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

До захисту допущено:
В. о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи і методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Інтелектуальна система підвищення роздільної здатності зображень на основі використання нейронних мереж SRGAN»

Виконав:
студент IV курсу, групи КІ-11
Головачов Артем Сергійович _____

Керівник:
професор кафедри ШІ, д.т.н., професор,
Синєглазов Віктор Михайлович _____

Консультант з економічного розділу:
доцент кафедри ЕК, к.е.н., доцент,
Рощина Надія Василівна _____

Консультант з нормоконтролю:
фахівець першої категорії кафедри ШІ, к.т.н., доцент,
Комариста Богдана Миколаївна _____

Рецензент:
доцент кафедри АКІК Державного
університету «Київський авіаційний інститут», к. т. н,
Василенко Микола Павлович _____

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

Київ – 2025 року

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

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В. о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«15» січня 2025 р.

ЗАВДАННЯ
на дипломну роботу студенту
Головачову Артему Сергійовичу

1. Тема роботи «Інтелектуальна система підвищення роздільної здатності зображень на основі використання нейронних мереж SRGAN», керівник роботи Синєглазов Віктор Михайлович, д.т.н, професор, затверджені наказом по НН ІПСА від «26» травня 2025 р. № 1759-с.

2. Термін подання студентом роботи «09» червня 2025 року.

3. Вихідні дані до роботи: модифікований набор аерофотознімків DOTA

4. Зміст роботи: дослідження предметної області, аналіз існуючих методів, побудова модифікованої мережі, аналіз результатів, функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): архітектури моделей, блок-схеми, приклади вхідних даних, результати.

6. Консультанти розділів роботи

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

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

Календарний план

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

Студент Артем ГОЛОВАЧОВ

Керівник Віктор СИНЄГЛАЗОВ

РЕФЕРАТ

Дипломна робота: 117 с., 24 рис., 10 табл., 37 посилань, 1 додаток.

ГЛИБОКЕ НАВЧАННЯ, ГЕНЕРАТИВНО ЗМАГАЛЬНІ МЕРЕЖІ, ПОДВІЙНИЙ ДИСКРИМІНАТОР, АЕРОФОТОЗНІМКИ, НЕЙРОННА МЕРЕЖА, ЗБІЛЬШЕННЯ РОЗДІЛЬНОЇ ЗДАТНОСТІ.

У цій роботі було розглянуто задачу збільшення роздільної здатності супутникових знімків. Спочатку було розглянуто варіанти існуючих методів збільшення роздільної здатності. Далі було запропоновано модифікований метод на базі генеративно змагальних мереж для заданого об’єкту дослідження. Хоча цей метод розроблений саме для збільшення роздільної здатності супутникових знімків, його також можна використовувати у багатьох інших сферах, наприклад медицина, безпека та багато інших.

Об’єкт дослідження – збільшення роздільної здатності супутникових знімків.

Предмет дослідження – генеративно змагальна мережа та її застосування для задачі збільшення роздільної здатності.

Мета роботи – розробка модифікованої генеративно змагальної мережі для збільшення роздільної здатності супутникових знімків.

Результати дослідження демонструють реалістичне відновлення, як для тренувального так і для тестового наборів даних. Хоча деякі зображення з великою кількістю деталей є доволі складними для відновлення, у загальному варіанті мережа видає дуже гарні результати. Новизна роботи полягає у модифікованій структурі генеративно змагальної мережі, а саме подвійній будові дискримінтаора, в якій окремі мережі фокусуються на локальних та глобальних ознаках.

ABSTRACT

Bachelor’s thesis: 117 p., 24 figures, 10 tables, 37 references, 1 appendix.

DEEP LEARNING, GENERATIVE ADVERSARIAL NETWORKS, DUAL DISCRIMINATOR, AERIAL IMAGERY, NEURAL NETWORKS, SUPER-RESOLUTION.

This paper considers the problem of increasing the resolution of satellite images. First, existing methods for increasing resolution were considered. Next, a modified method based on generative adversarial networks was proposed for the given research object. Although this method was developed specifically to increase the resolution of satellite images, it can also be used in many other areas, such as medicine, security, and many others.

The object of research is to increase the resolution of satellite images.

The subject of the study is a generative adversarial network and its application to the task of increasing resolution.

The purpose of the work is to develop a modified generative adversarial network to increase the resolution of satellite images.

The results of the study demonstrate realistic restoration for both training and test datasets. Although some images with a large amount of detail are quite difficult to restore, in general, the network produces very good results. The novelty of the work lies in the modified structure of the generative adversarial network, namely the dual structure of the discriminator, in which separate networks focus on local and global features.

ЗМІСТ

ВСТУП

Сучасний[1] розвиток технологій обробки зображень став важливим фактором в розвитку багатьох галузей науки та техніки. У численних прикладних областях, таких як медицина, астрономія, відеоспостереження, робототехніка, а також у сфері комп'ютерної графіки та віртуальної реальності, часто постає проблема низької роздільної здатності зображень, що суттєво обмежує їхнє використання для аналізу та прийняття рішень. Тому розробка ефективних методів підвищення роздільної здатності зображень є актуальним і важливим завданням у сучасних дослідженнях.

Підвищення роздільної здатності зображень є процесом реконструкції зображень з низькою роздільною здатністю у зображення високої якості з детальними і чіткими елементами. Раніше для цієї мети використовувались класичні методи, такі як інтерполяція або фільтрація, які хоча й давали певні результати, але не здатні були ефективно відновлювати деталі в зображеннях. З появою методів глибинного навчання, особливо за допомогою нейронних мереж, цей процес набув нового імпульсу. Одним із найбільш перспективних підходів є використання генеративних змагальних мереж (GAN, Generative Adversarial Networks).

Генеративні змагальні мережі – це потужний інструмент штучного інтелекту, який складається з двох нейронних мереж: генератора і дискримінатора. Завдяки своїй здатності створювати реалістичні зображення з шуму або інших зображень, GAN швидко здобули популярність у різних сферах. Спочатку використовувались для генерації зображень на основі

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

навчання з великої кількості прикладів, однак з часом ці мережі були адаптовані для розв’язання задачі підвищення роздільної здатності зображень.

Ця робота присвячена детальному аналізу використання GAN для підвищення роздільної здатності зображень. В рамках дослідження будуть розглянуті ключові принципи роботи генеративних змагальних мереж, зокрема їх архітектури та методи навчання. Особливу увагу буде приділено сучасним підходам до підвищення роздільної здатності, зокрема, аналізу ефективності різних архітектур, а також дослідженню їх застосування в реальних умовах. Крім теоретичного аналізу, у роботі буде проведено експериментальне порівняння моделей для оцінки їх продуктивності на практичних задачах підвищення роздільної здатності зображень.

У результаті дослідження ми прагнемо не лише оцінити переваги та недоліки застосування GAN у задачах підвищення роздільної здатності, але й визначити напрямки для подальших удосконалень у цій галузі. Оскільки технології глибинного навчання, зокрема генеративні моделі, продовжують розвиватися, наш аналіз може допомогти у виявленні нових можливостей для покращення якості зображень у різних галузях, від медицини до кінематографії.

РОЗДІЛ 1
ОГЛЯД ПІДХОДІВ ДО ЗБІЛЬШЕННЯ РОЗДІЛЬНОЇ ЗДАТНОСТІ

1.1 Сфера дослідження

Завдання збільшення роздільної здатності зображення – це інноваційна технологія обробки зображень, яка має на меті підвищити роздільну здатність і якість цифрових зображень. Цей метод передбачає реконструкцію зображення з одного або декількох вхідних зображень з низькою роздільною здатністю, ефективно підвищуючи рівень деталізації, різкості та візуальної інформації.

Концепція збільшення роздільної здатності з'явилася наприкінці 1970-х - на початку 1980-х років, спочатку розвиваючись з обробки сигналів [1] і досліджень комп'ютерного зору. Ранні підходи в основному базувалися на статистичних методах і техніці математичної інтерполяції. Однак з появою машинного навчання та штучного інтелекту ця сфера зазнала кардинальної трансформації.

Протягом багатьох років було розроблено кілька фундаментальних підходів до збільшення роздільної здатності:

Традиційні методи використовують математичну інтерполяцію для оцінки додаткових значень пікселів. Ці методи, такі як бікубічна [2] або білінійна [3] інтерполяція, оцінюють нові пікселі на основі значень навколишніх пікселів, створюючи ефект масштабування.

Багатокадровий підхід використовує декілька зображень однієї і тієї ж сцени з низькою роздільною здатністю для реконструкції зображення. Аналізуючи незначні варіації та зсуви між зображеннями, алгоритми можуть виділити та об'єднати додаткові деталі, невидимі на одному кадрі.

Сучасні методи були революціонізована завдяки глибокому навчанню, зокрема, згортковим нейронним мережам. Ці передові методи використовують навчені нейронні мережі для вивчення складних моделей реконструкції

зображень, часто досягаючи надзвичайно реалістичного та детального покращення зображень.

1.2 Актуальність

Проаналізуємо актуальність обраного об’єкту дослідження у контексті сучасного світу. Збільшення роздільної здатності застосовується у широкому переліку сфер. Розглянемо приклади застосування збільшення роздільної здатності у різних сферах.

У медичній сфері [4] лікарі та дослідники можуть отримувати набагато детальніші зображення з медичних сканів за допомогою менш інвазивних процедур. Наприклад, МРТ і КТ-сканування отримують величезну користь від цієї технології. Вона дозволяє медичним працівникам бачити найдрібніші деталі, які вони могли пропустити раніше, що потенційно підвищує точність діагностики. Ця технологія особливо цінна в таких сферах, як мікроскопічна візуалізація, де розуміння деталей на клітинному рівні може мати вирішальне значення для діагностики і досліджень.

Ще один приклад – супутникові та геопросторові зображення [5], завдяки яким екологи, містобудівники та дослідники тепер можуть отримувати набагато детальніші зображення земної поверхні. Ця технологія допомагає відстежувати зміни клімату, контролювати розвиток міст, прогнозувати врожайність сільськогосподарських культур і навіть підтримувати зусилля з ліквідації наслідків стихійних лих. Те, що раніше вимагало дорогого супутникового обладнання з високою роздільною здатністю, тепер можна досягти завдяки покращенню зображень.

У сфері розваг та медіа такий метод використовують для реставрації старих фільмів, покращення графіки у відеоіграх та підвищення якості потокового контенту. Старі фільми можна повернути до життя з покращеною

чіткістю, а творці контенту можуть підвищити якість відзнятого матеріалу, щоб відповідати сучасним стандартам відображення.

Системи безпеки та відеоспостереження [6] стали більш досконалими завдяки можливості збільшувати роздільну здатність зображення. Розпізнавання обличь, ідентифікація номерних знаків і моніторинг натовпу – все це виграло від можливості витягувати більш детальну інформацію із зображень і відеоматеріалів.

Разом ці застосування ілюструють, чому технологія збільшення роздільної здатності стала не просто корисною, а фактично незамінною в широкому діапазоні галузей. Оскільки наша залежність від інформації високої якості невпинно зростає, вміння досягати максимальної деталізації та чіткості в системах візуалізації стає критично важливою технологією, що суттєво впливає на розвиток науки, медицини, безпеки та повсякденного життя.

Сучасні підходи до збільшення роздільної здатності використовують генеративні змагальні мережі (GAN) [7] та моделі на основі трансформерів [8]. Ці вдосконалені алгоритми можуть генерувати високо деталізовані зображення, іноді додаючи реалістичні текстури та деталі, яких не було у вихідному зображенні.

Ця галузь продовжує стрімко розвиватися, а дослідники постійно розробляють більш досконалі алгоритми, які розширюють межі можливого у покращенні та реконструкції зображень.

У даній роботі буде розглянуто збільшення роздільної здатності зображень саме на прикладі супутникових знімків. Розглянемо детальніше наш об’єкт дослідження.

1.3 Об’єкт дослідження

Супутникові знімки сьогодні є одним з найкращих інструментів для спостереження та розуміння нашої планети. Вони надають унікальну перспективу, яка змінила те, як ми відстежуємо зміни довкілля, плануємо ресурси та реагуємо на глобальні проблеми.

Супутникові фотографії - це електронні зображення земної поверхні, зроблені високо розташованими супутниками. Складні космічні апарати оснащені науковими датчиками та камерами, які кількісно вимірюють електромагнітне випромінювання, відбите або випромінюване земною поверхнею, атмосферою та океанами. Інформація передається на наземні станції, де вона перетворюється на видимі зображення для вивчення та аналізу науковцями, дослідниками та представниками різних галузей промисловості.

На відміну від звичайної аерофотозйомки з літака, супутникові знімки дають рівномірне, широке покриття великих територій і можуть збиратися багаторазово протягом надзвичайно тривалих періодів часу. Це дозволяє відстежувати закономірності і тенденції, які було би неможливо отримати іншими шляхами.

Існує кілька різних типів супутникових знімків, кожен з яких використовується для різних цілей і застосувань. Розглянемо декілька розповсюджених варіантів.

Оптичні зображення є найбільш інтуїтивно зрозумілими, оскільки вони фіксують видиме світло та ближнє інфрачервоне випромінювання. Однак оптична зйомка має свої недоліки - для отримання якісних знімків оптимально використовувати її вдень або при ясному небі.

Інфрачервоні знімки функціонують по-іншому, фіксуючи теплові сигнатури та теплове випромінювання, що випромінюється з поверхні Землі. Цей тип знімків особливо корисний для температурного картографування, нічних знімків і моніторингу погоди. Інфрачервоні сенсори іноді проникають

крізь атмосферні умови, які приховують оптичні зображення, що робить їх особливо корисними для вивчення клімату та моніторингу теплових змін у часі.

Радіолокаційна технологія з синтезованою апертурою дозволяє створювати зображення, незважаючи на погоду чи денне світло. На відміну від оптичних систем, які покладаються на сонце, радар активно посилає сигнали в бік Землі і обчислює сигнали, що повертаються. Це підвищує ефективність радіолокаційних знімків у виявленні змін на поверхні, моніторингу руху та спостереженні за ділянками, які зазвичай затінені хмарами.

Супутникові знімки також значно різняться за роздільною здатністю, яка є мірою деталізації кожного зображення. Зображення з високою роздільною здатністю містять пікселі розміром, як правило, менше одного метра, і підходять для містобудування, моніторингу інфраструктури та точного картографування. Знімки середньої роздільної здатності, від 10 до 100 метрів, забезпечують компроміс між деталізацією та охопленням географічної території і підходять для регіонального моніторингу та картографування землекористування. Знімки низької роздільної здатності відображають великі регіони на одному зображенні з пікселями розміром понад 100 метрів, що ідеально підходить для міжнародного спостереження та великомасштабних кліматичних досліджень.

Застосування супутникових знімків охоплює майже всі сфери, які отримують користь від повітряного огляду процесів і змін на Землі. Моніторинг довкілля є одним з соновних застосувань, оскільки вчені покладаються на супутникові знімки для відстеження темпів вирубки лісів, спостереження за таненням полярних льодовиків і моніторингу стану екосистем по всьому світу. За допомогою цієї технології вчені можуть виявляти зміни в навколишньому середовищі, на реєстрацію яких за допомогою наземних спостережень можуть піти роки.

Під час стихійних лих супутникові знімки стають критично важливим помічником для рятувальників. Ця технологія дозволяє негайно оцінити

збитки після землетрусів, ураганів, повеней чи лісових пожеж, полегшуючи проведення рятувальних робіт і розподіл ресурсів. Рятувальники можуть швидко створювати карти пошкодженої місцевості, визначати безпечні шляхи і масштаби пошкоджень, що потенційно може врятувати десятки життів завдяки швидшому реагуванню і координації дій.

Міські планувальники все частіше звертаються до супутникових знімків для моніторингу зростання міст, планування розвитку інфраструктури та ефективного управління землекористуванням. Супутникові технології дозволяють відстежувати еволюцію міст у часі, визначати придатні для розвитку території і допомагають оцінити успішність стратегій планування. Оскільки урбанізація прискорюється в усьому світі, супутникові знімки забезпечують широку перспективу, необхідну для сталого розвитку міст, що зростають.

Дослідження клімату та прогнозування погоди значною мірою покладаються на використання супутникових даних для збору та обробки інформації. Метеорологи відстежують штормові системи, спостерігають за станом атмосфери і надають важливі дані для моделей прогнозування погоди. Дослідники клімату використовують супутникові дані дальнього радіусу дії для виявлення світових тенденцій, відстеження індикаторів зміни клімату і розробки більш точних кліматичних моделей.

Національна безпека і оборона використовує супутникові знімки для розвідки, моніторингу кордонів та оцінки загроз. Ця технологія надає важливу розвідувальну інформацію, водночас запобігаючи потенційній небезпеці для персоналу. Військовослужбовці та розвідувальні служби можуть відстежувати діяльність на великих відстанях і спостерігати за змінами, які можуть свідчити про загрози безпеці.

Існує кілька практичних аспектів, які визначають використання супутникових знімків і доступ до них. Компроміси щодо роздільної здатності є фундаментальним питанням, оскільки вища роздільна здатність, як правило, вимагає меншого покриття і менш частого перегляду. Організації повинні

знайти компроміс між потребою в деталях і вимогами до широкого покриття і частого оновлення.

Тому аналіз технології збільшення роздільної здатності на прикладі супутникових знімків є актуальним об’єктом дослідження. У даній роботі буде розглянуто саме супутникові оптичні знімки невеликої роздільної здатності та механізми збільшення їхньої якості для відновлення деталей знімку. Як було сказано раніше, така технологія може дозволити робити знімки меншої якості, які потім відновлюються до високої. Такі знімки є набагато легші у отриманні, що дає змогу збільшити об’єм знімків отриманих за період часу.

1.4 Короткі відомості

Для виконання даної роботи пропонується використати генеративно змагальні мережі, які зарекомендували себе як ефективний інструмент для задач відновлення зображень та збільшення роздільної здатності.

Генеративні змагальні мережі (GAN), запропоновані І. Гудфеллоу [9], стали революційною парадигмою для генеративного моделювання. На відміну від звичайних генеративних методів, які безпосередньо оптимізують функції правдоподібності або моделюють розподіли ймовірностей у явному вигляді, GAN використовують новий змагальний процес навчання, який виявився успішним для синтезу високоякісних синтетичних даних у широкому діапазоні областей.

Новизна GAN полягає в їхній змагальній структурі: генератор і дискримінатор нейронної мережі навчаються одночасно в мінімаксній грі. Генератор намагається створити синтетичні зразки, які неможливо відрізнити від реальних даних, а дискримінатор намагається відрізнити реальні зразки від синтезованих. Обидві мережі ітеративно покращують одна одну на

конкурентній основі, що призводить до все більш реалістичних синтетичних результатів.

Однією з найвагоміших переваг GAN є те, що вони можуть неявно вивчати складні, багатовимірні розподіли даних, не вимагаючи явного оцінювання щільності. Традиційні підходи, такі як варіаційні автокодувальники та авторегресійні моделі, або не справляються з високорозмірними розподілами ймовірностей, або роблять спрощені припущення. GAN обходять ці обмеження, навчаючись генерувати вибірки з цільового розподілу безпосередньо за допомогою навчання в змагальному режимі.

Процедура навчання змушує GAN вивчати значущі репрезентації та ознаки, не вимагаючи при цьому маркованих даних. Дискримінатор повинен ідентифікувати тонкі ознаки, за якими реальні та згенеровані зразки відрізняються, а генератор повинен навчитися точно відтворювати ці ознаки. Ця процедура змушує обидві мережі вивчати значущі представлення ознак, які можуть бути використані для подальших завдань.

1.5 Сфери використання GAN

GAN є гнучким інструментом, а саме тому може бути використаним у різних, та навіть не пов’язаних між собою сферах людської діяльності. Розглянемо детальніше приклади його використання.

У тих сферах, де процес збору даних є дорогим, тривалим або обмежений конфіденційністю, GAN є потужним інструментом, маючи можливість генерувати синтетичні зразки для доповнення навчальних даних. Цей аспект є особливо корисним у таких сферах.

1. Медична візуалізація [10]: використання генеративних мереж з метою генерації синтетичних даних для рідкісних випадків у

медицині, щоб збільшити датасети для навчання нейронних мереж.

2. Виявлення рідкісних подій [11]. у незбалансованих за класами наборах даних, GAN можна використовувати для створення більшої кількості прикладів класів менших об’ємів, щоб зробити модель більш надійною та менш упередженою.

Також GAN є важливим механізмом для створення синтетичних наборів даних, які зберігають статистичні властивості та корисність оригінальних даних, не розкриваючи окремих записів.

1. GAN з диференційованою конфіденційністю [12]: існують архітектури, які накладають гарантії конфіденційності під час навчання, щоб згенеровані зразки не могли бути використані для виведення фактів про конкретні навчальні приклади.
2. Синтетичні електронні медичні картки [13]: впроваджено нові методи для створення реалістичних електронних медичних карток для досліджень без порушення конфіденційності пацієнтів.
3. Синтез фінансових даних [14]: GAN використовуються для синтезу даних про фінансові транзакції, що зберігає статистичні зв'язки, захищаючи при цьому конфіденційні дані клієнтів.

Ще одне застосування, в якому GAN здійснили революцію – галузь перекладу "зображення в зображення".

1. Трансформація модальності [15]: GAN дозволяють синтезувати одну модальність зображення з іншої, наприклад, синтезувати ПЕТ-сканування з МРТ-зображень, що може запобігти проведенню декількох дорогих або інвазивних медичних тестів.
2. Перенесення стилю [16]: архітектури на кшталт StyleGAN дозволяють контролювати редагування візуальних атрибутів, уможливлюючи застосування його, від віртуальної примірки одягу, до змінення віку в криміналістиці.

