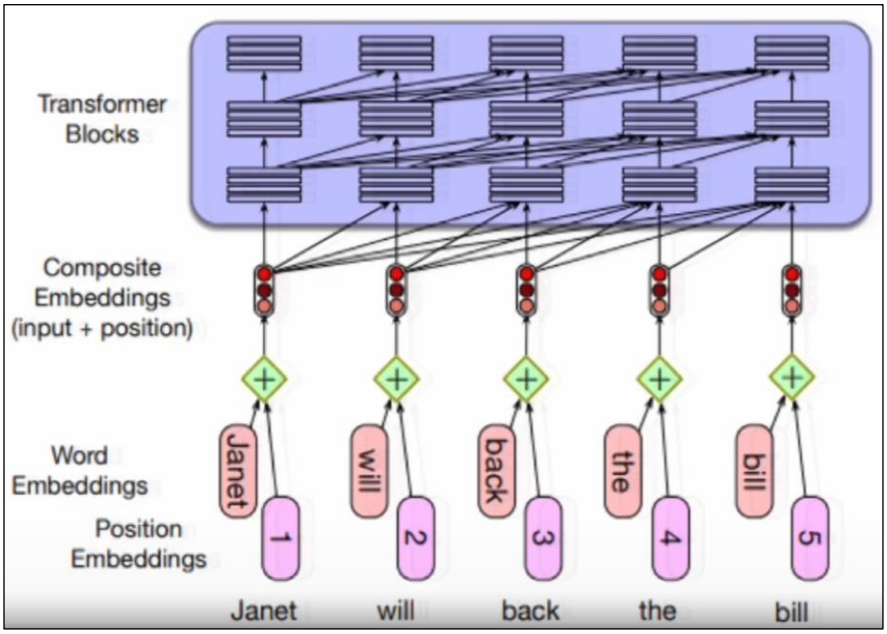

Рисунок 1.7 – Процес обробки вхідних даних та формування векторів контексту для кожного елемента послідовності [9]

Механізм уваги можна використовувати трьома різними способами: кодер-декодер, лише кодер, лише декодер. Увага кодера-декодера дозволяє декодеру враховувати вхідні послідовності під час генерації виходу, тоді як механізми уваги лише кодера та лише декодера дозволяють мережам враховувати всі частини попередньої та поточної послідовностей відповідно.

Побудову механізму уваги можна розділити на п’ять етапів наведених нижче.

1 Обчислення оцінки для всіх станів кодера.

2 Розрахунок ваг уваги.

3 Обчислення векторів контексту.

4 Оновлення вектора контексту з попереднім виведенням кроку часу.

5 Генерація виводу за допомогою декодера.

Перший крок полягає в тому, щоб декодер обчислив оцінку для всіх станів кодера. Це робиться шляхом навчання мережі декодера, яка є базовою нейронною мережею прямого зв’язку. Коли декодер навчається на першому слові вхідної послідовності, внутрішній/прихований стан ще не створено, тому останній стан кодера зазвичай використовується як попередній стан декодера.

Щоб обчислити вагові коефіцієнти уваги, використовується функція softmax для створення імовірнісного розподілу вагових коефіцієнтів уваги. Після того, як ваги уваги обчислено, потрібно обчислити вектор контексту. Це робиться шляхом множення вагових коефіцієнтів уваги та прихованого стану для кожного тимчасового кроку. Після обчислення вектора контексту він використовується поряд зі словом, згенерованим на попередньому часовому кроці, для створення наступного слова у вихідній послідовності. Оскільки декодер не має попереднього виводу, до якого можна було б звернутися на першому часовому кроці, замість нього часто використовується спеціальний маркер «початок».

Трансформери дедалі частіше застосовуються для генерації зображень завдяки їхній здатності ефективно працювати з послідовними даними та обробляти великі обсяги інформації. У задачах генерації зображень трансформери моделюють пікселі або патчі зображень як послідовності. Основна ідея полягає у використанні механізму уваги для захоплення довготривалих залежностей між різними частинами зображення.

1. Векторизація зображень: зображення розбивається на невеликі патчі (наприклад, 16x16 пікселів), і кожен патч перетворюється у вектор. Ці вектори зберігають інформацію про інтенсивність пікселів та просторове розташування.

2. Позиційне кодування: щоб зберегти інформацію про відносне розташування патчів, додається позиційне кодування. Це забезпечує збереження просторових зв'язків між різними частинами зображення.

3. Генеративний процес: трансформери, такі як Vision Transformer (ViT) або DALL-E, навчаються моделювати залежності між патчами. Наприклад, на етапі тренування трансформер вчиться, який наступний патч найбільш ймовірно підходить, базуючись на контексті попередніх патчів.

4. Обробка шуму: у стилі дифузійних моделей, трансформери можуть використовувати процеси додавання шуму та його видалення для поступової генерації зображень, починаючи з випадкових шумів.

5. Застосування в практиці: сучасні трансформерні моделі, як-от DALL-E або Imagen, дозволяють генерувати високоякісні зображення за текстовими підказками. Вони інтегрують текстові векторизації та зображення у спільний простір, що забезпечує високий рівень узгодженості між текстом і зображенням.

Таким чином, трансформери відкривають нові горизонти для генерації зображень, забезпечуючи високий рівень деталізації та точність у передачі задуму. Переваги трансформерів включають їхню здатність до генерації зображень на основі складних описів і гнучкість у навчанні. Проте вони є дуже ресурсозатратними, а їхнє навчання може бути економічно недоцільним для малих проектів.

Для створення системи генерації синтетичних рентгенівських знімків було обрано саме дифузійний підхід, оскільки він поєднує високу якість, стабільність навчання та гнучкість адаптації до специфічних типів медичних зображень.

Однією з найпоширеніших і технічно зрілих реалізацій цього підходу є Stable Diffusion, яка має низку переваг, описаних нижче.

1. Відкрита архітектура. Модель має відкритий вихідний код, що дозволяє модифікувати її під специфічні потреби конкретних досліджень, у тому числі медичних.

2. Висока якість результатів. Завдяки механізму латентної дифузії модель здатна генерувати зображення з високим рівнем деталізації та збереженням ключових особливостей об’єкта.

3. Оптимізація обчислень. Stable Diffusion може працювати на GPU середнього рівня, що робить її доступною для невеликих дослідницьких проєктів.

4. Гнучкість у використанні. Модель легко інтегрується з хмарними платформами, зокрема Google Colab або Kaggle, що дозволяє виконувати обчислення у віддаленому середовищі без потреби у власних потужних ресурсах.

Завдяки активній підтримці спільноти та наявності численних прикладів реалізацій, Stable Diffusion є одним із найкращих виборів для досліджень, пов’язаних із генерацією синтетичних медичних даних.

Таким чином, для подальшої роботи в межах цього проєкту пропонується використати дифузійну модель Stable Diffusion як базову технологію для створення системи генерації зображень на основі текстових описів. Це дозволить поєднати високу якість синтезованих результатів із практичністю впровадження та можливістю подальшого донавчання моделі на вузькоспеціалізованих медичних наборах даних.

1.3 Огляд останніх публікацій та систем генерації синтетичних рентгенівських знімків

Огляд останніх досліджень у сфері генерації синтетичних рентгенівських знімків є важливим кроком для розробки власної інтелектуальної системи, оскільки дозволяє ознайомитися з найновішими підходами, визначити ефективні архітектури та виявити технологічні тренди, що набувають популярності у медичній IT-галузі. Наприклад, у роботі Yuanfeng Ji *et al.* [10] наголошується на ефективності латентних дифузійних моделей, таких як Stable Diffusion, для синтезу chest X-ray (рентгенівський знімок грудної клітки). Автори виділяють кілька ключових технологій: використання масштабних текстово-візуальних датасетів для комплексного навчання; умовне керування генерацією за допомогою промптів, масок чи сегментацій; та технології spatial control (просторове керування) для точного завдання патологій. Дослідження також показують, що донавчання сучасних дифузійних моделей на відповідних медичних даних значно підвищує якість і різноманітність синтетичних знімків, а їх використання у тренувальних вибірках покращує ефективність і справедливість медичних AI-систем навіть при обмеженій кількості реально анотованих даних.

Водночас Junjie Shentu [11] підкреслює ефективність інтегрованих мультимодальних підходів у генерації chest X-ray, зокрема комбінації diffusion та GAN-технологій. Згідно результатів роботи, комбінування diffusion-модуля (для високої реалістичності й деталізації знімків) і GAN-компонента (для контрольованого редагування окремих ознак чи підвищення варіативності)

дозволяє значно підсилити якість синтетичних рентгенівських знімків, адаптуючи генерацію під текстовий опис з реальних клінічних сценаріїв. Такий гібридний підхід забезпечує гнучкість, універсальність і точність – саме те, чого часто бракує «чистим» моделям. Робота підтверджує, що для задач медичної синтетики найкращі результати досягаються у симбіозі diffusion та GAN, а не у виборі одного алгоритму.

В статті Tobias Weber *et al.* [12] пропонують потужний каскадний diffusion-підхід для генерації chest X-ray знімків із мегапіксельною роздільною здатністю, що об’єднує text-to-image генерацію та складний pre/postprocessing pipeline. Вперше було реалізовано узгоджене управління генерацією зображень за текстовим описом – через багаторівневий latent diffusion, кожний етап якого відповідає за деталізацію чи загальний зміст синтетичного chest X-ray. Автори демонструють, що каскадні дифузійні архітектури суттєво покращують якість, реалістичність та варіативність або «різноманітність» синтетичних даних для тренування, валідації та тестування AI-моделей. Науковці довели, що масштабовані diffusion-архітектури можуть гарантувати клієнтський рівень якості синтетики для медичних досліджень і практики.

В результаті роботи Gregory Schuit *et al.* [13] показали, що дифузійні моделі мають значну перевагу над GAN за критеріями візуального реалізму та оцінки непідготовлених і експертних радіологів при синтезі chest X-ray знімків. Водночас GAN продовжують демонструвати свою актуальність у специфічних сценаріях, де необхідно контролювати наявність чи відсутність окремих патологічних ознак. Робота підкреслює, що незалежний медичний аудит і багаторівнева перевірка якості залишаються фундаментальними для впровадження синтетичних рентгенівських даних у клінічну практику, а вибір моделі слід робити, виходячи з конкретних цілей синтезу.

В свою чергу, дослідження Muhammad Usman Akbar *et al.* [14] наголошує, що хоча diffusion-моделі забезпечують високу якість синтетичних медичних знімків, вони схильні запам’ятовувати приклади зі свого тренувального набору, що

особливо проявляється при малих обсягах даних. В порівнянні, GAN-архітектури демонструють більше «узагальнюючої» властивості – для малих датасетів синтетика від GAN менш схильна відтворювати оригінальні знімки один-в-один. Це бачення вказує на важливість уважного аналізу підбору моделі з точки зору приватності й справжньої новизни синтетичних chest X-ray для невеликих/специфічних задач, а також – на доцільність використання додаткових методів оцінювання результатів для diffusion-моделей.

Це підтверджують висновки Arwa H. Alshanbari та Salha M. Alzahrani [15], що хоча diffusion є лідером за якістю й стабільністю при роботі з великими датасетами, GAN залишаються незамінними для невеликих задач, де потрібна варіативність та швидка адаптація, а VAE – для explainable AI чи створення компактних синтетичних зображень. Усі підходи можуть бути впроваджені згідно ресурсних можливостей і завдань – рекомендація полягає у гібридизації методів для найліпшого результату, а для практики diffusion виправдовує себе як універсальний інструмент генерації chest X-ray, що стає галузевим стандартом.

Стаття Abdullah al Nomaan Nafi *et al.* [16] демонструє, що diffusion-моделі синтезу дозволяють створювати повноцінні та статистично релевантні датасети для навчання медичних AI, навіть у сферах із гострою нестачею реальних зображень. Автори показали, що CNN-класифікатори, навчені виключно на синтетичних даних, досягають конкурентної якості, коли тестуються на реальних кейсах по задачах класифікації – наприклад, діагностики пухлин мозку, ALL або COVID-19. Додатково, застосування explainable AI (LIME) підтверджує, що моделі фокусуються на релевантних ознаках зображення, а це підвищує довіру до системи в медичному контексті. Таким чином, diffusion-підходи – це не лише рішення для масштабування навчальних даних, а й аргументовано ефективний інструмент для створення навіть невеликих, але якісних систем медичного аналізу з мінімальною залежністю від реальних анотованих даних.

Дослідники Maya Pavlova *et al.* [17] акцентують увагу на значущості створення відкритих, масштабних chest X-ray датасетів та впровадженні explainable

AI для довіреної діагностики COVID-19. В результаті роботи було зібрано найбільшу на той момент колекцію chest X-ray знімків, верифіковану медичними експертами. Залучення перспективної і прозорої синтетики, а також пояснюваних моделей, довело важливість поєднання великої вибірки і зрозумілих рішень для підвищення довіри до сучасних медичних AI-систем.

Підсумовуючи результати проведеного огляду, можна зробити висновок, що дифузійні моделі на сьогодні є провідним підходом у генерації синтетичних рентгенівських знімків, демонструючи найвищу якість, стабільність і масштабованість серед усіх розглянутих архітектур. Більшість сучасних досліджень підтверджують, що саме латентні diffusion-підходи, такі як *Stable Diffusion*, забезпечують реалістичність і різноманітність синтетичних chest X-ray навіть при обмежених наборах реальних даних. Водночас відзначається потенціал гібридних рішень, що поєднують diffusion і GAN, які можуть підвищити керованість та узагальнювальні властивості системи.

1.4 Постановка задачі дослідження

На підставі поставленої мети дослідження було необхідно вирішити низку завдань, спрямованих на розробку інтелектуальної системи генерації синтетичних рентгенівських знімків грудної клітки на основі текстових описів із використанням дифузійних моделей. Реалізація запропонованого підходу сприятиме підвищенню ефективності підготовки медичних даних для навчання діагностичних моделей, зменшенню залежності від обмежених наборів реальних знімків та забезпеченню можливості створення додаткових синтетичних зразків для досліджень.

До основних завдань дослідження належать:

– аналіз сучасного стану проблеми генерації медичних зображень, зокрема синтетичних рентгенівських знімків, і визначення вимог до їхньої достовірності та якості;

– огляд існуючих підходів до генерації зображень за допомогою нейронних мереж та вибір архітектури, найбільш придатної для донавчання на медичних даних;

– формування та попередня обробка спеціалізованого датасету рентгенівських знімків для адаптації обраної моделі;

– розроблення архітектури інтелектуальної системи генерації, що включає модулі введення текстових описів, генерації зображень, оцінювання результатів і взаємодії з користувачем;

– реалізація програмної структури системи на основі Python із можливістю подальшого розширення функціональності та інтеграції у середовища досліджень медичних даних;

– проведення експериментів з генерації синтетичних рентгенівських знімків та аналіз отриманих результатів з точки зору візуальної достовірності й відповідності текстовим описам.

При формулюванні зазначених завдань враховано сучасні тенденції розвитку генеративних моделей у медицині, необхідність забезпечення високої якості зображень, а також потенційну можливість використання результатів у подальших дослідженнях для навчання систем підтримки діагностики.

Таким чином, постановка задачі дослідження передбачає комплексну розробку методів, алгоритмів та архітектури інтелектуальної системи, здатної здійснювати генерацію синтетичних рентгенівських знімків грудної клітки на основі текстових описів, що сприятиме підвищенню ефективності обробки та аналізу медичних зображень.

Висновки до розділу 1

У першому розділі було проведено аналіз сучасного стану проблеми генерації синтетичних рентгенівських знімків та сформульовано основні завдання дослідження. На основі вивчення предметної області встановлено, що одним із ключових викликів у медичній діагностиці є обмежена кількість якісних зображень

для навчання нейронних мереж, особливо у спеціалізованих галузях, таких як рентгенологія. Використання синтетичних даних є перспективним напрямом, який дозволяє підвищити ефективність навчання моделей без додаткових витрат на збір і розмітку великих обсягів реальних медичних даних.

У ході аналізу сучасних нейромережевих технологій генерації зображень було розглянуто підходи на основі GAN, трансформерів та дифузійних моделей. Визначено, що саме дифузійні моделі демонструють найвищу стабільність і якість при створенні зображень, зберігаючи деталі структури та текстури, що є критично важливим для медичних знімків. Серед існуючих реалізацій особливу увагу приділено моделі Stable Diffusion, яка поєднує відкриту архітектуру, можливість тонкого налаштування та ефективність у використанні обчислювальних ресурсів.

Під час огляду наукових публікацій та сучасних систем було виявлено, що більшість досліджень спрямовані на покращення якості синтезованих медичних зображень, зокрема рентгенівських, за допомогою дифузійних або гібридних моделей. Проаналізовано приклади існуючих підходів і доведено актуальність подальших досліджень у цьому напрямі, зокрема щодо адаптації моделей до специфіки медичних датасетів та розробки інтегрованих систем для генерації зображень за текстовими описами.

На підставі проведеного аналізу сформульовано мету дослідження та постановку задачі, яка полягає у розробці інтелектуальної системи генерації синтетичних рентгенівських знімків грудної клітки на основі текстових описів. Визначено комплекс завдань, що включає аналіз предметної області, вибір архітектури нейромережі, попередню обробку даних, проєктування структури системи та експериментальну перевірку результатів. Таким чином, перший розділ закладає теоретичну основу для подальшої реалізації системи, опис якої наведено у наступному розділі.

2 СТРУКТУРА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ГЕНЕРАЦІЇ СИНТЕТИЧНИХ РЕНТГЕНІВСЬКИХ ЗНІМКІВ ГРУДНОЇ КЛІТКИ

2.1 Елементи структури інтелектуальної системи генерації синтетичних рентгенівських знімків

Розроблена система призначена для генерації синтетичних рентгенівських знімків грудної клітки на основі текстових описів користувача з використанням дифузійної нейронної мережі Stable Diffusion XL. Архітектура системи базується на послідовному конвеєрі обробки даних – від введення текстового опису до формування та збереження згенерованого зображення. Передбачається, що модель, яка використовується в системі, проходить попереднє донавчання на спеціалізованому медичному наборі даних ще до етапу розгортання. Це дозволяє уникнути додаткових обчислювальних витрат під час запуску застосунку, зменшити час очікування користувача та забезпечити стабільну і відтворювану генерацію результатів.

Система побудована з п’яти основних підсистем, кожна з яких відповідає за виконання окремого набору функцій.

Інтерфейс користувача

Модуль взаємодії з користувачем реалізовано за допомогою фреймворку Streamlit, що дозволяє швидко створювати веб-інтерфейси у середовищі Python. Інтерфейс надає користувачу можливість вводити текстовий опис зображення (prompt), обирати або редагувати шаблони описів, а також керувати основними параметрами генерації, зокрема кількістю кроків дифузії, значенням guidance scale та seed. У центральній частині інтерфейсу відображається результат генерації, після чого користувач може зберегти зображення у форматі PNG. Поточна реалізація орієнтована на локальну демонстрацію та експериментальне використання без необхідності серверного розгортання.

Підготовка текстового опису

Після введення користувацького опису він передається до модуля підготовки текстового опису, який виконує базову перевірку коректності введених даних, зокрема обробку порожніх значень. Система не виконує автоматичного перекладу тексту, оскільки стабільна робота моделі передбачає використання англомовних описів. У разі відсутності введеного тексту застосовується стандартний опис за замовчуванням.

Ініціалізація та керування моделі

Даний модуль відповідає за завантаження та керування дифузійною моделлю Stable Diffusion XL. Він забезпечує підключення попередньо донавчених ваг (зокрема LoRA), ініціалізацію компонентів пайплайна (текстового енкодера, U-Net, VAE), а також розміщення моделі на доступному апаратному прискорювачі (GPU). Реалізовано механізм звільнення пам’яті та повторного завантаження моделі, що дозволяє змінювати активну модель без перезапуску всієї системи.

Модуль генерації зображення

Отриманий та валідований опис надходить до модуля генерації зображення, де відбувається основний дифузійний цикл. Текст кодується через текстовий енкодер (наприклад, CLIP) у векторне представлення, яке подається до U-Net із cross-attention блоками. Ініціалізований латентний шум поетапно денойзиться відповідно до обраного scheduler і параметрів (guidance scale, num_inference_steps). Після завершення ітерацій латентне представлення декодується через VAE у растрове зображення заданого розміру. Модуль також підтримує фіксацію seed, що забезпечує відтворюваність результатів при повторних запусках.

Модуль післяобробки та збереження результатів

Після завершення генерації зображення передається до модуля післяобробки, який у поточній реалізації виконує збереження результату у форматі PNG. Додатково до зображення вбудовуються службові метадані, що містять текстовий опис, параметри генерації та значення seed. Такий підхід дозволяє зберігати контекст створення зображення без необхідності окремих файлів опису. Функції

автоматичної візуальної корекції або фільтрації артефактів навмисно не застосовуються, оскільки вони можуть впливати на клінічну інформативність зображень.

Основні елементи структури інтелектуальної системи інтелектуальної системи генерації синтетичних рентгенівських знімків та їх функціональне призначення узагальнено в таблиці 2.1.

Таблиця 2.1 – Основні елементи системи інтелектуальної системи генерації синтетичних рентгенівських знімків

№	Назва елементу	Опис функцій
1	Інтерфейс користувача	Забезпечує взаємодію користувача з системою через веб-інтерфейс. Дозволяє обирати модель, задавати параметри генерації, вводити текстові описи та переглядати результати генерації в режимі реального часу.
2	Модуль підготовки текстового опису	Виконує базову валідацію користувацького опису, зокрема перевірку на порожні значення та формування стандартного опису за замовчуванням.
3	Модуль ініціалізації та керування моделі	Відповідає за завантаження та конфігурацію дифузійної моделі Stable Diffusion XL, підключення донавчених LoRA-ваг, ініціалізацію компонентів пайплайна (VAE, scheduler), а також розміщення моделі на GPU. Модуль підтримує динамічну заміну активної моделі без перезапуску системи та керування використанням пам’яті.
4	Модуль генерації зображення	Основний обчислювальний блок, де відбувається дифузійний процес. Текст кодується через CLIP, далі U-Net поступово денойзить латентне представлення. Після завершення ітерацій VAE декодує результат у готове зображення.
5	Модуль післяобробки та збереження результатів	Здійснює збереження згенерованого зображення у форматі PNG з додаванням службових метаданих, що містять текстовий опис. Такий підхід забезпечує збереження контексту генерації без використання зовнішніх файлів опису.

Для кращого розуміння принципу роботи розробленої системи на рис. 2.1 наведено її узагальнену структурну схему. В центрі архітектури розташовано

модуль ініціалізації та генерації зображення, який реалізує основний дифузійний процес на базі моделі Stable Diffusion. Інші підсистеми забезпечують підготовку, валідацію, післяобробку та збереження результатів.

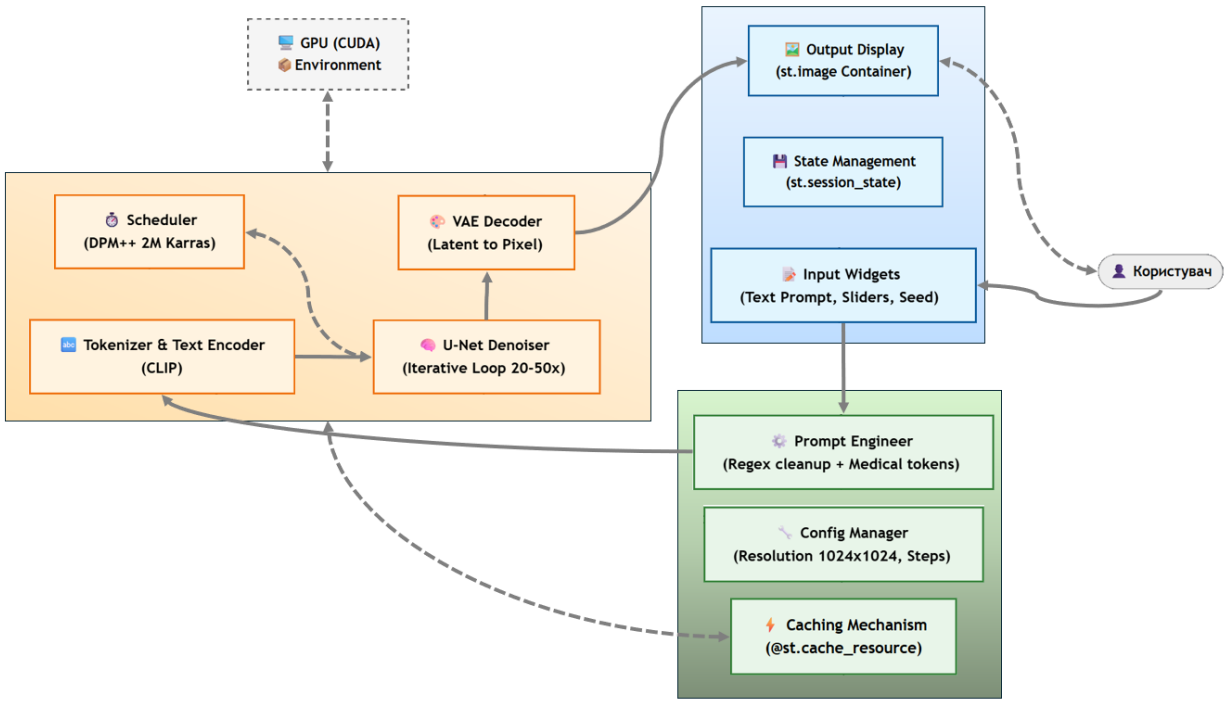

Рисунок 2.1 – Схема структури системи

Описана послідовність модулів формує практичний скелет системи, яка формується після етапу навчання моделі. На основі натренованої нейронної мережі реалізуються окремі компоненти для обробки запитів, генерації зображень, взаємодії з користувачем та збереженням результатів. Кожен із блоків розробляється як незалежний Python-модуль або набір функцій, що забезпечує гнучкість, зручність тестування й можливість подальшого розширення системи (наприклад, інтеграція механізмів автоматичного донавчання, логування помилок чи збору зворотного зв’язку від експертів).

2.2 Огляд архітектури дифузійної моделі Stable Diffusion

Stable Diffusion належить до класу дифузійних моделей, які генерують зображення шляхом поступового додавання та видалення гауссового шуму.

Основна ідея полягає в тому, щоб навчити модель відтворювати шлях зворотний до зашумлення – тобто навчити її «очищати» випадковий шум, покроково відновлюючи реалістичне зображення.

На відміну від класичних генеративних моделей, які безпосередньо працюють у просторі пікселів (наприклад, GAN чи VAE у початковій формі), Stable Diffusion використовує Latent Diffusion Model (LDM) – дифузійну модель, що функціонує у латентному (стисненому) просторі. Це означає, що всі обчислення виконуються не на рівні пікселів, а на рівні компактного представлення зображення. Такий підхід значно знижує обчислювальні витрати і робить модель доступною навіть для пристроїв із середніми апаратними ресурсами.

На етапі навчання дифузійна модель вчиться прогнозувати шум, який був доданий до латентного представлення зображення. Ключовим завданням є мінімізація різниці між передбаченим і фактичним шумом. Таким чином, замість безпосереднього відновлення пікселів модель опановує процес реконструкції зображення через поступове «розшумлення». Цей підхід виявився стабільнішим і більш контрольованим порівняно з генеративними змагальними мережами (GAN), оскільки усуває проблему нестійкої збіжності під час навчання двох суперницьких моделей – генератора та дискримінатора.