Найголовніше для нашого дослідження, GAN досягли найсучасніших результатів у вирішенні завдань пов’язаних з роздільною здатністю, коли зображення високої роздільної здатності генеруються з вхідних даних низької роздільної здатності.

1. Збільшення роздільної здатності [17]: мережі створюють зображення з високою роздільною здатністю порівняно з традиційними методами, з відновленими дрібними текстурними деталями, які інакше були б втрачені.
2. Розфарбовування та реставрація зображень [18]: GAN чудово доповнюють відсутні або пошкоджені частини зображень, вивчаючи контекстні та структурні зв'язки в даних.
3. 3D-реконструкція [19]: нещодавні дослідження також розширили можливості GAN до 3D-моделювання, що дозволяє реконструювати тривимірні об'єкти з одного 2D-зображення.

1.6 Обмеження альтернативних підходів

Хоча інші підходи можуть забезпечити виконання схожих завдань, але все одно саме мережі GAN дозволяють отримувати найкращі результати. Наприклад VAE [20] забезпечують стабільне навчання і явну оцінку ймовірності, але генерують більш розмиті результати, ніж GAN, через піксельні втрати при реконструкції, які вони використовують. Гаусові припущення в латентному просторі також обмежують їхню здатність до складного, мультимодального моделювання розподілу. Авторегресійні підходи [21], генерують чіткі зображення, але мають повільні процеси дискретизації через послідовний процес генерації. Вони також мають труднощі з фіксацією довготривалих залежностей у високорозмірних даних. Нещодавні дифузійні моделі [22] продемонстрували гарні результати, але вони вимагають декількох

кроків дискретизації, що вимагає великих обчислювальних затрат, на відміну від GAN, які роблять вибірку за один прохід вперед.

Незважаючи на свою фундаментальну природу, GAN продовжують мати проблеми [23], які спонукають до постійних досліджень. Розглянемо деякі приклади проблем, які можуть виникнути при навчанні та використанні моделей.

Кількісна оцінка ефективності роботи GAN залишається складним завданням. Такі показники оцінки, як Inception Score та Frechet Inception, є корисними проміжними показниками, але не обов'язково відображають якість сприйняття. Покращення системи оцінювання залишається активною темою для досліджень.

Також незважаючи на значне покращення, навчання GAN все ще може бути нестабільним у деяких випадках. Такі методи, як градієнтний штраф, спектральна нормалізація та адаптивне підсилення дискримінатора, пом'якшують ці проблеми, але вимагають точного налаштування гіперпараметрів.

Отже значення GAN полягає в їхній унікальній здатності генерувати високоякісні синтетичні дані, вивчаючи складні розподіли без явного моделювання. Їх використання в кожній галузі – від охорони здоров'я до мистецтва, від комп'ютерного зору до синтезу даних із збереженням конфіденційності - свідчить про їхню фундаментальну цінність для сучасних комп'ютерних досліджень.

Оскільки архітектури GAN продовжують розвиватися завдяки таким досягненням, як механізми самоуваги, адаптивні методи нормалізації та гібридні підходи, їхній основний внесок у галузі машинного навчання буде лише зростати.

Для виконання заданого завдання існують різні генеративно змагальні мережі, які можна використати, але в роботі буде проведено дослідження SRGAN(Super Resolution Generative Adversarial Network) [24]. Топологію та процес роботи буде розглянуто в наступних розділах.

1.7 Постановка задачі

Задача збільшення роздільної здатності зображень полягає в перетворенні зображення низької роздільної здатності (LR) у відповідне зображення високої роздільної здатності (HR) зі збереженням змістовної інформації та покращенням якості деталей.

Нехай $I^{LR} \in \mathbb{R}^{h \times w \times c}$ – вхідне зображення низької роздільної здатності, де h та w – відповідно висота та ширина зображення, а c – кількість каналів. Мета полягає у відновленні зображення високої роздільної здатності $I^{HR} \in \mathbb{R}^{\alpha h \times \alpha w \times c}$, де $\alpha > 1$ – коефіцієнт масштабування.

Формально, завдання можна представити, як пошук відображення $\mathcal{F}$:

$$\mathcal{F}: I^{LR} \mapsto I^{HR}.$$

Для реалізації цього відображення використовується архітектура генеративної змагальної мережі. На вхід генератору G подаються зображення низької роздільної здатності I^{LR}, на виході ми отримуємо зображення, з підвищеною роздільною здатністю I^{SR}.

Для реалізації відображення $\mathcal{F}$ використовується архітектура генеративної змагальної мережі, що складається з генератора G та дискримінатора D. Генератор G прагне створювати зображення, що візуально не відрізняються від реальних зображень високої роздільної здатності, в той час як дискримінатор D намагається відрізнити згенеровані зображення I^{SR} від реальних I^{HR}.

Для підготовки тренувальних даних необхідний процес зменшення роздільної здатності. У даній роботі це завдання буде виконано за допомогою бікубічної інтерполяції [37].

Отже, прописавши детально постановку задачі, можна переходити до описання модифікацій, які буде внесено у будову генеративно змагальної

мережі, для підвищення ефективності та її результатів збільшення роздільної здатності зображень.

Для побудови мережі та її навчання буде використано метрику MSE. Mean Squared Error (MSE) – вимірює середню квадратичну різницю між значеннями пікселів створеного зображення та основною правдою. Менше значення означає меншу різницю між зображеннями. MSE розраховують за наведеною формулою (1.1).

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(I_{SR}(i,j) - I_{GT}(i,j)\right)^2, \quad (1.1)$$

де I_{SR} – зображення з підвищеною роздільною здатністю;

I_{GT} – оригінальне зображення високої роздільної здатності;

m, n – розмірності зображення.

Для аналізу якості результатів підвищення роздільної здатності використовують такі метрики.

Peak Signal-to-Noise Ratio (PSNR) – вимірює співвідношення між максимально можливою потужністю сигналу та потужністю шуму. Вимірюється в децибелах (dB) та більші значення означають кращу якість. PSNR розраховують за наведеною формулою (1.2).

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right), \quad (1.2)$$

де MSE – середня квадратична помилка;

MAX_I – максимальне можливе значення пікселя для зображення.

Structural Similarity Index (SSIM) – вимірює подібність між двома зображеннями на основі яскравості, контрасту та структури. Знаходиться в проміжку -1 та 1, значення ближчі до 1 означають більшу подібність. SSIM розраховується за формулою (1.3).

$$\mathrm{SSIM}(x,y) = \frac{\left(2\mu_x\mu_y + (k_1L)^2\right)\left(2\sigma_{xy} + (k_2L)^2\right)}{\left(\mu_x^2 + \mu_y^2 + (k_1L)^2\right)\left(\sigma_x^2 + \sigma_y^2 + (k_2L)^2\right)}, \quad (1.3)$$

де μ_x, μ_y – середні значення для x та y відповідно;

σ_x^2, σ_y^2 – дисперсія для x та y відповідно;

σ_{xy} – коваріація;

k_1, k_2 – константи за замовчуванням відповідно 0.01 та 0.03;

L – динамічний діапазон значень пікселів.

Висновки до розділу 1

У першому розділі було проведено дослідження предметної області задачі збільшення роздільної здатності зображень. Розглянуто її актуальність та сучасні методи її вирішення. Було встановлено, що дана задача є актуальною у сучасності та має широке практичне застосування у різних галузях – від медицини та безпеки, закінчуючи супутниковими знімками.

Найсучаснішим методом вирішення даної проблеми є використання GAN – генеративно змагальні мережі. Використання такої архітектури до збільшення роздільної здатності зображень забезпечує реалістичну генерацію текстур та дрібних деталей, зберігаючи структуру глобального зображення.

Провівши аналіз альтернативних методів збільшення роздільної здатності зображення було визначено, що саме GAN мають найкращий баланс між якістю та продуктивністю. Але незважаючи на їхні переваги, такі мережі все ще мають низку обмежень та недоліків, які залишаються актуальними об’єктами досліджень.

Метою даної роботи є проведення модифікації існуючого підходу для збільшення роздільної здатності зображень – SRGAN(Super Resolution Generative Adversarial Network). У наступних розділах буде розглянуто її

архітектуру та проблеми, з якими вона стикається. Також буде запропоновано модифікації, які можна зробити для їх вирішення та покращення результатів у загальному.

РОЗДІЛ 2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

2.1 Топологія SRGAN

GAN(Generative Adversarial Networks) представляють собою концепцію в штучному інтелекті, де дві нейронні мережі змагаються між собою в конкурентному процесі. GAN були представлені І. Гудфеллоу та його колегами в 2014 році в їхній статті "Generative Adversarial Nets" [9]. Концепція стала значним відходом від попередніх генеративних моделей і швидко привернула увагу завдяки своєму новаторському підходу та вражаючим результатам.

Такі мережі використовуються в багатьох різних галузях, починаючи від генерації реалістичних зображень, модифікації стилю фотографій та навіть генерації нових семплів, для розширення існуючих датасетів, дозволяючи системам ШІ навчатися на штучно розширених наборах даних, що особливо корисне у випадках, коли реальні дані є важкодоступними.

Архітектура GAN включає в себе дві нейронні мережі: генератор та дискримінатор, які тренуються одночасно та взаємодіють між собою, що сприяє покращенню обох мереж.

Генератора створює фальшиві дані та намагається згенерувати їх, якомога реальніше. На вхід генератора подається випадковий шум, який він використовує для генерації фальшивих даних, що імітують розподіл цільових даних. Він постійно покращує свої результати, залежно від того, як дискримінатор відрізняє фальшиві та реальні дані. На ранніх етапах тренування згенеровані зразки часто мають погану якість і легко визначаються як фальшиві, але в міру прогресу тренування генератор вивчає основні шаблони та структури реальних даних, створюючи їх більш реалістично.

Дискримінатор отримує на вхід дані та намагається розпізнати їх реальність, тобто визначити, що вони походять з реального датасету, або були згенеровані генератором. Він має структуру класифікаційної мережі та на

виході має ймовірнісне значення того, наскільки реальним є поданий на його вхід семпл.

Дискримінатор навчається виявляти відмінності між реальними та фальшивими зразками, дедалі покращуючи свої можливості відрізняти їх між собою.

Головна особливість GAN полягає у постійній взаємодії між генератором та дискримінатором. Генератор намагається максимізувати ймовірність того, що дискримінатор помилково класифікує його виходи як реальні. Водночас, дискримінатор намагається мінімізувати свою помилку розпізнавання реальних і фальшивих семплів.

Саме цей зв’язок дозволяє моделям GAN демонструвати гарні результати. Коли дискримінатор покращує своє розпізнавання фальшивих зразків, це дає змогу надати генератору більш точний фідбек. Після цього генератор адаптується, щоб створювати кращі фальшиві зразки, що змушує дискримінатор навчатися розпізнавати їх. Таким чином обидві мережі можуть одночасно навчатися та покращувати свої результати.

Структура мережі GAN зображена на рис. 2.1.

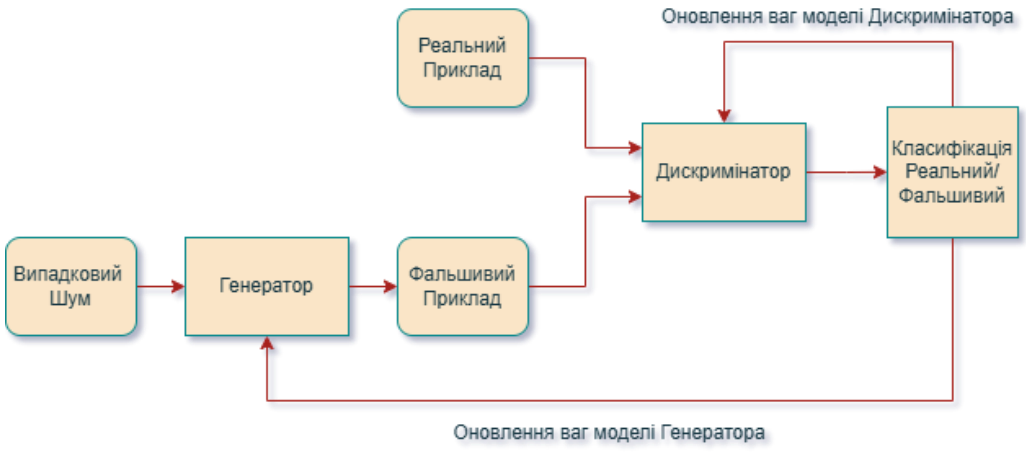

Рисунок 2.1 – Схема структури GAN

Процес навчання мережі GAN втілює ідею гри з нульовою сумою: для двох учасників гри, в умовах жорсткої конкуренції, якщо один виграє, це означає, що інший програє. Сума виграшів і збитків для обох учасників завжди нульова, і між ними немає можливості співпраці.

Алгоритм включає чергування оптимізації мереж дискримінатора і генератора.

1. Фаза навчання Дискримінатора D:
   - дискримінатор навчається на серії реальних семплів;
   - генератор створює серію фальшивих семплів;
   - дискримінатор навчається на цих фальшивих семплів;
   - ваги дискримінатора оновлюються для мінімізації помилки класифікації.
2. Фаза навчання генератора G:
   - генератор створює серію фальшивих зразків;
   - ці семпли подаються дискримінатору;
   - ваги генератора оновлюються для максимізації рівня помилки дискримінатора.

Мінімаксна цільова функція для GAN наведена у формулі (2.1).

$$\min_G \max_D V_{\text{GAN}}(D,G) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z\left[\log\left(1 - D\big(G(z)\big)\right)\right], \qquad (2.1)$$

де $D(x)$ – оцінка дискримінатору ймовірності того, що реальний екземпляр даних x є реальним;

$G(z)$ – вихідний сигнал генератору за заданого шуму z;

$D(G(z)$ – це оцінка дискримінатором ймовірності того, що фальшивий екземпляр є справжнім;

$\mathbb{E}_x$ – математичне сподівання для всіх реальних даних;

$\mathbb{E}_z$ – математичне сподівання для всіх випадкових вхідних даних генератора

Отже, дискримінатор натренований максимізувати правильність розпізнавання реальних та фальшивих даних. Водночас, генератор натренований мінімізувати правильне розпізнавання фальшивих даних дискримінатором. Цей процес змагального навчання дозволяє дискримінатору досягти рівноваги Неша. Тим часом Генератор може генерувати фальшиві дані, подібні до реальних.

Хоча в теорії, моделі GAN мають надійну структуру, вони все ще не ідеальні. При навчанні можуть виникнути наступні проблеми.

1. Проблема колапсу моделі. Колапс моделі відбувається, коли генератору не вдається повністю вловити розподіл тренувальних даних. Тобто, замість того, щоб навчитися генерувати вибірки по всьому розподілу, генератор знаходить кілька шаблонів, які постійно позначаються дискримінатором, як реальні, і починає створювати лише варіації цих шаблонів.
2. Нестабільність навчання. Навчання GAN часто страждає від нестабільності, зокрема від коливань функції втрат та нездатності досягти стабільного стану. Ця нестабільність може бути викликана кількома факторами: нерівновага між можливостями генератора та дискримінатора, чутливість до вибору гіперпараметрів, неконвексний характер оптимізаційної задачі.
3. Зникаючи градієнти. На ранніх етапах навчання дискримінатор може присвоїти дуже низькі ймовірності згенерованим зразкам. Це призводить до зникнення градієнтів для генератора, зупиняючи процес навчання, оскільки генератор отримує мінімальний фідбек. Ці проблеми призвели до численних інновацій в архітектурі та підходу до навчання моделей GAN.

SRGAN має звичайну структуру GAN: дискримінатор та генератор, без допоміжних моделей, проте моделі генератора та дискримінатора модифіковані під завдання підвищення роздільної здатності зображення та

побудовані з однотипних блоків. Структуру генератора та дискримінатору SRGAN можна побачити на рис. 2.2, рис. 2.3.

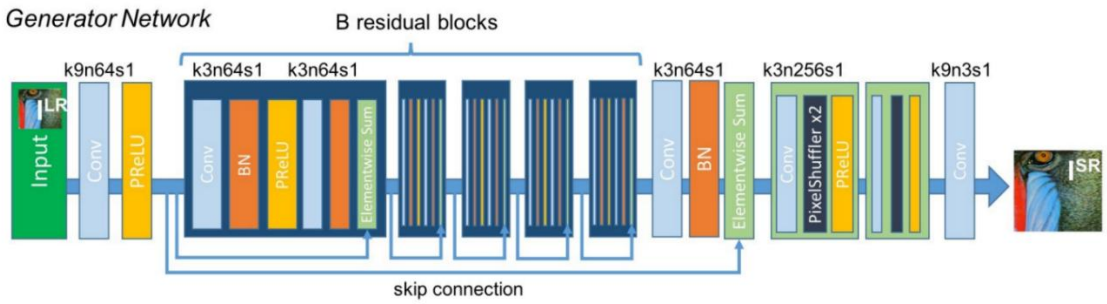

Рисунок 2.2 – Будова мережі генератора [24]

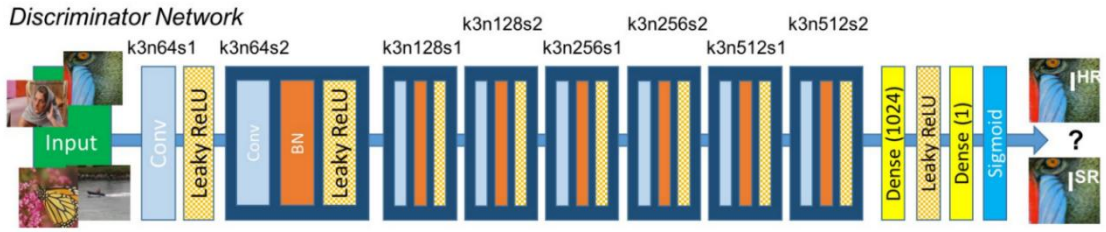

Рисунок 2.3 – Будова мережі дискримінатора [24]

Мережа генератора є адаптацією мережі SRResNet (Super Resolution ResNet) [24] та приймає на вхід зображення з низькою роздільною здатністю, яке пропускають через початковий згортковий шар розмірністю ядра 9x9 та 64 карти ознак, а після параметричний ReLU шар. Для генератора було обрано використовувати саме параметричну ReLU, через те, що вона найкраще підходить для зіставлення зображень із низькою роздільною здатністю з високою. При використанні звичайної ReLU можуть виникнути проблеми коли значення менші за нуль зіставляються з нулем. Тому в параметричній реалізації значення менші за нуль зіставляться з заданим значенням, яке обирається нейронною мережею.

Наступна частина складається з залишкових блоків. Кожен із залишкових блоків містить згортковий шар із 3x3 ядер і 64 карти ознак, за якими слідує шар batch normalization, параметрична функція активації ReLU, такий самий згортковий шар, batch normalization та кінцевий метод поелементної суми. Метод поелементної суми використовує вихід даного блоку разом із виходом skip connection для надання остаточного результату. Кожен із згорткових шарів використовує подібні заповнення, щоб розмір входів і виходів не змінювався.

В останній частині ми використовуємо pixel shuffle після чотирьох кратного збільшення згорткового шару для створення зображення з високою роздільною здатністю. Pixel shuffler беруть значення з розміру каналу та вставляють їх у розміри висоти та ширини. Розмірність висоти та ширини множаться на два, а кількість каналів ділиться на 2.

Перша частина дискримінатора складається з згорткового шару розмірністю ядра 3x3 та 64 карти ознак, а після Leaky ReLU шар. Наступна частина складається з декількох шарів, схожої будови, що складаються з згорткового шару, batch normalization та Leaky ReLU, але в кожному наступному шарі змінюються параметри згорткових шарів. В останній частині знаходяться два повнозв'язних шари, розділені Leaky ReLU та на виході стоїть сигмоїдна функція активації. Дискримінатора працює для розпізнавання зображень зі збільшеною роздільною здатністю та реальних зображень.

Функція втрати для даної моделі наведена у формулі (2.2).

$$l^{SR} = l_{\mathrm{X}}^{SR} + 10^{-3} l_{Gen}^{SR}, \tag{2.2}$$

де l_{X}^{SR} – VGG функція втрати;

l_{Gen}^{SR} – додаткова змагальна функція втрати

VGG функція втрати має важливу роль в генерації реалістичних зображень. Вона наведена у формулі (2.3).

$$l_{\frac{VGG}{i},j}^{SR} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}\left(G_{\theta_G}(I^{LR})\right)_{x,y} \right)^2, \quad (2.3)$$

де $\phi_{i,j}$ вказує на карту ознак, отриману j-м блоком згортки перед i-м max pooling шаром в структурі генератора;

$W_{i,j}H_{i,j}$ – розміри ширини та висоти відповідних карт ознак; I^{HR} – зображення високої роздільної здатності;

I^{LR} – зображення низької роздільної здатності.

Додаткова змагальна функція втрати спонукає модель GAN надавати перевагу рішенням, які базуються на різноманітності природних зображень. Ця функція наведена у формулі (2.4).

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -\log D_{\theta_D}\left(G_{\theta_G}(I^{LR})\right), \quad (2.4)$$

де I^{LR} – зображення низької роздільної здатності;

G_{θ_G} – вихідний сигнал генератору за заданого входу у вигляді зображення I^{LR};

D_{θ_D} – оцінка дискримінатору ймовірності поданого на нього прикладу на його реалістичність.

2.2 Огляд робіт у сфері збільшення роздільної здатності

В роботі “Improved SRGAN for Remote Sensing Image Super-Resolution Across Locations and Sensors” [25] було розглянуто покращення можливостей збільшення роздільної здатності зображень шляхом створення модифікованої версії SRGAN (Super-Resolution Generative Adversarial Network), з назвою ISRGAN. Дослідження присвячене вирішенню проблеми створення зображень

дистанційного зондування для різних географічних локацій і типів датчиків. Розглянемо основні інновації такого дослідження.

1. Модифікована функція втрат: стандартна SRGAN використовує втрату середньоквадратичної похибки та втрату на основі мережі VGG-19, ця робота розширює попередні дослідження.
2. Покращена архітектура мережі: структурні зміни в мережах генератора і дискримінатора для кращої справлятися з складністю зображень дистанційного зондування.
3. Міждоменна генералізація: процес навчання вдосконалено та посилено завдяки більшій здатності до узагальнення між різними локаціями та датчиками.

У своїй роботі автор пропонує діаграму робочого процесу мережі, яка наведена на рис. 2.4.

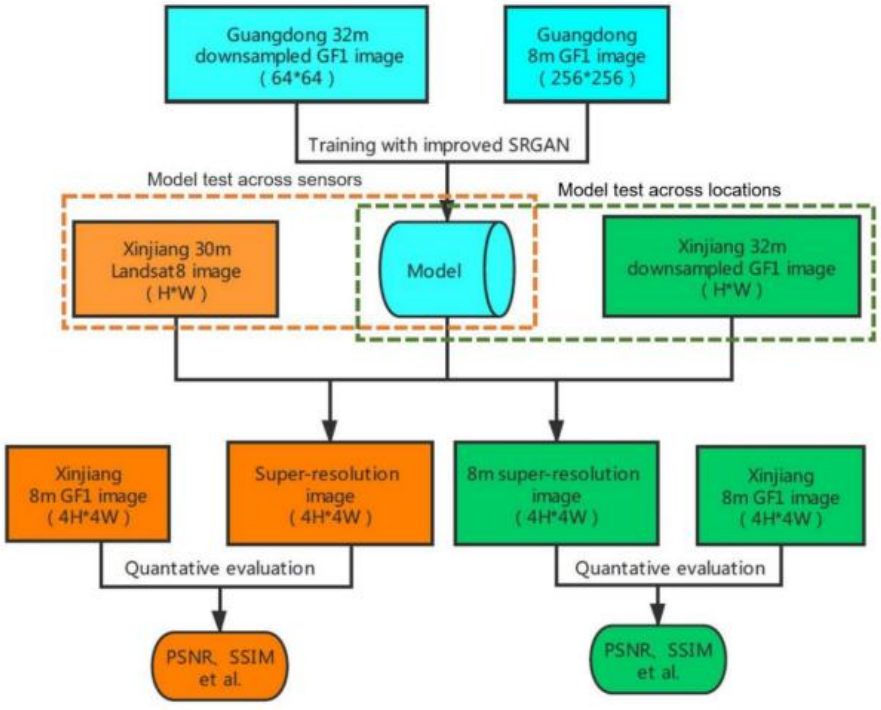

Рисунок 2.4 – Робочий процес мережі [25]

Розглянувши основну суть та мету дослідження перейдемо до аналізу переваг та недоліків такого методу. Автор відмічає такі переваги для свого методу.

1. Запропоновані нововведення стабілізують процес навчання краще, ніж оригінальна SRGAN.
2. Покращена здатність працювати з різними географічними локаціями та з різними типами датчиків, що є критично важливим у завданнях дистанційного зондування.
3. Автор зберігає напрацювання SRGAN у вдосконаленні функції втрат, але виправляє деякі недоліки.
4. Розроблена модель краще підходить для практичних застосувань дистанційного зондування в реальному світі, де зображення надходять з різних джерел і локацій.

Тепер розглянемо недоліки такої моделі.

1. Розроблена модель вимагає великих обчислювальних ресурсів під час навчання та інференції.
2. Незважаючи на вдосконалення, генеративно змагальні мережі є складними в навчанні і страждають від проблем колапсу режиму або нестабільністю.
3. Автором невказані критерії оцінки можливостей узагальнення розробленої мережі. Отже його твердження все ще потребують тестування з різними наборами даних і типами датчиків.

Робота “SRGAN for Super-Resolving Low-Resolution Food Images” [26] була присвячена вирішенню проблеми збільшення роздільної здатності зображень продуктів харчування, які мають низьку роздільну здатність або містять артефакти, зокрема для сфери застосування у веб-платформах з рецептами. У статті використовується стандартна архітектура SRGAN (Super-Resolution Generative Adversarial Network), але вона була модифікована для відновлення зображень продуктів харчування.

Автори внесли дві основні зміни до стандартного підходу SRGAN.

1. Розроблена процедура зменшення роздільної здатності з введенням шуму для отримання бажаних зображень для навчання моделі. Це вирішує проблему базової SRGAN, яка добре працює з штучно зменшеними зображеннями, але має труднощі з зображеннями, що містять реальні артефакти. Структура процесу зменшення роздільної здатності наведена на рис. 2.5.

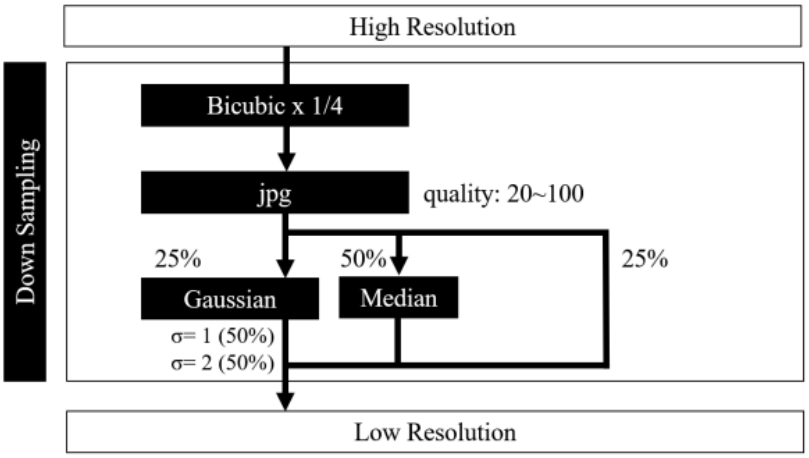

Рисунок 2.5 – Процедура створення зображень з низькою роздільною здатністю [26]

2. Було запропоновано новий метод аналізу результатів, що включає нові метрики для оцінки якості зображень їжі з збільшеною роздільною здатністю. Використання такого методу підвищує реалістичність згенерованих результатів роботи мережі.

Розглянувши основну суть та мету дослідження перейдемо до аналізу переваг та недоліків такого методу. Автор відмічає такі переваги для свого методу.

1. Модифікації, впроваджені автором, є вузько направленими саме для зображень їжі, тому вони набагато корисніші для використання у секторах пов'язаних з їжею.

2. Метод введення шуму для утворення зображень для тренування мережі створює більш реалістичне середовище для навчання, імітуючи тип артефактів, що зустрічаються в реальних зображеннях.
3. Новий метод оцінки результатів є кращим метричним показником у порівнянні з широко використовуваними метричними показниками PSNR та SSIM.

Тепер розглянемо недоліки нових модифікацій.

1. Хоча фокус на зображеннях їжі є плюсом для використання у даному секторі, але все одно це обмежує використання мережі для інших доменів зображень.
2. Впровадження шуму разом з новими протоколами оцінки збільшує загальну складність процесу навчання та тестування.
3. Процес навчання залишається складним і супроводжується такими проблемами, як колапс режиму та нестабільність.

Робота “A Super Resolution Algorithm Based on Attention Mechanism and SRGAN Network” [27] зосереджується на вдосконаленні архітектури SRGAN (Super Resolution Generative Adversarial Network) шляхом впровадження механізмів уваги для підвищення продуктивності збільшення роздільної здатності окремих зображень. Дослідження має на меті усунути обмеження існуючих мереж SRGAN, зробивши модель більш здатною до фіксації високочастотних ознак та покращивши загальну якість реконструкції зображень.

У роботі представлено три основні вдосконалення оригінальної архітектури SRGAN.

1. Автор запропонував модуль уваги до каналів (Channel Attention) для мережі SRGAN та збільшити глибину мережі, щоб краще виражати високочастотні особливості. Будова модуля уваги до каналів наведена на рис. 2.6.

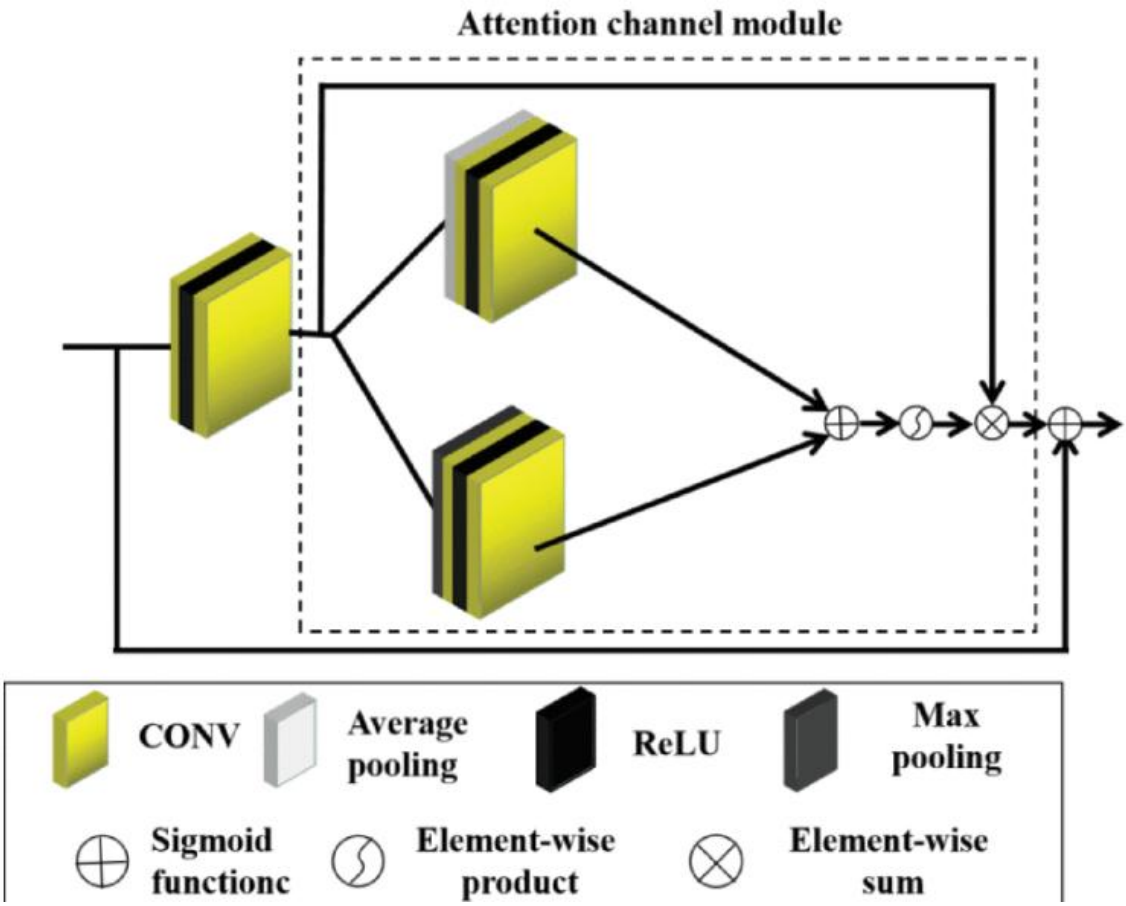

Рисунок 2.6 – Будова модуля уваги до каналів [27]

2. Оригінальний шар нормалізація пакетів було видалено для покращення продуктивності.
3. Покращена загальна архітектура мережі, де модифікації спрямовані на поліпшення здатності мережі фіксувати та відтворювати дрібні деталі за допомогою відбору ознак на основі уваги.

Розглянувши основну суть та мету дослідження перейдемо до аналізу переваг та недоліків такого методу. Автор відмічає такі переваги для свого методу.

1. Механізм уваги дозволяє мережі зосередитися на найбільш релевантних каналах та ознаках, що призводить до кращого відтворення високочастотних деталей.

2. Завдяки збільшенню глибини мережі та включенню модулів уваги алгоритм може краще зберігати та відтворювати дрібні текстури та краї.
3. Механізм уваги до каналів дозволяє мережі адаптивно зважувати різні канали ознак на основі їх важливості для відтворення.

Тепер розглянемо недоліки нових модифікацій.

1. Додавання модулів уваги та збільшення глибини мережі збільшує обчислювальні вимоги та загальний час навчання.
2. Потенційний ризик перенавчання через глибину мережі та більшої кількості параметрів.
3. Модифікована архітектура може вимагати більше пам'яті як під час навчання, так і під час використання.

Робота “ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks” [28] спрямована на поліпшення якості збільшення роздільної здатності окремих зображень. У цьому дослідженні розглядається проблема візуальних артефактів, які часто супроводжують результати, згенеровані SRGAN.

Розглянемо основні покращення, запропоновані автором дослідження.

1. Замінено стандартні залишкові блоки на нову архітектуру RRDB. Вона поєднує багаторівневі залишкові з'єднання з щільними з'єднаннями. Ключовою особливістю є видалення шарів нормалізації пакетів, щоб усунути артефакти, які вони можуть спричинити. Структура RRDB наведена на рис. 2.7.

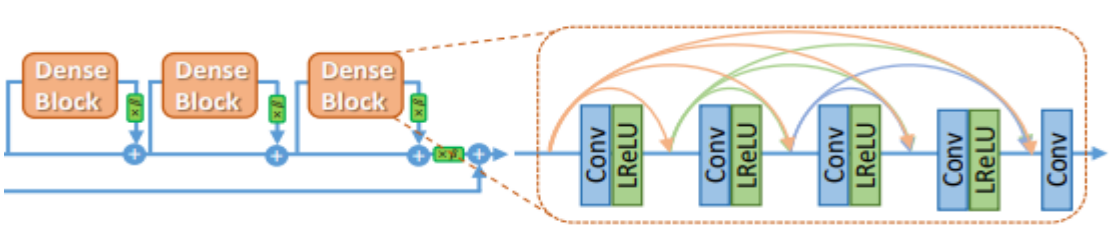

Рисунок 2.7 – Структура блоків RRDB [28]

2. Змінена архітектура дискримінатора, в якій оцінюється ймовірність того, що одне зображення є більш реалістичним, ніж інше замість абсолютних значень. Ці інновації забезпечує більш стабільне та ефективне змагальне навчання.
3. Проведено незначні зміни у структурі функції втрат, що доповнює інші внесені модифікації і тим самим впроваджує більш фотореалістичні результати з меншою кількістю спотворень.

Розглянувши основну суть та мету дослідження перейдемо до аналізу переваг та недоліків такого методу. Автор відмічає такі переваги для свого методу.

1. Значно покращена якість генерації порівняно з SRGAN та зменшена кількість візуальних артефактів, що виникали при відновленні роздільної здатності зображень.
2. Архітектура RRDB забезпечує більш ефективне вилучення особливостей завдяки щільним зв'язкам. Також блоки RRDB можна легко масштабувати та адаптувати до різних вимог.
3. Підвищена стабільність змагального навчання завдяки впровадженню нової архітектури для дискримінатора.

Тепер розглянемо недоліки нових модифікацій.

1. Такі зміни в структурі мережі, як архітектура RRDB та глибша мережа дискримінатора вимагають набагато більше ресурсів для обчислень та пам’яті.
2. Як результат попереднього пункту, мережа має збільшену кількість параметрів порівняно з оригінальною SRGAN. Таким чином вона вимагає набагато більше часу для навчання.
3. Більш складна архітектура може бути схильна до перенавчання на менших наборах даних. Також хоча видалення BN допомагає, мережа стає більш чутливою до інших гіперпараметрів.

Робота “CT Image Super Resolution Based On Improved SRGAN” [29] присвячена проблемі обмеженої просторової роздільної здатності КТ-

зображень, що використовуються для медичної діагностики. КТ-зображення широко застосовуються в клінічній практиці, але обмеження апаратного забезпечення та час сканування призводять до низької просторової роздільної здатності, що заважає лікарям точно аналізувати дрібні ділянки ураження та патологічні особливості. У дослідженні пропонується алгоритм підвищення роздільної здатності КТ-зображень на основі вдосконаленої версії SRGAN (Super-Resolution Generative Adversarial Network).

Головним внеском цього дослідження є розробка вдосконаленої SRGAN, спеціально оптимізованої для медичних КТ-зображень. На відміну від інших мереж, які більше зосереджуються на показниках оцінки зображень, автору вдалося досягти значного поліпшення у якості сприйняття згенерованих зображень.

Дослідження сприяє розвитку галузі медичної візуалізації за допомогою штучного інтелекту, вирішуючи конкретну проблему підвищення роздільної здатності КТ-зображень, що потенційно покращує діагностичну точність для невеликих патологічних ознак, які мають вирішальне значення для прийняття клінічних рішень.

Розглянувши основну суть та мету дослідження перейдемо до аналізу переваг та недоліків такого методу. Автор відмічає такі переваги для свого методу.

1. Вдосконалена SRGAN надає пріоритет якості візуального сприйняття над традиційними показниками якості зображення, що є надзвичайно важливим для медичної діагностики, де візуальна чіткість патологічних ознак має найбільше значення.
2. Модифікації спеціально розроблені для КТ-зображень, вирішуючи унікальні проблеми, такі як шумові шаблони та збереження анатомічних деталей.
3. Впроваджена модифікована модель дозволяє краще аналізувати дрібні ділянки ураження та патологічні ознаки, які раніше було важко досліджувати через низьку роздільну здатність.

Тепер розглянемо недоліки нових модифікацій.

1. Запропонована модель вимагає значних обчислювальних ресурсів як для навчання, так і для інференції, що може становити проблему для клінічних застосувань у реальному часі.
2. Хоча метод акцентує увагу на якості зображень для сприйняття, цей підхід може погіршити результати за традиційними кількісними показниками, які все ще є важливими для певних застосувань.

Робота “Super-Resolution for Hyperspectral Remote Sensing Images Based on the 3D Attention-SRGAN Network” [30] присвячена обробці гіперспектральних зображень дистанційного зондування, які мають високу спектральну роздільну здатність, але низьку просторову роздільну здатність через фізичні обмеження датчиків. У дослідженні пропонується новий підхід до підвищення просторової роздільної здатності гіперспектральних зображень із збереженням їхньої спектральної інформації.

Дослідники розробили тривимірну генеративно змагальну мережу 3D Attention-SRGAN для збільшення роздільної здатності на основі механізму уваги, спеціально призначену для гіперспектральних зображень. Інтеграція механізмів уваги дозволяє мережі зосередитися на найважливіших спектрально-просторових характеристиках під час процесу збільшення роздільної здатності, покращуючи якість відновлених зображень. Структура мережі наведена на рис. 2.8.

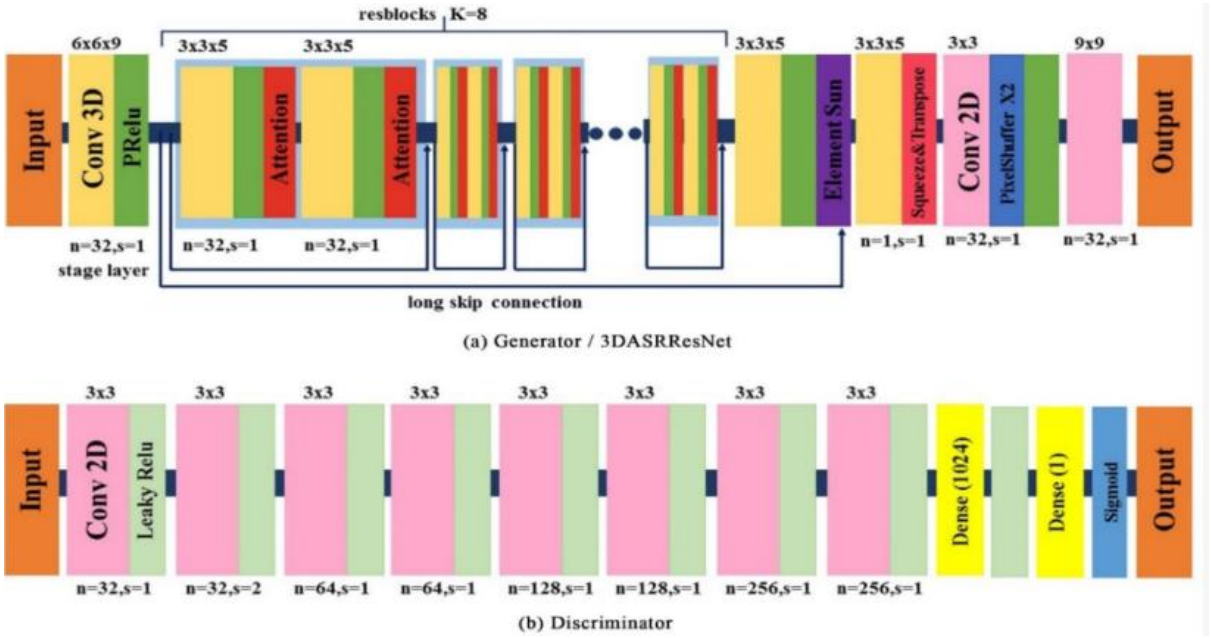

Рисунок 2.8 – Архітектура мережі 3D Attention-SRGAN [30]

На відміну від багатьох існуючих методів, які зосереджуються переважно на відновленні просторової інформації, цей підхід одночасно враховує як спектральний, так і просторовий виміри, забезпечуючи краще збереження гіперспектральних характеристик.