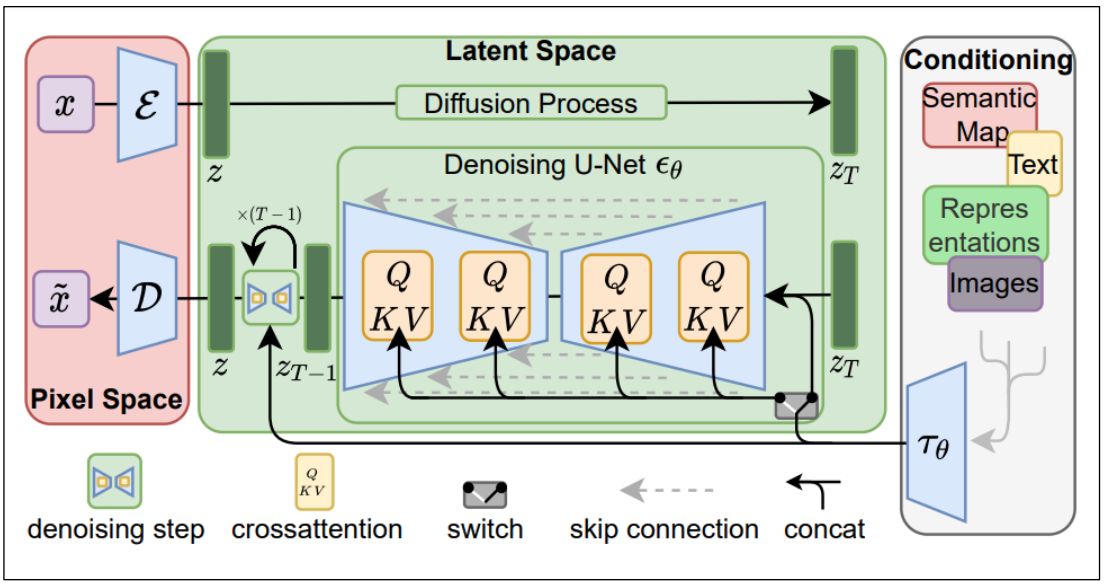

Рисунок 2.2 – Архітектура латентної дифузійної моделі [18]

Архітектура моделей Stable Diffusion постійно вдосконалюється, проте існують основні компоненти, які залишаються незмінними та визначають базовий підхід до генерації зображень.

Модель складається з трьох основних компонентів, описаних нижче.

1. Варіаційний автокодер (Variational Autoencoder, VAE). Цей модуль відповідає за перетворення зображення з простору пікселів у латентне представлення та навпаки:

– енкодер VAE стискає вхідне зображення (наприклад, 512×512 пікселів) у багатовимірний латентний тензор (зазвичай розміром 4×64×64). Це дозволяє зберегти семантичні ознаки, усуваючи надлишкові піксельні деталі;

– декодер VAE виконує зворотне перетворення – відновлює підсумкове зображення з латентного простору після завершення процесу генерації.

Таким чином, VAE виступає своєрідним «мостом» між світом пікселів і латентним простором, у якому працює ядро моделі (рис. 2.3).

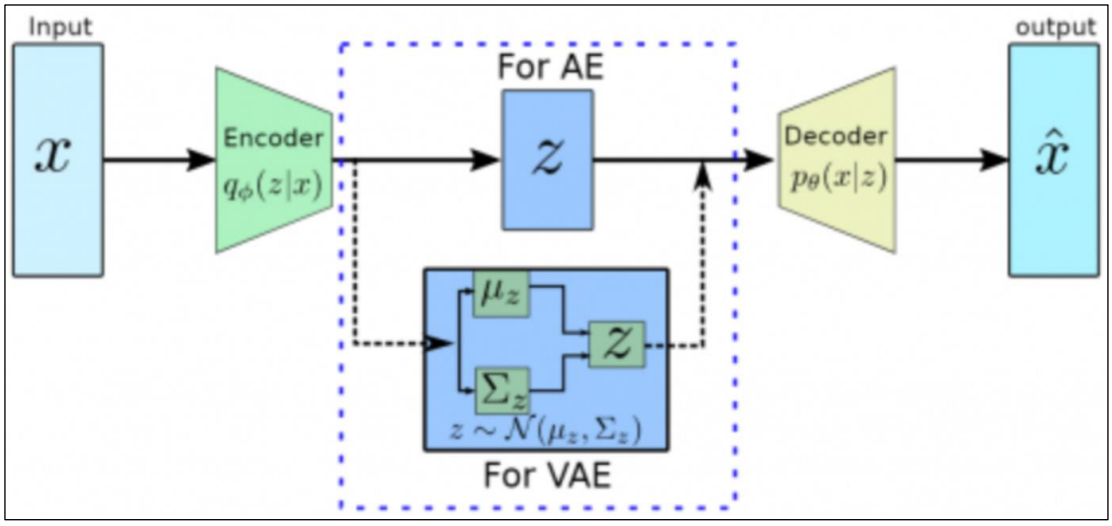

Рисунок 2.3 – Архітектура VAE [19]

2. U-Net. Це ядро моделі, яке виконує ітеративне видалення шуму з латентного представлення, поступово наближаючи його до структури реального зображення. U-Net у Stable Diffusion побудований на основі ResNet-блоків із пропускними з’єднаннями (skip connections), що дозволяє поєднувати детальну та

узагальнену інформацію (рис. 2.4). Важливим компонентом U-Net є Cross-Attention – механізм, який забезпечує узгодження між текстовим описом і генерованим зображенням. Кожен токен тексту після обробки трансформером формує вектор у семантичному просторі. Під час денойзінгу U-Net використовує ці вектори як «орієнтири», коригуючи процес відновлення зображення таким чином, щоб воно відповідало опису. Наприклад, якщо користувач вводить prompt «рентгенівський знімок грудної клітки», то Cross-Attention направляє генерацію на характерні форми та текстури, притаманні цій категорії зображень. Такий механізм забезпечує узгодженість між змістом тексту й просторовою структурою кінцевого зображення.

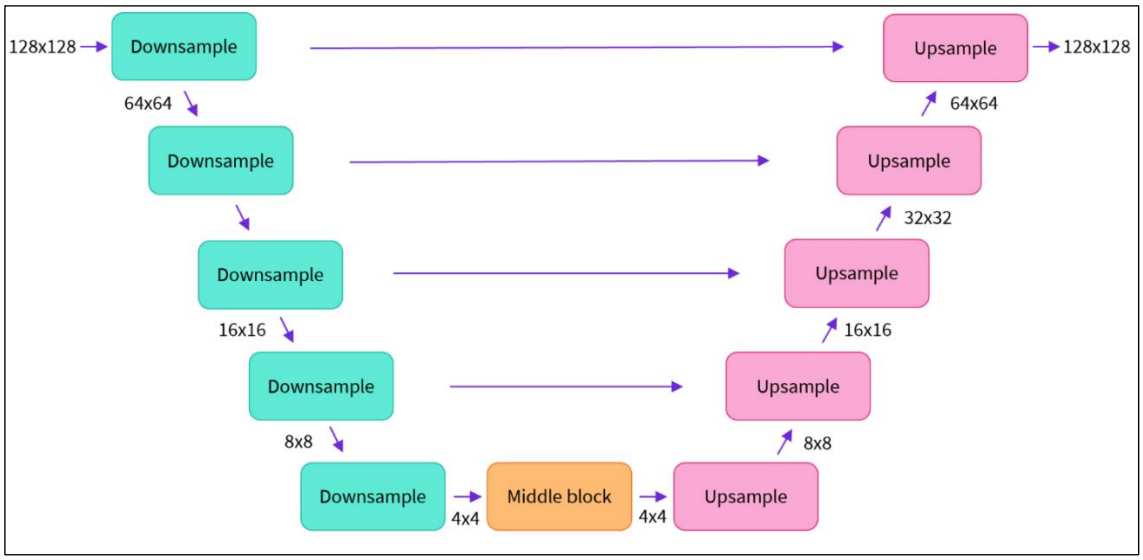

Рисунок 2.4 – Архітектура U-Net [20]

3. Текстовий енкодер (Text Encoder). Для обробки текстових підказок використовується CLIP (Contrastive Language–Image Pre-training) від OpenAI – модель, попередньо навчена на мільйонах пар «текст-зображення» (рис. 2.5). Модель складається з Text Encoder (Transformer) і Image Encoder (ViT).

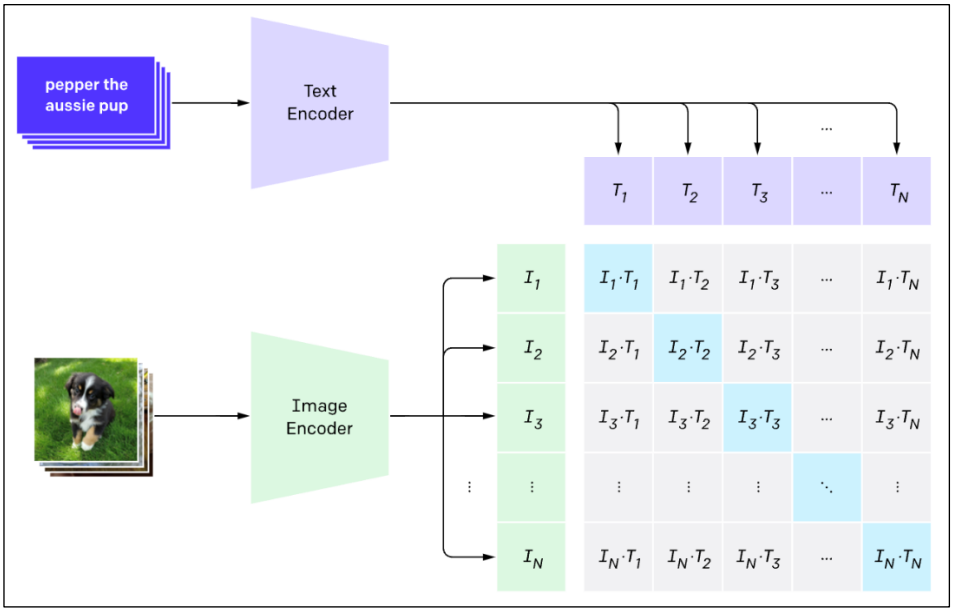

Рисунок 2.5 – Архітектура CLIP [21]

Завдання енкодера – перетворити текстовий опис (prompt) у векторне представлення, яке потім використовується у механізмах Cross-Attention всередині U-Net. Завдяки цьому система може формувати візуальні образи, що семантично відповідають заданому опису, навіть якщо такі поєднання раніше не зустрічалися в даних.

Додатковим компонентом є Scheduler, який контролює, як саме шум додається або видаляється під час процесу дифузії. Scheduler не має параметрів для навчання – він лише визначає траєкторію шуму та кількість кроків денойзінгу. Від його налаштувань залежить швидкість та якість генерації: наприклад, більша кількість кроків покращує деталізацію, але збільшує час обчислення.

Існує кілька варіантів Scheduler, що реалізують різні стратегії шумопригнічення. Серед найпоширеніших – DDIM (Denoising Diffusion Implicit Model), PLMS (Pseudo Linear Multistep) і Euler Ancestral. Вибір конкретного методу впливає на баланс між швидкістю і якістю генерації. Наприклад, DDIM дозволяє зменшити кількість кроків денойзінгу без суттєвої втрати якості, що робить його оптимальним для інтерактивних систем, де важливий час відгуку. Таким чином,

Scheduler виступає своєрідним «режисером» процесу дифузії, задаючи темп і характер еволюції латентного шуму у фінальне зображення.

Процес роботи Stable Diffusion можна описати послідовністю.

1. Початкове зображення стискається в латентне представлення за допомогою енкодера VAE.

2. До латентного представлення ітеративно додається гауссів шум, формуючи зашумлену версію.

3. U-Net разом із CrossAttention блоками і Text Encoder поступово видаляють шум, забезпечуючи відповідність між текстом і результатом.

4. Стиснене латентне представлення декодується в підсумкове зображення за допомогою декодера VAE.

Переваги латентної дифузійної моделі:

– ефективність: завдяки роботі у стисненому просторі суттєво зменшуються вимоги до пам’яті та продуктивності GPU;

– гнучкість: архітектура допускає використання додаткових модулів (наприклад, ControlNet, InstructPix2Pix, LoRA), що розширюють функціональність без повного перенавчання базової моделі;

– модульність: кожен компонент (VAE, U-Net, CLIP) може бути замінений або донавчений окремо, залежно від конкретного завдання;

– якість: навчання на масштабних мультимодальних наборах даних (текст+зображення) забезпечує здатність моделі створювати фотореалістичні або стилізовані зображення відповідно до контексту опису.

Процес навчання Stable Diffusion зазвичай відбувається попередньо (offline) на потужних обчислювальних кластерах і не є частиною основного користувацького циклу. У межах побудови системи, описаної у цьому проєкті, використовується вже навчена та донавчена модель, інтегрована у готову інфраструктуру.

Таким чином, архітектура Stable Diffusion забезпечує ефективне поєднання текстової семантики та візуальної генерації, що дозволяє створювати синтетичні

зображення високої якості навіть у задачах спеціалізованого характеру, таких як генерація медичних або технічних знімків.

2.3 Використані мови програмування та інструменти (Python, PyTorch, Diffusers, Kaggle)

Для реалізації системи генерації синтетичних рентгенівських знімків було використано сучасні інструменти та мови програмування, що забезпечують ефективну розробку, навчання та тестування моделей штучного інтелекту. Основною мовою програмування, на якій побудована система, є Python. Завдяки своїй простоті, гнучкості та величезній екосистемі бібліотек Python сьогодні є провідною мовою у сфері машинного навчання та глибинного навчання. Мова підтримується більшістю сучасних фреймворків (TensorFlow, PyTorch, Keras, Hugging Face, OpenCV тощо), що робить її незамінною для створення та дослідження нейромережевих моделей.

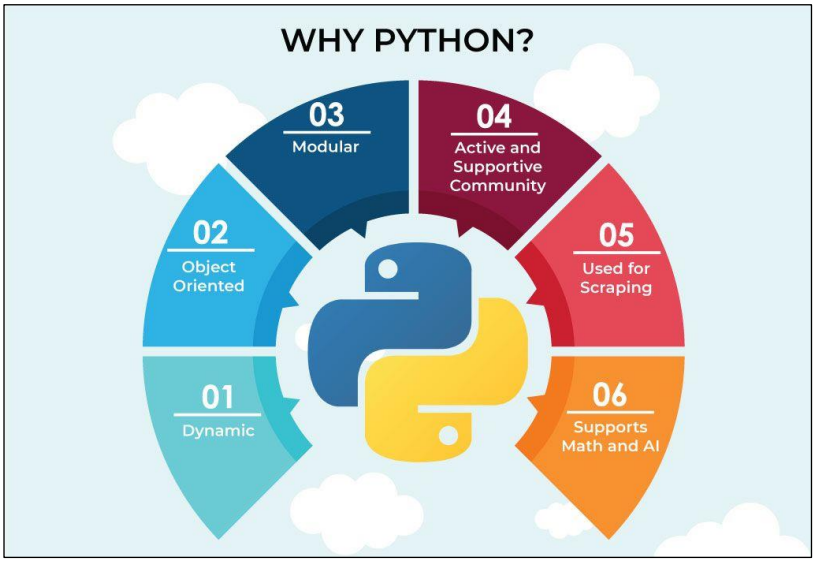

Рисунок 2.6 – Ключові переваги використання Python [22]

У контексті штучного інтелекту Python забезпечує високий рівень абстракції, дозволяючи зосередитися безпосередньо на алгоритмах і даних. Його бібліотеки, такі як NumPy, Pandas, Matplotlib та Seaborn, використовуються для обробки,

аналізу та візуалізації даних, що є важливою частиною підготовки медичних наборів зображень до донавчання. Гнучкість Python також дає змогу швидко експериментувати з різними архітектурами моделей і параметрами навчання без значних витрат часу на реалізацію.

Для реалізації самої генеративної моделі було обрано PyTorch – один із найпопулярніших фреймворків глибинного навчання з відкритим кодом. Його особливістю є динамічна побудова обчислювальних графів, що полегшує налагодження та гнучку зміну архітектури під час експериментів. PyTorch активно використовується у наукових дослідженнях і підтримується спільнотою розробників, що спрощує інтеграцію новітніх архітектур, зокрема дифузійних моделей.

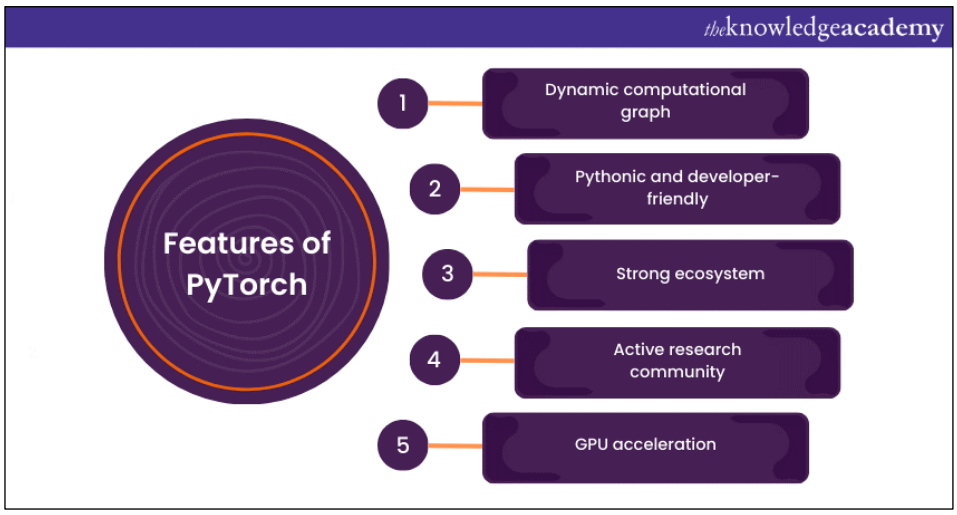

Рисунок 2.7 – Основні переваги PyTorch [23]

На базі PyTorch функціонує бібліотека Hugging Face Diffusers, яка надає зручні інструменти для роботи з дифузійними моделями, такими як Stable Diffusion. Вона дозволяє завантажувати попередньо навчені моделі, виконувати донавчання на власних наборах даних і здійснювати генерацію зображень із мінімальною кількістю коду. Це значно прискорює процес експериментів і забезпечує повторюваність результатів.

Для проведення тренувань і тестування моделі було використано Kaggle – хмарне середовище з підтримкою GPU, яке дозволяє запускати обчислювально інтенсивні завдання без необхідності власного апаратного забезпечення. Kaggle дає змогу працювати з великими наборами даних, зберігати результати та використовувати спільно доступні ресурси спільноти.

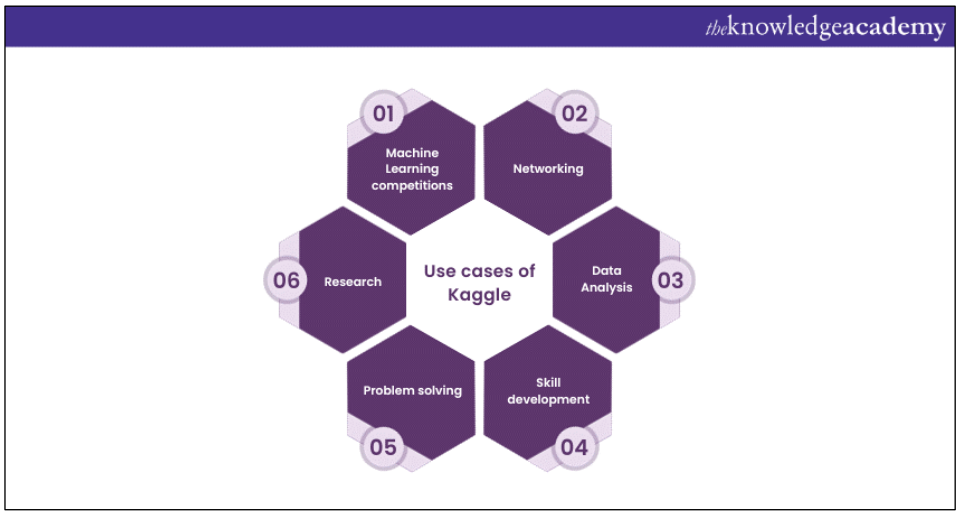

Рисунок 2.8 – Ключові можливості Kaggle [24]

Узагальнений перелік використаних засобів і їхнє призначення наведено в таблиці 2.1.

Таблиця 2.1 – Використані інструменти та їх призначення

Інструмент	**Призначення**
Python	Основна мова програмування, що забезпечує розробку та інтеграцію всіх модулів системи.
NumPy, Pandas, Matplotlib, Seaborn	Попередня обробка, аналіз і візуалізація медичних даних.
PyTorch	Створення, навчання та налагодження глибинних нейронних мереж.
Hugging Face Diffusers	Робота з дифузійними моделями (Stable Diffusion), донавчання та генерація зображень.
Kaggle	Хмарне середовище для обробки великих датасетів і проведення експериментів.

Застосування вищезгаданих інструментів дало змогу створити гнучке, масштабоване та відтворюване середовище для реалізації та тестування системи генерації зображень. Вибір відкритих технологій не лише спростив розробку, а й забезпечив можливість подальшого розширення системи, включно з інтеграцією нових архітектур або оптимізацією під конкретні обчислювальні ресурси.

Висновки до розділу 2

У цьому розділі було розглянуто структуру інтелектуальної системи генерації синтетичних рентгенівських знімків, її основні компоненти, підсистеми та архітектуру моделі, на основі якої реалізується процес генерації. Встановлено, що модульна структура системи забезпечує логічний розподіл функцій між окремими елементами, зокрема: модулем введення текстових описів, підсистемою попередньої обробки даних, ядром генеративної моделі, підсистемою візуалізації та блоком збереження результатів. Такий підхід підвищує гнучкість, масштабованість і дає змогу адаптувати систему для різних сценаріїв використання у медичних цілях.

Було детально проаналізовано архітектуру дифузійної моделі Stable Diffusion, яка становить основу системи. Ця модель поєднує можливості латентних дифузійних процесів і текстово-візуальної трансформації за допомогою попередньо навчених енкодерів (CLIP) та декодерів (VAE). Її використання дозволяє досягати високої якості синтетичних зображень при збереженні змістовної відповідності текстовим описам, що є ключовим для завдань медичної візуалізації.

Також у розділі наведено огляд основних інструментів і мов програмування, які використовуються для реалізації системи. Мова Python виступає центральним інструментом розробки завдяки своїй універсальності, підтримці численних бібліотек для глибокого навчання (зокрема PyTorch та Diffusers) і активній спільноті розробників у сфері штучного інтелекту. Серед інших інструментів

застосовуються Kaggle для проведення обчислювальних експериментів у хмарному середовищі, а також допоміжні бібліотеки для обробки та візуалізації результатів.

Таким чином, у розділі було проаналізовано та описано структуру інтелектуальної системи, архітектуру дифузійної моделі Stable Diffusion як її центрального компонента та розглянуто інструменти, що забезпечують її реалізацію. Проведений аналіз дозволяє краще зрозуміти взаємозв’язок між складовими системи та визначити напрями її подальшого розвитку – зокрема, у контексті донавчання моделі на спеціалізованих медичних наборах даних і вдосконалення якості синтетичних рентгенівських зображень.

3 АНАЛІЗ ТА ПОПЕРЕДНЯ ОБРОБКА ДАНИХ

3.1 Аналіз та опис набору даних Chest X-rays (Indiana University)

Для дослідження використано набір даних «Chest X-rays (Indiana University)», розміщений на платформі Kaggle [25]. Цей набір містить рентгенівські знімки грудної клітки, зібрані в межах клінічного дослідження Indiana University. Він використовується для завдань діагностики легеневих захворювань, класифікації патологій і побудови генеративних моделей у медичній галузі (рис. 3.1).

У датасеті міститься понад 7 тисяч зображень у форматі JPEG, а також супровідні текстові описи, що були сформовані на основі медичних звітів. Кожен запис містить рентгенівський знімок грудної клітки та короткий опис стану пацієнта. Це робить набір даних особливо корисним для завдань типу *text-to-image*.

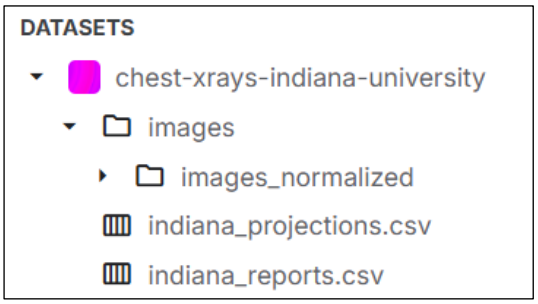

Рисунок 3.1 – Структура набору даних

Для подальшого дослідження було завантажено два основні файли з метаданими:

– indiana_projections.csv – містить інформацію про проєкції рентгенівських знімків;

– indiana_reports.csv – включає текстові медичні звіти, пов’язані із зображеннями.

```
base_path = "/kaggle/input/chest-xrays-indiana-university"
projections_path = os.path.join(base_path, "indiana_projections.csv")
reports_path = os.path.join(base_path, "indiana_reports.csv")
images_path = os.path.join(base_path, "images/images_normalized")
```

```
projections_df = pd.read_csv(projections_path)
reports_df = pd.read_csv(reports_path)

projections_df.info()
reports_df.info()
```

Шлях до цих файлів задається змінними *projections_path* і *reports_path*, після чого дані імпортуються у вигляді датафреймів за допомогою функції *pd.read_csv()*.

```
RangeIndex: 7466 entries, 0 to 7465
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   uid         7466 non-null   int64
 1   filename    7466 non-null   object
 2   projection  7466 non-null   object
dtypes: int64(1), object(2)
memory usage: 175.1+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3851 entries, 0 to 3850
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   uid         3851 non-null   int64
 1   MeSH        3851 non-null   object
 2   Problems    3851 non-null   object
 3   image       3851 non-null   object
 4   indication  3765 non-null   object
 5   comparison  2685 non-null   object
 6   findings    3337 non-null   object
 7   impression  3820 non-null   object
```

Рисунок 3.2 – Загальні характеристики обох наборів даних

За допомогою функції *DataFrame.info()* було отримано загальні характеристики кожного з наборів (рис. 3.2). Таблиця *projections_df* містить 7466 записів і три стовпці:

- uid – унікальний ідентифікатор знімка;
- filename – назва файлу у форматі .dcm.png;
- projection – тип проєкції (Frontal або Lateral).

У свою чергу, таблиця *reports_df* налічує 3851 запис і вісім стовпців, серед яких найважливішими є:

- MeSH – медичні ключові слова відповідно до системи Medical Subject Headings;
- problems – перелік виявлених патологій;
- findings – детальні результати спостереження;
- impression – підсумковий висновок лікаря.

```
display(projections_df.head(5))
display(reports_df.head(5))
```

Для первинного ознайомлення з вмістом наборів даних використано функцію *display()*, яка виводить перші п’ять записів кожного датафрейму. Це дозволяє візуально оцінити відповідність структури даних очікуваному формату та підтвердити наявність зв’язку між таблицями через спільний ідентифікатор *uid* (рис. 3.3).

	uid	filename	projection
0	1	1_IM-0001-4001.dcm.png	Frontal
1	1	1_IM-0001-3001.dcm.png	Lateral
2	2	2_IM-0652-1001.dcm.png	Frontal
3	2	2_IM-0652-2001.dcm.png	Lateral
4	3	3_IM-1384-1001.dcm.png	Frontal

	uid	MeSH	Problems	image	indication	comparison	findings	impression
0	1	normal	normal	Xray Chest PA and Lateral	Positive TB test	None.	The cardiac silhouette and mediastinum size ar...	Normal chest x-XXXX.
1	2	Cardiomegaly/borderline;Pulmonary Artery/enlarged	Cardiomegaly;Pulmonary Artery	Chest, 2 views, frontal and lateral	Preop bariatric surgery.	None.	Borderline cardiomegaly. Midline sternotomy XX...	No acute pulmonary findings.
2	3	normal	normal	Xray Chest PA and Lateral	rib pain after a XXXX, XXXX XXXX steps this XX...	NaN	NaN	No displaced rib fractures, pneumothorax, or p...
3	4	Pulmonary Disease, Chronic Obstructive;Bullous...	Pulmonary Disease, Chronic Obstructive;Bullous...	PA and lateral views of the chest XXXX, XXXX a...	XXXX-year-old XXXX with XXXX.	None available	There are diffuse bilateral interstitial and a...	1. Bullous emphysema and interstitial fibrosis...
4	5	Osteophyte/thoracic vertebrae/multiple/small;T...	Osteophyte;Thickening;Lung	Xray Chest PA and Lateral	Chest and nasal congestion.	NaN	The cardiomediastinal silhouette and pulmonary...	No acute cardiopulmonary abnormality.