Розглянувши основну суть та мету дослідження перейдемо до аналізу переваг та недоліків такого методу. Автор відмічає такі переваги для свого методу.

1. Трьох вимірна архітектура краще зберігає спектральну цілісність гіперспектральних зображень у порівнянні з існуючими двох вимірними методами.
2. Механізми уваги забезпечують більш ефективне навчання характеристик у спектральних діапазонах.
3. Впроваджений комплексний підхід, в якому одночасно вирішується питання просторового покращення та збереження спектральних характеристик.

Тепер розглянемо недоліки нових модифікацій.

1. Генеративно змагальні мережі є вимогливими до обчислювальної складності, а впровадження трьох вимірної обробки та механізму уваги, ще більше збільшують обчислювальні вимоги.
2. Складна архітектура вимагає ретельного налаштування гіперпараметрів для різних гіперспектральних наборів даних.
3. Оскільки тести були проведені тільки з деякими типами датчиків, тому невідомо яка продуктивність буде у мережі для різних типів гіперспектральних датчиків та умов зображення.

Висновки до розділу 2

У другому розділі дано загальну теорію роботи генеративно змагальних мереж та описано процес їх навчання. Також розглянуто, які саме проблеми можуть виникнути при процесі навчання. Було детально описано топологію, обраної за основу, генеративно змагальної мережі SRGAN. Надано характеристику її основних компонентів та проаналізовано взаємодію між ними.

Було ретельно досліджено схожі дослідження, в яких за основу було використано SRGAN. Розглянувши ці статті було проведено їх аналіз та виявлено плюси та мінуси запропонованих в них модифікацій. Такий аналіз може допомогти з визначенням остаточних модифікацій, які буде внесено у структуру даної мережі.

Задача збільшення роздільної здатності полягає у відновленні зображення з його варіанту з низькою роздільною здатністю, при тому зберігаючи ключову інформацію та якість деталей. Процес формування вхідних даних є зменшення роздільної здатності вхідних зображень, через застосування вище описаного оператору заниження роздільної здатності. Для оцінки точності відновлення використовують кількісні метрики, зокрема

середньоквадратична похибка (MSE), які дозволяють об’єктивно оцінити відмінності між реальним та згенерованими зображеннями.

РОЗДІЛ 3 ПОБУДОВА МОДИФІКОВАНОЇ МЕРЕЖІ

В цьому розділі буде розглянуто запропоновані мною модифікації, які можна внести в структуру генеративно змагальної мережі для покращення її стабільності та збільшення якості результатів, отриманих в процесі генерації.

Архітектурну модифікацію, яку я хочу запропонувати є підхід з подвійним дискримінатором. Така модифікація може значно покращити якість згенерованих зображень високої роздільної здатності.

Хоча поточний підхід із єдиним дискримінатором демонструє непогані результати, вони далекі від ідеальних, і він має низку недоліків. У нього є дві основні проблеми: підтримання глобальної зв’язності зображення при збереженні точних локальних особливостей. Я пропоную використати архітектуру подвійного дискримінатора, яка складається з двох елементів.

1. Глобальний дискримінатор – оцінює глобальну структуру, зв'язність і правдоподібність зображення.
2. Локальний дискримінатор – враховує лише високочастотні деталі та текстуру.

3.1 Теоретичні переваги

Перш за все, розглянемо, які саме переваги може надати підхід з архітектурою подвійного дискримінатора, зокрема в аспектах підвищення якості згенерованих результатів, кращого узагальнення моделі, а також можливості точнішої диференціації між реальними та штучно створеними зразками.

Структура подвійного дискримінатору дозволяє розділити вирішення проблем, тим самим забезпечити більш точні шляхи зворотного зв'язку до

генератора. Глобальний дискримінатор зменшуватиме структурні спотворення, які можуть з’являтися у процесі збільшення роздільної здатності зображень, а локальний дискримінатор відповідатиме за створення реалістичних текстур, що дозволить вирішити загальну проблему надмірного згладжування, характерну для більшості методів збільшення роздільної здатності.

Архітектура з двома дискримінаторами пропонує більш збалансоване середовище для навчання в умовах змагального навчання. Поточні SR-GAN схильні до колапсу моделі, якщо дискримінатор занадто потужний або занадто слабкий. Розділивши роботу між двома виділеними мережами, ми зможемо краще підтримувати тонкий баланс між генератором і дискримінатором, необхідний для стабільного навчання.

Також ця архітектура безпосередньо вирішує проблему компромісу між якістю сприйняття і спотвореннями. Спільна оптимізація обох дискримінаторів дозволяє досягти більш задовільного балансу між об'єктивною точністю та суб'єктивною візуальною якістю, створюючи зображення з підвищеною роздільною здатністю, які є не лише кількісно правильними, але й реалістичними для сприйняття.

3.2 Загальна архітектура мережі

За основу архітектури, яку буде використано для виконання поставленого завдання, взято генеративно змагальну модель SRGAN, детальний опис якої був вказаний у попередньому розділі.

Оскільки ми хочемо дізнатися, яким саме чином буде впливати подвійний дискримінатор на отримані результати та процес навчання, тому генератор моделі залишається без змін.

Дискримінатор моделі буде замінено на подвійний дискримінатор, який буде фокусуватись на різних ділянках зображень:

Глобальний дискримінатор, який фокусується на визначенні того, згенероване або реальне зображення подано на його вхід, а також на глобальній зв’язності зображення в цілому.

Локальний дискримінатор, який фокусується на точних деталях та текстурах зображення. Він відповідає за якість деталей зображення та запобігає згладженню його текстур.

Загальна структура модифікованої генеративно змагальної мережі зображено на рис. 3.1.

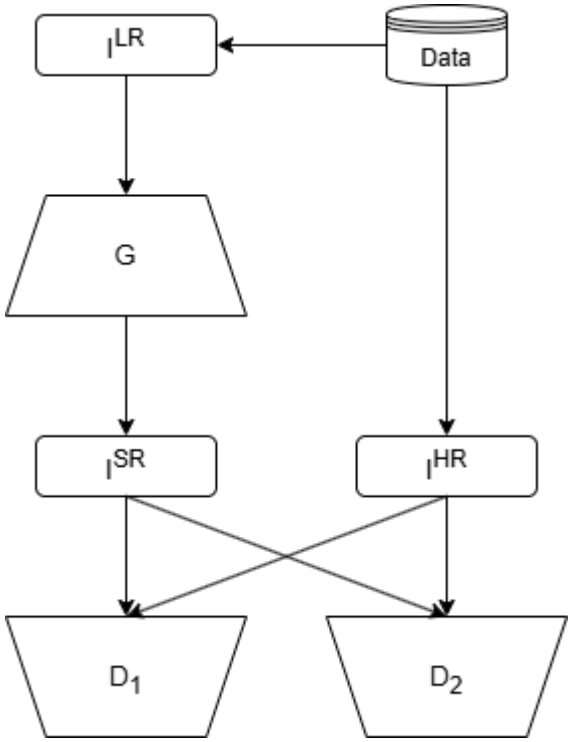

Рисунок 3.1 – Структура модифікованої генеративно змагальної мережі

На доданій вище діаграмі зображена загальна архітектура модифікованої генеративно змагальної мережі. На рисунку G – генератор, D_1, D_2 – дискримінатори, I^{LR}, I^{HR}, I^{SR} – відповідно зображення низької, високої, збільшеної роздільної здатності.

В наступних підрозділах буде детальніше розглянуто структури відповідних мереж дискримінаторів, їхню взаємодію та процес навчання модифікованої мережі.

3.3 Структура мережі глобального дискримінатору

Глобальний дискримінатор є головним компонентом генеративно змагальних мереж, які фокусуються на збільшенні роздільної здатності, таких як SRGAN [24] та інші приклади таких моделей. Його основне завдання полягає в розпізнаванні поданого на вхід зображення, а саме його походження: зображення високої роздільної здатності, згенероване генератором мережі або справжнє зображення, що входить у навчальний набір.

Глобальне сприйняття зображення дозволяє моделі краще розпізнавати артефакти генерації та загальну зв’язність зображення. У результаті, використання такої моделі сприяє генерації кращих результатів та зменшує ймовірності появи ярко виражених артефактів. Також врахування глобального контексту є важливим для збереження цілісності складних об’єктів, таких як будівлі, обличчя та інші.

Архітектура глобального дискримінатору складається з трьох основних частин: блок зменшення дискретизації (downsampling), блок просіювання (bottleneck) та залишковий блок (residual block). Їхня взаємодія дозволяє ефективно визначати реальність поданих на вхід зображень. Архітектура глобального дискримінатора зображена на рис. 3.2.

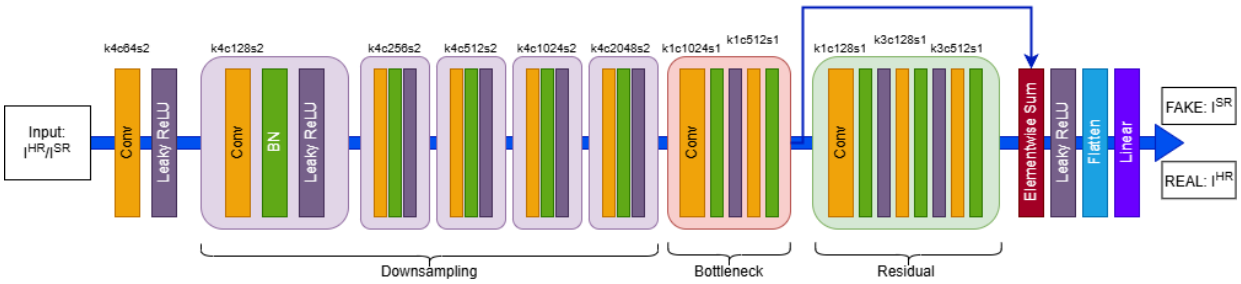

Рисунок 3.2 – архітектура мережі глобального дискримінатора

Мережа приймає на вхід зображення, яке проходить через початковий згортковий шар, розмірністю ядра 4x4 та 64 карт ознак, а після Leaky ReLU шар. Далі розглянемо детальніше структуру кожного блока та задачу, яку вони виконують при обробці вхідного зображення.

Перший частина – блок зменшення дискретизації. Він складається з 5 блоків схожої будови, що включають в себе послідовно згортковий шар, batch normalization, Leaky ReLU, але у кожному наступному блоці подвоюється кількість карт ознак. Така структура дозволяє поступово зменшувати дискретизацію зображення, що дозволяє мережі отримувати ознаки зображення на його різних розмірах. Тобто початкові блоки отримують базові шаблони такі, які краї та текстури, коли більш глибокі можуть розпізнавати семантичні ознаки вищого рівня.

Наступна частина мережі – блок просіювання. Він також складається з стандартної структури: згортковий шар з розмірністю ядра 1x1 та 1024 карт ознак, batch normalization, LeakyReLU, згортковий шар з розмірністю ядра 1x1 та 512 карт ознак та ще один batch normalization. Його задача полягає у стисненні інформації та утворенні міжканальної інтеграції. Він також зменшує кількість параметрів та тим самим вартість обрахунків.

Остання частина – залишковий блок. Він містить згортковий шар із 1x1 ядер і 128 карти ознак, за якими слідує шар batch normalization, Leaky ReLU, які повторюються два рази з єдиною різницею в розмірі ядра 3x3 згорткового шару. За ними знаходиться згортковий шар з 3x3 ядер і 512 карт ознак, batch

normalization та метод поелементної суми. Його впровадження допомагає запобігати проблемі зникаючих градієнтів. В останній частині мережі результат, отриманий у попередніх блоках, проходить через LeakyReLU, шар випрямлення та лінійну функцію активації.

Розмір ядра 4x4 та крок 2 у параметрах згорткових шарів має гарний баланс між складністю розрахунків та захватом локальних ознак. Також такий розмір згорток дозволяє запобігти жодним втратам ознак, оскільки вони будуть перетинатися на 50%. Згорткові шари у яких використовується розмір ядра 1x1 використовуються для зменшення кількості каналів без втрати просторової інформації.

У результаті дана мережа видає результат в залежності від аналізу поданого на вхід зображення: Fake – зображення є згенерованим, або Real – зображення є реальним. У запропонованій модифікації структури цей дискримінатор працює у поєднанні з локальним дискримінатором, який буде розглянутий у наступному підрозділі.

3.4 Структура мережі локального дискримінатору

На відміну від глобального дискримінатору, локальний розглядає маленькі ділянки замість зображення в цілому. Таке впровадження дозволяє моделі детальніше аналізувати зображення, а саме тонкі локальні деталі. До таких відносяться такі компоненти: текстури, структури поверхонь, краї предметів та інші. Детальний аналіз зображень дозволяє краще розпізнавати зображення на предмет їхньої реалістичності, що в свою чергу покращує згенеровані моделлю результати.

Локальний аналіз зображень може надати генератору інформацію щодо якості кожної окремої області, сприяючи підвищенню рівня локальної деталізації. Це особливо корисно при відновленні зображень з великою

кількістю дрібних об’єктів та деталей, деталізація яких може втратитися під час збільшення роздільної здатності.

Локальний дискримінатор має значно меншу архітектуру за глобальний дискримінатор та включає лише один основний блок – блок вилучення ознак. На вхід до даної мережі подається декілька частин заданого розміру, визначеного у параметрах мережі. Загальна структура дискримінатору зображена на рис. 3.3.

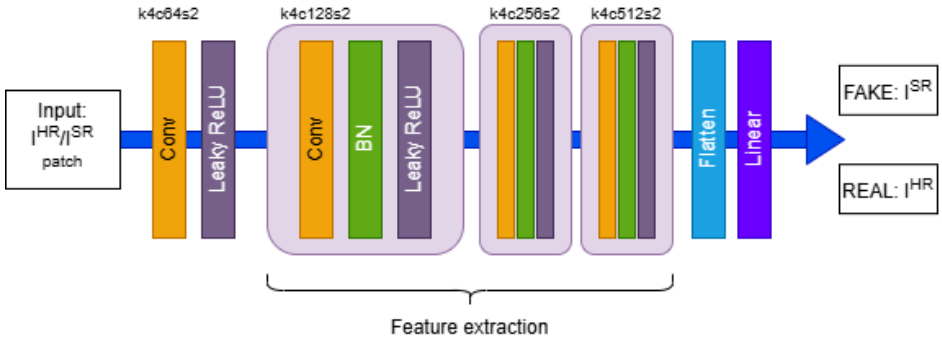

Рисунок 3.3 –Будова локального дискримінатору

Задані частини зображень подаються на вхід локального дискримінатору, які проходять через такі початкові шари: згортковий з розмірами ядра 4x4 і 64 картами ознак та Leaky ReLU шар. Далі починається основна частина в якій відбувається вилучення ознак з вхідних даних. Вона складається з трьох блоків схожої будови: згортковий шар з розмірами ядра 4x4, batch normalization та Leaky ReLU. Єдина різниця між ними у розмірах карт ознак, які відповідно при збільшенні глибини мають такі розміри – 128, 256 та 512. У кінці отримані ознаки проходять через шар випрямлення та лінійну функцію активації.

Аналогічно до глобального дискримінатору розміри ядра 4x4 дозволяють підтримувати баланс між складністю розрахунків та захватом локальних ознак. Також такий розмір згорток дозволяє запобігти жодним втратам ознак.

У результаті дана мережа видає результат в залежності від аналізу поданих на вхід частин зображення: Fake – зображення є згенерованим, або Real – зображення є реальним. Така локальний аналіз дозволяє детально аналізувати текстуру та деталі зображення та відповідно до цього визначати його реальність.

Поєднання глобального та локального дискримінаторів покращують аналіз зображення та тим самим збільшують відсоток правильно визначених зображень. Отже у генератора з'являється набагато більше відгуків, а тому збільшується реалістичність його генерацій. Далі розглянемо яким саме чином дискримінатори та генератор взаємодіють між собою.

3.5 Взаємодія у мережі

Визначення функції втрат є вирішальним елементом для якісної роботи мережі. Для великої кількості схожих робіт поширеним варіантом є використання MSE функції втрат [31],[32]. В даній мережі буде використано модифікований варіант такої функції втрат. Запропонована модифікація, окрім MSE, буде включати в себе аналіз зображень за допомогою іншої мережі VGG19 [33] та має загальний вигляд, наведений на формулі (3.1).

$$\mathcal{L}^{SR} = \mathcal{L}^{MSE} + \mathcal{L}^{VGG} + \mathcal{L}^{ADV}, \tag{3.1}$$

де $\mathcal{L}^{MSE}$ – звичайна MSE функція втрат;

$\mathcal{L}^{VGG}$ – функція втрат пов'язана з мережею VGG19;

$\mathcal{L}^{ADV}$ – змагальна функція втрат, формується під час змагального навчання.

Розглянемо детальніше кожну частину нашої функції втрат.

3.5.1 MSE функція втрат

По піксельна функція втрат MSE описується формулою (3.2).

$$\mathcal{L}^{MSE} = \frac{1}{WH}\sum_{x=1}^{W}\sum_{y=1}^{H}\left(I_{x,y}^{HR} - \mathrm{G}(I^{LR})_{x,y}\right)^2, \tag{3.2}$$

де I^{HR} – зображення високої роздільної здатності;

I^{SR}– отримане зображення із збільшеною роздільною здатністю;

H, W – відповідно висота та ширина зображення.

Оскільки така функція втрат не може надати достатньо інформації, щоб підтримувати відновлення малих деталей зображень, що приводить до згладжування текстур та поганого вигляду згенерованого зображення, тоді ми вводимо додаткову функцію втрат сприйняття, що базується на використанні VGG19 [34], яка дозволить наблизити результати до генерації реалістичних зображень.

3.5.2 Функція втрат сприйняття

Як було сказано раніше дана функція втрат використовує натреновану нейронну мережу VGG19 [33], для аналізу зображень. Дана функція втрат виражена у вигляді евклідової відстані між результатами мережі VGG з поданими на вхід реальним та згенерованим зображеннями наведена у формулі (3.3).

$$\mathcal{L}^{VGG} = 2 * 10^{-6} \cdot (\|\phi(I^{HR}) - \phi(I^{SR})\|_2)^2, \tag{3.3}$$

де I^{HR} – зображення високої роздільної здатності;

I^{SR}– отримане зображення із збільшеною роздільною здатністю;

$\phi()$ – карта ознак, отримана з використанням VGG19 для вхідного зображення.

3.5.3 Змагальна функція втрат

Додатково до інших компонентів функції втрат ми також використовуємо генеративну частину нашої мережі GAN. Ця функція втрат спонукає генератор G надавати перевагу більш натуральній генерації зображень, намагаючись обдурити подвійний дискримінатор мережі. Оскільки наша мережа має два дискримінатори, то функція втрат наведена на формулах (3.4), (3.5) та (3.6):

$$\mathcal{L}^{ADV} = \mathcal{L}^{LOC} + \mathcal{L}^{GLOB}, \tag{3.5}$$

$$\mathcal{L}^{LOC} = 10^{-3}\mathcal{L}^{BCE}(D_{Local}(I^{SR}), \tag{3.6}$$

$$\mathcal{L}^{GLOB} = 10^{-3}\mathcal{L}^{BCE}(D_{Global}(I^{SR}), \tag{3.7}$$

де D_{Local}, D_{Global} – відповідно результати локального та глобального дискримінаторів;

I^{SR} – отримане зображення із збільшеною роздільною здатністю;

$\mathcal{L}^{BCE}$ – BCE(Binary Cross-Entropy) [35] функція втрат, що має формулу (3.8).

$$\mathcal{L}^{BCE} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \cdot \log(\widehat{y_i}) + (1 - y_i) \cdot \log(1 - \widehat{y_i})), \tag{3.8}$$

де N – кількість прикладів датасету;

y_i – реальна позначка (0 або 1) прикладу;

$\widehat{y_i}$ – ймовірність того, що передбачений результат є позитивним.

Отже, розроблена комбінована функція втрат є ефективним рішенням для підвищення якості генерації зображень високої роздільної здатності. Запропонована функція втрат використовує три компоненти: MSE функція втрат, VGG функція втрат та змагальна функція втрат. Синергетичний ефект від поєднання всіх трьох компонентів дозволяє досягти оптимального балансу між точністю відтворення, збереженням деталей та реалістичністю згенерованих зображень. Це робить запропоновану функцію втрат ефективним інструментом для вирішення таких задач з високою якістю результатів.

3.6 Датасет

За основу набору даних, який буде використано для навчання мережі було вирішено обрати датасет DOTA [36]. Цей датасет містить 2806 аерофотознімків, зібраних з різних датчиків і платформ, кожен з яких має розмір приблизно 4000×4000 пікселів, хоча набір даних включає зображення розміром від 800×800 до 20 000×20 000 пікселів.

Зображення в наборі даних взяті з декількох джерел таких, як Google Earth, супутників GF-2 та JL-1 та інших. Набір даних містить як RGB-зображення, так і зображення в відтінках сірого, надаючи різноманітну спектральну інформацію для аналізу.

Хоча цей датасет було створено з метою використання у машинному навчанні для аналізу супутникових зображень, він містить велику кількість знімків з різними об’єктами, що дозволяє використовувати його для поставленої задачі.

Датасет включає велику кількість зображень з різними локацями такими, як аеропорти, поля, міста, порти та інші. Також відповідно попередньому пункту, у наборі даних велика кількість об’єктів, які знаходяться на зображеннях: літаки, кораблі, машини, дома та багато інших. Таке різноманіття

датасету дозволяє ефективніше тренувати модель та запобігати перенавчання для якогось конкретного випадку.

Об'єкти на зображеннях мають широкий діапазон масштабів, орієнтацій та форм, що дозволяє ефективно навчатися відновленню зображень як з великою кількістю деталей, так і маленькою деталізацією.

На рис. 3.4 – рис. 3.6 наведено приклади зображень з початкового датасету.

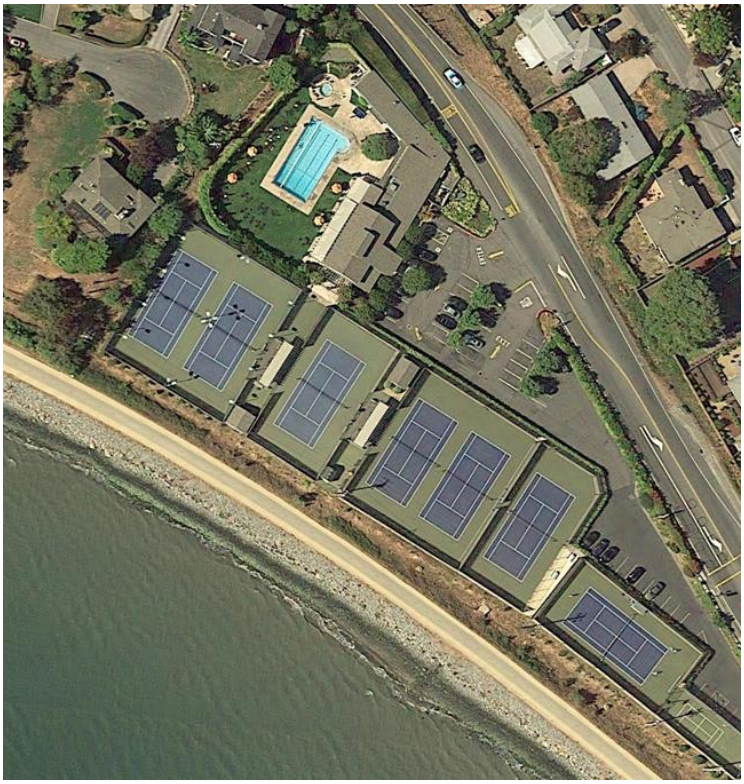

Рисунок 3.4 – Приклад зображень з датасету

Рисунок 3.5 – Приклад зображень з датасету

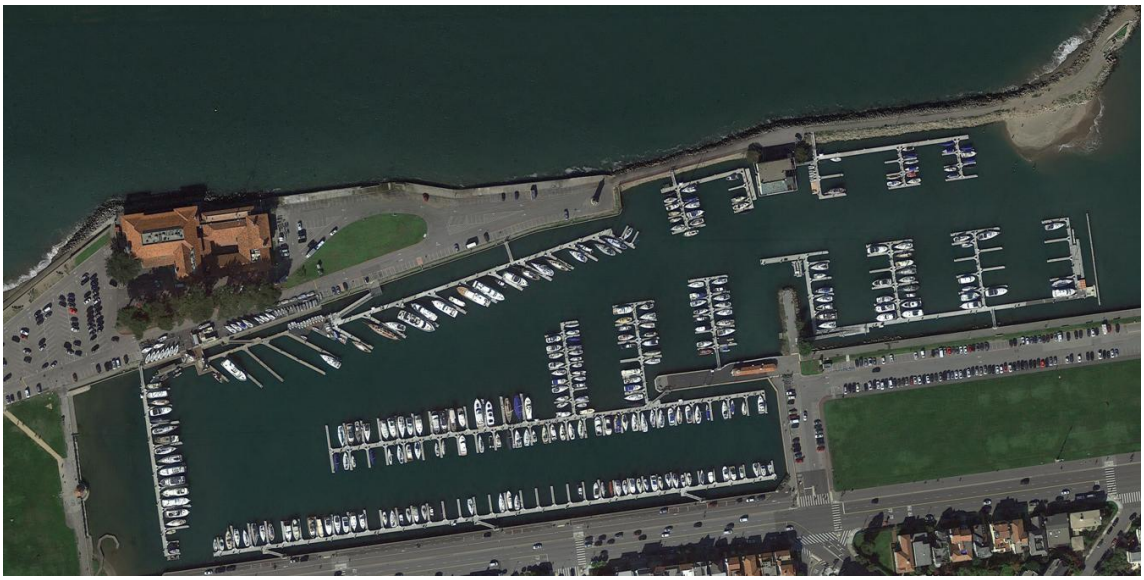

Рисунок 3.6 – Приклад зображень з датасету

Оскільки базовий датасет не повністю відповідає критеріям для зображень, на яких буде проведено навчання мережі, тому буде проведено

попередню обробку. Для кожного зображення буде проведено операцію кадрування до розміру 1024x1024 пікселя. Оскільки в наборі даних також були зображення з чорними краями (рис 3.6), то після кадрування буде також виконано перевірку, щоб уникнути проблем під час навчання. Приклади перетворених зображень наведено на рис. 3.7.

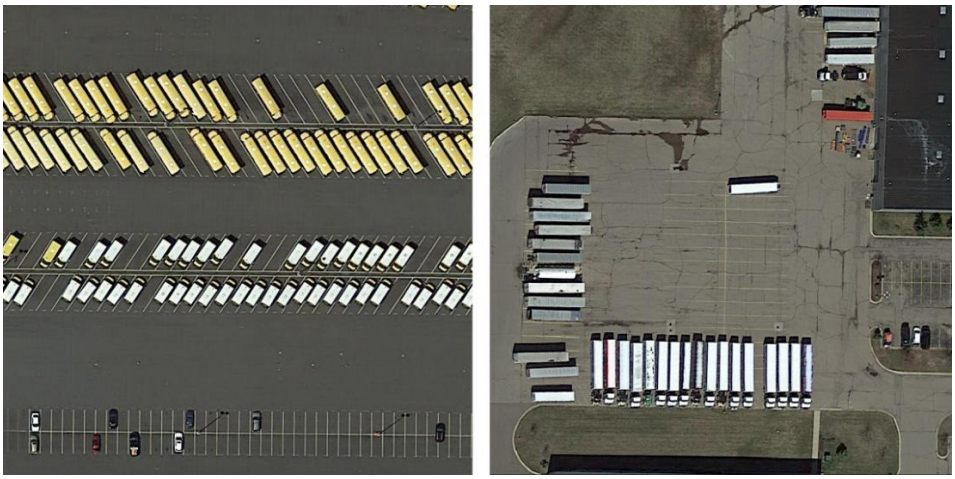

Рисунок 3.7 – Приклад перетворених зображень за датасету

Отже, отримуємо 1900 зображень розміром 1024x1024 пікселя. Остаточно розбиваємо на тренувальну, валідаційну та тестову вибірки з розмірами 70, 20 та 10 відсотків відповідно. Такий датасет дозволить ефективно проводити навчання моделі та аналізувати результати її роботи після цього.

3.7 Навчання

Генеративні змагальні мережі для збільшення роздільної здатності використовують специфічний процес навчання для створення високоякісних, фотореалістичних реконструйованих зображень. Процес навчання

відбувається за двофазною архітектурою, яка поступово нарощує функціональність моделі від базової реконструкції до прогресивного покращення реалістичності згенерованих зображень.

Навчання даної мережі складається з двох різних етапів: попереднє навчання генератора та змагальне навчання. Кожен з цих етапів є критично важливим для вдосконалення якості згенерованих зображень. Розглянемо кожен з етапів більш детально.

1. Перший етап. Навчання починається з фази ініціалізації, яка присвячена виключно навчанню мережі генератору базовим навичкам реконструкції зображень. Під час цієї фази генератор навчається відновленню роздільної здатності зображень з низькою роздільною здатністю, використовуючи лише середньоквадратичну похибку (MSE) як функцію втрат. Навчання в такий спосіб закладає базові можливості генератора відновлювати роздільну здатність зображень, поданих на вхід, перед початком фази змагального навчання. Етап попереднього навчання є критично важливою процедурою стабілізації, яка утримує генератор від створення абсолютно випадкових результатів на початку змагального навчання. Без фази попереднього навчання генератора, дискримінатор міг би легко визначити реконструйовані зображення, що призвело б до колапсу навчання.
2. Другий етап. Друга фаза включає повну динаміку навчання генеративно змагальних систем, де дискримінатор і генератор навчаються одночасно. На цьому етапі модель перетворюється з простої мережі, що масштабується, на складну систему, яка здатна генерувати реалістичні зображення з високою роздільною здатністю. Під час навчання було використано багатокомпонентну функцію втрат, яка враховує різні аспекти якості зображення. Детальна інформація про структуру використаної функції втрат

подана у попередньому підрозділі. Також різні механізми використані для досягнення стабільного та ефективного навчання даної мережі.

Планувальник швидкості навчання використовує експоненціальний спад для поступового зниження швидкості навчання, сприяючи збіжності і запобігаючи коливанням на пізніх етапах навчання.

Адаптивна оптимізація використовує оптимізатор Адама з ретельно налаштованими параметрами для забезпечення стабільного оновлення градієнта в обох мережах.

Для спостереження за стабільністю процесу навчання, використано використано мультиметричне оцінювання, яке одночасно відстежує точність відновлення та якість сприйняття, щоб забезпечити комплексне уявлення про продуктивність моделі.

Також, кожні 10 епох модель генерує зразки зображень, для візуальної оцінки прогресу і виявлення можливих проблеми, таких як збій режиму або утворення візуальних артефактів. В той самий час зберігається стан моделі для можливості відновлення навчання з будь-якого його етапу або, за потреби, для порівняння продуктивності та результатів різних етапів навчання.

Отже навчання мережі починається з простіших завдань на основі MSE і поступово вводить більш складні змагальні та перцептивні завдання, дозволяючи моделі поступово набувати можливостей.

Конкурентний баланс у змагальному навчанні підтримує рухому рівновагу, де ні дискримінатор, ні генератор не домінують над іншим, забезпечуючи постійне вдосконалення двох мереж.

Така повна методика навчання дозволяє мережі генерувати зображення високої роздільної здатності, які перевершують як кількісну оцінку, так і суб'єктивну якість, долаючи розрив між традиційними методами оптимізації та людськими візуальними уподобаннями. Результатом є ефективна система, здатна генерувати фотореалістичні високоякісні зображення з вхідних даних низької роздільної здатності.

Висновок до розділу 3

У даному розділі було представлено модифікацію генеративно змагальної мережі для збільшення роздільної здатності зображень, що базується на використанні архітектури подвійного дискримінатора. Запропонований підхід демонструє значний потенціал для подолання ключових обмежень традиційних SR-GAN методів.

Розроблена архітектура з подвійним дискримінатором успішно вирішує фундаментальну проблему балансу між глобальною зв'язністю зображення та збереженням локальних деталей. Розділення функцій між глобальним і локальним дискримінаторами дозволяє покращити стабільність навчання за рахунок більш збалансованого змагального середовища, підвищити якість згенерованих зображень через детальний аналіз як структурних особливостей, так і текстурних характеристик, а також запобігти проблемі надмірного згладжування, характерній для більшості методів збільшення роздільної здатності.

Запропонована комбінована функція втрат, що включає MSE, VGG та змагальні компоненти, забезпечує оптимальний баланс між об'єктивною точністю та суб'єктивною візуальною якістю. Двофазний процес навчання – від попереднього навчання генератора до повноцінного змагального навчання – гарантує стабільну збіжність моделі. Використання датасету DOTA з різноманітними аерофотознімками для навчання моделі дозволяє ефективно працювати з широким спектром зображень.

РОЗДІЛ 4 АНАЛІЗ РЕЗУЛЬТАТІВ

Після проведення навчання модифікованої генеративно змагальної моделі для збільшення роздільної здатності супутникових зображень, буде проведено оцінка її ефективності та аналіз результатів її роботи.

4.1 Використані бібліотеки

Дана реалізація модифікації GAN була написана з використанням таких основних бібліотек: PyTorch, Torchvision, NumPy, Matplotlib та PIL. Кожна бібліотека є важливою частиною проекту, оскільки вони надають функціонал, потрібний для роботи коду. Розглянемо детальніше кожну бібліотеку та її внесок у роботі побудованої моделі.

PyTorch – це основний пакет глибокого навчання для побудови нейронних мереж. Він надає основні інструменти, такі як тензорні обчислення, автоматичне диференціювання, будівельні блоки для нейронних мереж та методи оптимізації. PyTorch особливо добре підходить для побудови генеративно змагальних мереж завдяки динамічному графіку обчислень, який легко використовувати для побудови складних архітектур та моніторингу процесів навчання. Модулі Dataset та DataLoader автоматично обробляють пакетування та завантаження даних, що є критично важливою функцією для навчання мереж з великими наборами даних зображень.

Torchvision використовується для попередньої обробки зображень у датасеті. Ця бібліотека стандартизує вхідні зображення за допомогою таких операцій, як зміна розміру, нормалізація та перетворення між форматами. Розширення даних за допомогою перетворень збільшує ефективність навчання

мережі, забезпечуючи більшу різноманітність навчальних зразків для уникнення перенавчання.

NumPy забезпечує високошвидкісні можливості чисельних обчислень. Він є дуже потужним інструментом та використовується у інших бібліотеках, використаних у даній роботі.

Matplotlib використовується для візуалізації та спостереження за процесом навчання. Це особливо важливо, оскільки навчання може бути нестабільним, а візуалізація зразків дозволяє ефективно стежити за його якістю. На прикладах, згенерованих під час навчання, можна перевірити якість моделі, виявити збій режиму та визначити, коли слід зупинити навчання. Він також використовується для побудови графіків втрат та порівняння реальних і згенерованих зображень.

PIL (Python Imaging Library) виконує операції вводу-виводу файлів зображень, такі як завантаження, збереження та базова обробка зображень. У даному проекті ця бібліотека є важливою частиною читання навчальних зображень з диска, перетворення формату зображень та збереження згенерованих зображень.

Поєднання цих бібліотек забезпечує всі потреби для побудови модифікованої мережі та її ефективного навчання, а також для виведення результатів.

4.2 Аналіз отриманих результатів

Початковий процес навчання для генератора тривав 50 епох, після цього було проведено змагальне навчання довжиною у 1200 епох та такими заданими параметрами:

- batch size: 16;
- learning rate decay: 0.1;

- optimizer: Adam з параметрами $weight\ decay = 10^{-4}; \beta_1 = 0.9,; \beta_2 = 0.999$.

Під час навчання модель оцінювала ефективність своєї роботи кожні 5 епох на валідаційному наборі, що був утворений з 20 відсотків початкового датасету. За візуальною оцінкою модель продемонструвала достатньо гарні результати. Приклади результатів збільшення роздільної здатності зображень можна побачити на рис. 4.1.

Рисунок 4.1 – Приклад результатів збільшення роздільної здатності

На наведеному зображені ліва колонка включає в себе початкові зображення низької роздільної здатності, утворені при проектуванні датасету.

Центральна колонка включає в себе результати роботи модифікованої генеративно змагальної мережі. Права колонка включає в себе оригінальні зображення високої роздільної здатності.

Як можна побачити мережа успішно навчилася відновлювати зображення низької роздільної здатності. Незважаючи на високий фактор збільшення роздільної здатності, що складає чотирикратне збільшення початкового зображення, мережа має результати високої точності та мінімальну кількість візуальних артефактів у загальному випадку. Але модель все ще має деякі проблеми, наприклад відновлення зображень з великою кількістю малих деталей. Результат обробки таких зображень наведено на рис. 4.2.

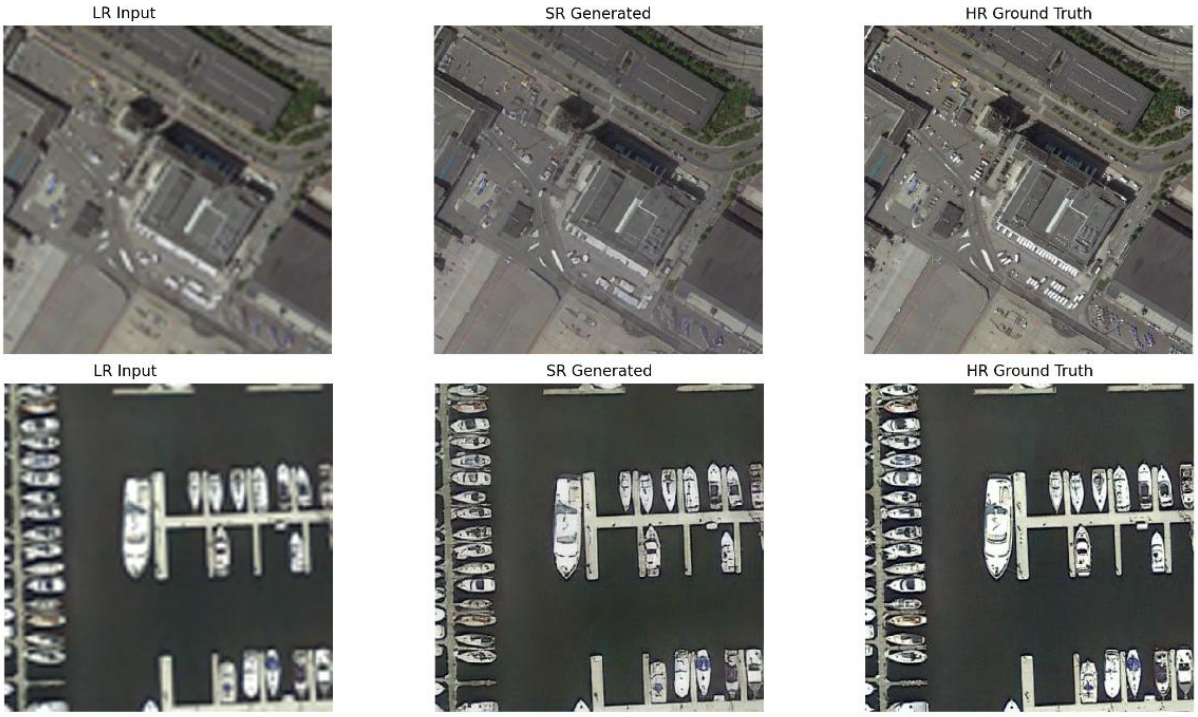

Рисунок 4.2 – Результати роботи мережі на зображеннях з великою кількістю деталей

Як можна побачити, для даного випадку, хоча мережа і надає достатньо якісні та реалістичні зображення, але все одно результат відрізняється від

зображень великої роздільної здатності. При детальному аналізі наведених зображень можна побачити достатню кількість візуальних дефектів у зонах з великою кількістю дрібних деталей.

Також під час навчання проводилися числові обчислення попередньо вказаних метрик таких, як MSE, PSNR та SSIM. Порівняння числових значень метрик на початку навчання та на останніх епохах наведено у табл. 4.1.

Таблиця 4.1 – Порівняння значень метрик

Епоха	MSE	PSNR	SSIM
10	0.0142	24.72 dB	0.6491
200	0.0105	25.56 dB	0.6978
1200	0.0082	27.06 dB	0.7696

Як можна побачити, під час фази тренування відбувалося поступове покращення всіх метрик, що є показником збалансованого навчання. Для завдання збільшення роздільної здатності зображень за фактором 4, такі значення для всіх метрик є гарними результатами.

Далі модель було перевірено на тестовій вибірці для остаточного аналіз її продуктивності. Результати обробки супутникових знімків наведені нижче, на рис. 4.3.

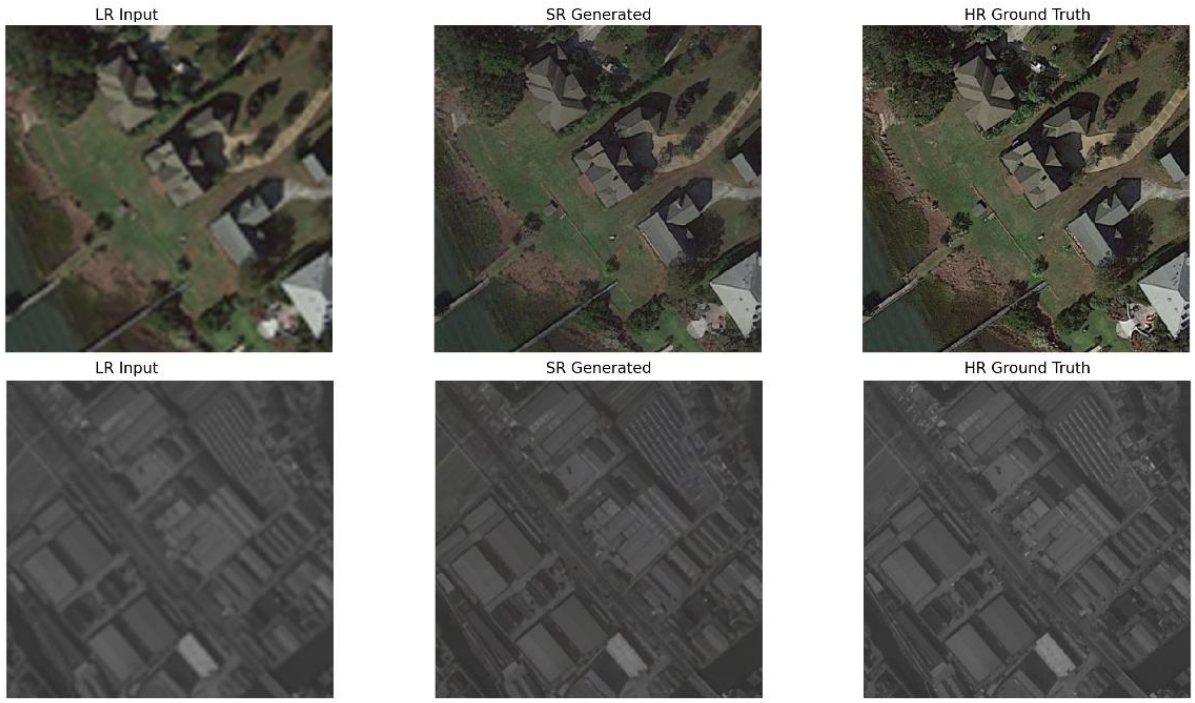

Рисунок 4.3 – Приклад роботи моделі на тестовій вибірці

Як можна побачити, модель має однакові результати на тестовій та тренувальних вибірках. Це означає, що процес навчання моделі було проведено успішно та модель видає стабільні результати. Також можна побачити, що вдалося запобігти перенавчанню моделі. Для остаточного підтвердження оцінені метрики моделі на тестовому датасеті наведені у табл. 4.2.