Рисунок 3.3 – Демонстрація вмісту обох наборів даних

Для перевірки цілісності даних та узгодженості між таблицями проведено аналіз кількості унікальних ідентифікаторів *uid* у кожному наборі.

```
print("Унікальних UID у reports:", reports_df['uid'].nunique())
print("Унікальних UID у projections:", projections_df['uid'].nunique())

common_uid=set(reports_df['uid']).intersection(set(projections_df['uid']))
print("Спільних UID:", len(common_uid))
```

За допомогою функції *nunique()* визначено, що як у файлі *reports_df*, так і в *projections_df* міститься по 3851 унікальному *uid*, що свідчить про відсутність дублювання записів на рівні пацієнтів або досліджень.

Далі, за допомогою операції перетину множин (*set.intersection()*), встановлено, що кількість спільних ідентифікаторів становить 3851, тобто кожен запис у звітах має відповідні рентгенівські знімки у таблиці з проєкціями (рис. 3.4).

```
Унікальних UID у reports: 3851
Унікальних UID у projections: 3851
Спільних UID: 3851
```

Рисунок 3.4 – Результат підрахунку спільних ідентифікаторів

Це підтверджує структурну цілісність даних і дозволяє коректно об’єднати обидві таблиці.

```
merged_df = pd.merge(projections_df, reports_df, on='uid', how='inner')
print("Кількість записів після об’єднання:", len(merged_df))
```

Злиття було виконано за допомогою функції *pd.merge()* із параметром how='inner', що забезпечує поєднання лише тих записів, які мають спільні uid. У результаті отримано таблицю *merged_df*, яка містить 7466 записів – це пояснюється тим, що більшість пацієнтів мають дві проєкції (Frontal та Lateral), що дублює записи у злитому наборі.

Отже, на цьому етапі сформовано об’єднаний датафрейм, який поєднує медичні зображення та відповідні текстові описи. Це створює основу для подальшої аналітики та попередньої обробки даних перед навчанням моделі.

На етапі перевірки відповідності між метаданими та фактичними зображеннями було виконано звірку кількості елементів у папці із зображеннями та у зведеному CSV-файлі. Для цього було сформовано множини імен файлів зображень і тих, що використовуються в таблиці, після чого визначено відсутні та зайві елементи.

```
all_images = set(os.listdir(images_path))
```

```
      used_images = set(merged_df["filename"])
      missing_images = used_images - all_images
      extra_images = all_images - used_images

      print("Зображень у папці:", len(all_images))
      print("Використаних у CSV:", len(used_images))
      print("Відсутніх зображень:", len(missing_images))
      print("Зайвих зображень:", len(extra_images))

      projections_frontal = merged_df[merged_df['projection'].str.lower() ==
'frontal'].copy()
      print("Фронтальних знімків:", len(projections_frontal))
```

Результати показали (рис. 3.5), що всі зображення, згадані у CSV, присутні у вихідній директорії (відсутніх файлів нуль), а також виявлено чотири зайвих файли, не пов’язаних із жодним записом у таблиці. Загальна кількість зображень у папці становить 7470, а в CSV – 7466.

```
Зображень у папці: 7470
Використаних у CSV: 7466
Відсутніх зображень: 0
Зайвих зображень: 4
Фронтальних знімків: 3818
```

Рисунок 3.5 – Результат перевірки зображень в наборі даних

Додатково було перевірено розподіл за типом проєкції; у результаті встановлено, що до класу фронтальних знімків належать 3818 зображень (рис. 3.5).

Після перевірки цілісності даних та аналізу структури проєкцій постало завдання відібрати лише ті знімки, які відповідають умовам подальшої роботи моделі. У межах дослідження буде використовуватися тільки фронтальна проєкція грудної клітки (Frontal/PA), оскільки саме вона є стандартом у більшості клінічних сценаріїв і найпоширенішою у відкритих медичних датасетах.

```
      merged_frontal_df = merged_df[merged_df['projection'].str.lower() ==
'frontal'].copy()
      print("Залишилось фронтальних знімків:", len(merged_frontal_df))
```

Зі зведеної таблиці було відібрано всі записи з позначкою *frontal*, у результаті чого отримано 3818 знімків. Після цього проведено аналіз на дублікати – деякі пацієнти мали по два фронтальні знімки, що не відповідає вимогам моделі, яка

повинна працювати з одним зображенням на пацієнта для уникнення перекосу в розподілі даних. Було виявлено 122 пацієнти з дубльованими фронтальними проєкціями, які були виключені з подальшої вибірки.

```
      duplicates = merged_frontal_df['uid'].value_counts()
      multiple_frontal = duplicates[duplicates > 1]
      print("Пацієнтів з кількома фронтальними знімками:", len(multiple_frontal))

      unique_frontal_df =
merged_frontal_df[~merged_frontal_df['uid'].isin(multiple_frontal.index)]
      print("Кількість записів після очищення:", len(unique_frontal_df))
```

Після очищення фінальний набір даних для подальшої обробки та генерації синтетичних рентгенівських знімків містить 3567 унікальних фронтальних зображень (рис. 3.6).

```
Залишилось фронтальних знімків: 3818
Пацієнтів з кількома фронтальними знімками: 122
Кількість записів після очищення: 3567
```

Рисунок 3.6 – Результат перевірки декількох знімків для одного запису

Для додаткової перевірки коректності сформованої вибірки було виконано аналіз одного випадкового запису з очищеного набору даних. Така операція дозволяє впевнитися, що:

- усі стовпці коректно об’єднані після злиття таблиць;
- метадані (MeSH, Problems, Findings, Impression тощо) відповідають конкретному рентгенівському знімку;
- файли зображень доступні у файловій структурі та їх можна завантажити без помилок.

```
      sample_uid = unique_frontal_df["uid"].sample(1).values[0]
      sample_row = unique_frontal_df[unique_frontal_df["uid"] ==
sample_uid].iloc[0]

      W = 70

      print("\n=== Інформація по пацієнту ===")
      print(f"UID: {sample_row.uid}")
      print(textwrap.fill(f"MeSH: {sample_row.MeSH}", width=W))
```

```
    print(textwrap.fill(f"Problems: {sample_row.Problems}", width=W))
    print(textwrap.fill(f"Indication: {sample_row.indication}", width=W))
    print(textwrap.fill(f"Findings: {sample_row.findings}", width=W))
    print(textwrap.fill(f"Impression: {sample_row.impression}", width=W))
    print(f"Файл: {sample_row.filename}")
```

Було випадковим чином обрано один *uid*, після чого програмно виведено повну текстову інформацію щодо стану пацієнта та відповідне зображення грудної клітки. Така вибіркова перевірка є стандартною практикою під час роботи з медичними датасетами, оскільки дозволяє наочно оцінити якість розмітки, структуру даних та переконатися, що зображення відповідає описаним у звіті діагностичним висновкам.

У результаті перевірки встановлено, що система коректно зчитує зображення, а текстова медична інформація (MeSH-теги, показані проблеми, індикація, опис знаходжень та фінальна інтерпретація) узгоджується з рентгенівським знімком (рис. 3.7).

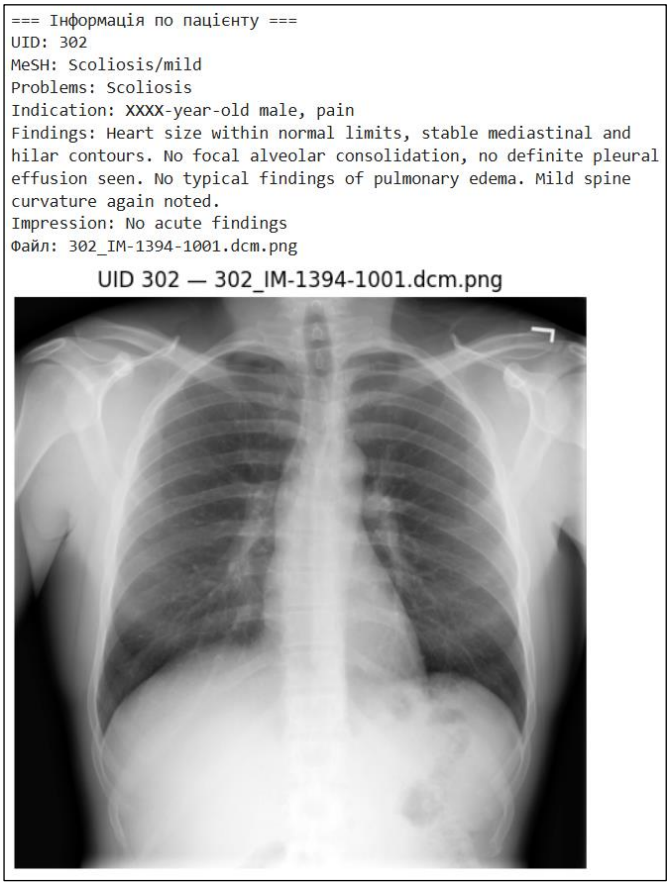

Рисунок 3.7 – Результат перевірки узгодженості зображення з описом

Подальший аналіз був спрямований на дослідження змістовної структури описів у стовпці MeSH, який містить ключові ознаки, виявлені на рентгенівських знімках. Ці мітки мають чітку ієрархічну структуру: окрема патологія або ознака подається як перший елемент, після чого через символ «/» перелічуються різноманітні уточнення – ступінь вираженості, локалізація або додаткові характеристики. Якщо в одному записі присутні кілька незалежних патологій, вони розділяються символом «;».

Саме тому першою важливою задачею стало проаналізувати базові категорії ознак, тобто підрахувати частоти появи основних MeSH-термінів без їх уточнень. За допомогою поетапного розділення рядків були виокремлені перші елементи кожної ознаки, після чого підраховано їхню поширеність у датасеті.

```
      stats =
unique_frontal_df['MeSH'].str.split(';').explode().str.split('/').str[0].str.strip
().value_counts()

      print(stats.to_string())
```

Такий підхід дозволяє:

- оцінити розподіл патологій у наборі даних;
- визначити найчастіші діагностичні шаблони;
- зрозуміти, наскільки різноманітні випадки представлені у вибірці;
- отримати попереднє уявлення про складність майбутнього генеративного моделювання.

Попередній аналіз частот MeSH-міток дав загальне уявлення про розподіл ключових ознак у наборі даних (Рис. 3.8). Проте варто зауважити, що структура цих міток є досить неоднорідною. Не завжди першим елементом у записі виступає власне патологія – у багатьох випадках це може бути анатомічний орган (Lung, Aorta, Diaphragm тощо), після чого лише далі вказуються конкретні зміни, ступені вираженості або локалізація. Через це автоматичне групування лише за першим елементом не завжди дає повну та однозначну картину змісту знімка.

MeSH	
normal	1300
Lung	511
Opacity	462
Cardiomegaly	307
Calcinosis	306
Pulmonary Atelectasis	296
Calcified Granuloma	260
Thoracic Vertebrae	238
Cicatrix	179
Spine	154
Markings	154
Aorta	150
Pleural Effusion	143
Diaphragm	128
Density	119
Atherosclerosis	117
Deformity	110
Nodule	109
Airspace Disease	108
Catheters, Indwelling	108
Scoliosis	105
Granulomatous Disease	100
Surgical Instruments	91
Fractures, Bone	84
No Indexing	83
Aorta, Thoracic	81
Costophrenic Angle	72
Technical Quality of Image Unsatisfactory	71

Рисунок 3.8 – Підрахунок частот найпоширеніших ознак

Разом із тим, під час аналізу розподілу MeSH-категорій були виявлені окремі значення, які не характеризують медичний стан пацієнта, а натомість вказують на неякісні або непридатні для моделювання знімки. Зокрема, були знайдені мітки, що прямо описують технічні проблеми або невідповідність дослідження поставленим задачам (*Technical Quality of Image Unsatisfactory*, *Contrast Media*, *Abdomen*).

Це стало підставою для цілеспрямованого пошуку таких записів у стовпці MeSH, а також у текстових описах *Findings* та *Impression*, де інколи згадуються технічні недоліки, такі як *limited exam*, *rotated examination* або *motion artifact*. Для цього були сформовані два окремі списки ключових слів: один – для MeSH-тегів, інший – для аналізу текстових полів.

```
# Слова для пошуку в MeSH
bad_mesh_phrases = [
    "Technical Quality of Image Unsatisfactory",
    "Contrast Media",
    "Abdomen"
]
```

```
    # Слова для пошуку в описі (Findings + Impression)
    bad_text_keywords = [
        "limited exam",
        "rotated examination",
        "rotated",
        "motion artifact"
    ]
```

Результати перевірки показали (рис. 3.9), що:

- 75 записів містять небажані категорії в MeSH;
- 22 записи містять ключові слова технічних дефектів у текстових описах;
- загалом 76 знімків (≈2,1% вибірки) підлягають видаленню як такі, що не відповідають вимогам до якості або змісту.

```
=== НЕЯКІСНІ ЗНІМКИ ===
1. За тегами MeSH (Technical/Contrast/Abdomen): 75
2. За текстом опису (Limited/Rotated/Motion): 22
--------------------------------------------------
ВСЬОГО записів до видалення: 76 (з 3567)
```

Рисунок 3.9 – Виявлена кількість зображень з незадовільною якістю

Це очищення є важливим підготовчим кроком, оскільки наявність знімків поганої якості або таких, що аналізують інші ділянки тіла, може негативно впливати на стабільність тренування дифузійної моделі та призводити до небажаних артефактів у процесі генерації синтетичних рентгенівських зображень.

Після видалення технічно несправних та нерелевантних записів очищений датасет налічує 3491 спостереження замість початкових 3567. Враховуючи подальшу мету – підготовку пари «медичний текст → рентгенівське зображення» для донавчання дифузійної моделі – було вирішено звузити набір атрибутів до тих, що безпосередньо використовуватимуться у процесі навчання.

```
    columns_to_keep = ['uid', 'filename', 'MeSH', 'findings', 'impression']

    final_dataset = clean_df.loc[:, [col for col in columns_to_keep if col in
clean_df.columns]].copy()
```

У вихідних даних доступні різні текстові поля, однак їхнє призначення та структура відрізняються:

Findings: містить найбільш детальний опис виявлених аномалій, локалізацій, морфологічних ознак та інших клінічно значущих деталей. Це основне поле, яке найкраще підходить як текст для генеративної моделі.

Impression: є узагальненим висновком радіолога, який часто дублює ключові елементи *Findings*, але може містити додаткову інтерпретацію. У випадках, коли *Findings* відсутній або надто короткий, саме *Impression* може братися як основне текстове поле.

MeSH: містить структуровані медичні категорії («Main Heading» та уточнення), які дають стислий формалізований опис патологій. Хоча MeSH-теги рідко використовуються безпосередньо як вхід для генеративної моделі, вони є цінним джерелом додаткової інформації. У ситуаціях, коли описів *Findings* або *Impression* недостатньо, MeSH може бути використаний для формування компактного текстового доповнення (наприклад, як «словник ключових ознак» для кожного прикладу).

Після попереднього аналізу стало очевидно, що частина MeSH-тегів, які слугують короткими формалізованими описами патологій, є надто узагальненими. Наприклад, категорії *Lung*, *Diaphragm*, *Spine* або *Aorta* можуть позначати значно ширший спектр патологій, які стають зрозумілими лише після розбору їхніх уточнень.

Для систематизації цієї інформації було розроблено допоміжну програму, яка:

а) розбиває MeSH-записи на ієрархічні рівні:
   - основна категорія (Main Heading);
   - локалізація;
   - тип патології;
   - ступінь важкості тощо;

б) фільтрує записи за заданою категорією;

в) підраховує частоти зустрічання уточнень на кожному рівні;

г) формує зручний для перегляду деревоподібний список, що дозволяє відкривати структуру MeSH-тегів у глибину.

Приклад аналізу структури MeSH для категорії «Diaphragm»:

```
explore_mesh_structure(final_dataset, "Diaphragm")
```

```
Пошук терміну: 'diaphragm' | Знайдено записів: 128

right                    → 47
    └─ elevated            → 46
          └─ mild           → 12
          └─ chronic        → 1
    └─ posterior           → 1
          └─ elevated       → 1

bilateral                → 31
    └─ flattened           → 27
          └─ mild           → 1
    └─ elevated            → 2
    └─ obscured            → 1
    └─ posterior           → 1
          └─ flattened      → 1

left                     → 30
    └─ elevated            → 26
          └─ mild           → 5
          └─ chronic        → 1
    └─ obscured            → 3
    └─ flattened           → 1

flattened                → 16

posterior                → 4
    └─ flattened           → 4
          └─ mild           → 1
```

Рисунок 3.10 – Аналіз структури MeSH для категорії «Diaphragm»

Аналіз показав (рис. 3.10), що найбільш поширеними патологічними змінами діафрагми є elevated та flattened, тоді як інші уточнення (mild, chronic, posterior) зустрічаються значно рідше.

На основі отриманих структур і частот зустрічання було сформовано компактний словник анотацій, який дозволяє:

- уніфікувати назви категорій для подальшої візуалізації;
- спростити інтерпретацію статистики MeSH;
- використовувати короткі коментарі у графіках або таблицях.

```
category_annotations = {
    "Lung": "(hypoinflation/hyperdistention)",
    "Thoracic Vertebrae": "(degenerative)",
    "Spine": "(degenerative)",
    "Aorta": "(tortuous)",
    "Diaphragm": "(elevated/flattened)",
    "Aorta, Thoracic": "(tortuous)",
```

```
        "Costophrenic Angle": "(blunted)",
        "Cardiac Shadow": "(enlarged)",
        "No Indexing": None
    }
```

У процесі аналізу MeSH-термінів було виявлено спеціальну категорію «*No Indexing*», яка позначає записи без призначених індексів MeSH. Ця категорія не несе корисного змістовного навантаження, оскільки не відображає жодної рентгенологічної ознаки.

Тому було прийнято рішення виключити категорію *"No Indexing"* зі статистичних розрахунків і частотного аналізу. Але не видаляти відповідні записи з датасету, оскільки вони все одно можуть містити текстові описи у полях *findings* або *impression*. Таким чином зберігається інформаційна повнота датасету, але статистика MeSH відображає лише реальні рентгенологічні категорії.

Для виявлення найпоширеніших рентгенологічних ознак було виконано аналіз частот згадувань MeSH-категорій у всіх записах датасету. Спочатку з кожного значення MeSH виділявся перший елемент кожної рубрики (до символу «/»), що відповідає основній категорії терміну. Після цього всі категорії були зібрані та підраховані з урахуванням виключення «*No Indexing*».

Задля наочності було встановлено поріг 31 згадування:

- категорії з частотою ≥31 формують список основних;
- усі категорії, що трапляються рідше, об’єднуються в групу «*Others*».

Такий підхід дозволяє уникнути надмірної фрагментації діаграми та водночас зберігає інформацію про рідкісні ознаки.

Було побудовано горизонтальну діаграму (рис. 3.11), яка відображає 40 найчастіше зустрічуваних MeSH-категорій. До діаграми додатково включено інформацію про сумарну кількість елементів, що увійшли до групи «*Others*».

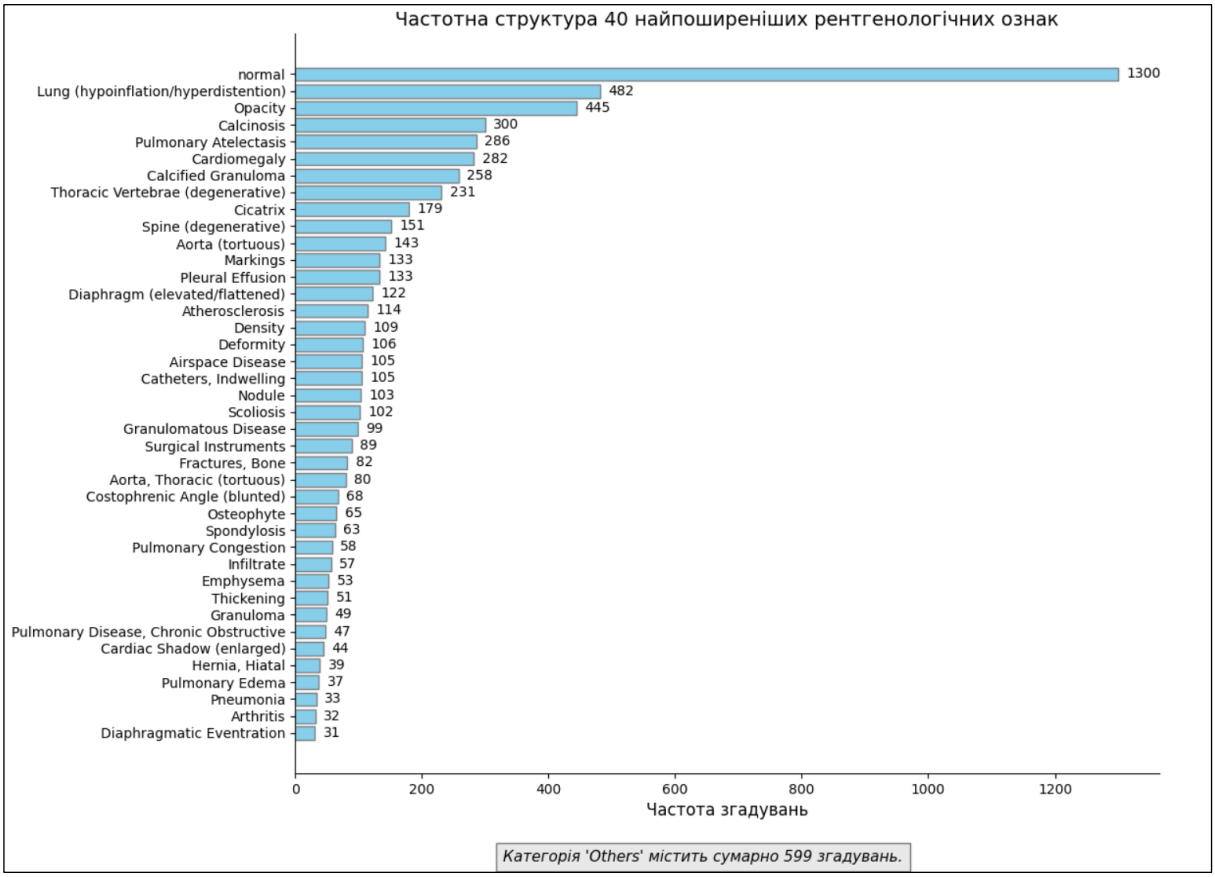

Рисунок 3.11 – Частотна структура 40 найпоширеніших рентгенологічних ознак

Частотна структура MeSH-позначень демонструє суттєвий дисбаланс між нормальними та патологічними випадками. Найпоширеніша категорія «*normal*» зустрічається у понад тисячі записів, що значно перевищує частоту будь-яких патологій. Така нерівномірність є типовою для клінічних наборів знімків, проте вона може негативно позначитися на навчанні моделі, оскільки призводить до переорієнтації моделі на генерацію «нормальних» зображень.

Після аналізу загальної частотної структури патологій було виконано більш детальне дослідження поширеності рентгенологічних ознак на рівні окремого знімка. На відміну від підрахунку частоти окремих MeSH-позначень, у даному випадку враховувалися комбінації ознак, які реально зустрічаються в межах одного рентгенівського зображення. Такий підхід дозволяє точніше відобразити клінічну картину, оскільки на практиці один знімок часто містить кілька супутніх змін, що формують складний і неоднорідний візуальний шаблон.

Рисунок 3.12 – Частотна структура 18 найпоширеніших ознак на одному знімку

Отримані результати підтверджують наявність значного домінування нормальних знімків, однак водночас демонструють широку варіативність патологічних поєднань (рис. 3.12). Серед найбільш поширених одиничних ознак після категорії «*normal*» виділяються кальцифіковані гранульоми, порушення вентиляції легень (гіпоінфляція або гіпердистензія), дегенеративні зміни хребта та різні форми кальцинозу. Окрему групу становлять менш численні, але клінічно важливі стани, зокрема легеневі ущільнення (opacity), кардіомегалія, вузлики та наслідки хірургічних втручань. Наявність значної кількості різнорідних та малочастотних комбінацій, об’єднаних у категорію з низькою частотою, свідчить про високу структурну складність датасету та додатково ускладнює задачу навчання генеративної моделі.

Збір статистики на рівні одного знімка створює основу для подальшого цілеспрямованого формування тренувальних підвибірок, зокрема для відбору патологій із низькою представленістю або для фокусованого донавчання моделі на окремих рентгенологічних ознаках. Такий підхід є особливо важливим у контексті

генерації синтетичних медичних зображень, де точність відтворення локальних і малоконтрастних змін напряму залежить від збалансованості та структури навчальних даних.

Після завершення всіх етапів очищення та структурування даних планується збалансувати датасет шляхом зменшення кількості нормальних випадків до приблизно 500 знімків. Це дозволить уникнути статистичного перекосу та покращити якість подальшого донавчання моделі за патологічними ознаками.

Описана вище первинна аналітика дозволила встановити якість даних, визначити структуру MeSH-термінів та окреслити необхідні кроки щодо балансування класів.

3.2 Підготовка та очищення даних

Після завершення первинного аналізу та структуризації датасету Chest X-rays (Indiana University) доцільним етапом стає перехід до поглибленої роботи з текстовими даними. Саме текстові описи – поля findings, impression та, за потреби, MeSH як додаткове контекстуальне джерело – формують основу для подальшого донавчання дифузійної моделі. Однак у вихідному вигляді ці поля містять суттєву варіативність формулювань, шаблонні фрази, неінформативні конструкції, а також анонімізовані сегменти, позначені маркером «XXXX».

Оскільки анонімізація була здійснена автоматичними методами, виникає необхідність перевірити, чи не приховують замінені фрагменти персональних даних або внутрішньолікарняних маркерів (таких як ідентифікатори пристроїв, номерні позначення чи службова документація). Для цього виконано спеціалізований пошук контекстних шаблонів – аналіз найближчого оточення кожного токена «XXXX», який дозволяє виявити, які саме слова типово передують або слідують за замаскованими елементами.

Задля цього проведено обробку об’єднаного тексту полів findings та impression з подальшим виділенням двох змістовних слів ліворуч та праворуч від кожного випадку заміненої інформації. Під час аналізу було враховано список

стоп-слів (артиклі, сполучники, службові дієслова), що дозволило зосередитися лише на значущих мовних одиницях і мінімізувати шум.

```
NOISE_WORDS = {
    'of', 'the', 'is', 'are', 'and', 'in', 'to', 'with', 'a', 'an',
    'or', 'for', 'at', 'on', 'be', 'as', 'that', 'which', 'by',
    'have', 'has', 'been', 'it', 'this', 'there', 'from',
    'dr', 'md', 'please', 'thanks'
}
```

Отриманий частотний перелік контекстних фрагментів демонструє (рис. 3.13), що в переважній більшості випадків маскування стосується анатомічних структур, радіологічних ознак або стандартних шаблонних описів, притаманних медичним звітам. У найпоширеніших шаблонах анонімізовані сегменти зустрічаються поруч з термінами «*heart*», «*pulmonary*», «*mediastinum*», «*aorta*», «*vasculature*», тощо. Це свідчить про те, що маскування не містить персональних або чутливих даних – воно, ймовірно, пов’язане з особливостями вихідного корпусу, де певні частини тексту були некоректно розпізнані або автоматично замінені.

```
=== ВСЬОГО ЗАЛИШИЛОСЯ СЛІВ XXXX: 2945 ===

Count | Deep Context (2 Words Left ... 2 Words Right)
----------------------------------------------------------------
60    | heart and lungs have xxxx xxxx in the interval. both
45    | lungs are clear. xxxx are normal. no
35    | heart, pulmonary xxxx and mediastinum are within
23    | pleural fluid. the xxxx are grossly normal
18    | effusion or pneumothorax. the xxxx are intact. no
16    | comparison chest x-xxxx. well-expanded and clear
16    | contour, pulmonary xxxx and vasculature, central
15    | heart and pulmonary xxxx appear normal
12    | heart and pulmonary xxxx are normal. the pleural
11    | normal limits. xxxx and soft tissues
11    | contour. aortic xxxx appear unremarkable
11    | normal limits. xxxx are unremarkable. no
10    | pleural effusions. the xxxx are intact. no
10    | edema. no xxxx of pleural effusions
10    | consolidation. there are no xxxx of a large pleural
```

Рисунок 3.13 – Результат аналізу шаблонів анонімізації

Для цього на наступному етапі був застосований перший набір правил деанонімізації, спрямованих на реконструкцію тих фрагментів, що достовірно відповідають типовим анатомічним описам. Зокрема, на основі виявлених шаблонів були створені регулярні вирази, які дозволяють повернути коректні терміни там, де контекст очевидно вказує на анатомічні структури (наприклад, замість «XXXX are normal» → «The osseous structures are normal», або «pulmonary XXXX» → «pulmonary parenchyma/vasculature»). При цьому реконструкція ніколи не застосовувалася до можливих персональних чи адміністративних фрагментів – останні залишалися маркованими для подальшого аналізу.

Після застосування першої хвилі реконструкції вдалося значно скоротити кількість маркерів «XXXX» (рис. 3.14). Як засвідчили результати повторного підрахунку, їх число зменшилося з 2945 до 2064, тобто було відновлено 881 фрагмент.

Цей етап дозволив відновити всі найбільш очевидні шаблонні анатомічні конструкції, що повторювалися десятки разів. Хоча підхід продемонстрував високу ефективність для частотних шаблонів, подальший аналіз показав, що він практично вичерпав свій потенціал: залишилися здебільшого низькочастотні, структурно різнорідні та часто шумові конструкції.

```
=== ВСЬОГО ЗАЛИШИЛОСЯ СЛІВ XXXX: 2064 ===

Count | Deep Context (2 Words Left ... 2 Words Right)
--------------------------------------------------------------------------------
5     | cardiac and mediastinal xxxx appear normal
5     | effusion. normal xxxx. negative for acute
4     | abnormality. comparison xxxx, xxxx. well-expanded and clear
4     | lung volumes are xxxx. xxxx opacities are present
4     | contour. no xxxx acute abnormalities
4     | apparent cardiomegaly xxxx at xxxx partially accentuated
3     | were discussed xxxx. xxxx by dr. xxxx xxxx telephone at xxxx p
3     | vasculature are normal. xxxx change. hypoinflation
3     | were discussed xxxx. xxxx in the xxxx department at xxxx a. m
3     | calcified lymph xxxx and granuloma are noted
3     | critical results at xxxx on xxxx, xxxx by telephone and acknowledged
3     | monitor leads xxxx the thorax. the cardiomediastinal
3     | osseous injury xxxx demonstrated. no
3     | mediastinal lymph xxxx. the skeletal structures
3     | about this examination please xxxx. xxxx, xxxx certified radiologist
```

Рисунок 3.14 – Результат аналізу шаблонів після застосування деанонімізації

Оскільки повторювані шаблони вже були повністю вилучені в першій фазі, подальша реконструкція вимагала переходу до більш гнучкого підходу, здатного аналізувати навіть короткі або асиметричні контексти.

Зокрема, було очевидно, що у багатьох фразах одне з сусідніх слів є пунктуацією, або службовим словом, яке не несе медичного змісту, або фрагмент стоїть на початку/кінці речення. Це означало, що попередня стратегія більше не підходить. Багато шаблонів залишалися поза зоною охоплення.

Для подолання цих обмежень було застосовано мікроконтекстний аналіз, заснований на пошуку одного інформативного слова перед анонімним сегментом, або одного інформативного слова після нього, якщо лівий контекст виявлявся пунктуаційним, службовим або відсутнім.

Другий рівень пошуку дозволив виділити значну кількість нових шаблонів, які не були доступні при першій спробі (рис. 3.15).

```
=== ВСЬОГО ЗАЛИШИЛОСЯ СЛІВ XXXX: 2064 ===

Count | Context
-------------------------------------------
43    | lymph xxxx.
26    | , xxxx.
25    | no xxxx focal
24    | . xxxx opacities
23    | comparison xxxx,
19    | . xxxx sternotomy
19    | . xxxx xxxx and lateral
18    | . xxxx right
14    | , xxxx atelectasis
13    | no xxxx acute
13    | no xxxx infiltrates
13    | normal xxxx.
12    | lung xxxx.
12    | . xxxx change
12    | , xxxx subsegmental
```

Рисунок 3.15 – Результат мікроконтекстного аналізу шаблонів анонімізації

Ці шаблони демонструють все ще чітко анатомічні конструкції, які піддаються реконструкції з високою достовірністю. Мікроконтекстний аналіз став ключовим доповненням до першої хвилі реконструкції, оскільки дозволив

ідентифікувати залишкові анатомічні фрази, приховані за пунктуацією або службовими словами, відкрив додаткові шаблони, особливо у випадках, де контекст був дуже коротким, забезпечив відбір лише медично значущих конструкцій, оминаючи адміністративні фрагменти.

Після застосування другого набору правил частина таких шаблонів була успішно реконструйована, здебільшого у випадках, коли контекст однозначно вказував на морфологічні структури або типові для рентгенології терміни. Загальна кількість маскованих елементів зменшилася до 1115, що майже вдвічі менше, ніж після попереднього етапу (рис. 3.16). Водночас значна частина тих, що залишилися, містила шумові вставки – службові коментарі лікарів, часові позначення, згадки про телефонні повідомлення, інструкції або порівняння з попередніми обстеженнями. Подібні конструкції не підлягають реконструкції за контекстом, оскільки вони не містять значущих для моделі даних і підлягатимуть повному видаленню на наступному етапі очищення тексту.

```
=== ВСЬОГО ЗАЛИШИЛОСЯ СЛІВ XXXX: 1115 ===

Count | Context
-------------------------------------------
26    | , xxxx.
23    | comparison xxxx,
19    | . xxxx xxxx and lateral
5     | . xxxx,
5     | , xxxx related
4     | injury xxxx demonstrated
4     | remain in xxxx.
4     | , xxxx seen
4     | . xxxx at xxxx.
4     | cardiomegaly xxxx at xxxx partially
4     | since xxxx.
3     | discussed xxxx.
3     | left xxxx,
3     | . xxxx interstitial
3     | , xxxx chronic
```

Рисунок 3.16 – Результат аналізу шаблонів після застосування деанонімізації

Таким чином, друга хвиля деанонімізації майже повністю вичерпала потенціал автоматичного відновлення анатомічних та радіологічних термінів,

залишивши лише ті фрагменти, які не є корисними в рамках основного завдання й будуть вилучені в процесі фінального нівелювання шумових конструкцій.

Після завершення етапів деанонімізації та відновлення значущих клінічних формулювань стало зрозуміло, що у текстах залишається велика кількість шумових фрагментів, які не несуть користі для донавчання моделі та навіть можуть порушити семантичну узгодженість описів. До них належали службові конструкції, пов’язані з порівнянням поточного знімка з минулими дослідженнями, журналами комунікації, технічними уточненнями, а також короткі рекомендаційні фрази, які не описують анатомічних змін. Тому для очищення текстів було застосовано алгоритм, який працював із ключовими словами та регулярними виразами, що давало змогу знайти як окремі речення, так і цілі групи фраз, характерних для технічних та бюрократичних вставок (рис. 3.17).

```
================================================================================================
ТОП-3 Найпоширеніші шаблони з кожної категорії
================================================================================================
```

	Category	Pattern_Start	Count	Sample_Sentence
0	COMMUNICATION (Log)	critical result notification documented	6	Critical result notification documented through Primordial.
1	COMMUNICATION (Log)	result notification xxxx primordial	2	Result notification XXXX Primordial.
2	COMMUNICATION (Log)	xxxx xxxx i discussed	2	XXXX XXXX I discussed the findings and further workup suggestions by telephone approximately XXXX hours XXXX, XXXX.
3	COMPARISON (Technical)	no comparison chest x	22	No comparison chest x-ray.
4	COMPARISON (Technical)	comparison xxxx xxxx	17	Comparison XXXX, XXXX.
5	COMPARISON (Technical)	recommend comparison with prior	3	Recommend comparison with prior images to document stability.
6	RECOMMENDATION (Action)	please note that fractures	4	Please note that fractures may not be demonstrated and consider additional imaging as clinically warranted.
7	RECOMMENDATION (Action)	abnormal pulmonary opacities most	3	Abnormal pulmonary opacities most suggestive of pulmonary edema, differential diagnosis includes infectious and inflammatory processes.
8	RECOMMENDATION (Action)	hyperexpanded lungs suggestive of	3	Hyperexpanded lungs, suggestive of obstructive lung disease.

Рисунок 3.17 – Найпоширеніші шаблони шумових конструкцій

На першому кроці кожен опис розбивався на окремі речення, і для кожного з них перевірялося, чи містить воно шаблони, притаманні трьом основним категоріям шуму: технічному порівнянню із попередніми знімками, службовим логам комунікацій між лікарем та іншими фахівцями, а також рекомендаційним формулюванням, які стосуються майбутніх дій або додаткових методів обстеження. Саме ці типи інформації зазвичай не є безпосередньою характеристикою рентгенологічного стану, а тому не повинні потрапляти у навчальні дані. На основі такого аналізу будувалася проміжна статистика, що

давала змогу оцінити, які саме шаблони трапляються найчастіше, й адаптувати правила очищення, щоб максимально звузити вибірку нерелевантних фрагментів.

На певному етапі очищення категорія *communication* повністю зникла зі статистики повторюваних шаблонів, що свідчило про ефективне відсікання бюрократичних записів, які не мають жодного відношення до структури або стану органів грудної клітки. Категорія рекомендацій також значною мірою очистилася від шуму, хоч деякі речення продовжували з'являтися у вибірці, але вже містили змістовні згадки про патологічні стани, як-от «suggestive of obstructive» або «suggest atelectasis» (рис. 3.18).

```
================================================================================================
ТОП-3 Найпоширеніші шаблони з кожної категорії
================================================================================================
```

	Category	Pattern_Start	Count	Sample_Sentence
0	COMPARISON (Technical)	no comparisons	1	No comparisons.
1	COMPARISON (Technical)	no comparisons are available	1	No comparisons are available to evaluate stability.
2	COMPARISON (Technical)	no interval change in	1	No interval change in the appearance of the opacities in the bilateral lower lobes.
3	RECOMMENDATION (Action)	abnormal pulmonary opacities most	3	Abnormal pulmonary opacities most suggestive of pulmonary edema, differential diagnosis includes infectious and inflammatory processes.
4	RECOMMENDATION (Action)	hyperexpanded lungs suggestive of	3	Hyperexpanded lungs, suggestive of obstructive lung disease.
5	RECOMMENDATION (Action)	the appearance suggests atelectasis	3	The appearance suggests atelectasis.

Рисунок 3.18 – Найпоширеніші шаблони шумових конструкцій після декількох етапів очищення

Таким чином, видалення шумових фрагментів стало ітеративним процесом: після кожного набору правил тексти знову перевірялися на предмет повторюваних конструкцій, що дозволяло поступово зменшити кількість небажаних фраз та зберегти тільки ті речення, які містять реальні медичні ознаки. Остаточна статистика показала, що значна частина технічних і бюрократичних вставок була усунута, а в категоріях, які раніше містили сотні випадків шуму, залишилися лише поодинокі фрази, що підлягають фінальному перегляду. Завдяки цьому етапу вдалося сформувати чистий і структурований корпус текстів, який відображає лише медичний зміст і є придатним для подальшого донавчання моделі.

Після видалення шумових конструкцій у датасеті залишалася лише невелика кількість анонімних токенів «XXXX», які вже не утворювали повторюваних

шаблонів і переважно були поодинокими випадками. Частина з них не мала жодної інформативної цінності, а решта могла впливати на зміст лише у дуже вузьких контекстах. Подальше їх відновлення вимагало би непропорційно великих витрат часу, тому на цьому етапі було ухвалено рішення повністю видалити всі залишки анонімних слів, але зробити це акуратно, щоб не пошкодити важливі речення.

Перед фінальним очищенням потрібно було визначити ті випадки, у яких анонімне слово виконувало суттєву роль і його видалення залишало б речення семантично порожнім. Для цього було здійснено спеціальний пошук по всіх текстах у полях *findings* та *impression*, виокремлюючи лише ті речення, де одночасно містилися принаймні одне інформативне слово, та анонімний токен «XXXX».

```
Found 20 sentences.

1. [IMPRESSION] No XXXX abnormalities as. (Info: abnormalities)
2. [FINDINGS] The airway is XXXX. (Info: airway)
3. [FINDINGS] This appears XXXX. (Info: appears)
4. [IMPRESSION] This appears XXXX from XXXX. (Info: appears)
5. [FINDINGS] Broken of the 4XXXX XXXX XXXX. (Info: broken)
6. [FINDINGS] No chest XXXX XXXX. (Info: chest)
7. [FINDINGS] XXXX cholecystectomy. (Info: cholecystectomy)
8. [FINDINGS] XXXX cholecystectomy. (Info: cholecystectomy)
9. [FINDINGS] No XXXX edema. (Info: edema)
10. [IMPRESSION] No XXXX edema. (Info: edema)
11. [FINDINGS] epicardial XXXX XXXX. (Info: epicardial)
12. [FINDINGS] Inferior XXXX XXXX XXXX. (Info: inferior)
13. [FINDINGS] Intact XXXX XXXX. (Info: intact)
14. [FINDINGS] Lungs are XXXX. (Info: lungs)
15. [FINDINGS] The lungs are XXXX. (Info: lungs)
16. [FINDINGS] XXXX XXXX normal. (Info: normal)
17. [FINDINGS] XXXX are osteopenic. (Info: osteopenic)
18. [IMPRESSION] This may be related to XXXX XXXX. (Info: related)
19. [IMPRESSION] XXXX since XXXX.. (Info: since)
20. [FINDINGS] XXXX XXXX of the spine. (Info: spine)
```

Рисунок 3.19 – Результат перевірки анонімізованих неінформативних речень

У результаті було знайдено 20 таких речень (рис. 3.19). Частина з них підлягала повному видаленню, оскільки після усунення анонімного слова зміст перетворювався на порожній або некорисний. Інші ж, де анонімізація не впливала на загальну інформативність (наприклад, коли залишалося достатньо контексту), були збережені.

Після цього можна було виконати фінальне очищення. На цьому етапі було застосовано функцію, яка забезпечувала повне видалення всіх залишків «XXXX», виправлення пунктуаційних спотворень, нормалізацію пробілів і приведення речень до коректного регістру. Завдяки цьому у фінальній версії тексту більше не залишалося анонімних фрагментів або технічних артефактів попередніх етапів обробки. Усі речення набули акуратного, читабельного вигляду, придатного як для подальшого аналізу, так і для використання у навчанні генеративної моделі.

3.3 Комплексна підготовка та структурування датасету для донавчання

Після проведення повного очищення текстових описів та приведення їх до уніфікованого формату наступним кроком стала фіналізація датасету для подальшого донавчання моделі. Оскільки кінцева мета системи – генерація рентгенівських знімків на основі текстового опису – важливо забезпечити репрезентативність навчальних даних, їх збалансованість за діагностичними категоріями та відповідність структурним вимогам дифузійної моделі.

На цьому етапі виконано комплекс робіт, що включав балансування класів, аналіз параметрів зображень, формування єдиного текстового опису для кожного знімка та приведення всіх даних до стандартизованого формату перед навчанням. Першим важливим завданням було вирішення проблеми домінування категорії *normal*, яка непропорційно переважала в наборі даних та потенційно могла змістити модель у бік генерації здебільшого здорових рентгенівських знімків.

Початковий аналіз розподілу діагнозів показав суттєву диспропорцію: категорія *normal* містила 1300 знімків, тоді як друга за чисельністю категорія – *Lung (hypoinflation/hyperdistention)* – мала 482 приклади. Такий перекіс міг негативно вплинути на навчання моделі, особливо в контексті генеративної задачі, де надлишок одного класу веде до надмірного «упередженого» відтворення саме цього типу зображень.

Замість випадкового видалення знімків було прийняте рішення виконати відбір за змістовністю опису. Для категорії *normal* наявні текстові пояснення є

класично короткими та менш інформативними, тому логічним способом зменшення впливу цього класу стало видалення прикладів з найкоротшими описами. Для цього всі записи категорії *normal* були відсортовані за довжиною об'єднаного текстового поля (findings + impression), після чого перші 800 записів із найменшим обсягом інформації було видалено (рис. 3.20).

```
Було записів 'normal': 1300
Видалено записів: 800
Залишилося записів 'normal': 500
Загальна кількість записів у новому датасеті: 2691

Максимальна довжина видаленого запису: 211.0 символів
Мінімальна довжина залишеного запису: 211.0 символів
```

Рисунок 3.20 – Результат видалення надмірної кількості нормальних знімків

Таким чином було вирішено 2 задачі одночасно:

– зберігаються найінформативніші приклади, які можуть містити корисні формулювання для моделі.

– видаляються малозмістовні записи, які не впливають на діагностичну різноманітність та лише підсилюють дисбаланс.

У результаті кількість знімків *normal* було зменшено з 1300 до 500. Статистика підтвердила коректність порогового значення: найдовший запис серед видалених і найкоротший серед залишених мали однакову довжину – 211 символів, що свідчить про чітку межу відбору. Після балансування загальна кількість прикладів у датасеті склала 2691 записи, що створило більш рівномірні умови для подальшого навчання LoRA-модуля.

Після формування збалансованого набору даних наступним кроком став аналіз вихідних рентгенівських знімків, оскільки на подальші етапи генерації синтетичних зображень важливо подавати однорідні за структурою та роздільною здатністю дані.

Насамперед було вирішено дослідити геометричні параметри всіх зображень: їхню фактичну роздільну здатність, пропорції та розподіл співвідношення сторін.

Для цього було завантажено весь збалансований датасет, після чого для кожного запису визначались ширина, висота та співвідношення сторін. Інформація про зображення збиралася в додатковий датафрейм, що дозволило проаналізувати не лише окремі приклади, але й отримати цілісну статистичну картину (рис. 3.21).

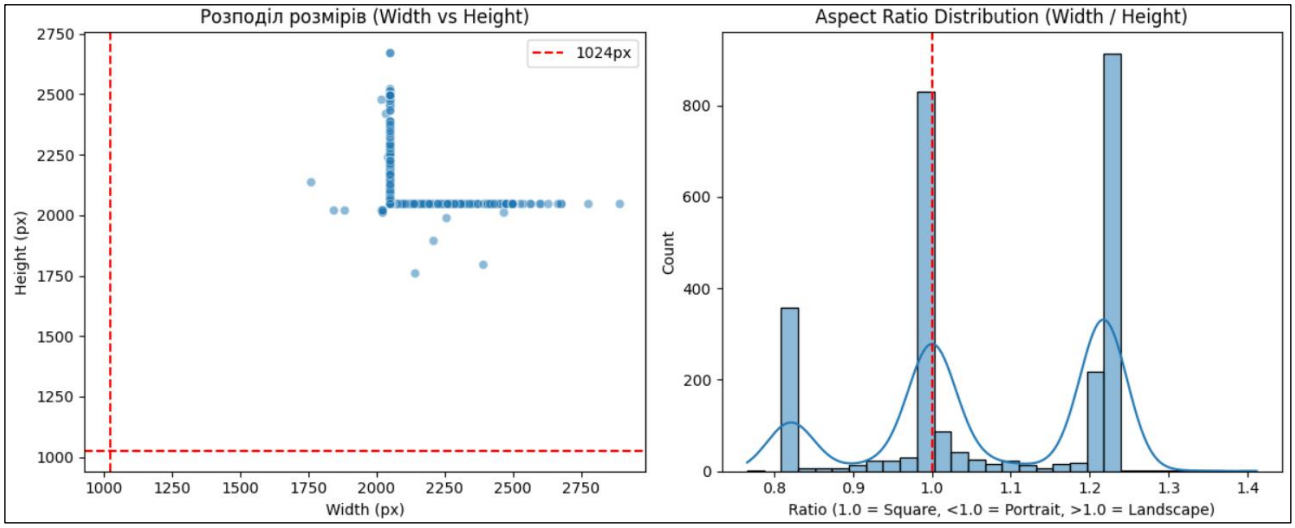

Рисунок 3.21 – Візуалізація розміру та співвідношення сторін зображень

Аналіз показав, що всі файли датасету присутні – жоден із записів не містив відсутніх або пошкоджених зображень. Діапазон розмірів виявився доволі широким: найменший знімок мав роздільну здатність 1760×1760, а найбільший – 2891×2674 пікселів. У середньому зображення складали приблизно 2249×2114 пікселів, що свідчить про те, що вихідні дані були отримані з різних джерел або обладнання з різними технічними характеристиками. Окрему увагу було приділено співвідношенню сторін, оскільки цей параметр є критичним при подальшому приведенні всіх зображень до єдиного формату (рис 3.22).

```
--- СТАТИСТИКА ЗОБРАЖЕНЬ ---
Всього перевірено: 2691
Відсутні файли: 0

Min Size: 1760 x 1760
Max Size: 2891 x 2674
Mean Size: 2249 x 2114

--- ГРУПИ ЗА СПІВВІДНОШЕННЯМ ---
Квадратні (~1:1): 1002 (37.2%)
Портретні (Високі): 433 (16.1%)
Ландшафтні (Широкі): 1256 (46.7%)
```

Рисунок 3.22 – Зведена статистика розміру та співвідношення сторін зображень

Результати показали, що близько третини знімків є майже квадратними: близько 37,2% мали співвідношення сторін у межах 0.95-1.05. Портретні (вертикально орієнтовані) зображення становили 16,1%, тоді як ландшафтні (горизонтальні) – більшість, а саме 46,7%. Така нерівномірність за типами пропорцій підтвердила необхідність подальшого перетворення всіх файлів до єдиного квадратного формату з фіксованим розміром 1024×1024, що було важливим для забезпечення стабільності в роботі моделі генерації та запобігання появі артефактів під час тренування дифузійної мережі.

Останнім етапом підготовки навчального набору даних стало формування фінальної структури, в якій кожному зображенню відповідав один стандартизований текстовий опис. На основі очищених текстових полів було побудовано єдине узгоджене текстове представлення для кожного запису. Воно складалося з трьох ключових компонентів, об’єднаних у суворо визначеному порядку: MeSH-міток, розділу findings та розділу impression. Однак перед їх злиттям застосовувався фіксований стилістичний тригер – коротке вступне речення *«Chest x-ray.»*, яке задавало структуру подальшого опису й водночас уніфікувало стиль текстів у всьому датасеті. Поле MeSH перетворювалося в діагностичний фрагмент, причому для нормальних випадків створювався компактний варіант *«Normal.»*, тоді як для патологій здійснювалося очищення пунктуації та роздільників, щоб отримати однорідні діагнозні фрази. Далі до цього діагностичного блоку додавалися тексти findings та impression, які проходили

попереднє очищення під час інших етапів обробки, що дало змогу уникнути залишкових артефактів деанонімізації та шумових фрагментів.

Паралельно з текстовою частиною виконувалась обробка всіх зображень із приведенням їх до квадратного формату 1024×1024 пікселі. Спершу проводилося масштабування зберігаючи пропорції, після чого застосовувалася адаптивна логіка обрізання. Для ландшафтних зображень обрізання виконувалося симетрично з боків після зменшення висоти, що дозволяло зберегти центральну частину знімка, яка зазвичай містить грудну клітку. Для портретних зображень використовувалася зміщена обрізка із пропорцією 1/3 зверху та 2/3 знизу. Такий підхід вибрано не випадково: рентгенівські знімки грудної клітки з вертикальною орієнтацією часто включають нижні ділянки тіла, зокрема частину черевної порожнини, яка не є інформативною для аналізу. Тому більш глибоке обрізання нижньої частини впорядковує кадр та концентрує модель на області легень і серця. Після масштабування та кропу всі зображення переводилися в RGB-формат та зберігалися у вихідній директорії з високою якістю стискання, що дозволило уникнути втрати деталей, важливих для генеративного моделювання.

Коли кожне зображення отримало свою фінальну форму та відповідний текстовий опис, дані були зібрані у метафайл, який містив назви файлів та їхні текстові описи. Ця таблиця стала основою для навчання дифузійної моделі, оскільки забезпечувала прямий зв’язок між візуальною та текстовою інформацією. Після перевірки цілісності всі файли – зображення формату 1024×1024 та *metadata.csv* – були упаковані в один архів. Це дало можливість отримати завершений, структурований і повністю підготовлений датасет, який можна безпосередньо використовувати під час навчання моделі генерації синтетичних рентгенівських знімків.

Висновки до розділу 3

У третьому розділі було реалізовано повний цикл підготовки набору даних для донавчання генеративної дифузійної моделі, що включав очищення,

стандартизацію, балансування, структурування та нормалізацію як текстової, так і візуальної складових. На першому етапі виконано багаторівневу обробку текстів, спрямовану на усунення службових фрагментів, залишків підстановок, артефактів деанонімізації та дублювань. Вміст полів *findings* та *impression* було приведено до змістово цілісної та формально узгодженої форми, що забезпечило мінімізацію шуму та підвищення інформативності текстової частини датасету.

Подальшим кроком стало балансування класів, що особливо важливо для моделей, які працюють у режимі текст-до-зображення та чутливі до диспропорції частот діагнозів. Клас *normal* був цілеспрямовано зменшений шляхом видалення знімків із найкоротшими описами, що дозволило вирівняти розподіл патологій без втрати інформативності даних і сформувати репрезентативну вибірку. Окремо здійснено аналіз вихідних рентгенограм, зокрема їхніх розмірів і пропорцій, що дозволило побудувати коректну логіку масштабування та адаптивного кропу для подальшої стандартизації зображень.

На завершальному етапі всі текстові описи було об’єднано в уніфікованому форматі з використанням фіксованого префікса *«Chest x-ray.»* та суворим порядком конфігурації діагностичного блоку й клінічного опису. Зображення приведено до квадратного формату 1024×1024 із застосуванням «розумного» кропу, який враховує особливості рентгенографії грудної клітки та дозволяє зберегти найбільш значущі області. Підготовлені тексти й зображення об'єднано у структуру з метафайлом та сформовано фінальний архів для використання у процесі донавчання моделі.

Таким чином, сформовано повністю очищений, збалансований і стандартизований датасет, який відповідає вимогам високоякісного навчального матеріалу для задач генерації медичних рентгенівських зображень на основі текстових описів. Результати підготовки створили надійну основу для подальшого експериментального навчання та аналізу якості генерації.

4 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Методологія донавчання моделі генерації зображень

Процес донавчання моделі розпочався з підготовки робочого середовища та завантаження сформованого у попередньому розділі датасету Chest X-rays (Indiana University), який було попередньо перетворено в архів та розгорнуто у вигляді структурованої директорії *iu-xray-1024-training-ready*. Цей набір даних містить нормалізовані зображення у форматі 1024×1024 та очищені текстові описи, що забезпечує його готовність до безпосереднього використання у навчальному циклі.