Таблиця 4.2 – Значення метрик для тестової вибірки

MSE	PSNR	SSIM
0.0089	26.95 dB	0.7663

Отримані результати підтверджують ефективність розробленої модифікованої генеративно змагальної мережі для задачі збільшення роздільної здатності супутникових зображень. Близькість значень метрик між тренувальною та тестовою вибірками свідчить про хорошу узагальнюючу здатність моделі та відсутність перенавчання. Незважаючи на виявлені

обмеження у відновленні дрібних деталей на зображеннях високої складності, модель демонструє стабільну якість роботи та може бути рекомендована для практичного застосування в задачах покращення якості супутникових знімків з чотирикратним збільшенням роздільної здатності.

4.3 Порівняння моделей

Щоб підтвердити ефективність запропонованої моделі з подвійним дискримінатором, було проведено додаткове порівняння з версією мережі, що містить тільки глобальний дискримінатор. Такий аналіз дозволяє оцінити внесок локального дискримінатора в загальну продуктивність модифікованої генеративно змагальної мережі.

У порівнянні приймали участь дві моделі:

- модель А: вже протестована модель з подвійним дискрімінатором;
- модель Б: генеративно змагальна мережа тільки з глобальним дискрімінатором.

Для точності експерименту, обидві моделі мають однакову конфігурацію та були натреновані однакову кількість епох.

Провівши навчання мережі тільки з глобальним дискримінатором протягом 1200 епох було отримано результати, що наведені у табл. 4.3.

Таблиця 4.3 – Результати навчання моделі Б

Епоха	MSE	PSNR	SSIM
10	0.0200	23.05 dB	0.6001
200	0.0203	22.98 dB	0.5867
1200	0.0240	22.24 dB	0.4656

Як можна побачити, з збільшенням кількості епох, значення метрик не змінилось суттєво, а в кінці навіть погіршилось. Такі результати свідчать про нестабільність навчання та неможливість досягнути гарних результатів.

Для повного аналізу проведемо порівняння результатів для обох моделей. Значення метрик для обох моделей після навчання на 1200 епох наведено у табл.4.4.

Таблиця 4.4 – Порівняння результатів навчання моделей

Модель	MSE	PSNR	SSIM
А	0.0082	27.06 dB	0.7696
Б	0.0240	22.24 dB	0.4656
	-65%	21%	40%

Результати чітко вказують на те, що модель А перевершує модель Б за всіма метриками. Окрім того, різниця в отриманих значеннях є досить великою. Для повного підтвердження, розглянемо результати, отримані для моделі Б, які зображені на рис.4.4.

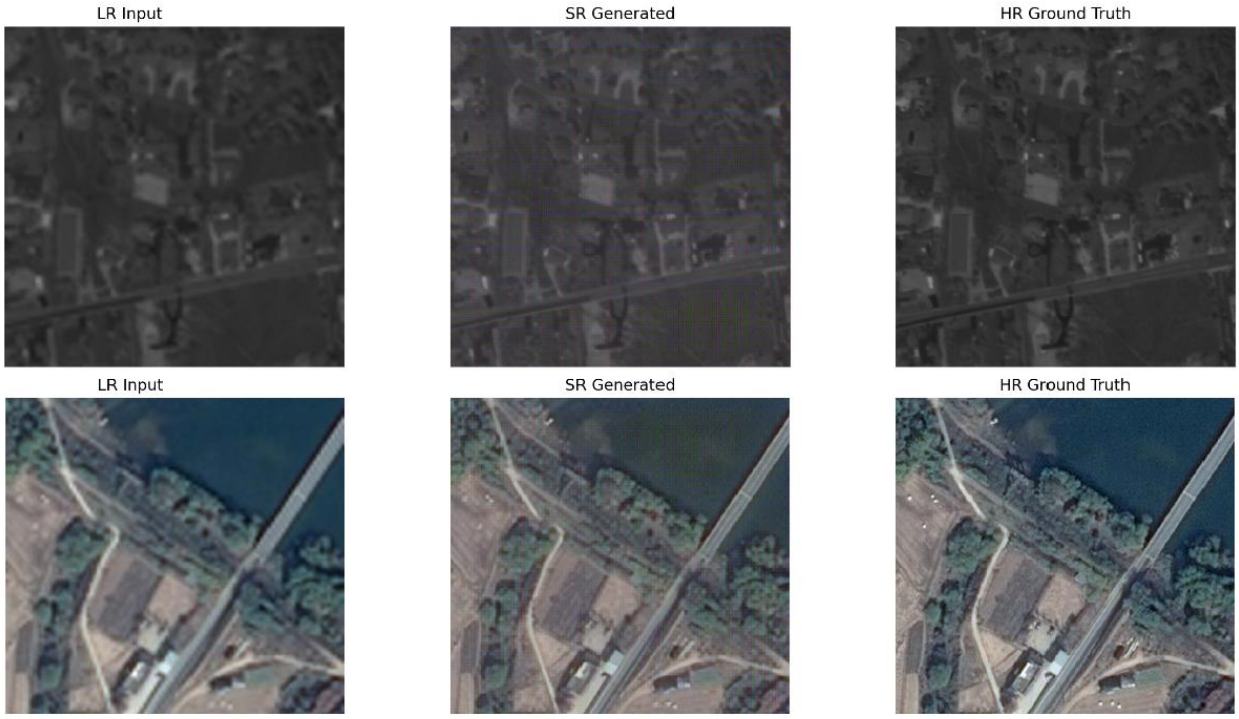

Рисунок 4.4 – Результати роботи моделі тільки з глобальним дискримінатором

Як можна побачити на зображеннях, результати відновлення є набагато гіршими за модель з подвійним дискримінатором. Мережа погано відновлює не тільки малі деталі, а й зображення у загальному. Окрім цього, текстура зображення має багато артефактів та не відображає оригінальну текстуру зображення з великою роздільною здатністю.

Проведене порівняння демонструє результативність запропонованої модифікованої архітектури генеративно змагальної мережі з подвійним дискримінатором. Експериментальні результати підтверджують значну перевагу такої моделі над моделлю тільки з глобальним дискримінатором за всіма ключовими метриками якості.

Аналіз отриманих результатів підтвердив ефективність такої архітектури. Таким чином глобальний дискримінатор забезпечує загальну відповідність зображення, тоді як локальний дискримінатор концентрується на точності відтворення локальних деталей та текстур. Ефект від їх спільної

роботи забезпечує вищу якість кінцевого результату порівняно з традиційними підходами.

Висновки до розділу 4

Запропонована модифікована генеративно змагальна мережа продемонструвала високу ефективність у задачі збільшення роздільної здатності супутникових зображень. Модель успішно навчилася відновлювати деталі та текстури на зображеннях низької роздільної здатності. Протягом процесу навчання вдалось досягти значних покращень у всіх ключових метриках якості: MSE знизилася з 0.0142 до 0.0082, PSNR змінився з 24.72 dB до 27.06 dB, а SSIM збільшився з 0.6491 до 0.7696. Ці результати свідчать про стабільне та збалансоване навчання моделі.

Порівняння результатів на тренувальній та тестовій вибірках підтверджує хорошу узагальнюючу здатність розробленої моделі. Також, близькість значень метрик свідчать про відсутність перенавчання та стабільність роботи моделі на нових даних.

Незважаючи на загальну високу якість результатів, було виявлено певні обмеження моделі. Під час відновлення зображень з великою кількістю дрібних деталей, спостерігається наявність візуальних артефактів та деякі розбіжності з оригінальними зображеннями високої роздільної здатності.

Порівняння моделей показало, що використання локального дискримінатора в поєднанні з глобальним забезпечує покращення показників на 65% за метрикою MSE, на 21% за PSNR та на 40% за SSIM. Такі суттєві відмінності свідчать про важливість локального дискримінатора для досягнення високої якості відновлення зображень.

РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі буде здійснено аналіз ключових характеристик майбутнього вбудованого програмного забезпечення. Дослідження також охоплює різні варіанти реалізації, з метою вибору найбільш ефективної та доцільної стратегії, яка впливає як на економічні аспекти, так і на сумісність із майбутнім продуктом. Для цього використано метод функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це методологія, що дає змогу об'єктивно оцінити фактичну вартість продукту або послуги незалежно від структури підприємства. Основна мета ФВА полягає у виявленні шляхів зниження витрат за допомогою ефективніших рішень, що забезпечують оптимальне співвідношення між споживчою цінністю продукту та витратами на його створення. Для аналізу використовуються дані економічного, технічного та інженерного характеру.

Процес ФВА включає визначення послідовних етапів розробки програмного продукту, розрахунок повних (річних) витрат і робочого часу, ідентифікацію джерел витрат та остаточне обчислення вартості розробки.

5.1 Постановка задачі проектування

У роботі використовується метод функціонально-вартісного аналізу (ФВА) для проведення техніко-економічного обґрунтування розробки системи прогнозування стійкості фінансових показників. Оскільки рішення щодо проєктування та реалізації окремих компонентів впливають на функціонування всієї системи, кожна з підсистем повинна відповідати

загальним вимогам. Таким чином, аналіз зосереджений на функціональному призначенні програмного забезпечення, що створюється для збору, обробки та аналізу даних по компанії.

Технічні вимоги до програмного продукту є такими:

- висока якість збільшення роздільної здатності зображень;
- функціонування на персональних комп'ютерах із звичайним набором компонентів;
- швидкість навчання та обробки даних моделі;
- можливість масштабування та обслуговування;
- розробка продукту має бути з мінімальними витратами на розробку та подальше обслуговування.

5.2 Обґрунтування функцій програмного продукту

Головна функція F_0 відповідає за розробку програмного продукту, який підтримує процес збільшення роздільної здатності зображень та дозволяє аналізувати ключові характеристики, що впливають на стійкість підприємства. На основі цієї функції можна виокремити такі підфункції:

- F_1 – вибір мови програмування;
- F_2 – вибір фреймворку для розробки генеративно змагальної мережі;
- F_3 – вибір середовища розробки

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python;

б) C++.

Функція F_2:

а) Torch;

б) Tensorflow.

Функція F_3:

а) Pycharm;

б) Visual studio.

Варіанти реалізації основних функцій наведені у морфологічній карті системи, зображеній на рис. 5.1.

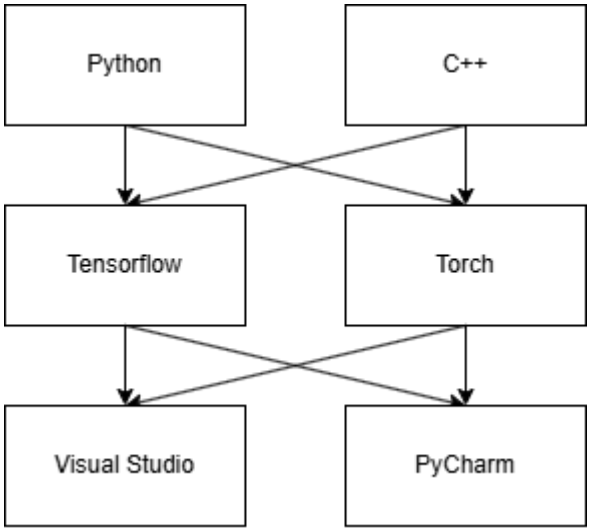

Рисунок 5.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіанти основних функцій. Тепер є можливість побудувати позитивно-негативну матрицю з відображенням усіх варіантів основних функцій наведену у таблиці 5.1.

Таблиця 5.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	А	Простий та легкий для вивчення синтаксис, велика кількість бібліотек	Низька швидкість роботи через те, що це інтерпретована мова
	Б	Висока продуктивність, має контроль над пам’яттю	Складний синтаксис, велика ймовірність виникнення помилок при розробці
F_2	А	Зручний та інтуїтивно зрозумілий API, підходить для досліджень та розробки прототипів	Значно менше оптимізований, можуть виникати проблеми з стабільністю
	Б	Велика екосистема рішень, можна працювати як з CPU, так і з GPU	Досить складний у розумінні, немає механізмів спрощення процесу дебагу
F_3	А	Спеціалізовано заточений для python, є багато в будованих потужних інструментів для спрощення задач	Важкий та потребує багато ресурсів, через фокус на мові python менш зручний для інших мов
	Б	Спеціалізовано заточений для C++, велика кількість розширень	Важкий та потребує багато ресурсів, через фокус на мові C++ менш зручний для інших мов

На основі аналізу позитивно-негативної матриці можна зробити висновок, що при розробці програмного продукту деякі варіанти реалізації функцій не відповідають поставленим перед програмним продуктом задачам, тому їх потрібно відкинути.

Функція F_1. У порівнянні мова Python є набагато кращою для розробки потрібного продукту за рахунок спрощеного синтаксису та великої кількості доступних бібліотек.

Функція F_2. Обидва варіанти задовольняють потреби завдання, оскільки вони надають хорошу базу для розробки, але кожен має свою плюси та мінуси.

Функція F_3. Оскільки вибір для функція F_1 є мова Python, то логічним вибором буде варіант заточений для розробки на цій мові – Pycharm.

Отже, буде розглядатися такий варіанти реалізації ПП:

$$F_1\text{а} - F_2\text{а} - F_3\text{а},$$
$$F_1\text{а} - F_2\text{б} - F_3\text{а}.$$

Для оцінювання якості розглянутих функцій буде використана описана нижче система параметрів.

5.3 Обґрунтування системи параметрів програмного продукту

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

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

- X_1– швидкодія мови програмування;
- X_2 – об’єм пам’яті для обчислень та збереження даних;
- X_3 – час навчання даних;

- X_4 – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у табл. 5.2.

Таблиця 5.2 – Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X_1	оп/мс	9000	12000	15000
Об’єм пам’яті	X_2	Мб	512	256	128
Час попередньої обробки даних	X_3	мс	10	5	2
Потенційний об’єм програмного коду	X_4	Кількість рядків коду	4000	2000	1000

За даними табл. 5.2 будуються графічні характеристики параметрів – рис. 5.2 – рис. 5.5.

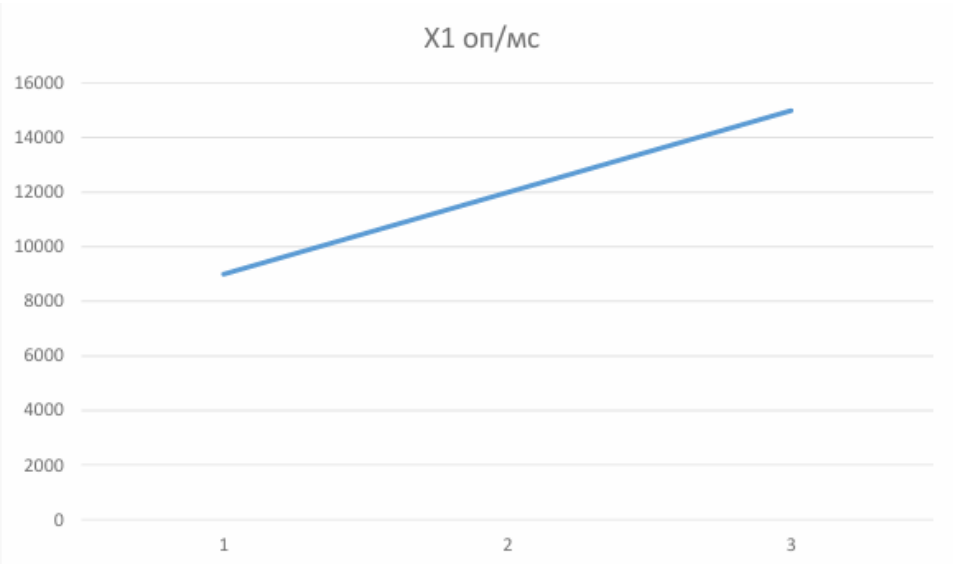

Рисунок 5.2 – X_1, швидкодія мови програмування

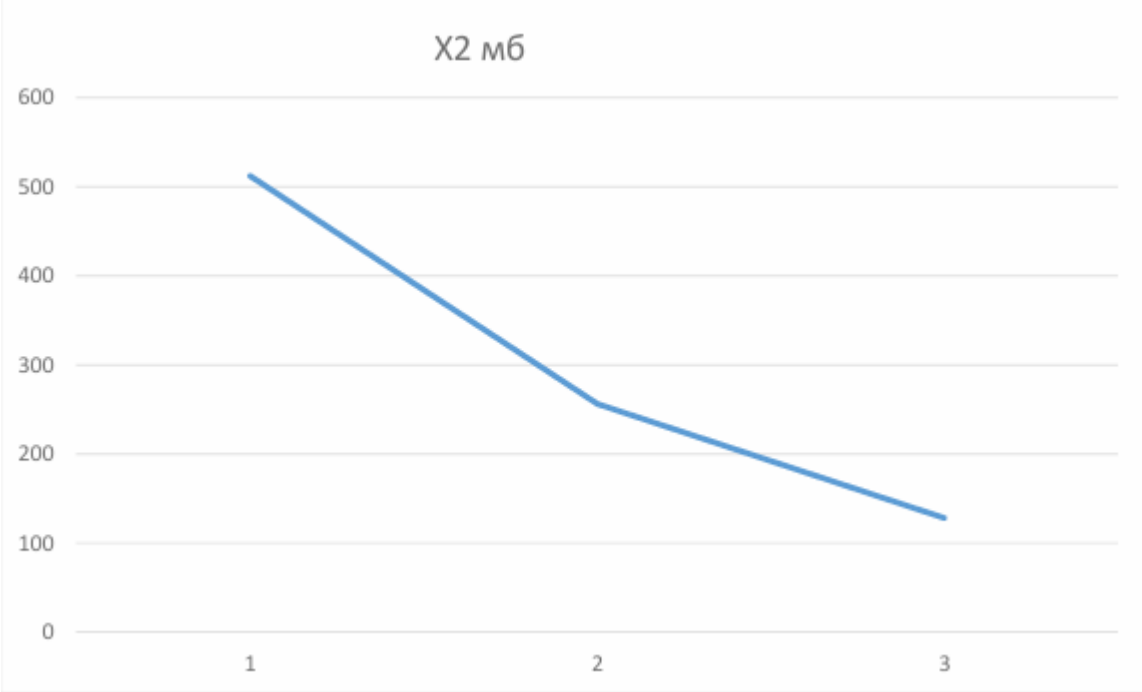

Рисунок 5.3 – X_2, об’єм пам’яті

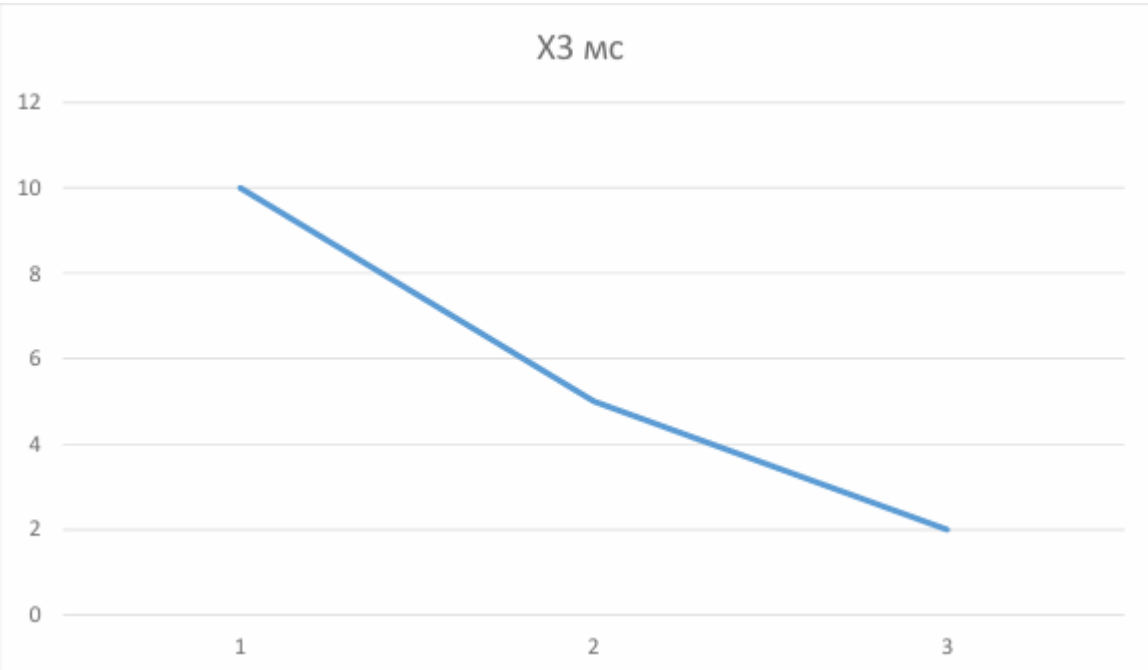

Рисунок 5.4 – X_3, час попередньої обробки даних

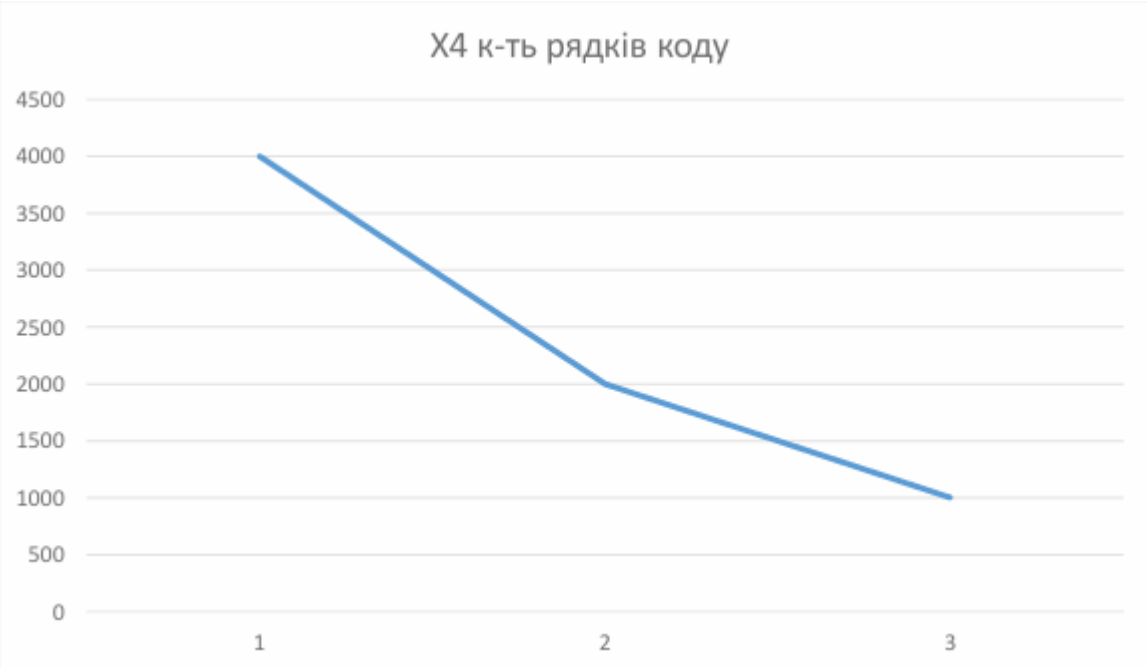

Рисунок 5.5 – X_4, потенційний об’єм програмного коду

5.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;

- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у табл. 5.3.