Для забезпечення сумісності з обраною моделлю Stable Diffusion XL та уникнення конфліктів між залежностями, було встановлено та оновлено необхідні бібліотеки, зокрема *diffusers*, *transformers*, *accelerate*, *peft*, *bitsandbytes* та додаткові допоміжні пакети. Після встановлення залежностей у середовище було імпортовано основні модулі, необхідні для роботи з дифузійними моделями, адаптерами LoRA та обчисленнями з підтримкою змішаної точності (fp16).

Оскільки доступ до моделей і базового коду Stable Diffusion потребує автентифікації в сервісі Hugging Face [26], у середовище було інтегровано збережений токен доступу «*HF_TOKEN*». Після успішного входу виконувалася перевірка автентифікації, що забезпечує можливість завантаження моделі SDXL напряму з офіційного репозиторію.

```
from kaggle_secrets import UserSecretsClient
from huggingface_hub import login

user_secrets = UserSecretsClient()
hf_token = user_secrets.get_secret("HF_TOKEN")

login(token=hf_token)
```

Датасет, завантажений у системний каталог вхідних файлів платформи Kaggle, було перенесено до робочого каталогу, що дозволило виконувати повноцінні операції запису та обробки даних у процесі навчання. Після цього

виконано завантаження базової моделі *stable-diffusion-xl-base-1.0* у форматі fp16 з використанням функції *snapshot_download*, що дало змогу зменшити обсяг файлів і зберегти обмежений дисковий простір робочого середовища.

```
from huggingface_hub import snapshot_download

MODEL_DIR = "/kaggle/working/sdxl-base-1.0-fp16"
os.makedirs(MODEL_DIR, exist_ok=True)

print(f"⬇ Завантаження моделі SDXL у {MODEL_DIR}...")

# Завантажуємо тільки потрібні fp16 файли
snapshot_download(
    repo_id="stabilityai/stable-diffusion-xl-base-1.0",
    local_dir=MODEL_DIR,
    local_dir_use_symlinks=False,
    allow_patterns=["*.json", "*.fp16.safetensors", "*.txt"],
    ignore_patterns=["*.bin", "*.pth", "*.onnx"]
)
print("✅ Модель завантажено локально!")
```

Для перевірки коректності завантаження моделі та початкової якості генеративного ядра було виконано тестове створення зображення за текстовим запитом *«Chest X-ray»* (рис. 4.1). Отримане зображення мало значні анатомічні спотворення, характерні для моделей загального призначення, що підтвердило необхідність донавчання моделі на спеціалізованому медичному датасеті.

Рисунок 4.1 – Згенероване зображення рентгену базовою моделлю SDXL

Для реалізації процесу донавчання моделі Stable Diffusion XL із застосуванням методу Low-Rank Adaptation (LoRA) було завантажено офіційний скрипт *train_text_to_image_lora_sdxl.py*, розроблений командою Hugging Face та рекомендований для адаптації моделей SDXL під задачі генерації зображень у високій роздільній здатності. Скрипт містить оптимізовану навчальну логіку, що включає підтримку LoRA-адаптерів, fp16-обчислень, використання оптимізатора Adam у 8-бітному форматі та механізми економії пам’яті (gradient checkpointing).

```
!wget
https://raw.githubusercontent.com/huggingface/diffusers/main/examples/text_to_image/train_text_to_image_lora_sdxl.py
```

Після завантаження навчального скрипту у робоче середовище було виконано налаштування базового конфігураційного файлу для бібліотеки *Accelerate*, яка забезпечує узгоджену роботу обчислювальних ресурсів і коректний запуск процесу навчання у змішаній точності. Створення конфігурації дозволило уникнути несумісностей між середовищем виконання Kaggle та параметрами навчального циклу.

```
from accelerate.utils import write_basic_config
write_basic_config()
```

На наступному етапі було визначено основні середовищні змінні, що описують розташування базової моделі SDXL, підготовленого датасету Chest X-ray та директорії для збереження результатів донавчання. Це забезпечило узгодженість шляхів і спростило запуск навчального процесу.

Фінальним кроком стало формування та виконання команди запуску навчання за допомогою *accelerate launch*. Під час навчання було використано такі ключові параметри:

- роздільна здатність зображень 1024×1024, що відповідає формату підготовленого датасету;
- навчання з малим розміром батчу та акумуляцією градієнтів, що дозволяє працювати в умовах обмеженої оперативної пам’яті GPU;
- швидкість навчання 1e-4 та константний граф планування LR;

– максимальна кількість кроків – 1500, що забезпечує адаптацію моделі без перенавчання;

– застосування оптимізатора Adam у 8-бітному режимі (*use_8bit_adam*), що зменшує навантаження на пам’ять;

– періодичне збереження проміжних чекпоінтів для подальшого аналізу стабільності навчання.

```
!accelerate launch train_text_to_image_lora_sdxl.py \
#  --pretrained_model_name_or_path=$MODEL_NAME \
#  --train_data_dir=$DATA_DIR \
#  --caption_column="text" \
#  --resolution=1024 \
#  --train_batch_size=1 \
#  --gradient_accumulation_steps=4 \
#  --learning_rate=1e-4 \
#  --lr_scheduler="constant" \
#  --lr_warmup_steps=0 \
#  --mixed_precision="fp16" \
#  --use_8bit_adam \
#  --max_train_steps=1500 \
#  --checkpointing_steps=500 \
#  --seed=42 \
#  --output_dir=$OUTPUT_DIR \
#  --gradient_checkpointing
```

Після запуску навчальний процес розпочався негайно, і модель поступово адаптувалася до структури, текстур та анатомічних особливостей медичних рентгенівських зображень. Саме на цьому етапі система почала формувати ознаки спеціалізації, що суттєво відрізняють донавчену модель від оригінальної версії SDXL загального призначення.

4.2 Перші результати генерації та їх якісний аналіз

Після завершення процесу донавчання моделі було виконано початкове тестування її якості на основі генерації окремих рентгенівських знімків із включенням LoRA-адаптера. Для цього було сформовано пайплайн Stable Diffusion XL, завантажений у конфігурації fp16, після чого до нього було підключено отримані ваги LoRA, що містили адаптовані параметри мережі для домену рентгенографії органів грудної клітки.

Перше тестове завдання передбачало генерацію зображень для двох типових запитів: *«Normal chest x-ray»* та патологічного стану – кардіомегалії. Генерація виконувалася з використанням медично орієнтованих описів та негативних підказок, покликаних уникнути появи здорових структур у випадку патології.

Перші результати засвідчили, що модель успішно відтворює загальну структуру рентгенівського знімка, коректно передає положення легеневих полів, тінь серця, реберні дуги та характерну монохроматичну текстуру медичної візуалізації. Порівняно з базовим SDXL, який до донавчання генерував стилізовані та неприродні «рентгеноподібні» зображення, адаптована модель демонструє істотне покращення анатомічної достовірності. Така поведінка свідчить про те, що дифузійна архітектура SDXL добре адаптується до медичної доменної специфіки та не втрачає стабільності після інтеграції LoRA.

Рисунок 4.2 – Результат генерації рентгену здорової грудної клітки

Промпт для кардіомегалії включав підвищення ваги ключового терміна, наприклад: *«(Cardiomegaly:1.5). The cardiac silhouette is significantly enlarged. Unfolding of the aorta. Pulmonary vasculature are within normal limits. No focal*

airspace opacity. No pleural effusion. Chest x-ray, monochrome», у поєднанні з негативними характеристиками: *«healthy heart, normal size heart»*.

```
      prompt = "(Cardiomegaly:1.5). The cardiac silhouette is significantly
enlarged. Unfolding of the aorta. Pulmonary vasculature are within normal limits.
No focal airspace opacity. No pleural effusion. Chest x-ray, monochrome."

      image = pipe(
          prompt=prompt,
          negative_prompt="healthy heart, normal size heart",
          num_inference_steps=40,
          guidance_scale=6,
          cross_attention_kwargs={"scale": 1.0}
      ).images[0]
```

Водночас під час аналізу патологічних генерацій було виявлено ключове обмеження (рис. 4.3): попри наявність LoRA, модель має схильність відтворювати зображення, близькі до норми, навіть коли опис містить підсилені маркери патології. Зокрема, при генерації кардіомегалії на багатьох зображеннях розмір тіні серця лише незначно відрізнявся від нормального, або модель взагалі відтворювала стандартний здоровий рентген.

Рисунок 4.3 – Результат синтетичної генерації стану відповідному кардіомегалії

Це явище має комплексну природу. На перший погляд може здатися, що основною причиною є класичний дисбаланс даних, властивий більшості медичних наборів: справді, нормальних або майже нормальних знімків зазвичай значно більше, ніж випадків із вираженими патологіями. Проте навіть повне вирівнювання такого дисбалансу на етапі підготовки даних навряд чи усунуло б головну проблему оскільки основна складність полягає в тому, що велика частина знімків, промаркованих як патологічні, фактично містять лише слабко виражені або початкові прояви захворювань, які важко виокремити навіть фахівцеві.

До цього додається важливий фактор того, що навіть рентгенівські знімки з патологіями здебільшого містять великий обсяг здорових структур – легеневі поля, кісткові елементи, окремі контури органів. Таким чином, патологія представлена лише частково, вона ніколи не домінує в зображенні повністю. На рівні латентного простору це означає, що модель під час навчання отримує суперпозицію великої кількості нормальних ознак і лише невелику частку патологічних, які до того ж часто варіюють за інтенсивністю та візуальною вираженістю. У результаті дифузійна модель починає інтерпретувати запит на патологію як запит на нормальну грудну клітину з невеликою, майже непомітною варіацією – що і пояснює, чому у відповідь на опис із кардіомегалією модель усе одно генерує здебільшого здорові варіанти.

Суттєву роль відіграє і сама стратегія донавчання. Було обрано мультикатегорійний підхід, коли модель одночасно вивчала велику кількість патологій з різною частотою, різною складністю та різним ступенем вираженості. За такого підходу LoRA не встигає достатньо чітко сформувати узагальнений шаблон для кожної конкретної патології, оскільки різні класи конкурують за параметри адаптації. Внаслідок цього опис патологій у латентному просторі стає розмитим, а домінуючим шаблоном залишається типовий нормальний рентген, який є найпоширенішим у наборі.

Не менш суттєвим чинником є й доменний зсув між початковою моделлю SDXL та медичними зображеннями. SDXL навчався переважно на фотографіях та

ілюстраціях, тобто на даних, що мають принципово іншу структуру, ніж рентгенівські знімки. Попри здатність до загальної адаптації, модель не володіє вбудованою фізичною чи анатомічною інтерпретацією, яка притаманна спеціалізованим медичним моделям. Тому формування патологічних шаблонів у такому середовищі стає складнішим і вимагає або більшої кількості відповідних зразків, або більш вузького фокусу навчання.

Аналіз результатів дозволив сформувати подальшу стратегію. Найперспективнішим напрямом виглядає донавчання моделі на окремих, вузько визначених патологіях – наприклад, лише на кардіомегалії. У такому випадку модель зможе побудувати чіткі, глибокі представлення одного типу патології та навчитися відтворювати її у різних формах і ступенях вираженості. Інший можливий підхід полягає в обмеженні кількості одночасно використаних класів до невеликої групи споріднених станів, що також дасть змогу покращити збалансованість та зменшити шум у даних. Обидві стратегії потенційно підвищують точність моделі та створюють основу для побудови спеціалізованих інструментів генерації синтетичних рентгенівських знімків.

4.3 Донавчання моделі для окремої патології (Opacity)

Логічним продовженням аналізу результатів попереднього донавчання стало впровадження альтернативної стратегії, що полягає у фокусуванні навчання на одній конкретній патології. Такий підхід безпосередньо випливає з виявлених обмежень мультикласового навчання та дозволяє перевірити гіпотезу про підвищення якості генерації за умови звуження предметної області. У межах даної роботи для експерименту було обрано категорію *Opacity* як одну з найбільш репрезентативних та водночас складних для візуального сприйняття.

У клінічному контексті термін *Opacity* використовується для опису ділянок підвищеної щільності на рентгенівських знімках легень, які проявляються у вигляді світлих або затемнених зон порівняно з нормальною легеневою тканиною. Такі зміни можуть бути спричинені широким спектром патологічних процесів, зокрема

пневмонією, набряком легень, інфільтративними ураженнями або ателектазом. Важливою особливістю є те, що підвищена щільність легень не завжди має чіткі межі чи виражену форму, а в багатьох випадках є слабо помітною для неспеціаліста, що додатково ускладнює її моделювання.

Аналіз початкового датасету показав, що загальна кількість знімків, промаркованих як такі, що містять підвищену щільність, є відносно значною (445 зображень). Проте детальніше дослідження структури анотацій виявило, що у переважній більшості випадків ця патологія поєднується з кількома іншими діагнозами. Якщо ж розглядати лише знімки, де підвищена щільність легень є єдиною або домінантною патологією, їх кількість зменшується до критично малого значення. Така ситуація створює додатковий шум у даних і заважає моделі сформувати чітке уявлення саме про характерні шаблони непрозорості легеневої тканини.

З огляду на це було реалізовано спеціалізований алгоритм фільтрації датасету, спрямований на формування більш «чистої» вибірки. Відбір здійснювався на основі текстових медичних анотацій (MeSH-термінів) із застосуванням кількох логічних правил. По-перше, обмежувалася максимальна кількість одночасно присутніх патологій на одному знімку, що дозволяло зменшити складність візуального контексту. По-друге, із вибірки виключалися знімки з небажаними або протилежними за візуальними ознаками станами, такими як пневмоторакс, емфізема або наявність сторонніх тіл і медичних пристроїв. По-третє, використовувався розширений список ключових слів, що охоплює суміжні діагнози, пов’язані з проявами підвищеної щільності легень, зокрема інфільтрацію, консолідацію, набряк або ателектаз.

Застосування такого підходу дозволило сформувати спеціалізований піднабір даних із 230 рентгенівських знімків, які з високою ймовірністю відображають патологію без надмірного домішку сторонніх візуальних факторів. Отриманий датасет став основою для окремого донавчання LoRA-моделі, метою

якого було перевірити, чи здатна дифузійна модель при вузькому фокусі навчання відтворювати більш виразні та стабільні патологічні шаблони.

Наступним етапом підготовки даних стало формування групи зображень, що відповідають умовно «нормальному» стану органів грудної клітки. Необхідність такого кроку зумовлена прагненням забезпечити збалансованість спеціалізованого датасету та надати моделі можливість навчитися чітко відрізняти патологічні прояви підвищеної щільності легень від відсутності виражених змін. Наявність обох типів зображень у межах одного навчального набору створює більш стійке представлення в латентному просторі та зменшує ризик надмірної генералізації патологічних ознак.

У результаті було відібрано 230 рентгенівських знімків, які відповідають нормальному стану без очевидних патологічних змін та без сторонніх об’єктів. Таким чином, кількість зображень у двох підмножинах – патологічній та контрольній – склала 460 та була свідомо вирівняна, що є важливою передумовою для стабільного донавчання дифузійної моделі.

Особливу увагу було приділено формуванню текстових описів для фінального датасету. Для всіх знімків із проявами підвищеної щільності було введено єдиний семантичний маркер у вигляді ключової фрази *«Lung opacity.»*. Цей маркер додавався на початок об’єднаного текстового опису, виконуючи роль домінантного сигналу для моделі під час навчання. Такий підхід дозволяє явно закріпити зв’язок між конкретним візуальним шаблоном і відповідним текстовим тригером, що є критично важливим для подальшої керованої генерації зображень за допомогою описів.

Для нормальних знімків, навпаки, спеціальний маркер не застосовувався, що підкреслює відсутність патологічного стану та формує для моделі контрастний приклад «базового» рентгенівського зображення. Подальші етапи обробки – нормалізація зображень, адаптивне обтинання, масштабування до роздільної здатності 1024×1024 та структурування даних – повністю відтворювали процедури,

використані на попередніх етапах підготовки, що забезпечує узгодженість і відтворюваність експерименту.

Сформований навчальний набір даних був збережений як окремий датасет у середовищі Kaggle та використаний для донавчання спеціалізованої LoRA-моделі. Це дозволило перейти від загального експериментального підходу до цілеспрямованого дослідження можливостей дифузійних моделей у генерації конкретної патології з чітко контрольованим семантичним маркером.

Після донавчання спеціалізованої LoRA-моделі на підмножині даних, що включала лише знімки з ознаками легеневої непрозорості та нормальних рентгенограм, основну увагу було зосереджено на аналізі того, які саме візуальні шаблони модель здатна відтворювати після такого звуження домену. Для цього було застосовано підхід прямого порівняння генерацій за контрольованих умов.

Генерація двох зображень – умовно «здорового» та з ознаками *Opacity* – здійснювалася з фіксованим значенням зерна генерації (seed). Такий прийом дозволяє мінімізувати стохастичну варіативність дифузійного процесу та забезпечити максимально схожі базові структури зображення, змінюючи при цьому лише семантичний вміст текстового опису. Обидва описи були сформульовані лаконічно, з акцентом на ключові ознаки: у першому випадку – на чистоту легеневих полів, у другому – на локалізовану легеневу непрозорість у базальному відділі правої легені.

Рисунок 4.4 – Аналіз впливу опису на генерацію з фіксованим зерном генерації

Отримані результати продемонстрували високий рівень структурної стабільності генерації (рис. 4.4). Загальна геометрія грудної клітки, контури легенів, серцевої тіні та кісткових структур у двох зображеннях майже повністю збігалися. Це підтверджується і побудованою тепловою картою різниць, яка показала мінімальні відхилення на більшій частині зображення. Основні розбіжності були зафіксовані поза межами легеневих полів, де спостерігалися незначні генеративні артефакти, що не мають діагностичного значення.

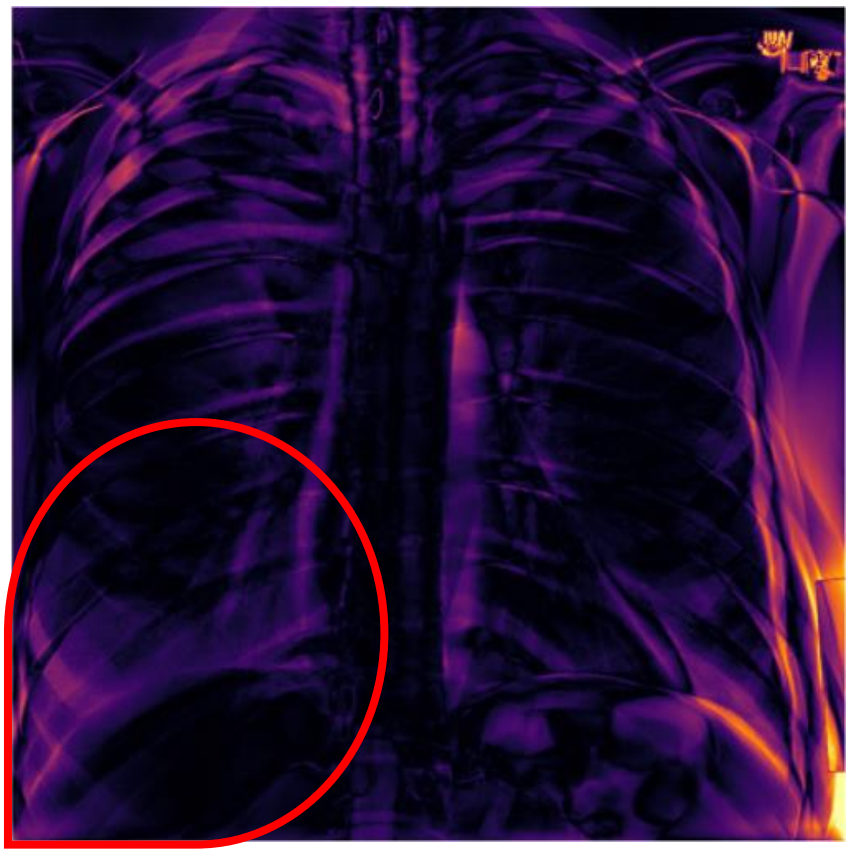

Рисунок 4.5 – Теплова карта з виділенням зони потенційно патологічних змін

Водночас у нижньому відділі правої легені було виявлено локальну ділянку відмінностей, яка корелює з текстовим описом патології (рис. 4.5). На тепловій карті ця зона проявляється спочатку як слабка дифузна структура з низькими значеннями інтенсивності, що поступово посилюється у напрямку периферії легені та досягає вищих значень у кутовій ділянці. Хоча на візуальному рівні така зміна

може здаватися маловираженою, її локалізація та форма узгоджуються з характерними проявами легеневої непрозорості на ранніх або легких стадіях.

Подібний результат можна інтерпретувати як ознаку того, що модель не намагається штучно «перемалювати» зображення або вводити грубі, неприродні зміни, а натомість зберігає загальний реалізм рентгенограми, модифікуючи лише ті області, які безпосередньо пов’язані з патологічним описом. Це узгоджується з природою навчальних даних, оскільки навіть знімки, марковані як патологічні, у більшості випадків містять лише локальні та часто слабко виражені зміни на тлі переважно здорових анатомічних структур.

Порівняння результатів генерації універсальної та спеціалізованої моделі для легеневої непрозорості показало суттєву різницю в характері відтворення патологічних ознак. Універсальна модель після загального донавчання здатна генерувати реалістичні рентгенівські зображення грудної клітки з коректною анатомією, однак патологічні зміни в таких зображеннях часто мають узагальнений характер. За навчання на великій кількості різнорідних патологій модель змушена формувати компромісні представлення, через що тонкі відмінності між нормальним і патологічним станом можуть згладжуватися, особливо у випадку слабко виражених або локальних змін, таких як opacity.

Натомість спеціалізоване донавчання на одній категорії дозволяє моделі зосередитися на конкретному типі патології та краще зберігати дрібні структурні відмінності. Це проявляється у більш стабільній генерації базової анатомії та появі локалізованих змін, що відповідають текстовому опису. Такий підхід створює передумови для точнішого відтворення клінічно значущих деталей і є більш придатним для задач генерації синтетичних медичних зображень.

Отже, проведений аналіз підтверджує доцільність використання точкового, вузькоспеціалізованого донавчання замість універсального підходу для задач генерації синтетичних медичних зображень. У контексті клінічних даних, де патологічні зміни часто є локальними та слабо контрастними, саме спеціалізація моделі виступає ключовим чинником збереження діагностично значущих деталей.

4.4 Реалізація інтелектуальної системи «SynTheX-Ray»

Реалізація системи «SynTheX-Ray» передбачає створення повноцінного користувацького інтерфейсу, який працює поверх обчислювальних потужностей Kaggle і забезпечує можливість керування моделлю генерації зображень без необхідності взаємодії з програмним кодом. Для цього було обрано фреймворк Streamlit, який дозволяє розгортати простий, але функціональний веб-інтерфейс безпосередньо в середовищі обчислювального ядра. Завдяки цьому підхід поєднує зручність фронтенду та високу продуктивність бекенду, де безпосередньо відбувається обробка запитів і виконання генерації зображень.

Оскільки Kaggle за замовчуванням не відкриває зовнішні порти, для забезпечення доступу до Streamlit поза межами середовища було використано сервіс ngrok, що дозволив створити захищений тунель і сформувати унікальну URL-адресу доступу. Перед запуском інтерфейсу потрібно було завантажити токен ngrok для роботи сервісу, а також токен Hugging Face, який дає можливість завантажувати та використовувати базову модель Stable Diffusion XL з репозиторію. На цьому етапі також імпортуються всі необхідні бібліотеки та інші залежності, необхідні для коректної роботи системи.

Увесь код інтерфейсу було структуровано в окремому файлі app.py, який містить усі необхідні компоненти: логіку очищення пам’яті GPU, функції ініціалізації або звільнення пайплайна, елементи інтерфейсу для вибору моделі, поля для введення текстових описів, механізми вибору гіперпараметрів генерації, а також функції збереження згенерованого зображення у форматі PNG зі службовими метаданими.

Інтерфейс системи було спроєктовано таким чином (рис 4.6), щоб він залишався максимально інтуїтивним і водночас відображав логіку роботи генеративної моделі. Основою візуальної частини став поділ простору на два функціональні блоки: панель налаштувань та робочу область. Бічна панель, що розташована ліворуч, містить елементи, пов’язані з конфігурацією системи – вибір

моделі, а також параметри генерації, такі як кількість ітерацій, коефіцієнт керованості та сила LoRA-модуляції. Панель може бути прихована за потреби, що забезпечує компактність і дозволяє зосередитися на основному екрані.

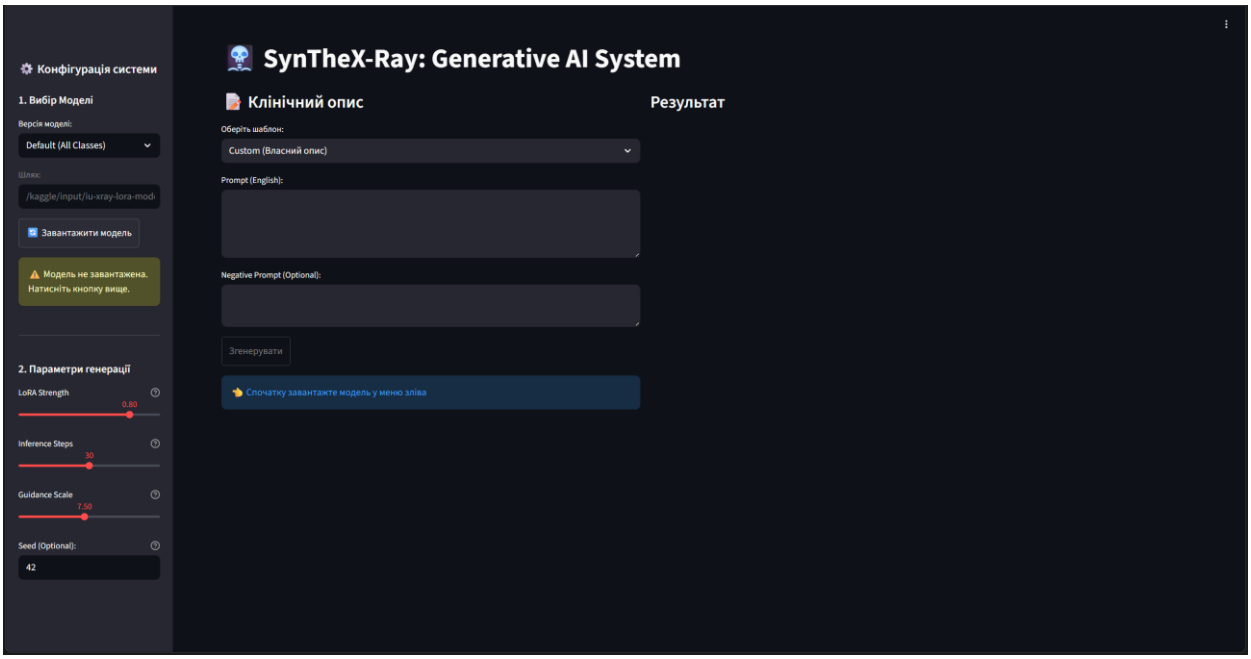

Рисунок 4.6 – Інтерфейс інтелектуальної системи SynTheX-Ray

Центральний простір інтерфейсу поділено на дві колонки: поле введення текстових описів та область відображення результату. Користувач спочатку формує клінічний опис або обирає одну з доступних заготовок, після чого має змогу відредагувати або розширити деталі у вигляді *Prompt* і *Negative Prompt*. Праворуч у реальному часі з’являється результат генерації, що дає змогу швидко оцінити якість моделі та за потреби повторити генерацію зі зміненими параметрами. Така структура дозволила зробити взаємодію зі складною нейронною системою зручною навіть для користувачів, які не мають спеціальної підготовки.

Основна ідея інтерфейсу полягає в тому, щоб надати користувачу можливість у кілька кроків вибрати відповідну модель, завантажити її в оперативну пам’ять та перейти безпосередньо до процесу генерації зображень із текстових описів.

У системі реалізовано випадаючий список в меню конфігурації системи (рис. 4.7), де користувачу пропонуються доступні варіанти попередньо

підготовлених моделей, включаючи дефолтну модель *Default (All Classes)* – базовий варіант, що був донавчений на всіх категоріях датасету. Також система підтримує спеціалізовані моделі, наприклад окремо створену модель для патологій типу *Opacity*, яка дозволяє протестувати можливість більш вузької адаптації моделі під конкретний підтип захворювань.

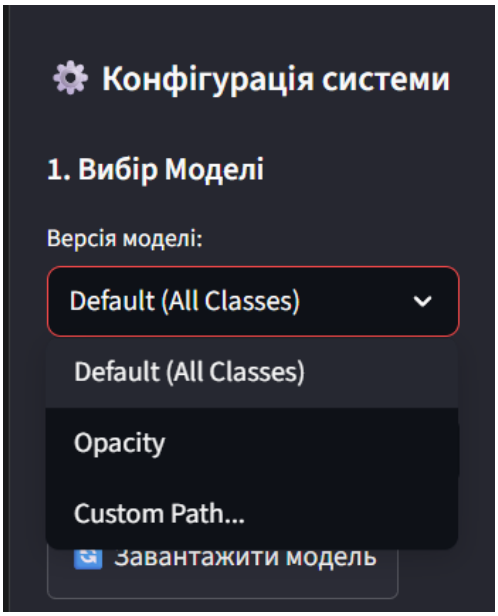

Рисунок 4.7 – Ілюстрація меню вибору моделі

Якщо ж користувач бажає використати власні ваги або самостійно донавчену LoRA-модель, передбачено режим *Custom Path*. Після вибору цього пункту інтерфейс автоматично активує поле для введення шляху до моделі вручну. Це дозволяє легко підключати моделі з локального середовища або зовнішніх джерел, зберігаючи гнучкість системи та її придатність для подальших експериментів (рис. 4.8).

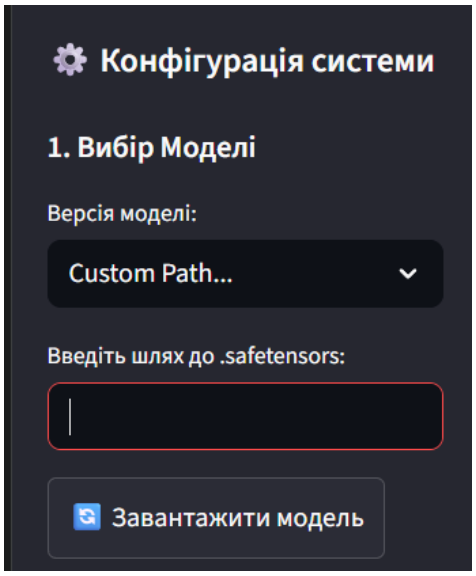

Рисунок 4.8– Ілюстрація самостійного додавання моделі

У бічній панелі інтерфейсу зосереджено основні параметри керування процесом генерації синтетичних рентгенівських знімків, що дозволяють користувачу впливати на поведінку дифузійної моделі та характеристики сформованого зображення (рис. 4.9).

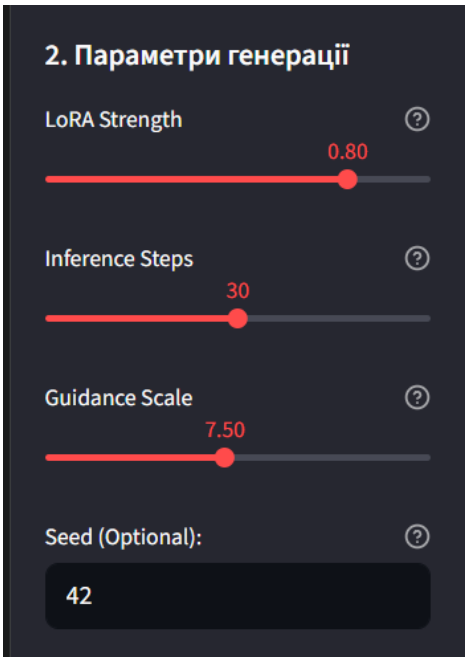

Рисунок 4.9 – Ілюстрація меню параметрів генерації

Основні параметри та їх призначення наведено в таблиці 4.1.

Таблиця 4.1 – Параметри генерації та їх призначення

Параметр	**Призначення**
LoRA Strength	Визначає інтенсивність впливу попередньо натренованих LoRA-ваг на фінальний результат генерації. Дозволяє керувати тим, наскільки сильно спеціалізована модель формуватиме стиль та особливості синтетичного рентгенівського знімка.
Inference Steps	Кількість кроків дифузійного процесу. Чим більше значення, тим детальнішим і чистішим може бути зображення, однак це збільшує час генерації.
Guidance Scale	Регулює ступінь відповідності результату заданому текстовому опису. Вищі значення посилюють «слухняність» моделі щодо промпта, але можуть зменшити різноманітність генерованих зображень.
Seed	Забезпечує контроль над відтворюваністю результатів. Якщо встановити конкретне значення seed, модель генеруватиме одні й ті самі зображення при однакових умовах. Якщо залишити seed порожнім, результат буде випадковим.

Особливістю інтерфейсу є наявність невеликих іконок-підказок поряд з кожним параметром. При наведенні курсора на таку іконку користувач отримує стислий опис того, для чого призначений відповідний повзунок і як його зміна впливає на результати генерації. Усі пояснення створені так, щоб користувач міг швидко зорієнтуватися та експериментувати з параметрами не боячись помилитися.

Після успішного завантаження обраної моделі користувачу стає доступною основна робоча частина інтерфейсу системи. Центральним елементом цієї частини є поле введення текстового опису (prompt), який має формуватися англійською мовою, оскільки саме така мова використовується в процесі навчання та інференсу моделі Stable Diffusion XL.

Користувач може створювати текстовий опис самостійно або скористатися одним із заздалегідь підготовлених шаблонів клінічних описів, доступних у випадаючому списку (рис. 4.10).

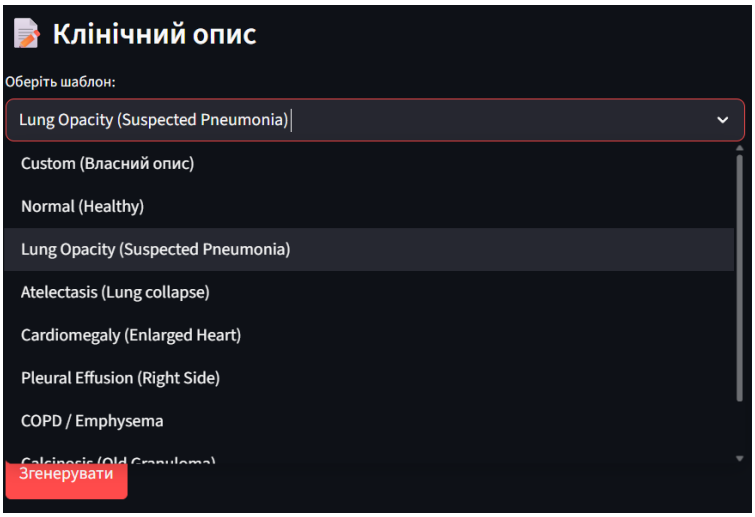

Рисунок 4.10 – Ілюстрація вибору шаблонів клінічних описів

Після вибору шаблону відповідний повний текст автоматично підставляється в поле промпту, де за потреби може бути відредагований (рис. 4.11). Зазначені шаблони не обмежують користувача у формуванні опису, а виконують допоміжну роль, демонструючи приклади коректних клінічних формулювань та структуру описів рентгенівських знімків.

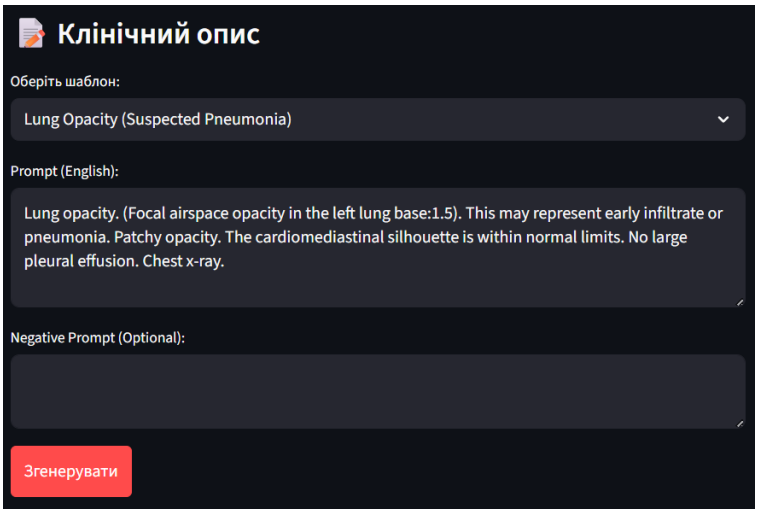

Рисунок 4.11 – Ілюстрація опису за обраним шаблоном

Додатково інтерфейс передбачає поле для введення негативного промпту (Negative Prompt), яке є опціональним, але дозволяє уточнити процес генерації шляхом виключення небажаних візуальних характеристик або артефактів. Такий

механізм підвищує контроль над результатом та сприяє отриманню більш релевантних зображень.

Після заповнення текстових полів користувач може ініціювати генерацію натисканням кнопки «Згенерувати». Тривалість процесу генерації зазвичай становить від 30 секунд до півтори хвилини й залежить від обраних параметрів дифузії та обчислювальних ресурсів. У випадку, якщо поле текстового опису залишено порожнім, система використовує базовий опис *«Chest x-ray.»*, що дозволяє згенерувати нейтральний знімок без додаткових уточнень.

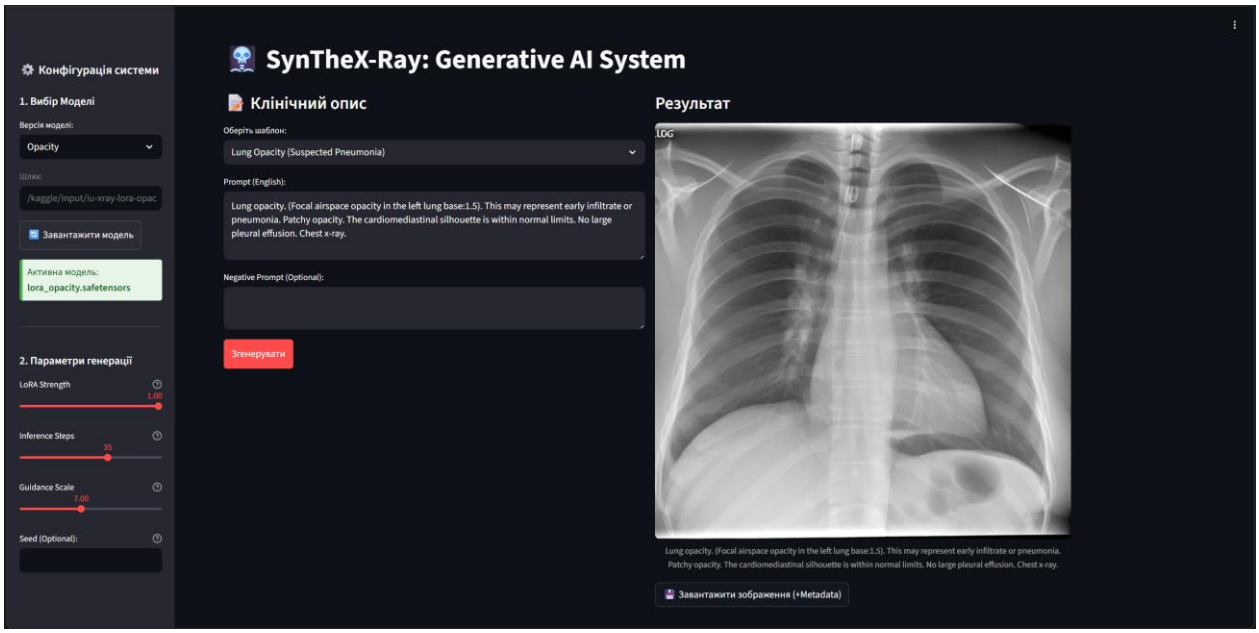

Рисунок 4.12 – Ілюстрація результату генерації синтетичного рентгенівського знімку за текстовим описом

Після завершення генерації отримане зображення відображається у правій частині інтерфейсу (рис. 4.12). Під ним виводиться текстовий опис, який було використано для генерації, а також надається можливість завантаження зображення. Під час збереження результату відповідний текстовий опис автоматично вбудовується у файл зображення у вигляді метаданих формату PNG (рис. 4.13). Такий підхід забезпечує збереження семантичного зв’язку між зображенням та його описом і дозволяє надалі зчитувати ці дані за допомогою

спеціалізованих скриптів, що є особливо корисним у контексті формування та аналізу навчальних датасетів.

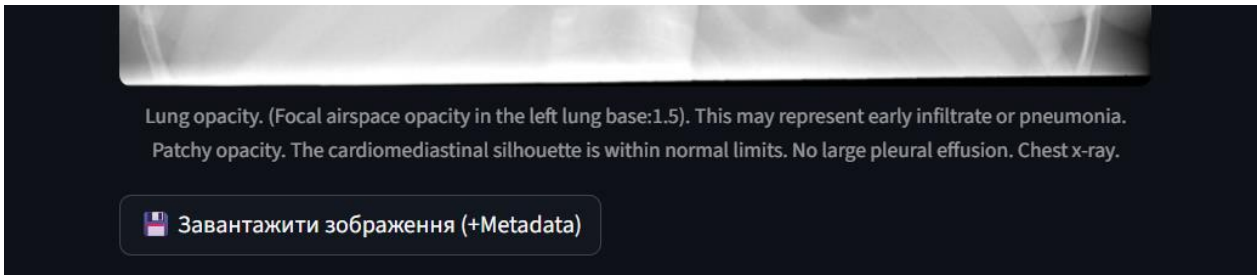

Рисунок 4.13 – Ілюстрація опису згенерованого синтетичного знімку яке буде збережено разом із зображенням

Таким чином, розроблений інтерфейс системи «SynTheX-Ray» забезпечує зручну та послідовну взаємодію користувача з донавченою моделлю генерації зображень, поєднуючи гнучкість налаштувань із простотою використання. Реалізація механізмів вибору моделі, керування параметрами генерації, формування текстових і негативних описів, а також збереження результатів разом із вбудованими метаданими дозволяє розглядати інтерфейс не лише як демонстраційний інструмент, а як повноцінний приклад прикладної інтелектуальної системи для генерації синтетичних рентгенівських знімків, орієнтованої на подальше використання в навчальних та дослідницьких цілях.

Висновки до розділу 4

У четвертому розділі було реалізовано та експериментально досліджено інтелектуальну систему генерації синтетичних рентгенівських знімків грудної клітки на основі текстових описів із використанням дифузійної моделі Stable Diffusion XL. Описано методологію донавчання моделі за допомогою підходу LoRA, що дозволило адаптувати загальну генеративну модель до медичного домену без повного перенавчання та з обмеженими обчислювальними ресурсами.

Перші результати генерації показали, що донавчена модель здатна відтворювати загальну анатомічну структуру рентгенівських знімків грудної клітки та формувати візуально коректні, реалістичні зображення. Водночас було виявлено, що універсальне донавчання на широкому наборі патологій призводить до домінування ознак здорових органів навіть у випадках, коли текстовий опис містить патологічні стани. Це пояснюється як дизбалансом класів у навчальних даних, так і специфікою медичних зображень, де патологія часто є локальною та малоконтрастною на фоні переважно здорових структур.

Подальший експеримент із донавчанням моделі на окремій патології (*opacity*) підтвердив доцільність вузькоспеціалізованого підходу. Зосередження навчання на обмеженому наборі знімків із контрольованою кількістю супутніх патологій дозволило моделі краще зберігати стабільність базової анатомії та водночас формувати тонкі відмінності між нормальним і патологічним станами. Порівняльний аналіз з використанням фіксованого зерна генерації (seed) та теплових карт різниці продемонстрував, що модель навчилася вносити локальні зміни, узгоджені з текстовим описом, без суттєвого порушення загальної структури зображення.

Завершальним етапом розділу стала реалізація повноцінної інтелектуальної системи «SynTheX-Ray» з веб-інтерфейсом на основі Streamlit. Система забезпечує вибір донавчених моделей, керування параметрами генерації, введення текстових описів та збереження результатів із вбудованими метаданими. Реалізований інтерфейс дозволяє гнучко експериментувати з моделями та параметрами генерації, що робить систему зручною платформою для подальших досліджень у сфері синтетичної генерації медичних зображень.

Таким чином, у межах четвертого розділу не лише реалізовано працездатну інтелектуальну систему, але й отримано важливі експериментальні висновки щодо впливу стратегії донавчання на якість та клінічну релевантність синтетичних рентгенівських знімків, що створює основу для подальшого розвитку спеціалізованих генеративних моделей у медичній візуалізації.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було проведено комплексне дослідження та реалізовано інтелектуальну систему генерації синтетичних рентгенівських знімків грудної клітки на основі текстових описів із використанням дифузійних нейронних мереж. Робота поєднує аналітичний, теоретичний і прикладний підходи, що дозволило не лише створити працездатну програмну систему, але й отримати науково обґрунтовані висновки щодо ефективності застосування сучасних генеративних моделей у медичній візуалізації.

У межах дослідження проаналізовано особливості предметної області рентгенодіагностики, зокрема проблему обмеженої доступності медичних даних, їх нерівномірного розподілу та високих вимог до якості зображень і супровідних описів. Досліджено сучасні підходи до генерації зображень із використанням глибинного навчання, а також проаналізовано їхні переваги та обмеження в контексті медичних застосувань. Особливу увагу приділено дифузійним моделям як одному з найперспективніших класів генеративних методів, здатних відтворювати складні, малоконтрастні анатомічні структури, характерні для рентгенівських знімків грудної клітки.

На основі аналізу наукових публікацій та існуючих рішень у сфері генерації синтетичних медичних зображень було виявлено ключові проблеми універсальних підходів, зокрема схильність моделей до втрати локальних патологічних ознак та домінування візуальних характеристик здорових тканин. Це дозволило обґрунтувати доцільність застосування спеціалізованого підходу до навчання генеративних моделей, орієнтованого на конкретні патологічні стани.

Важливим етапом стало формування та підготовка навчальних даних. Було проведено комплексну попередню обробку датасету рентгенівських знімків, яка включала очищення, фільтрацію та структуризацію як зображень, так і текстових описів. Особливу увагу приділено якості текстових анотацій, оскільки саме вони виступають умовною інформацією для генерації зображень. Виконано видалення

надлишкових, суперечливих і шумових фрагментів тексту, уніфіковано формулювання клінічних описів та забезпечено відповідність між текстовими даними й візуальним вмістом знімків. Проведена робота підтвердила, що якість і структурованість текстових описів є критично важливими чинниками для стабільного навчання дифузійних моделей і формування клінічно релевантних результатів генерації.

На основі проведеного аналізу та підготовлених даних було реалізовано інтелектуальну систему генерації синтетичних рентгенівських знімків із використанням моделі Stable Diffusion XL та підходу точкового донавчання LoRA. У процесі експериментальних досліджень виконано якісний аналіз результатів генерації, що продемонстрував здатність моделі відтворювати анатомічну структуру грудної клітки та формувати візуально правдоподібні рентгенівські знімки.

Окрему увагу було приділено дослідженню ефективності вузькоспеціалізованого донавчання моделі на окремій патології – легеневій непрозорості (*opacity*). Отримані результати показали, що спеціалізоване навчання дозволяє зберігати тонкі локальні патологічні ознаки та краще узгоджувати згенероване зображення з текстовим описом. Це підтверджує доцільність відмови від універсального підходу на користь точкового донавчання в задачах генерації медичних зображень, де навіть незначні візуальні деталі можуть мати діагностичне значення.

Завершальним етапом роботи стала реалізація користувацького інтерфейсу інтелектуальної системи «SynTheX-Ray», який забезпечує взаємодію з моделлю, введення та редагування текстових описів, керування параметрами генерації та збереження результатів разом із метаданими. Реалізована система демонструє практичну придатність для дослідницьких і навчальних задач та може бути використана як експериментальна платформа для подальшого розвитку методів синтетичної генерації медичних зображень.

Отримані результати підтверджують перспективність використання дифузійних нейронних мереж у поєднанні з якісною підготовкою текстових даних та спеціалізованим донавчанням для задач генерації синтетичних рентгенівських знімків. Запропонований підхід створює підґрунтя для подальших досліджень, зокрема розширення спектра патологій, підвищення клінічної достовірності зображень і інтеграції генеративних моделей у системи комп’ютерної підтримки медичних рішень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Adnane Ait Nasser, Moulay A Akhloufi. (2023). A Review of Recent Advances in Deep Learning Models for Chest Disease Detection Using Radiography. *Diagnostics, 13(1), 159.* URL: https://doi.org/10.3390/diagnostics13010159 (дата звернення: 28.08.2025)

2. Saeed Iqbala *et al*. Dynamic learning for imbalanced data in learning chest X-ray and CT images. *Heliyon Volume 9, Issue 6, June 2023, e16807.* URL: https://doi.org/10.1016/j.heliyon.2023.e16807 (дата звернення: 28.08.2025)

3. Dominik Cywiński. (2025). Creating Synthetic Training Data for Healthcare AI: What to Do When You Don't Have Big Datasets. *Momentum.* URL: https://www.themomentum.ai/blog/creating-synthetic-training-data-for-healthcare-ai (дата звернення: 02.09.2025).

4. Згенероване зображення. ChatGPT. URL: https://chatgpt.com (дата звернення: 04.09.2025).

5. Що таке GAN - генеративно-змагальні нейронні мережі і як їх застосовувати для генерації зображень. (2019). *Evergreen.* URL: https://evergreens.com.ua/ua/articles/gan.html (дата звернення: 12.09.2025).

6. Що таке Generative Adversarial Network (GAN)? (2020). *Unite.AI.* URL: https://www.unite.ai/uk/what-is-a-generative-adversarial-network-gan/ (дата звернення: 12.09.2025).

7. Дифузійні моделі в ШІ. (2023). *Unite.AI.* URL: https://www.unite.ai/uk/diffusion-models-in-ai-everything-you-need-to-know/ (дата звернення: 12.09.2025).

8. Що таке трансформаторні нейронні мережі?. (2021). *Unite.AI.* URL: https://www.unite.ai/uk/what-are-transformer-neural-networks/ (дата звернення: 12.09.2025).

9. Трансформерні архітектури та переднавчені мовні моделі. (2023). *TIB AV-PORTAL*. URL: https://av.tib.eu/media/63458. (дата звернення: 12.09.2025).

10. Yuanfeng Ji *et al.* (2025). A Generative Foundation Model for Chest Radiography. *arXiv:2509.03903*. URL: https://doi.org/10.48550/arXiv.2509.03903 (дата звернення: 17.09.2025).

11. Junjie Shentu. (2023). CXR-IRGen: An Integrated Vision and Language Model for the Generation of Clinically Accurate Chest X-Ray Image-Report Pairs. URL: https://openaccess.thecvf.com/content/WACV2024/papers/Shentu_CXR-IRGen_An_Integrated_Vision_and_Language_Model_for_the_Generation_WACV_2024_paper.pdf (дата звернення: 17.09.2025).

12. Tobias Weber, Michael Ingrisch, Bernd Bischl, David Rügamer. (2023). Cascaded Latent Diffusion Models for High-Resolution Chest X-ray Synthesis. *Advances in Knowledge Discovery and Data Mining: 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD, May 25–28, Proceedings, Part III, 180-191*. URL: https://doi.org/10.48550/arXiv.2303.11224 (дата звернення: 17.09.2025).

13. Gregory Schuit, Denis Parra, Cecilia Besa. (2025). Perceptual Evaluation of GANs and Diffusion Models for Generating X-rays. *arXiv:2508.07128*. URL: https://doi.org/10.48550/arXiv.2508.07128 (дата звернення: 25.09.2025).

14. Muhammad Usman Akbar, Wuhao Wang, Anders Eklund. (2023). Beware of Diffusion Models for Synthesizing Medical Images - a Comparison with Gans in Terms of Memorizing Brain MRI and Chest X-Ray Images. URL: https://ssrn.com/abstract=4611613 (дата звернення: 25.09.2025).

15. Arwa H. Alshanbari, Salha M. Alzahrani. (2025). Generative AI for Diagnostic Medical Imaging: A Review. *Current Medical Imaging, Volume 21, Issue 1*. URL: https://doi.org/10.2174/0115734056369157250212095252 (дата звернення: 25.09.2025).

16. Abdullah al Nomaan Nafi *et al.* (2024). Diffusion-Based Approaches in Medical Image Generation and Analysis. *arXiv:2412.16860*. URL: https://doi.org/10.48550/arXiv.2412.16860 (дата звернення: 02.10.2025).

17. Maya Pavlova *et al.* (2022). COVID-Net CXR-2: An Enhanced Deep Convolutional Neural Network Design for Detection of COVID-19 Cases From Chest X-

ray Images. *Front. Med., Sec. Pulmonary Medicine Volume 9*. URL: https://doi.org/10.3389/fmed.2022.861680 (дата звернення: 02.10.2025).

18. Stable Diffusion. (2022). *WikipediA*. URL: https://en.wikipedia.org/wiki/Stable_Diffusion (дата звернення: 04.10.2025).

19. Rayan Yassminh. (2025). Contrastive Learning vs. Generative Modeling: What's the Difference and When to Use Each (with Examples). *Medium*. URL: https://blog.gopenai.com/contrastive-learning-vs-1253368ed8a5 (дата звернення: 04.10.2025).

20. Jie Wu *et al.* (2025). Enhancing Bolt Object Detection via AIGC-Driven Data Augmentation for Automated Construction Inspection. *3D Computer Vision and Smart Building and City, 3rd Edition, 15(5), 819*. URL: https://doi.org/10.3390/buildings15050819. (дата звернення: 04.10.2025).

21. Purnasai Gudikandula. (2023). Contrastive Language Image Pretraining (CLIP) Summary. *Medium*. URL: https://purnasaigudikandula.medium.com/ contrastive-language-image-pretraining-clip-summary-f2adacb95366 (дата звернення: 13.10.2025).

22. Python Web Development: The Reasons Why Should be Choose. (2023). *Britwise Technologies*. URL: https://www.britwise.com/blog/python-web-development (дата звернення: 15.10.2025).

23. Lily Turner. (2025). Pytorch vs Tensorflow: A Complete Breakdown. *The Knowledge Academy*. URL: https://www.theknowledgeacademy.com/blog/pytorch-vs-tensorflow/ (дата звернення: 15.10.2025).

24. Lily Turner. (2025). How to use Kaggle for Data Science: A Comprehensive Guide. *The Knowledge Academy*. URL: https://www.theknowledgeacademy.com/blog/how-to-use-kaggle-for-data-science/ (дата звернення: 15.10.2025).

25. Chest X-rays (Indiana University). *Kaggle*. URL: https://www.kaggle.com/datasets/raddar/chest-xrays-indiana-university (дата звернення: 15.10.2025).

26. Stable Diffusion XL. *Hugging Face*. URL: https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0 (дата звернення: 25.10.2025).

27. SDLX VAE fp16 Fix. *Hugging Face*. URL: https://huggingface.co/madebyollin/sdxl-vae-fp16-fix (дата звернення: 25.10.2025).

28. Diffusers. *GitHub*. URL: https://github.com/huggingface/diffusers (дата звернення: 12.11.2025).

29. Diffusers Documentation: Stable Diffusion XL. *Hugging Face*. URL: https://huggingface.co/docs/diffusers/api/pipelines/stable_diffusion/stable_diffusion_xl (дата звернення: 12.11.2025).