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

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

$$\mathrm{R_i} = \sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{r_{ij}} \mathrm{R_{ij}} = \frac{\mathrm{Nn(n+1)}}{2} = 77,$$

де N – число експертів;

n – кількість параметрів.

Таблиця 5.3 – Результати ранжування параметрів

Позначення параметра	*Назва параметра*	*Одиниці виміру*	*Ранг параметра за оцінкою експерта*							*Сума рангів* R_i	*Відхилення* Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X_1	Швидкодія мови програмування	Оп/мс	1	2	3	3	2	1	2	14	-5.25	27.562
X_2	Об’єм пам’яті	Mb	4	4	4	3	5	4	5	29	9.75	95.062
X_3	Час попередньої обробки даних	мс	2	2	1	2	1	2	1	11	-8.25	68.062
X_4	Потенційний об’єм програмного коду	Кількість рядків коду	4	3	3	3	3	4	3	23	3.75	14.062
	Разом		11	11	11	11	11	11	11	77	0	204.75

б) середня сума рангів:

$$\mathrm{T} = \frac{1}{\mathrm{n}} \mathrm{R_{ij}} = 19{,}25.$$

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

$$\Delta_i = R_i - T.$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 204{,}75.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 204{,}75}{7^2(4^3 - 4)} = 0{,}83 > W_k = 0{,}67.$$

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

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у табл. 5.4.

Таблиця 5.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 і X_2	<	<	<	=	<	<	<	<	0.5
X_1 і X_3	<	=	>	>	>	<	>	>	1.5
X_1 і X_4	<	<	=	=	<	<	<	<	0.5
X_2 і X_3	>	>	>	>	>	>	>	>	1.5
X_2 і X_4	=	>	>	=	>	=	>	>	1.5
X_3 і X_4	<	<	<	<	<	<	<	<	0.5

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

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

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

Для кожного параметра зробимо розрахунок вагомості $K_{\text{в}i}$ за наступними формулами:

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

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

Відносні оцінки розраховуються декілька разів до моменту, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$\mathrm{K_{вi}} = \frac{\mathrm{b_i'}}{\sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{b_i'}},$$

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

Як видно з табл. 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітерація		Друга ітерація		Третя ітерація	
	X_1	X_2	X_3	X_4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X_1	1	0.5	1.5	0.5	3.5	0.22	12.25	0.21	44.875	0.21
X_2	1.5	1	1.5	1.5	5.5	0.34	21.25	0.36	77.875	0.36
X_3	0.5	0.5	1	0.5	2.5	0.16	9.25	0.16	34.125	0.16
X_4	1.5	0.5	1.5	1	4.5	0.28	16.25	0.27	59.125	0.27
					16	1	59	1	216	1

5.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X_2 (об'єм пам'яті), X_3 (час попередньої обробки даних) та X_4 (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X_1 (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (табл. 5.6):

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

де n – кількість параметрів;

$K_{\text{ві}}$ – коефіцієнт вагомості і–го параметра;

B_i – оцінка і–го параметра в балах.

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	*Варіант реалізації функції*	*Параметри*	*Абсолютне значення параметра*	*Бальна оцінка параметра*	*Коефіцієнт вагомості параметра*	*Коефіцієнт рівня якості*
F_1	А	X_1	12000	19	0.21	3.99
F_2	А	X_2	128	22	0.36	7.92
	Б	X_3	2	18	0.16	2.88
F_3	А	X_4	2000	25	0.27	6.75

За даними з табл. 5.6 та за формулою:

$$\mathrm{K_K} = \mathrm{K_{TY}[F_{1k}]} + \mathrm{K_{TY}[F_{2k}]} + \cdots + \mathrm{K_{TY}[F_{zk}]},$$

визначаємо рівень якості кожного з варіантів:

$$\mathrm{K_{Kl}} = 3{,}99 + 7{,}92 + 6{,}75 = 18{,}66,$$
$$\mathrm{K_{K2}} = 3{,}99 + 2{,}88 + 6{,}75 = 13{,}62.$$

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

5.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання.

1. Розробка проекту програмного продукту.
2. Розробка програмної оболонки.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

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

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_О = T_Р \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М},$$

де $T_Р$ – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

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

$K_М$ – коефіцієнт рівня мови програмування;

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

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

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_Р$ =35 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.9. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.7. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 35 \cdot 1.9 \cdot 0.7 = 46.55 \text{ людино} - \text{днів}.$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_Р = 30$ людино-днів, $К_П = 1.2$, $К_{СК} = 1$, $К_{СТ} = 0.8$:

$$T_1 = 30 \cdot 1.2 \cdot 0.8 = 28.8 \text{ людино} - \text{днів}.$$

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

$$T_I = (46{,}55 + 28{,}8 + 4{,}72 + 28{,}8) \cdot 8 = 870{,}96 \text{ людино} - \text{днів},$$
$$T_{II} = (46{,}55 + 28{,}8 + 6{,}35 + 28{,}8) \cdot 8 = 884 \text{ людино} - \text{днів}.$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 34698 грн., один аналітик в області даних з окладом 47709. Визначимо середню зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} \text{ грн},$$

де М – місячний оклад працівників;

T_m – кількість робочих днів на тиждень;

t – кількість робочих годин в день.

$$С_ч = \frac{34698+34698+47709}{3 \cdot 21 \cdot 8} = 232{,}35.$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{ЗП} = C_ч \cdot T_i \cdot K_д,$$

де $C_ч$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$K_д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$I.\ C_{ЗП} = 232.35 \cdot 870.96 \cdot 1.2 = 242841.07 \text{ грн.}$$
$$II.\ C_{ЗП} = 2232.35 \cdot 884 \cdot 1.2 = 246476.88 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$I.\ C_{ВІД} = C_{ЗП} \cdot 0.22 = 242841.07 \cdot 0.22 = 53425.04 \text{ грн.}$$
$$II.\ C_{ВІД} = C_{ЗП} \cdot 0.22 = 246476.88 \cdot 0.22 = 54224.91 \text{ грн.}$$

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

Так як одна ЕОМ обслуговує одного програміста з окладом 34698 грн., з коефіцієнтом зайнятості 0.2 то для однієї машини отримаємо:

$$C_Г = 12 \cdot М \cdot K_З = 12 \cdot 34698 \cdot 0.2 = 83275.2 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{ЗП} = C_Г \cdot (1 + K_З) = 83275.2 \cdot (1 + 0.2) = 99930.24 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{ВІД} = C_{ЗП} \cdot 0.22 = 99930.24 \cdot 0.22 = 21984.65 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 28000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.2 \cdot 0.25 \cdot 28000 = 8400 \text{ грн},$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_р = 1.2 \cdot 28000 \cdot 0.05 = 1680 \text{ грн},$$

де $К_р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t_З \cdot К_В = (365 - 104 - 11 - 14) \cdot 8 \cdot 0.8 = 1510.4 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1510.4 \cdot 0.5 \cdot 0.3 \cdot 9.43 = 2136.46 \text{ грн},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_{А} + С_{Р} + С_{ЕЛ} + С_{Н},$$
$$С_{ЕКС} = 99930.24 + 21984.65 + 8400 + 1680 + 2136.46 + 18760 = 153491.35 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М-Г} = \frac{С_{ЕКС}}{Т_{ЕФ}} = \frac{153491,35}{1510.4} = 101.62 \text{ грн/год.}$$

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

$$С_{М} = С_{М-Г} \cdot Т,$$
$$\text{I. } С_{М} = 101.62 \cdot 870.96 = 88509.55 \text{ грн.}$$
$$\text{II. } С_{М} = 101.62 \cdot 884 = 89832.08 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$С_{Н} = С_{ЗП} \cdot 0.67,$$
$$\text{I. } С_{Н} = 242841.07 \cdot 0.67 = 162703.52 \text{ грн.}$$
$$\text{II. } С_{Н} = 246476.88 \cdot 0.67 = 165139.51 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н},$$
$$\text{I. } С_{ПП} = 242841,07 + 53425,04 + 88509.55 + 162703,52 = 547479,18 \text{ грн.}$$
$$\text{II. } С_{ПП} = 246476,88 + 54224,91 + 89832.08 + 165139,51 = 555673,38 \text{ грн.}$$

5.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj}/С_{Фj},$$

$$К_{ТЕР1} = 18{,}66/546465{,}27 = 3{,}4147 \cdot 10^{-5},$$

$$К_{ТЕР2} = 13{,}62/554647{,}74 = 2{,}4556 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 3{,}4147 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР2} = 2{,}4556 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- фреймворку машинного навчання – Torch;
- середовище програмування – PyCharm.

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

Висновки до розділу 5

Була поставлена задача проектування програмного продукту, де були визначені цілі та область його застосування. Обґрунтовані функції

програмного продукту, визначено необхідні можливості та сервіси для користувачів. Виконано обґрунтування системи параметрів програмного продукту, включаючи ключові налаштування та вимоги до продукту.

Проведено аналіз експертного оцінювання параметрів, де експерти оцінили важливість окремих параметрів та їх вплив на функціональність продукту. Здійснено аналіз рівня якості варіантів реалізації функцій, порівняно різні варіанти реалізації та оцінено їх ефективність. Проведено економічний аналіз варіантів розробки програмного продукту, включаючи оцінку фінансових витрат та прибутковості, здійснено вибір кращого варіанту програмного продукту на основі техніко-економічного аналізу

ВИСНОВКИ

У ході виконання роботи було проведено аналіз предметної області, а саме збільшення роздільної здатності супутникових знімків. Було розглянуто альтернативні методи, їхні недоліки та остаточно обрано генеративно змагальні мережі для подальшого аналізу.

Було запропоновано новий метод збільшення роздільної здатності супутникових зображень із використанням моделі генеративної змагальної мережі з подвійними дискримінаторами. Конструкція з подвійним дискримінатором вирішує існуючи проблеми поліпшення якості супутникових зображень за допомогою використання спеціалізованих дискримінаторів для різних ознак якості зображень.

Перший дискримінатор спеціалізується на глобальній структурній цілісності та просторових взаємозв'язках, а другий дискримінатор аналізує дрібні локальні деталі. Поєднання цих двох мереж дає змогу отримати зображення з високою роздільною здатністю, які зберігають загальну візуальну якість та реалістичність.

Модель подвійного дискримінатора має покращену стабільність навчання, зменшуючи випадки колапсу режиму. Також навчання моделі було проведено на датасеті з великою кількістю варіацій зображень, включаючи різні рельєфи та об’єкти, тому модель має підвищену стабільність роботи в різних умовах супутникових зображень, включаючи зміну умов освітлення, сезонні коливання та географічний рельєф.

За експериментальними результатами на як тренувальному, так і тестовому наборах супутникових зображень, мережа демонструє гарні значення метрик оцінки якості, наприклад для тестової вибірки було отримано такі значення: MSE – 0.0089, PSNR – 26.95dB, SSIM – 0.7663. Також згенеровані зображення виглядають реалістично за візуальною оцінкою. Отже

мережа показала хороші результати для чотирикратного фактора збільшення роздільної здатності.

Практичне значення цього дослідження є досить великим, маючи потенційну користь для моніторингу навколишнього середовища, міського планування, реагування на надзвичайні ситуації та оцінки врожаю, де супутникові знімки з високою роздільною здатністю мають велике значення для процесу прийняття рішень.

У висновку, архітектура генеративно змагальної мережі з подвійним дискримінатором є ефективним методом збільшення роздільної здатності зображень, забезпечуючи стабільну модель, яка балансує між обчислювальною ефективністю та покращеною якістю зображення.

СПИСОК ЛІТЕРАТУРИ

1. URL: https://uk.wikipedia.org/wiki/Обробка_зображень
2. Keys, R. (2003). Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, *29*(6), 1153-1160.
3. Press, W. H. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.
4. Greenspan, H. (2009). Super-resolution in medical imaging. *The computer journal*, *52*(1), 43-63.
5. Deudon, M., Kalaitzis, A., Goytom, I., Arefin, M. R., Lin, Z., Sankaran, K., ... & Bengio, Y. (2020). Highres-net: Recursive fusion for multi-frame super-resolution of satellite imagery. *arXiv preprint arXiv:2002.06460*.
6. Gohshi, S. (2015, July). Real-time super resolution algorithm for security cameras. In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)* (Vol. 5, pp. 92-97). IEEE.
7. Vo, K. D., & Bui, L. T. (2023). StarSRGAN: Improving real-world blind super-resolution. *arXiv preprint arXiv:2307.16169*.
8. Lu, Z., Li, J., Liu, H., Huang, C., Zhang, L., & Zeng, T. (2022). Transformer for single image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 457-466).
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139-144.
10. Murtaza, H., Ahmed, M., Khan, N. F., Murtaza, G., Zafar, S., & Bano, A. (2023). Synthetic data generation: State of the art in health care domain. *Computer Science Review*, *48*, 100546.

11. Mariani, G., Scheidegger, F., Istrate, R., Bekas, C., & Malossi, C. (2018). Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*.
12. Ghane, S., Kulik, L., & Ramamohanarao, K. (2019). TGM: A generative mechanism for publishing trajectories with differential privacy. *IEEE Internet of Things Journal*, *7*(4), 2611-2621.
13. Faisal, F., Mohammed, N., Leung, C. K., & Wang, Y. (2022, October). Generating privacy preserving synthetic medical data. In *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)* (pp. 1-10). IEEE.
14. Assefa, S. A., Dervovic, D., Mahfouz, M., Tillman, R. E., Reddy, P., & Veloso, M. (2020, October). Generating synthetic data in finance: opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance* (pp. 1-8).
15. Sikka, A., Virk, J. S., & Bathula, D. R. (2021). MRI to PET cross-modality translation using globally and locally aware GAN (GLA-GAN) for multi-modal diagnosis of Alzheimer's disease. *arXiv preprint arXiv:2108.02160*.
16. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8110-8119).
17. Wang, X., Sun, L., Chehri, A., & Song, Y. (2023). A review of GAN-based super-resolution reconstruction for optical remote sensing images. *Remote Sensing*, *15*(20), 5062.
18. Foun, M. H. Application and Analysis of Black and White Image Coloring Based on Generative Adversarial Networks (GANs).
19. Dundar, A., Gao, J., Tao, A., & Catanzaro, B. (2023). Progressive learning of 3d reconstruction network from 2d gan data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(2), 793-804.

20. Chira, D., Haralampiev, I., Winther, O., Dittadi, A., & Liévin, V. (2022, October). Image super-resolution with deep variational autoencoders. In *European Conference on Computer Vision* (pp. 395-411). Cham: Springer Nature Switzerland.
21. Li, K., Zhu, Y., Yang, J., & Jiang, J. (2016). Video super-resolution using an adaptive superpixel-guided auto-regressive model. *Pattern Recognition*, *51*, 59-71.
22. Li, H., Yang, Y., Chang, M., Chen, S., Feng, H., Xu, Z., ... & Chen, Y. (2022). Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, *479*, 47-59.
23. Saxena, D., & Cao, J. (2021). Generative adversarial networks (GANs) challenges, solutions, and future directions. *ACM Computing Surveys (CSUR)*, *54*(3), 1-42.
24. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4681-4690).
25. Xiong, Y., Guo, S., Chen, J., Deng, X., Sun, L., Zheng, X., & Xu, W. (2020). Improved SRGAN for remote sensing image super-resolution across locations and sensors. *Remote Sensing*, *12*(8), 1263.
26. Nagano, Y., & Kikuta, Y. (2018, July). SRGAN for super-resolving low-resolution food images. In *Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management* (pp. 33-37).
27. Liu, B., & Chen, J. (2021). A super resolution algorithm based on attention mechanism and srgan network. *IEEE Access*, *9*, 139138-139145.
28. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., ... & Change Loy, C. (2018). Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops* (pp. 0-0).

29. Jiang, X., Xu, Y., Wei, P., & Zhou, Z. (2020, May). CT image super resolution based on improved SRGAN. In *2020 5th international conference on computer and communication systems (ICCCS)* (pp. 363-367). IEEE.
30. Dou, X., Li, C., Shi, Q., & Liu, M. (2020). Super-resolution for hyperspectral remote sensing images based on the 3D attention-SRGAN network. *Remote Sensing*, *12*(7), 1204.
31. Dong, C., Loy, C. C., He, K., & Tang, X. (2015). Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, *38*(2), 295-307.
32. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., ... & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1874-1883).
33. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
34. Gatys, L., Ecker, A. S., & Bethge, M. (2015). Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, *28*.
35. Ruby, U., & Yendapalli, V. (2020). Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, *9*(10).
36. Xia, G. S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., ... & Zhang, L. (2018). DOTA: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3974-3983).
37. Zhang, Y., Zhao, D., Zhang, J., Xiong, R., & Gao, W. (2011). Interpolation-dependent image downsampling. *IEEE Transactions on Image Processing*, *20*(11), 3291-3296.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

config.py

```
from easydict import EasyDict as edict

config = edict()
config.TRAIN = edict()

config.TRAIN.batch_size = 16
config.TRAIN.lr_init = 1e-4
config.TRAIN.beta1 = 0.9

config.TRAIN.n_epoch_init = 50

config.TRAIN.n_epoch = 2000
config.TRAIN.lr_decay = 0.1
config.TRAIN.decay_every = int(config.TRAIN.n_epoch / 2)

config.TRAIN.hr_img_path = 'dataset/train/'
config.TRAIN.shuffle_buffer_size = True

config.VALID = edict()
config.VALID.hr_img_path = 'dataset/valid/'
```

datalodader.py

```
import os
import glob
import numpy as np
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as transforms
```

```
from PIL import Image

from config import config


class TrainDataset(Dataset):
    def __init__(self, hr_img_path):
        super(TrainDataset, self).__init__()

        self.hr_image_paths = sorted(glob.glob(os.path.join(hr_img_path, '*.png')))
        self.hr_transforms = transforms.Compose([
            transforms.RandomCrop((384, 384)),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),  # Converts to [0,1]
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
        ])

        self.lr_transforms = transforms.Resize((96, 96),
interpolation=transforms.InterpolationMode.BICUBIC)

    def __getitem__(self, index):
        hr_img = Image.open(self.hr_image_paths[index]).convert('RGB')

        hr_patch = self.hr_transforms(hr_img)
        lr_patch = self.lr_transforms(hr_patch)

        return lr_patch, hr_patch

    def __len__(self):
        return len(self.hr_image_paths)


def get_train_data():
    train_dataset = TrainDataset(hr_img_path=config.TRAIN.hr_img_path)
    batch_size = config.TRAIN.batch_size
```

```
    train_loader = DataLoader(
        dataset=train_dataset,
        batch_size=batch_size,
        shuffle=True,
        num_workers=0,
        drop_last=True,
        pin_memory=True
    )

    return train_loader


class ValDataset(Dataset):
    def __init__(self, hr_img_path):
        super(ValDataset, self).__init__()

        self.hr_image_paths = sorted(glob.glob(os.path.join(hr_img_path, '*.png')))
        self.hr_transforms = transforms.Compose([
            transforms.RandomCrop((384, 384)),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),  # Converts to [0,1]
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))  # Normalizes to [-1,1]
        ])

        self.lr_transforms = transforms.Resize((96, 96),
interpolation=transforms.InterpolationMode.BICUBIC)

    def __getitem__(self, index):
        hr_img = Image.open(self.hr_image_paths[index]).convert('RGB')

        hr_patch = self.hr_transforms(hr_img)
        lr_patch = self.lr_transforms(hr_patch)

        return lr_patch, hr_patch
```

```
    def __len__(self):
        return len(self.hr_image_paths)


def get_val_data():
    val_dataset = ValDataset(hr_img_path=config.VALID.hr_img_path)
    batch_size = config.TRAIN.batch_size

    val_loader = DataLoader(
        dataset=val_dataset,
        batch_size=batch_size,
        shuffle=True,
        num_workers=0,
        drop_last=True,
        pin_memory=True
    )

    return val_loader
```

model.py

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


class ResidualBlock(nn.Module):
    def __init__(self, channels):
        super(ResidualBlock, self).__init__()
        self.conv1 = nn.Conv2d(channels, channels, kernel_size=3, stride=1, padding=1,
bias=False)
```