30. PEFT Documentation: Low-Rank Adaptation (LoRA). *Hugging Face*. URL: https://huggingface.co/docs/diffusers/training/lora (дата звернення: 18.11.2025).

ДОДАТОК А

Вміст data_preprocessing.py

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
import warnings
import textwrap
import random
import re
import string
import shutil

from collections import Counter
from wordcloud import WordCloud
from PIL import Image
from tqdm.auto import tqdm
from IPython.display import display

base_path = "/kaggle/input/chest-xrays-indiana-university"
projections_path = os.path.join(base_path, "indiana_projections.csv")
reports_path = os.path.join(base_path, "indiana_reports.csv")
images_path = os.path.join(base_path, "images/images_normalized")

projections_df = pd.read_csv(projections_path)
reports_df = pd.read_csv(reports_path)

projections_df.info()
reports_df.info()

display(projections_df.head(5))
display(reports_df.head(5))

# Перевірка кількості унікальних UID
print("Унікальних UID у reports:", reports_df['uid'].nunique())
print("Унікальних UID у projections:", projections_df['uid'].nunique())

# Перевірка перетину UID
common_uid = set(reports_df['uid']).intersection(set(projections_df['uid']))
print("Спільних UID:", len(common_uid))

merged_df = pd.merge(projections_df, reports_df, on='uid', how='inner')
print("Кількість записів після об’єднання:", len(merged_df))
merged_df.head(5)

all_images = set(os.listdir(images_path))
used_images = set(merged_df["filename"])
missing_images = used_images - all_images
extra_images = all_images - used_images

print("Зображень у папці:", len(all_images))
print("Використаних у CSV:", len(used_images))
print("Відсутніх зображень:", len(missing_images))
print("Зайвих зображень:", len(extra_images))

projections_frontal = merged_df[merged_df['projection'].str.lower() == 'frontal'].copy()
print("Фронтальних знімків:", len(projections_frontal))
```

```
# Залишаємо тільки фронтальні знімки
merged_frontal_df = merged_df[merged_df['projection'].str.lower() == 'frontal'].copy()
print("Залишилось фронтальних знімків:", len(merged_frontal_df))

# Чи є пацієнти з кількома фронтальними знімками
duplicates = merged_frontal_df['uid'].value_counts()
multiple_frontal = duplicates[duplicates > 1]
print("Пацієнтів з кількома фронтальними знімками:", len(multiple_frontal))

unique_frontal_df = merged_frontal_df[~merged_frontal_df['uid'].isin(multiple_frontal.index)]
print("Кількість записів після очищення:", len(unique_frontal_df))

# Слова для пошуку в MeSH
bad_mesh_phrases = [
    "Technical Quality of Image Unsatisfactory",
    "Contrast Media",
    "Abdomen"
]

# Слова для пошуку в описі (Findings + Impression)
bad_text_keywords = [
    "limited exam",
    "rotated examination",
    "rotated",
    "motion artifact"
]

def is_bad_mesh(mesh_str):
    if pd.isna(mesh_str): return False

    # Розбиваємо на окремі діагнози (через ;)
    diagnoses = mesh_str.split(';')

    for diag in diagnoses:
        # Беремо тільки першу частину (Main Heading) до '/'
        main_heading = diag.split('/')[0].strip()

        # Перевіряємо, чи цей заголовок у чорному списку (точний збіг, ігноруємо регістр)
        for bad_term in bad_mesh_phrases:
            if main_heading.lower() == bad_term.lower():
                return True
    return False

# Пошук в MeSH
mask_mesh_bad = unique_frontal_df['MeSH'].apply(is_bad_mesh)

# Пошук в описі (Findings + Impression)
full_text_series = unique_frontal_df['findings'].fillna('') + " " + unique_frontal_df['impression'].fillna('')

mask_text_bad = full_text_series.apply(
    lambda x: any(kw in x.lower() for kw in bad_text_keywords)
)

# Зведення статистики
mask_to_remove = mask_mesh_bad | mask_text_bad

print("=== НЕЯКІСНІ ЗНІМКИ ===")
print(f"1. За тегами MeSH (Technical/Contrast/Abdomen): {mask_mesh_bad.sum()}")
print(f"2. За текстом опису (Limited/Rotated/Motion): {mask_text_bad.sum()}")
print("-" * 50)
print(f"ВСЬОГО записів до видалення: {mask_to_remove.sum()} (з {len(unique_frontal_df)})")
```

```
clean_df = unique_frontal_df[~mask_to_remove].reset_index(drop=True)

print(f"=== РЕЗУЛЬТАТ ОЧИЩЕННЯ ===")
print(f"Було записів: {len(unique_frontal_df)}")
print(f"Видалено:     {mask_to_remove.sum()}")
print(f"Стало:        {len(clean_df)}")

columns_to_keep = ['uid', 'filename', 'MeSH', 'findings', 'impression']

final_dataset = clean_df.loc[:, [col for col in columns_to_keep if col in clean_df.columns]].copy()

print("\nФінальна структура датасету:")
print(final_dataset.info())

def first_deonymization(text):
    if pd.isna(text) or text == "": return text

    replacements = [
        # "(The) XXXX are normal" -> "The osseous structures are normal")
        (r"(?i)(^|[.?!]\s+)(?:the\s+)?(?:xxxx\s*)+(?=\s+are\s+(?:grossly\s+)?normal)", r"\1The osseous structures "),
…
…
…
        # "lungs have xxxx xxxx in the interval" -> ""
        (r"(?i)(?:(?<=^)|(?<=\.\s))[^.]*?lungs\s+have\s+(?:xxxx\s*)+\s+in\s+the\s+interval[^.]*(?:\.|$)", ""),
    ]

    for pattern, repl in replacements:
        text = re.sub(pattern, repl, text)

    text = re.sub(r'\s+', ' ', text).strip()
    text = re.sub(r'\s+([.,;?!])', r'\1', text)

    return text

final_dataset['findings'] = final_dataset['findings'].apply(first_deonymization)
final_dataset['impression'] = final_dataset['impression'].apply(first_deonymization)

def second_deonymization(text):
    if pd.isna(text) or text == "": return text

    replacements = [
        # "lymph XXXX" -> "lymph nodes"
        (r"(?i)(lymph)\s+(?:xxxx\s*)+", r"\1 nodes "),

        # "hilar (lymph) XXXX" -> "hilar lymph nodes"
        (r"(?i)(hilar)\s+(?:lymph\s+)?(?:xxxx\s*)+", r"\1 lymph nodes"),
…
…
…
        (r"(?i)(bilateral\s+upper\s+lobe)\s+(?:xxxx\s*)+(?=\s*,?\s*fibronodular)", r"\1 scarring"),

        #-----------------------------------------------------------
        (r"(?i)(cerclage)\s+(?:xxxx\s*)+(?=\s+appear\s+intact)", r"\1 wires"),
        (r"(?i)(?:xxxx\s*)+(?=\s+appear\s+intact)", "osseous structures"),

    ]

    for pattern, repl in replacements:
        text = re.sub(pattern, repl, text)
```

```
    text = re.sub(r'\s+', ' ', text).strip()
    text = re.sub(r'\s+([.,;?!])', r'\1', text)

    return text

final_dataset['findings'] = final_dataset['findings'].apply(second_deonymization)
final_dataset['impression'] = final_dataset['impression'].apply(second_deonymization)

def findings_optimizing(text):
    if pd.isna(text) or text == "": return text

    replacements = [
        # --- FINDINGS ---

        # --- Видалення всього реченння ---
        (r"(?i)^(?:the\s+)?(?:prior\s+)?(?:examination|study)\s+consists\s+of.*$", ""),
        (r"(?i)^three\s+images\s+submitted\.?$", ""),
        (r"(?i)(?:(?<=^)|(?<=\.\s))\d+\s+images?\.?", ""),
    ]

    for pattern, repl in replacements:
        text = re.sub(pattern, repl, text)

    text = re.sub(r'\s+', ' ', text).strip()
    text = re.sub(r'\s+([.,;?!])', r'\1', text)

    return text

final_dataset['findings'] = final_dataset['findings'].apply(findings_optimizing)

def text_optimizing(text):
    if pd.isna(text) or text == "": return text

    replacements = [
        # --- Перетворення ---

        # is unchanged -> is normal
        (r"(?i)\b(heart(?:\s+size)?|cardiomediastinal\s+silhouette|appearance|mediastinum|vasculature)\s+is\s+unchanged", "\1
is normal"),

        # are unchanged -> are normal
        (r"(?i)\b(contours|tissues|parenchyma|structures|hemidiaphragms)\s+are\s+unchanged", "\1 are normal"),
…
…
…
        # ct/scan/considered/advised
        (r"(?i)(?:(?<=^)|(?<=\.\s))[^.]*?\b(?:ct|scan|considered|advised)\b[^.]*(?:\.|$)", ""),
    ]

    for pattern, repl in replacements:
        text = re.sub(pattern, repl, text)

    text = re.sub(r'\s+', ' ', text).strip()
    text = re.sub(r'\s+([.,;?!])', r'\1', text)

    return text

final_dataset['impression'] = final_dataset['impression'].apply(text_optimizing)
final_dataset['findings'] = final_dataset['findings'].apply(text_optimizing)

def final_optimizing(text):
```

```
    if pd.isna(text) or text == "": return text

    replacements = [
        # 1. "No XXXX abnormalities as."
        (r"(?i)(?:(?<=^)|(?<=\.\s))no\s+(?:xxxx\s*)+\s+abnormalities\s+as[^.]*(?:\.|$)", ""),
…
…
…
        # 12. "XXXX XXXX of the spine."
        (r"(?i)(?:(?<=^)|(?<=\.\s))(?:xxxx\s*)+\s+of\s+the\s+spine\.?", ""),
    ]

    for pattern, repl in replacements:
        text = re.sub(pattern, repl, text)

    text = re.sub(r'\s+', ' ', text).strip()
    text = re.sub(r'\s+([.,;?!])', r'\1', text)

    return text

final_dataset['findings'] = final_dataset['findings'].apply(final_optimizing)
final_dataset['impression'] = final_dataset['impression'].apply(final_optimizing)

def final_polish(text):
    if pd.isna(text) or str(text).strip() == "":
        return ""

    text = str(text)

    # 1. Глобальне видалення ВСІХ залишків XXXX
    text = re.sub(r"(?i)xxxx", "", text)

    # 2. Виправлення пунктуації та пробілів
    # Видаляємо пробіли перед розділовими знаками ("word ." -> "word.")
    text = re.sub(r'\s+([.,;?])', r'\1', text)

    # Видаляємо дублікати розділових знаків (".." -> ".", ",," -> ",")
    text = re.sub(r'\.{2,}', '.', text)
    text = re.sub(r',{2,}', ',', text)

    # Виправляємо комбінації коми та крапки ("word,." -> "word.")
    text = re.sub(r'[,;]\s*\.', '.', text)

    # Видаляємо розділові знаки на початку рядка (якщо видалили перше слово)
    text = re.sub(r'^\s*[.,;]\s*', '', text)

    # Нормалізуємо пробіли (видаляємо подвійні, та пробіли на краях)
    text = re.sub(r'\s+', ' ', text).strip()

    # 3. Відновлення регістру (Sentence Case)
    if text:
        # Розбиваємо на речення по крапці/знаку оклику/питання
        sentences = re.split(r'(?<=[.!?])\s+', text)
        capitalized_sentences = []
        for s in sentences:
            if s:
                # Робимо першу літеру великою, решту не чіпаємо (щоб не зіпсувати "CT", "pH", "COPD")
                cap_s = s[0].upper() + s[1:]
                capitalized_sentences.append(cap_s)
        text = " ".join(capitalized_sentences)
```

```
    return text

final_dataset['impression'] = final_dataset['impression'].apply(final_polish)
final_dataset['findings'] = final_dataset['findings'].apply(final_polish)

df = final_dataset

# 1. Знаходимо записи з діагнозом 'normal'
# Створюємо маску (фільтр): шукаємо рядки, де в колонці MeSH написано 'normal' (незалежно від регістру)
normal_mask = df['MeSH'].str.lower() == 'normal'

# 2. Рахуємо довжину тексту для цих записів
# Об'єднуємо 'findings' та 'impression', прибираємо зайві пробіли та рахуємо кількість символів.
df.loc[normal_mask, 'text_length'] = (
    df.loc[normal_mask, 'findings'].fillna('') + " " +
    df.loc[normal_mask, 'impression'].fillna('')
).str.strip().str.len()

# 3. Визначаємо, які саме 800 записів треба видалити
# Сортуємо нормальні записи за довжиною тексту (від найкоротших до найдовших)
normal_indices_sorted = df[normal_mask].sort_values('text_length').index

# Беремо перші 800 індексів (це будуть найкоротші записи)
drop_indices = normal_indices_sorted[:800]

# 4. Видаляємо вибрані індекси з оригінального датафрейму
balanced_dataset = df.drop(drop_indices).copy()

# Прибираємо тимчасову колонку 'text_length'
if 'text_length' in balanced_dataset.columns:
    balanced_dataset = balanced_dataset.drop(columns=['text_length'])

# 5. Перевірка результатів
original_normal_count = len(df[normal_mask])
new_normal_count = len(balanced_dataset[balanced_dataset['MeSH'].str.lower() == 'normal'])
removed_count = len(drop_indices)

print(f"Було записів 'normal': {original_normal_count}")
print(f"Видалено записів: {removed_count}")
print(f"Залишилося записів 'normal': {new_normal_count}")
print(f"Загальна кількість записів у новому датасеті: {len(balanced_dataset)}")

# 6. Збереження результату
output_filename = 'balanced_dataset.csv'
balanced_dataset.to_csv(output_filename, index=False)
print(f"\nЗбалансований датасет збережено у файл: '{output_filename}'")

# --- НАЛАШТУВАННЯ ШЛЯХІВ ---
# Вхідні картинки
SOURCE_IMAGES_DIR = "/kaggle/input/chest-xrays-indiana-university/images/images_normalized"
# Вхідний CSV
CSV_PATH = "/kaggle/input/iu-xray-final/balanced_dataset.csv"

# Вихідна папка
OUTPUT_DIR = "/kaggle/working/train_data"
os.makedirs(OUTPUT_DIR, exist_ok=True)

TARGET_SIZE = 1024

# --- 1. ФУНКЦІЯ ПІДГОТОВКИ ТЕКСТУ ---
def prepare_final_prompt(row):
```

```
    # ГЛОБАЛЬНИЙ ПРЕФІКС (Тригер стилю)
    base_prefix = "Chest x-ray."

    # Обробка MeSH (Теги діагнозів)
    mesh_raw = str(row['MeSH'])

    if mesh_raw.lower() == 'normal':
        # Для норми: "Chest x-ray. Normal."
        mesh_text = "Normal."
    else:
        # Для патології: чистимо роздільники
        # "Cardiomegaly/borderline" -> "Cardiomegaly borderline."
        mesh_text = mesh_raw.replace(';', '. ').replace('/', ' ').replace('_', ' ')
        mesh_text = mesh_text.strip()
        if not mesh_text.endswith('.'):
            mesh_text += '.'

    # Об'єднання Findings + Impression
    findings = str(row.get('final_findings', row.get('findings', '')))
    impression = str(row.get('final_impression', row.get('impression', '')))

    if findings.lower() == 'nan': findings = ''
    if impression.lower() == 'nan': impression = ''

    # ЗБИРАЄМО ВСЕ РАЗОМ
    # Порядок: [Trigger] [Diagnoses] [Details]
    # Приклад: "Chest x-ray. Cardiomegaly. Heart size is enlarged..."
    full_prompt = f"{base_prefix} {mesh_text} {findings} {impression}"

    # Чистимо зайві пробіли
    full_prompt = " ".join(full_prompt.split())

    return full_prompt

# --- 2. ЗАВАНТАЖЕННЯ ДАНИХ ---
print("Loading dataset...")
if os.path.exists(CSV_PATH):
    df = pd.read_csv(CSV_PATH)
else:
    # Резервний варіант (якщо файл у поточній сесії)
    df = pd.read_csv('balanced_dataset.csv')

# Створюємо колонку з текстом
print("Processing text prompts...")
df['text'] = df.apply(prepare_final_prompt, axis=1)

# --- 3. ОБРОБКА ЗОБРАЖЕНЬ (SMART CROP) ---
print(f"Starting image processing for {len(df)} records...")

valid_metadata = []

for index, row in tqdm(df.iterrows(), total=len(df)):
    img_name = row['filename']
    src_path = os.path.join(SOURCE_IMAGES_DIR, img_name)
    dst_path = os.path.join(OUTPUT_DIR, img_name)

    try:
        with Image.open(src_path) as img:
            # Конвертація в RGB
            if img.mode != "RGB":
                img = img.convert("RGB")
```

```
        width, height = img.size
        aspect_ratio = width / height

        # Логіка зміни розміру (Resize)
        if width < height:
            # ПОРТРЕТ (Високе): Ширина стає 1024, висота > 1024
            new_width = TARGET_SIZE
            new_height = int(TARGET_SIZE / aspect_ratio)
        else:
            # ЛАНДШАФТ (Широке): Висота стає 1024, ширина > 1024
            new_height = TARGET_SIZE
            new_width = int(TARGET_SIZE * aspect_ratio)

        img = img.resize((new_width, new_height), Image.LANCZOS)

        # Логіка обрізки (CROP)
        if width < height:
            # --- ПОРТРЕТНИЙ РЕЖИМ (Smart Crop: 1/3 зверху, 2/3 знизу) ---
            # Зайва висота:
            diff = new_height - TARGET_SIZE

            # Зміщуємо вікно вгору: відрізаємо зверху менше (1/3 від зайвого),
            # щоб зберегти верхівки легень.
            top = int(diff / 3)
            bottom = top + TARGET_SIZE

            # По горизонталі - центр (хоча ширина і так 1024, це формальність)
            left = 0
            right = TARGET_SIZE

        else:
            # --- ЛАНДШАФТНИЙ РЕЖИМ (Center Crop) ---
            # Тут класика: ріжемо боки порівну
            left = (new_width - TARGET_SIZE) // 2
            top = 0
            right = left + TARGET_SIZE
            bottom = TARGET_SIZE

        # Виконуємо обрізку
        img = img.crop((left, top, right, bottom))

        # Зберігаємо
        img.save(dst_path, quality=95)

        # Додаємо в метадані
        valid_metadata.append({'file_name': img_name, 'text': row['text']})

    except Exception as e:
        # print(f"Error processing {img_name}: {e}")
        pass

# --- 4. ЗБЕРЕЖЕННЯ METADATA.CSV ---
meta_df = pd.DataFrame(valid_metadata)
meta_df.to_csv(os.path.join(OUTPUT_DIR, "metadata.csv"), index=False)
```

ДОДАТОК Б

Вміст filtering_opacity.py

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
import warnings
import random
import re
import string
import shutil

from collections import Counter
from wordcloud import WordCloud
from PIL import Image
from tqdm.auto import tqdm
from IPython.display import display

dataset_dir = "/kaggle/input/iu-xray-final"

# 2. Повний шлях до файлу
csv_path = os.path.join(dataset_dir, "balanced_dataset.csv")

# 3. Перевірка
if not os.path.exists(csv_path):
    print(f"Файл не знайдено за шляхом: {csv_path}")
    print("Ось що є в папці /kaggle/input/:")
    for root, dirs, files in os.walk("/kaggle/input"):
        for file in files:
            print(os.path.join(root, file))
else:
    print(f"Файл знайдено: {csv_path}")

    # 4. Завантаження
    balanced_dataset = pd.read_csv(csv_path)
    print(f"Успішно завантажено {len(balanced_dataset)} рядків.")
    display(balanced_dataset.head(3))

df = balanced_dataset.dropna(subset=['MeSH']).copy()

def clean_mesh(mesh_str):
    items = str(mesh_str).split(';')
    clean_items = [item.split('/')[0].strip().lower() for item in items]
    return clean_items

df['pathology_list'] = df['MeSH'].apply(clean_mesh)

# --- 2. НАЛАШТУВАННЯ ФІЛЬТРІВ ---

# Що ми шукаємо (Синоніми білих плям)
core_keywords = {
    'opacity', 'lung opacity', 'pneumonia', 'infiltrate',
    'consolidation', 'density', 'airspace disease',
    'pulmonary congestion', 'congestion',
    'pulmonary edema', 'edema',
    'pulmonary atelectasis', 'atelectasis'
}
```

```
# Що для нас не бажано
blacklist_keywords = {
    # Метал та сторонні тіла
    'catheter', 'device', 'pacemaker', 'hardware', 'wire', 'clip', 'stents',
    'foreign bodies', 'surgical instruments', 'tube', 'implanted', 'monitoring',

    # Геометрія кісток і травми
    'fracture', 'fractures', 'vertebrae', 'spine', 'scoliosis', 'deformity',
    'spondylosis', 'kyphosis', 'ribs', 'osteophyte', 'bone',

    # Дрібні точки та пухлини
    'granuloma', 'calcinosis', 'nodule', 'mass', 'nodules',
    'nipple shadow',

    # Протилежність Opacity
    'pneumothorax', 'emphysema', 'hyperlucent', 'bulla', 'mastectomy',
    'chronic obstructive', 'copd'
}

# --- 3. ФУНКЦІЯ ВІДБОРУ ---
def select_pure_opacity(path_list):
    path_set = set(path_list)

    # Правило 1: Не більше 3 діагнозів
    if len(path_set) > 3:
        return False

    # Правило 2: Перевірка на Чорний список
    for path in path_set:
        for black_word in blacklist_keywords:
            if black_word in path:
                return False

    # Правило 3: Перевірка на Білий список
    has_core = False
    for path in path_set:
        for core_word in core_keywords:
            if core_word in path:
                has_core = True
                break

    if not has_core:
        return False

    return True

# Застосовуємо фільтр
selected_df = df[df['pathology_list'].apply(select_pure_opacity)].copy()

# Шлях до зображень
IMAGES_PATH = "/kaggle/input/chest-xrays-indiana-university/images/images_normalized"

# Словник заборонених слів
blacklist_keywords = {
    'catheter', 'device', 'pacemaker', 'hardware', 'wire', 'clip', 'stents',
    'implanted', 'mastectomy', 'scoliosis', 'deformity', 'spine', 'artifact',
    'spondylosis', 'calcified', 'granuloma', 'sternotomy', 'sutures'
}

# --- КРОК 1: Текстовий фільтр ---
```

```
# Беремо тільки ті, де MeSH містить 'normal'
normal_candidates = balanced_dataset[
    balanced_dataset['MeSH'].str.strip().str.lower() == 'normal'
].copy()

def is_text_clean(row):
    full_text = str(row['findings']) + " " + str(row['impression'])
    full_text = full_text.lower()

    for word in blacklist_keywords:
        if word in full_text:
            return False
    return True

clean_text_normals = normal_candidates[normal_candidates.apply(is_text_clean, axis=1)].copy()

print(f"Кандидатів після текстового відбору: {len(clean_text_normals)}")


# --- КРОК 2: Розрахунок геометрії (оновлений) ---

def get_aspect_ratio(filename):
    try:
        path = os.path.join(IMAGES_PATH, filename)
        with Image.open(path) as img:
            width, height = img.size
            return width / height
    except Exception as e:
        return None # Якщо файл битий

print("Розрахунок співвідношення сторін для всіх кандидатів...")

clean_text_normals['ratio'] = clean_text_normals['filename'].apply(get_aspect_ratio)

clean_text_normals = clean_text_normals.dropna(subset=['ratio'])


# --- КРОК 3: Розумний баланс і добір ---

TARGET_COUNT = 230

# 1. Спочатку беремо - квадратні (±15%)
square_mask = (clean_text_normals['ratio'] >= 0.85) & (clean_text_normals['ratio'] <= 1.15)
square_normals = clean_text_normals[square_mask].copy()

print(f"Знайдено ідеальних квадратних (1:1): {len(square_normals)}")

if len(square_normals) >= TARGET_COUNT:
    # Якщо квадратних вистачає - беремо 230 випадкових
    final_normal_df = square_normals.sample(n=TARGET_COUNT, random_state=42)
    print("Квадратних вистачило, добір не потрібен.")

else:
    # Якщо квадратних мало
    current_count = len(square_normals)
    needed = TARGET_COUNT - current_count
    print(f"Не вистачає {needed} зображень. Починаємо добір з широкоформатних...")

    # Шукаємо серед тих, що не потрапили в квадратні
    # Беремо тільки ті, що ширші (ratio > 1.15)
```

```
    wide_candidates = clean_text_normals[clean_text_normals['ratio'] > 1.15].copy()

    # Сортуємо за зростанням ratio
    wide_candidates = wide_candidates.sort_values(by='ratio', ascending=True)

    if len(wide_candidates) >= needed:
        # Беремо топ необхідної кількості
        extra_normals = wide_candidates.head(needed).copy()

        # Об'єднуємо: Всі квадратні + Добрані широкі
        final_normal_df = pd.concat([square_normals, extra_normals])
        print(f"Успішно добрано {len(extra_normals)} широких зображень.")
    else:
        # Якщо навіть з широкими не вистачає
        print(f"Увага! Доступно всього {len(wide_candidates)} широких. Беремо все що є.")
        final_normal_df = pd.concat([square_normals, wide_candidates])

print("-" * 30)
print(f"Фінальний розмір датасету Normal: {len(final_normal_df)} записів")

# 1.1. Для Opacity (selected_df)
selected_df['label'] = "Lung opacity."
# 1.2. Для Normal (final_normal_df)
final_normal_df['label'] = ""

# --- КРОК 2: Об'єднання та перемішування ---

# Вибираємо тільки потрібні колонки
cols_to_keep = ['filename', 'MeSH', 'findings', 'impression', 'label']

df_opacity = selected_df[cols_to_keep].copy()
df_normal = final_normal_df[cols_to_keep].copy()

joined_df = pd.concat([df_opacity, df_normal], ignore_index=True)

# Перемішуємо
joined_df = joined_df.sample(frac=1, random_state=42).reset_index(drop=True)

# --- КРОК 3: Оновлена функція генерації тексту ---

def prepare_final_prompt(row):
    # 1. ГЛОБАЛЬНИЙ ПРЕФІКС
    parts = ["Chest x-ray."]

    # 2. ДОДАТКОВИЙ ЛЕЙБЛ (Lung opacity або пусто)
    if row['label']:
        parts.append(row['label'])

    # 3. ОБРОБКА MeSH
    mesh_raw = str(row['MeSH'])

    if mesh_raw.strip().lower() == 'normal':
        # Для норми просто пишемо Normal
        parts.append("Normal.")
    else:
        # Для патології:
        # "Cardiomegaly/borderline; Other" -> "Cardiomegaly borderline. Other."
        mesh_text = mesh_raw.replace(';', '. ').replace('/', ' ').replace('_', ' ')

        # Видаляємо зайві пробіли і додаємо крапку в кінці, якщо треба
        mesh_text = " ".join(mesh_text.split())
```

```
        if not mesh_text.endswith('.'):
            mesh_text += '.'

        parts.append(mesh_text)

    # 4. Findings + Impression
    findings = str(row.get('findings', ''))
    impression = str(row.get('impression', ''))

    if findings.lower() == 'nan': findings = ''
    if impression.lower() == 'nan': impression = ''

    # Додаємо, якщо вони не пусті
    if findings: parts.append(findings)
    if impression: parts.append(impression)

    # 5. ЗБИРАЄМО РАЗОМ
    full_prompt = " ".join(parts)

    # Фінальна чистка від подвійних пробілів
    full_prompt = " ".join(full_prompt.split())

    return full_prompt

# --- КРОК 4: Застосування ---

print("Генерація описів...")
joined_df['text'] = joined_df.apply(prepare_final_prompt, axis=1)

# --- ПЕРЕВІРКА РЕЗУЛЬТАТУ ---
print("-" * 30)
print("Приклад Opacity:")
print(joined_df[joined_df['label'] != ""].iloc[0]['text'])

print("\n" + "-" * 30)
print("Приклад Normal:")
print(joined_df[joined_df['label'] == ""].iloc[0]['text'])

# --- НАЛАШТУВАННЯ ШЛЯХІВ ---
# Вхідні картинки (де лежить весь датасет Indiana)
SOURCE_IMAGES_DIR = "/kaggle/input/chest-xrays-indiana-university/images/images_normalized"
# Вихідна папка (тимчасова, перед архівацією)
OUTPUT_DIR = "/kaggle/working/opacity_train_data"
os.makedirs(OUTPUT_DIR, exist_ok=True)
TARGET_SIZE = 1024

# --- 1. ПІДГОТОВКА ТЕКСТУ ---
print("Копіювання підготовленого датафрейму...")
opacity_train_df = joined_df.copy()


# --- 2. ОБРОБКА ЗОБРАЖЕНЬ ---
print(f"Початок обробки {len(opacity_train_df)} зображень...")

valid_metadata = []
errors = 0

for index, row in tqdm(opacity_train_df.iterrows(), total=len(opacity_train_df)):
    img_name = row['filename']
    src_path = os.path.join(SOURCE_IMAGES_DIR, img_name)
    dst_path = os.path.join(OUTPUT_DIR, img_name)
```

```
    try:
        with Image.open(src_path) as img:
            # 1. Конвертація в RGB (SDXL не любить RGBA/Greyscale)
            if img.mode != "RGB":
                img = img.convert("RGB")

            width, height = img.size
            aspect_ratio = width / height

            # 2. Логіка зміни розміру (Resize)
            # Ми масштабуємо так, щоб менша сторона стала 1024.
            # Більша сторона стане > 1024, і ми її обріжемо.

            if width < height:  # Портрет (Високе)
                new_width = TARGET_SIZE
                new_height = int(TARGET_SIZE / aspect_ratio)
            else: # Ландшафт (Широке) або Квадрат
                new_height = TARGET_SIZE
                new_width = int(TARGET_SIZE * aspect_ratio)

            # Використовуємо LANCZOS для найкращої якості при зміні розміру
            img = img.resize((new_width, new_height), Image.LANCZOS)

            # 3. Логіка обрізки (CROP) до квадрата 1024x1024

            # Логіка Smart Crop
            if new_width < new_height: # Портрет (зсув вгору)
                diff = new_height - TARGET_SIZE
                top = int(diff / 3)
                bottom = top + TARGET_SIZE
                left = 0
                right = TARGET_SIZE
            else: # Ландшафт (центр)
                diff = new_width - TARGET_SIZE
                left = int(diff / 2)
                right = left + TARGET_SIZE
                top = 0
                bottom = TARGET_SIZE

            # Виконуємо обрізку
            img = img.crop((left, top, right, bottom))

            # 4. Зберігаємо
            img.save(dst_path, quality=95)

            # Додаємо в список для CSV тільки якщо збереження пройшло успішно
            valid_metadata.append({'file_name': img_name, 'text': row['text']})

    except Exception as e:
        # print(f"Skipping {img_name}: {e}")
        errors += 1

print(f"✅ Обробка завершена. Успішно: {len(valid_metadata)}, Помилок: {errors}")


# --- 3. ЗБЕРЕЖЕННЯ METADATA.CSV ---
meta_df = pd.DataFrame(valid_metadata)
meta_df.to_csv(os.path.join(OUTPUT_DIR, "metadata.csv"), index=False)
```