```
        self.bn1 = nn.BatchNorm2d(channels)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = nn.Conv2d(channels, channels, kernel_size=3, stride=1, padding=1,
bias=False)
        self.bn2 = nn.BatchNorm2d(channels)

    def forward(self, x):
        residual = x
        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)
        out = self.conv2(out)
        out = self.bn2(out)
        out += residual
        return out


class SubpixelConv2d(nn.Module):
    def __init__(self, scale, channels):
        super(SubpixelConv2d, self).__init__()
        self.conv = nn.Conv2d(channels, channels * scale * scale, kernel_size=3, stride=1,
padding=1)
        self.pixel_shuffle = nn.PixelShuffle(scale)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        x = self.conv(x)
        x = self.pixel_shuffle(x)
        x = self.relu(x)
        return x


class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
```

```
    def weights_init(m):
        if isinstance(m, nn.Conv2d):
            nn.init.normal_(m.weight.data, 0.0, 0.02)
            if m.bias is not None:
                nn.init.constant_(m.bias.data, 0)
        elif isinstance(m, nn.BatchNorm2d):
            nn.init.normal_(m.weight.data, 1.0, 0.02)
            nn.init.constant_(m.bias.data, 0)

    self.initial_conv = nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1)
    self.initial_relu = nn.ReLU(inplace=True)

    self.residual_blocks = nn.ModuleList([ResidualBlock(64) for _ in range(16)])

    self.post_conv = nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1, bias=False)
    self.post_bn = nn.BatchNorm2d(64)

    self.upscale1 = SubpixelConv2d(scale=2, channels=64)
    self.upscale2 = SubpixelConv2d(scale=2, channels=64)

    self.final_conv = nn.Conv2d(64, 3, kernel_size=1, stride=1, padding=0)
    self.tanh = nn.Tanh()

    self.apply(weights_init)

def forward(self, x):
    initial = self.initial_conv(x)
    x = self.initial_relu(initial)

    for block in self.residual_blocks:
        x = block(x)

    x = self.post_conv(x)
    x = self.post_bn(x)
```

```
        x = x + initial

        x = self.upscale1(x)
        x = self.upscale2(x)

        x = self.final_conv(x)
        x = self.tanh(x)

        return x


def get_G():
    return Generator()


class GlobalDiscriminator(nn.Module):
    def __init__(self, input_shape):
        super(GlobalDiscriminator, self).__init__()
        self.channels = input_shape[1]

        df_dim = 64

        def weights_init(m):
            classname = m.__class__.__name__
            if classname.find('Conv') != -1:
                nn.init.normal_(m.weight.data, 0.0, 0.02)
            elif classname.find('BatchNorm') != -1:
                nn.init.normal_(m.weight.data, 1.0, 0.02)
                nn.init.constant_(m.bias.data, 0)

        self.lrelu = nn.LeakyReLU(0.2, inplace=True)

        self.conv1 = nn.Conv2d(self.channels, df_dim, kernel_size=4, stride=2, padding=1)

        self.conv2 = nn.Conv2d(df_dim, df_dim * 2, kernel_size=4, stride=2, padding=1,
```

bias=False)

self.bn2 = nn.BatchNorm2d(df_dim * 2)

self.conv3 = nn.Conv2d(df_dim * 2, df_dim * 4, *kernel_size*=4, *stride*=2, *padding*=1, *bias*=False)

self.bn3 = nn.BatchNorm2d(df_dim * 4)

self.conv4 = nn.Conv2d(df_dim * 4, df_dim * 8, *kernel_size*=4, *stride*=2, *padding*=1, *bias*=False)

self.bn4 = nn.BatchNorm2d(df_dim * 8)

self.conv5 = nn.Conv2d(df_dim * 8, df_dim * 16, *kernel_size*=4, *stride*=2, *padding*=1, *bias*=False)

self.bn5 = nn.BatchNorm2d(df_dim * 16)

self.conv6 = nn.Conv2d(df_dim * 16, df_dim * 32, *kernel_size*=4, *stride*=2, *padding*=1, *bias*=False)

self.bn6 = nn.BatchNorm2d(df_dim * 32)

self.conv7 = nn.Conv2d(df_dim * 32, df_dim * 16, *kernel_size*=1, *stride*=1, *padding*=0, *bias*=False)

self.bn7 = nn.BatchNorm2d(df_dim * 16)

self.conv8 = nn.Conv2d(df_dim * 16, df_dim * 8, *kernel_size*=1, *stride*=1, *padding*=0, *bias*=False)

self.bn8 = nn.BatchNorm2d(df_dim * 8)

self.conv9 = nn.Conv2d(df_dim * 8, df_dim * 2, *kernel_size*=1, *stride*=1, *padding*=0, *bias*=False)

self.bn9 = nn.BatchNorm2d(df_dim * 2)

self.conv10 = nn.Conv2d(df_dim * 2, df_dim * 2, *kernel_size*=3, *stride*=1, *padding*=1, *bias*=False)

self.bn10 = nn.BatchNorm2d(df_dim * 2)

```
        self.conv11 = nn.Conv2d(df_dim * 2, df_dim * 8, kernel_size=3, stride=1,
padding=1, bias=False)
        self.bn11 = nn.BatchNorm2d(df_dim * 8)

        self.flatten = nn.Flatten()

        feature_size = self._calculate_feature_size(input_shape)
        self.linear = nn.Linear(df_dim * 8 * feature_size * feature_size, 1)

        self.apply(weights_init)

    def _calculate_feature_size(self, input_shape):
        h = input_shape[2]
        for _ in range(6):
            h = (h + 2 - 4) // 2 + 1
        return h

    def forward(self, x):
        x = self.lrelu(self.conv1(x))

        x = self.lrelu(self.bn2(self.conv2(x)))
        x = self.lrelu(self.bn3(self.conv3(x)))
        x = self.lrelu(self.bn4(self.conv4(x)))
        x = self.lrelu(self.bn5(self.conv5(x)))
        x = self.lrelu(self.bn6(self.conv6(x)))

        x = self.lrelu(self.bn7(self.conv7(x)))
        x = self.bn8(self.conv8(x))
        residual = x

        x = self.lrelu(self.bn9(self.conv9(residual)))
        x = self.lrelu(self.bn10(self.conv10(x)))
        x = self.bn11(self.conv11(x))

        x = self.lrelu(x + residual)
```

```
        x = self.flatten(x)
        x = self.linear(x)

        return x


class LocalDiscriminator(nn.Module):
    def __init__(self, patch_size=96):
        super(LocalDiscriminator, self).__init__()
        self.patch_size = patch_size

        channels = 3
        df_dim = 64

        def weights_init(m):
            classname = m.__class__.__name__
            if classname.find('Conv') != -1:
                nn.init.normal_(m.weight.data, 0.0, 0.02)
            elif classname.find('BatchNorm') != -1:
                nn.init.normal_(m.weight.data, 1.0, 0.02)
                nn.init.constant_(m.bias.data, 0)

        self.lrelu = nn.LeakyReLU(0.2, inplace=True)

        self.conv1 = nn.Conv2d(channels, df_dim, kernel_size=4, stride=2, padding=1)

        self.conv2 = nn.Conv2d(df_dim, df_dim * 2, kernel_size=4, stride=2, padding=1,
bias=False)
        self.bn2 = nn.BatchNorm2d(df_dim * 2)

        self.conv3 = nn.Conv2d(df_dim * 2, df_dim * 4, kernel_size=4, stride=2,
padding=1, bias=False)
        self.bn3 = nn.BatchNorm2d(df_dim * 4)
```

```
        self.conv4 = nn.Conv2d(df_dim * 4, df_dim * 8, kernel_size=4, stride=2,
padding=1, bias=False)
        self.bn4 = nn.BatchNorm2d(df_dim * 8)

        feature_size = patch_size // 16

        self.flatten = nn.Flatten()
        self.linear = nn.Linear(df_dim * 8 * feature_size * feature_size, 1)

        self.apply(weights_init)

    def forward(self, x):
        x = self.lrelu(self.conv1(x))

        x = self.lrelu(self.bn2(self.conv2(x)))
        x = self.lrelu(self.bn3(self.conv3(x)))
        x = self.lrelu(self.bn4(self.conv4(x)))

        x = self.flatten(x)
        x = self.linear(x)

        return x

    def extract_patches(self, images, num_patches=4):
        batch_size, channels, height, width = images.shape
        patches = []

        for i in range(batch_size):
            for _ in range(num_patches):
                h_start = torch.randint(0, height - self.patch_size + 1, (1,)).item()
                w_start = torch.randint(0, width - self.patch_size + 1, (1,)).item()

                patch = images[i:i + 1, :, h_start:h_start + self.patch_size, w_start:w_start +
self.patch_size]
                patches.append(patch)
```

```
        return torch.cat(patches, dim=0)


class DualDiscriminator(nn.Module):
    def __init__(self, global_input_shape, local_patch_size=96):
        super(DualDiscriminator, self).__init__()
        self.global_discriminator = GlobalDiscriminator(global_input_shape)
        self.local_discriminator = LocalDiscriminator(patch_size=local_patch_size)

    def forward(self, x, get_features=False):
        global_out = self.global_discriminator(x)

        patches = self.local_discriminator.extract_patches(x)
        local_out = self.local_discriminator(patches)

        batch_size = x.shape[0]
        num_patches = local_out.shape[0] // batch_size
        local_out = local_out.view(batch_size, num_patches, -1).mean(dim=1)  # Average
across patches

        return global_out, local_out


def get_dual_D(global_input_shape, local_patch_size=96):
    return DualDiscriminator(global_input_shape, local_patch_size)
```

train.py

```
import os
import time

import matplotlib.pyplot as plt
```

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.models as models
from skimage.metrics import structural_similarity as ssim
from torch.multiprocessing import freeze_support
from torchvision.utils import make_grid
from tqdm import tqdm

from config import config
from dataloader import get_train_data, get_val_data
from dual_model import get_G, get_dual_D


def calculate_psnr(img1, img2, max_value=1.0):
    mse = torch.mean((img1 - img2) ** 2)
    if mse == 0:
        return float('inf')
    return 20 * torch.log10(max_value / torch.sqrt(mse))


def calculate_ssim(img1, img2):
    img1_np = img1.detach().cpu().numpy()
    img2_np = img2.detach().cpu().numpy()

    ssim_values = []
    for i in range(img1_np.shape[0]):
        img1_hwc = np.transpose(img1_np[i], (1, 2, 0))
        img2_hwc = np.transpose(img2_np[i], (1, 2, 0))

        if img1_hwc.shape[2] == 1:
            img1_hwc = img1_hwc.squeeze(2)
            img2_hwc = img2_hwc.squeeze(2)
            ssim_val = ssim(img1_hwc, img2_hwc, data_range=2.0)
```

```
        else:
            ssim_val = ssim(img1_hwc, img2_hwc, data_range=2.0, channel_axis=2)

        ssim_values.append(ssim_val)

    return np.mean(ssim_values)


def validate_model(G, val_loader, device, epoch):
    G.eval()
    total_mse = 0.0
    total_psnr = 0.0
    total_ssim = 0.0
    num_batches = 0

    with torch.no_grad():
        for lr_patches, hr_patches in val_loader:
            lr_patches = lr_patches.to(device)
            hr_patches = hr_patches.to(device)

            fake_patches = G(lr_patches)

            mse = nn.MSELoss()(fake_patches, hr_patches)
            total_mse += mse.item()

            psnr = calculate_psnr(fake_patches, hr_patches, max_value=2.0)
            total_psnr += psnr.item()

            ssim_val = calculate_ssim(fake_patches, hr_patches)
            total_ssim += ssim_val

            num_batches += 1

    save_dir = './samples_val'
    save_image(fake_patches, lr_patches, hr_patches,
```

```
            os.path.join(save_dir, 'val_g_{}.png'.format(epoch)))

    avg_mse = total_mse / num_batches
    avg_psnr = total_psnr / num_batches
    avg_ssim = total_ssim / num_batches

    G.train()
    return avg_mse, avg_psnr, avg_ssim


def save_image(generated_images, low_res_inputs, high_res_targets, save_path,
num_samples=4):
    os.makedirs(os.path.dirname(save_path), exist_ok=True)

    gen_imgs = generated_images.detach().cpu()
    lr_imgs = low_res_inputs.detach().cpu()
    hr_imgs = high_res_targets.detach().cpu()

    gen_imgs = gen_imgs[:num_samples]
    lr_imgs = lr_imgs[:num_samples]
    hr_imgs = hr_imgs[:num_samples]

    fig, axes = plt.subplots(num_samples, 3, figsize=(15, 4 * num_samples))

    if num_samples == 1:
        axes = [axes]

    kwargs = {'interpolation': 'none'}

    for i in range(num_samples):

        lr_img = torch.nn.functional.interpolate(
            lr_imgs[i:i + 1], scale_factor=4, mode='bicubic', align_corners=False
        )[0].permute(1, 2, 0)
        lr_img = (lr_img + 1) / 2
```

```
            lr_img = lr_img.clamp(0, 1).numpy()

            gen_img = gen_imgs[i].permute(1, 2, 0)
            gen_img = (gen_img + 1) / 2
            gen_img = gen_img.clamp(0, 1).numpy()

            hr_img = hr_imgs[i].permute(1, 2, 0)
            hr_img = (hr_img + 1) / 2
            hr_img = hr_img.clamp(0, 1).numpy()

            axes[i][0].imshow(lr_img, **kwargs)
            axes[i][0].set_title('LR Input (Upscaled)')
            axes[i][0].axis('off')

            axes[i][1].imshow(gen_img, **kwargs)
            axes[i][1].set_title('SR Generated')
            axes[i][1].axis('off')

            axes[i][2].imshow(hr_img, **kwargs)
            axes[i][2].set_title('HR Ground Truth')
            axes[i][2].axis('off')

        plt.tight_layout()
        plt.savefig(save_path, dpi=200, bbox_inches='tight')
        plt.close(fig)

        print(f"Saved sample comparison to {save_path}")


batch_size = config.TRAIN.batch_size
lr_init = config.TRAIN.lr_init
beta1 = config.TRAIN.beta1
n_epoch_init = config.TRAIN.n_epoch_init
n_epoch = config.TRAIN.n_epoch
lr_decay = config.TRAIN.lr_decay
```

```
decay_every = config.TRAIN.decay_every
shuffle_buffer_size = 128


def train():
    freeze_support()

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    G = get_G().to(device)
    D = get_dual_D((8, 3, 384, 384), local_patch_size=96).to(device)

    VGG = models.vgg19(pretrained=True).features[:21]
    VGG.eval()
    for param in VGG.parameters():
        param.requires_grad = False
    VGG = VGG.to(device)

    lr_v = config.TRAIN.lr_init
    beta1 = 0.9
    g_optimizer_init = optim.Adam(G.parameters(), lr=lr_v, betas=(beta1, 0.999))
    g_optimizer = optim.Adam(G.parameters(), lr=lr_v, betas=(beta1, 0.999))
    d_optimizer = optim.Adam(D.parameters(), lr=lr_v, betas=(beta1, 0.999))

    G.train()
    D.train()

    train_loader = get_train_data()
    val_loader = get_val_data()
    batch_size = train_loader.batch_size

    n_epoch_init = config.TRAIN.n_epoch_init
    for epoch in range(n_epoch_init):
        n_step_epoch = len(train_loader)
        for step, (lr_patches, hr_patches) in tqdm(enumerate(train_loader)):
```

```
            if lr_patches.shape[0] != batch_size:
                break

            step_time = time.time()

            lr_patches = lr_patches.to(device)
            hr_patches = hr_patches.to(device)

            g_optimizer_init.zero_grad()
            fake_hr_patches = G(lr_patches)

            mse_loss = nn.MSELoss()(fake_hr_patches, hr_patches)

            mse_loss.backward()
            g_optimizer_init.step()

        print("Epoch: [{}/{}] step: [{}/{}] time: {:.3f}s, mse: {:.3f} ".format(
            epoch, n_epoch_init, step, n_step_epoch, time.time() - step_time,
mse_loss.item()))



    n_epoch = config.TRAIN.n_epoch
    decay_every = config.TRAIN.decay_every
    lr_init = config.TRAIN.lr_init
    lr_decay = config.TRAIN.lr_decay
    best_mse = 10
    checkpoint_dir = './models'

    for epoch in range(n_epoch):
        n_step_epoch = len(train_loader)
        for step, (lr_patches, hr_patches) in tqdm(enumerate(train_loader)):
            if lr_patches.shape[0] != batch_size:
                break

            step_time = time.time()
```

```
        lr_patches = lr_patches.to(device)
        hr_patches = hr_patches.to(device)

        d_optimizer.zero_grad()

        global_logits_real, local_logits_real = D(hr_patches)

        fake_patches = G(lr_patches)
        global_logits_fake, local_logits_fake = D(
            fake_patches.detach())

        d_global_loss_real = nn.BCEWithLogitsLoss()(global_logits_real,
torch.ones_like(global_logits_real))
        d_global_loss_fake = nn.BCEWithLogitsLoss()(global_logits_fake,
torch.zeros_like(global_logits_fake))
        d_global_loss = d_global_loss_real + d_global_loss_fake

        d_local_loss_real = nn.BCEWithLogitsLoss()(local_logits_real,
torch.ones_like(local_logits_real))
        d_local_loss_fake = nn.BCEWithLogitsLoss()(local_logits_fake,
torch.zeros_like(local_logits_fake))
        d_local_loss = d_local_loss_real + d_local_loss_fake

        d_loss = d_global_loss + d_local_loss

        d_loss.backward()
        d_optimizer.step()

        g_optimizer.zero_grad()

        global_logits_fake, local_logits_fake = D(fake_patches)

        feature_fake = VGG((fake_patches + 1) / 2.)
        feature_real = VGG((hr_patches + 1) / 2.)
```

```
        g_global_gan_loss = 1e-3 * nn.BCEWithLogitsLoss()(global_logits_fake,
torch.ones_like(global_logits_fake))
        g_local_gan_loss = 1e-3 * nn.BCEWithLogitsLoss()(local_logits_fake,
torch.ones_like(local_logits_fake))
        g_gan_loss = g_global_gan_loss + g_local_gan_loss

        mse_loss = 2e-4 * nn.MSELoss()(fake_patches, hr_patches)
        vgg_loss = nn.MSELoss()(feature_fake, feature_real)
        g_loss = mse_loss + vgg_loss + g_gan_loss

        g_loss.backward()
        g_optimizer.step()

    print(
        "Epoch: [{}/{}] step: [{}/{}] time: {:.3f}s, "
        "g_loss(mse:{:.3f}, vgg:{:.3f}, global_adv:{:.3f}, local_adv:{:.3f}) "
        "d_loss(global:{:.3f}, local:{:.3f})".format(
            epoch, n_epoch, step, n_step_epoch, time.time() - step_time,
            mse_loss.item(), vgg_loss.item(),
            g_global_gan_loss.item(), g_local_gan_loss.item(),
            d_global_loss.item(), d_local_loss.item()))

    if epoch != 0 and (epoch % decay_every == 0):
        new_lr_decay = lr_decay ** (epoch // decay_every)
        for param_group in g_optimizer.param_groups:
            param_group['lr'] = lr_init * new_lr_decay
        for param_group in d_optimizer.param_groups:
            param_group['lr'] = lr_init * new_lr_decay
        print(" ** new learning rate: %f (for GAN)" % (lr_init * new_lr_decay))

    if (epoch != 0) and (epoch % 5 == 0):
        print("Running validation...")
        val_mse, val_psnr, val_ssim = validate_model(G, val_loader, device, epoch)
        print("Validation Results - MSE: {:.4f}, PSNR: {:.2f} dB, SSIM: {:.4f}".format(
```

```
            val_mse, val_psnr, val_ssim))


        if val_mse < best_mse:
            best_mse = val_mse
            torch.save(G.state_dict(), os.path.join(checkpoint_dir,
f'best_{epoch}_{best_mse}_g.pth'))
            torch.save(D.state_dict(), os.path.join(checkpoint_dir,
f'best_{epoch}_{best_mse}_d.pth'))

    if (epoch != 0) and (epoch % 10 == 0):
        save_dir = './samples'
        save_image(fake_patches, lr_patches, hr_patches,
                os.path.join(save_dir, 'train_g_{}.png'.format(epoch)))

        torch.save(G.state_dict(), os.path.join(checkpoint_dir, f'{epoch}_g.pth'))
        torch.save(D.state_dict(), os.path.join(checkpoint_dir, f'{epoch}_d.pth'))


if __name__ == '__main__':
    train()
```