ДОДАТОК В

Вміст файлу train_lora_stage1.py

```
!pip install -q git+https://github.com/huggingface/diffusers
!pip install -q -U transformers accelerate peft bitsandbytes datasets
!pip install -U "protobuf<4" –quiet

import torch
import diffusers

import os
import shutil
import gc
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

from kaggle_secrets import UserSecretsClient
from huggingface_hub import login

user_secrets = UserSecretsClient()
hf_token = user_secrets.get_secret("HF_TOKEN")

login(token=hf_token)

!hf auth whoami

#!cp -r /kaggle/input/iu-xray-1024-training-ready /kaggle/working/train_data

from huggingface_hub import snapshot_download

# Куди качаємо (тимчасова папка)
MODEL_DIR = "/kaggle/working/sdxl-base-1.0-fp16"
os.makedirs(MODEL_DIR, exist_ok=True)

print(f"⬇ Завантаження моделі SDXL у {MODEL_DIR}...")

# Качаємо тільки потрібні fp16 файли
snapshot_download(
    repo_id="stabilityai/stable-diffusion-xl-base-1.0",
    local_dir=MODEL_DIR,
    local_dir_use_symlinks=False,
    allow_patterns=["*.json", "*.fp16.safetensors", "*.txt"]
    ignore_patterns=["*.bin", "*.pth", "*.onnx"]
)

print("✅ Модель завантажено локально!")

!wget
https://raw.githubusercontent.com/huggingface/diffusers/main/examples/text_to_image/train_text_to_image_lora_sdxl.py

from accelerate.utils import write_basic_config
write_basic_config()

#os.environ["MODEL_NAME"] = "/kaggle/working/sdxl-base-1.0-fp16"
#os.environ["DATA_DIR"] = "/kaggle/working/train_data"
#os.environ["OUTPUT_DIR"] = "/kaggle/working/sdxl-xray-lora-v1"

#print("Починаю навчання ...")
```

```
!accelerate launch train_text_to_image_lora_sdxl.py \
  --pretrained_model_name_or_path=$MODEL_NAME \
  --train_data_dir=$DATA_DIR \
  --caption_column="text" \
  --resolution=1024 \
  --train_batch_size=1 \
  --gradient_accumulation_steps=4 \
  --learning_rate=1e-4 \
  --lr_scheduler="constant" \
  --lr_warmup_steps=0 \
  --mixed_precision="fp16" \
  --use_8bit_adam \
  --max_train_steps=1500 \
  --checkpointing_steps=500 \
  --seed=42 \
  --output_dir=$OUTPUT_DIR \
  --gradient_checkpointing

TRAIN_OUTPUT = "/kaggle/working/lora-opacity-v1"
ARCHIVE_NAME = "/kaggle/working/Lora_Opacity_V1"

if os.path.exists(TRAIN_OUTPUT):
    print("Архівація...")
    shutil.make_archive(ARCHIVE_NAME, 'zip', TRAIN_OUTPUT)
    print(f"Архів створено: {ARCHIVE_NAME}.zip")

    shutil.rmtree(TRAIN_OUTPUT)
    print("Тимчасові файли видалено.")
else:
    print("Папку з моделлю не знайдено.")
```

ДОДАТОК Г

Вміст файлу train_lora_stage2_opacity.py

```
!pip install -q git+https://github.com/huggingface/diffusers
!pip install -q -U transformers accelerate peft bitsandbytes datasets
!pip install -U "protobuf<4" –quiet

import torch
import diffusers

import os
import shutil
import gc
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

from kaggle_secrets import UserSecretsClient
from huggingface_hub import login

user_secrets = UserSecretsClient()
hf_token = user_secrets.get_secret("HF_TOKEN")

login(token=hf_token)

!hf auth whoami

!cp -r /kaggle/input/iu-xray-1024-opacity-train-data /kaggle/working/opacity_train_data

from huggingface_hub import snapshot_download

# Куди качаємо (тимчасова папка)
MODEL_DIR = "/kaggle/working/sdxl-base-1.0-fp16"
os.makedirs(MODEL_DIR, exist_ok=True)

print(f"⬇ Завантаження моделі SDXL у {MODEL_DIR}...")

# Качаємо тільки потрібні fp16 файли
snapshot_download(
    repo_id="stabilityai/stable-diffusion-xl-base-1.0",
    local_dir=MODEL_DIR,
    local_dir_use_symlinks=False,
    allow_patterns=["*.json", "*.fp16.safetensors", "*.txt"]
    ignore_patterns=["*.bin", "*.pth", "*.onnx"]
)

print("✅ Модель завантажено локально!")

wget
https://raw.githubusercontent.com/huggingface/diffusers/main/examples/text_to_image/train_text_to_image_lora_sdxl.py

from accelerate.utils import write_basic_config
write_basic_config()

os.environ["MODEL_NAME"] = "stabilityai/stable-diffusion-xl-base-1.0"
os.environ["VAE_NAME"] = "madebyollin/sdxl-vae-fp16-fix"

os.environ["LOCAL_DATA_DIR"] = "/kaggle/working/opacity_train_data"
os.environ["OUTPUT_DIR"] = "/kaggle/working/lora-opacity-v1"
```

```
#print("Починаю навчання ...")

!accelerate launch train_text_to_image_lora_sdxl.py \
  --pretrained_model_name_or_path=$MODEL_NAME \
  --pretrained_vae_model_name_or_path=$VAE_NAME \
  --train_data_dir=$LOCAL_DATA_DIR \
  --caption_column="text" \
  --resolution=1024 \
  --train_batch_size=1 \
  --gradient_accumulation_steps=4 \
  --learning_rate=1e-4 \
  --lr_scheduler="constant" \
  --lr_warmup_steps=0 \
  --mixed_precision="fp16" \
  --use_8bit_adam \
  --gradient_checkpointing \
  --num_train_epochs=15 \
  --checkpointing_steps=500 \
  --validation_prompt="Chest x-ray. Lung opacity. Opacity lung base right." \
  --validation_epochs=2 \
  --seed=42 \
  --output_dir=$OUTPUT_DIR

TRAIN_OUTPUT = "/kaggle/working/lora-opacity-v1"
ARCHIVE_NAME = "/kaggle/working/Lora_Opacity_V1"

if os.path.exists(TRAIN_OUTPUT):
    print("Архівація...")
    shutil.make_archive(ARCHIVE_NAME, 'zip', TRAIN_OUTPUT)
    print(f"Архів створено: {ARCHIVE_NAME}.zip")

    shutil.rmtree(TRAIN_OUTPUT)
    print("Тимчасові файли видалено.")
else:
    print("Папку з моделлю не знайдено.")

from diffusers import StableDiffusionXLPipeline, AutoencoderKL

# 1. Очищення пам'яті перед стартом
gc.collect()
torch.cuda.empty_cache()

# 2. Шляхи до моделей
# Шлях до SDXL завантаженого за допомогою snapshot_download
base_model_path = "/kaggle/working/sdxl-base-1.0-fp16"
# Шлях до файлу LoRA
lora_path = "/kaggle/input/iu-xray-lora-opacity/lora_opacity.safetensors"

print("Завантажую пайплайн...")

# 3. Завантаження базової моделі

vae = AutoencoderKL.from_pretrained(
    "madebyollin/sdxl-vae-fp16-fix",
    torch_dtype=torch.float16
)

pipe = StableDiffusionXLPipeline.from_pretrained(
    base_model_path,
    vae=vae,
```

```
    torch_dtype=torch.float16,
    variant="fp16",
    use_safetensors=True
)

pipe.to("cuda")

# 4. Підключення LoRA
print("Підключаю LoRA...")
try:
    pipe.load_lora_weights(lora_path)
    print("LoRA успішно підключена та активована!")
except Exception as e:
    print(f"Помилка підключення LoRA: {e}")

def plot_difference_heatmap(image_normal, image_pathology, save_path="heatmap_report.png"):

    # 1. Завантаження та конвертація в ч/б (Grayscale)
    # Якщо передано шляхи, завантажуємо. Якщо об'єкти - використовуємо як є.
    if isinstance(image_normal, str):
        img1 = Image.open(image_normal).convert('L')
        img2 = Image.open(image_pathology).convert('L')
    else:
        img1 = image_normal.convert('L')
        img2 = image_pathology.convert('L')

    # Переконаємося, що розміри однакові
    img2 = img2.resize(img1.size)

    # 2. Перетворення в масиви чисел (NumPy arrays)
    arr1 = np.array(img1, dtype=np.int16) # int16, щоб уникнути переповнення при відніманні
    arr2 = np.array(img2, dtype=np.int16)

    # 3. Обчислення абсолютної різниці
    # abs(Норма - Патологія) покаже всі змінені пікселі
    diff = np.abs(arr1 - arr2)

    # Можна трохи підсилити контраст різниці для наочності (множник x2 або x3)
    diff_visual = np.clip(diff * 2, 0, 255).astype(np.uint8)

    # 4. Візуалізація (Створюємо фігуру з 3 частин)
    fig, axes = plt.subplots(1, 3, figsize=(15, 5))

    # Зображення 1: Норма
    axes[0].imshow(arr1, cmap='gray')
    axes[0].set_title("Normal (Prompt A)")
    axes[0].axis('off')

    # Зображення 2: Патологія
    axes[1].imshow(arr2, cmap='gray')
    axes[1].set_title("Opacity (Prompt B)")
    axes[1].axis('off')

    # Зображення 3: Теплова карта різниці
    # cmap='inferno' або 'hot' робить різницю яскравою (вогняною)
    im = axes[2].imshow(diff_visual, cmap='inferno')
    axes[2].set_title("Difference Map")
    axes[2].axis('off')

    # Додаємо шкалу кольорів
    plt.colorbar(im, ax=axes[2], fraction=0.046, pad=0.04)
```

```
    plt.suptitle(f"Аналіз впливу промпту на генерацію (Seed Fixed)", fontsize=16)
    plt.tight_layout()

    # Збереження для звіту
    #plt.savefig(save_path, dpi=300)
    #print(f"Графік збережено як {save_path}")
    #plt.show()

prompt_healthy = "Normal. The lungs are (clear:1.5). Chest x-ray."
prompt_pneumonia = "Lung opacity. (White opacity:1.5) lung base right. Chest x-ray."
neg_prompt = ""

# 2. Фіксація Seed
seed = 12345

print("Генерація здорових легень...")
generator = torch.Generator("cuda").manual_seed(seed)
image_healthy = pipe(
    prompt=prompt_healthy,
    negative_prompt=neg_prompt,
    num_inference_steps=35,
    guidance_scale=7,
    cross_attention_kwargs={"scale": 1},
    generator=generator
).images[0]

print("Генерація opacity...")
generator = torch.Generator("cuda").manual_seed(seed)
image_pneumonia = pipe(
    prompt=prompt_pneumonia,
    negative_prompt=neg_prompt,
    num_inference_steps=35,
    guidance_scale=7,
    cross_attention_kwargs={"scale": 1},
    generator=generator
).images[0]

print("Генерація завершена. Побудова теплової карти...")

try:
    plot_difference_heatmap(image_healthy, image_pneumonia, save_path="comparison_report.png")
except NameError:
    print("Помилка")
```

ДОДАТОК Д

Вміст файлу app.py

```
import streamlit as st
import torch
from diffusers import StableDiffusionXLPipeline, AutoencoderKL
from io import BytesIO
import gc
import random
from PIL.PngImagePlugin import PngInfo

# --- 1. СИСТЕМНІ ФУНКЦІЇ ---
def flush_memory():
    """Примусове очищення пам'яті."""
    try:
        gc.collect()
        torch.cuda.empty_cache()
        torch.cuda.ipc_collect()
    except:
        pass

def free_pipeline():
    """Повне видалення моделі з пам'яті перед завантаженням нової"""
    if "pipeline" in st.session_state and st.session_state.pipeline is not None:
        del st.session_state.pipeline
        st.session_state.pipeline = None
    flush_memory()

# --- 2. НАЛАШТУВАННЯ СТОРІНКИ ---
st.set_page_config(page_title="SynTheX-Ray", page_icon="□", layout="wide")

st.markdown("""
<style>
    div.block-container { padding-top: 1.5rem; padding-bottom: 0rem; }
    .stButton>button { width: 100%; border-radius: 5px; height: 3em; font-weight: bold; }
    div[data-testid="stVerticalBlock"] > div { gap: 0.8rem; }
    .stTextArea textarea { font-size: 16px !important; }
    section[data-testid="stSidebar"] { width: 270px; }
    /* Стиль для відображення активної моделі */
    .model-status {
        padding: 8px;
        background-color: #e8f5e9;
        border-left: 5px solid #4CAF50;
        border-radius: 4px;
        margin-bottom: 10px;
        color: #1b5e20;
    }
</style>
""", unsafe_allow_html=True)

# --- 3. ІНІЦІАЛІЗАЦІЯ СТАНУ ---
if "pipeline" not in st.session_state:
    st.session_state.pipeline = None
if "current_model_name" not in st.session_state:
    st.session_state.current_model_name = "None"
if "generated_image" not in st.session_state:
    st.session_state.generated_image = None
if "final_caption" not in st.session_state:
    st.session_state.final_caption = ""
```

```
if "image_metadata" not in st.session_state:
    st.session_state.image_metadata = None

# --- 4. ФУНКЦІЯ ЗАВАНТАЖЕННЯ ---
def load_model_manual(lora_path):
    # КРОК 1: Очищаємо пам'ять від попередньої моделі
    free_pipeline()

    base_model = "/kaggle/working/sdxl-base-1.0-fp16"
    vae_fix = "/kaggle/working/sdxl-vae-fp16-fix"
    status_msg = ""
    is_error = False

    try:
        vae = AutoencoderKL.from_pretrained(
            vae_fix,
            torch_dtype=torch.float16
        )

        pipe = StableDiffusionXLPipeline.from_pretrained(
            base_model,
            vae=vae,
            torch_dtype=torch.float16,
            variant="fp16",
            use_safetensors=True
        )

        pipe.to("cuda")

        try:
            pipe.load_lora_weights(lora_path)
            # Успіх: Нічого не записуємо в status_msg, воно залишається пустим ("")
        except Exception as e:
            # Помилка LoRA: Записуємо попередження
            status_msg = f"⚠ Base Model OK, LoRA failed: {e}"
            is_error = True

        return pipe, status_msg, is_error, lora_path.split('/')[-1]

    except Exception as e:
        # КРИТИЧНА ПОМИЛКА
        return None, f"❌ Critical Error: {e}", True, "Error"

# --- 5. САЙДБАР ---
with st.sidebar:
    st.header("⚙ Конфігурація системи")

    # === БЛОК ВИБОРУ МОДЕЛІ ===
    st.subheader("1. Вибір Моделі")

    # Список пресетів
    presets = {
        "Default (All Classes)": "/kaggle/input/iu-xray-lora-model/pytorch_lora_weights.safetensors",
        "Opacity": "/kaggle/input/iu-xray-lora-opacity/lora_opacity.safetensors",
        "Custom Path...": "custom"
    }

    model_choice = st.selectbox("Версія моделі:", list(presets.keys()))

    if model_choice == "Custom Path...":
```

```
        lora_path_input = st.text_input("Введіть шлях до .safetensors:", value="")
    else:
        lora_path_input = presets[model_choice]
        # Показуємо шлях, але робимо його неактивним для редагування
        st.text_input("Шлях:", value=lora_path_input, disabled=True)

    # Кнопка завантаження
    load_btn = st.button("🔄 Завантажити модель")

    if load_btn:
        with st.spinner("Звільнення пам'яті та завантаження нової моделі..."):
            pipe, msg, err, name = load_model_manual(lora_path_input)

            if pipe:
                # Успішне завантаження (або часткове з помилкою LoRA)
                st.session_state.pipeline = pipe
                st.session_state.current_model_name = name

                # Виводимо повідомлення тільки якщо воно не пусте (тобто була помилка LoRA)
                if msg:
                    st.warning(msg)
            else:
                # Критична помилка (модель не завантажилась взагалі)
                st.error(msg)

    # Відображення активної моделі
    if st.session_state.pipeline is not None:
        st.markdown(f"<div class='model-status'>Активна
модель:<br><b>{st.session_state.current_model_name}</b></div>", unsafe_allow_html=True)
    else:
        st.warning("⚠️ Модель не завантажена. Натисніть кнопку вище.")

    st.divider()

    # === НАЛАШТУВАННЯ ГЕНЕРАЦІЇ ===
    st.subheader("2. Параметри генерації")
    lora_scale = st.sidebar.slider("LoRA Strength", 0.0, 1.0, 0.8, 0.1,
                                   help="Інтенсивність впливу навченої моделі на результат")
    steps = st.slider("Inference Steps", 10, 50, 30,
                      help="Кількість кроків генерації, впливає на деталізацію")
    guidance = st.slider("Guidance Scale", 1.0, 15.0, 7.5, 0.1,
                         help="Точність слідування тексту")

    seed_input = st.text_input("Seed (Optional):", value="42",
                               help="Залиште пустим для випадкової ренерації")

# --- 6. ОСНОВНИЙ ЕКРАН ---
st.title("🩻 SynTheX-Ray: Generative AI System")

# --- СТВОРЕННЯ МАКЕТУ (LAYOUT) ---
col1, col2 = st.columns([1, 1.3])

# --- ЛІВА КОЛОНКА (ВВІД) ---
with col1:
    st.subheader("📝 Клінічний опис")

    examples = {
    "Custom (Власний опис)": "",
    "Normal (Healthy)": (
        "Normal. (The lungs are clear:1.5) bilaterally. "
```

```
        "There is no focal consolidation, pleural effusion, or pneumothorax. "
        "The cardiomediastinal silhouette is within normal limits. "
        "The osseous structures are unremarkable. Chest x-ray."
    ),
…
…
}

    selected_example = st.selectbox("Оберіть шаблон:", list(examples.keys()))
    default_text = examples[selected_example] if selected_example != "Custom (Власний опис)" else ""

    prompt_text = st.text_area("Prompt (English):", value=default_text, height=140)
    neg_prompt = st.text_area("Negative Prompt (Optional):", value="", height=70)

    # Кнопка активна тільки якщо модель завантажена
    is_ready = st.session_state.pipeline is not None
    generate_btn = st.button("Згенерувати", type="primary", disabled=not is_ready)

    if not is_ready:
        st.info("👈 Спочатку завантажте модель у меню зліва")

# --- ПРАВА КОЛОНКА (ПІДГОТОВКА) ---
with col2:
    st.subheader("Результат")
    result_container = st.empty()

# --- 7. ЛОГІКА ТА ВІДОБРАЖЕННЯ ---

# Крок 1: Якщо є збережена картинка в стані - показуємо її відразу
if st.session_state.generated_image is not None:
    with result_container.container():
        st.image(st.session_state.generated_image, caption=st.session_state.final_caption, width=680)

        # Кнопка скачування
        buf = BytesIO()
        if st.session_state.image_metadata:
            st.session_state.generated_image.save(buf, format="PNG", pnginfo=st.session_state.image_metadata)
        else:
            st.session_state.generated_image.save(buf, format="PNG")
        byte_im = buf.getvalue()

        st.download_button(
            label="💾 Завантажити зображення (+Metadata)",
            data=byte_im,
            file_name=f"synthetic_xray_{random.randint(1000,9999)}.png",
            mime="image/png"
        )

# Крок 2: Обробка натискання кнопки
if generate_btn and st.session_state.pipeline is not None:
    # Очищаємо попередній результат візуально перед новою генерацією
    result_container.empty()

    # Підготовка змінних
    process_seed = None
    if seed_input.strip() == "":
        process_seed = random.randint(0, 2**32 - 1)
    else:
        try:
            process_seed = int(seed_input)
```

```
    except ValueError:
        st.error("❌ Invalid Seed")
        st.stop()

generator = torch.Generator("cuda").manual_seed(process_seed)
flush_memory()

final_prompt = prompt_text
# Ваша логіка додавання тексту
if "chest x-ray" not in final_prompt.lower():
    final_prompt = final_prompt + "Chest x-ray."

st.session_state.final_caption = final_prompt

# ГЕНЕРАЦІЯ
try:
    with col2:
        with st.spinner("Generating Image..."):
            # Використовуємо pipeline зі стану сесії
            image = st.session_state.pipeline(
                prompt=final_prompt,
                negative_prompt=neg_prompt,
                num_inference_steps=steps,
                guidance_scale=guidance,
                cross_attention_kwargs={"scale": lora_scale},
                generator=generator
            ).images[0]

            # Метадані
            metadata = PngInfo()
            metadata.add_text("Description", final_prompt)
            metadata.add_text("Seed", str(process_seed))
            metadata.add_text("Model", st.session_state.current_model_name)
            metadata.add_text("Software", "SyntheX-Ray AI")

            # Оновлюємо стан
            st.session_state.generated_image = image
            st.session_state.image_metadata = metadata

            # Видаляємо локальну копію
            del image

            # Примусове перезавантаження скрипта
            st.rerun()

except Exception as e:
    st.error(f"⚠ Error: {e}")
    flush_memory()
    # Якщо сталася помилка пам'яті - скидаємо пайплайн для безпеки
    if "out of memory" in str(e).lower():
        free_pipeline()
finally:
    flush_memory()
```

ДОДАТОК Е

Вміст файлу system_launch.py

```
!pip install -q streamlit pyngrok diffusers transformers accelerate
!pip install -U "protobuf<4" –quiet

from kaggle_secrets import UserSecretsClient
from huggingface_hub import login

user_secrets = UserSecretsClient()
hf_token = user_secrets.get_secret("HF_TOKEN")

login(token=hf_token)

!hf auth whoami

from huggingface_hub import snapshot_download
import os

# 1. Налаштування
base_model_repo = "stabilityai/stable-diffusion-xl-base-1.0"
base_model_dir = "/kaggle/working/sdxl-base-1.0-fp16"

# Створення папки (обов'язково!)
os.makedirs(base_model_dir, exist_ok=True)

print(f"⬇ Завантажую {base_model_repo} у {base_model_dir}...")

# 2. Завантаження БАЗОВОЇ МОДЕЛІ
snapshot_download(
    repo_id=base_model_repo,
    local_dir=base_model_dir,
    local_dir_use_symlinks=False, # Важливо для Kaggle/Colab
    # Качаємо: конфіги (json), fp16 ваги (safetensors), текстові файли (ліцензії/токени)
    allow_patterns=["*.json", "*.fp16.safetensors", "*.txt", "*.model"],
    ignore_patterns=["*.bin", "*.pth", "*.onnx", "*.png"]
)

print("✅ Базова модель завантажена!")

# --- ДОДАТОК: НЕ ЗАБУДЬТЕ ПРО VAE ---
# Оскільки ми вирішили використовувати madebyollin/sdxl-vae-fp16-fix,
# його теж бажано завантажити локально, щоб не качати щоразу при старті.

vae_repo = "madebyollin/sdxl-vae-fp16-fix"
vae_dir = "/kaggle/working/sdxl-vae-fp16-fix"
os.makedirs(vae_dir, exist_ok=True)

print(f"⬇ Завантажую VAE Fix...")
snapshot_download(
    repo_id=vae_repo,
    local_dir=vae_dir,
    local_dir_use_symlinks=False,
    allow_patterns=["*.json", "*.safetensors"]
)

print("✅ VAE завантажено!")

import subprocess
```

```
from pyngrok import ngrok
from kaggle_secrets import UserSecretsClient
import time

# 1. Авторизація ngrok
user_secrets = UserSecretsClient()
try:
    ngrok_token = user_secrets.get_secret("NGROK_TOKEN")
    ngrok.set_auth_token(ngrok_token)
except:
    print("NGROK_TOKEN not found!")

# 2. Вбиваємо старі процеси
print("Перезапуск сесії: завершення старих процесів...")

# Вбиваємо процес Python-обгортки ngrok
ngrok.kill()

# Вбиваємо будь-які системні процеси ngrok та streamlit
os.system("pkill ngrok")
os.system("pkill streamlit")

# Даємо системі 2 секунди, щоб звільнити порти
time.sleep(2)

# 3. Відкриваємо тунель на порт 8501
try:
    public_url = ngrok.connect(8501).public_url
    print(f"🔗 CLICK HERE TO OPEN APP: {public_url}")
except Exception as e:
    print(f"❌ Помилка ngrok: {e}")

# 4. Запускаємо Streamlit у фоновому режимі
process = subprocess.Popen(["streamlit", "run", "app.py"])

print(f"✅ Streamlit запущено у фоні (PID: {process.pid})")
```