

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ЗБІЛЬШЕННЯ ЯКОСТІ ТЕРМАЛЬНИХ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ
ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ
RESEARCH AND SOFTWARE IMPLEMENTATION OF THERMAL IMAGE
SUPER-RESOLUTION METHODS USING NEURAL NETWORKS

Виконав: студент 2 курсу, групи КНТ-111м

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Діденко А.Є.

(прізвище та ініціали)

Керівник Олійник. А.О.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Діденка Андрія Євгеновича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів збільшення якості термальних зображень з використанням штучних нейронних мереж. Research and Software Implementation of Thermal Image Super-Resolution Methods Using Neural Networks

керівник проєкту (роботи) Олійник Андрій Олександрович, д.т.н., професор,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “04” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) 12 грудня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури мереж і програми. 3. Аналіз результатів та реалізація програмного забезпечення. 4. Експлуатація програмного забезпечення. Висновки. Перелік джерел посилання. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Олійник А.О., професор		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “05” вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-5 тижні	Розділ 1
3	Розробка архітектури програми	6-8 тиждень	Розділ 2
4	Розробка програми	9-11 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	12 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	13 тиждень	Додатки
7	Захист роботи	14 тиждень	

Студент

(підпис) Діденко А.Є.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Олійник А.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
88 с., 55 рис., 5 табл., 3 дод., 51 джерело.

ЗБІЛЬШЕННЯ РОЗДІЛЬНОЇ ЗДАТНОСТІ, ГЛИБИННЕ НАВЧАННЯ,
ОБРОБКА ЗОБРАЖЕНЬ, ТЕРМОГРАФІЯ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ.

Об'єкт дослідження – процес збільшення роздільної здатності термальних зображень.

Предмет дослідження – методи збільшення роздільної здатності термальних зображень з використанням штучних нейронних мереж.

Мета роботи – програмна реалізація методів збільшення роздільної здатності термальних зображень з використанням штучних нейронних мереж.

Матеріали, методи та технічні засоби: мова програмування Python, фреймворк глибинного навчання PyTorch, хмарних сервіс AWS з використанням серверу з відеокартою Nvidia A10G.

Результати. Обрано та навчено моделі для збільшення роздільної здатності термальних зображень. Створено та навчено власну модифікацію існуючої моделі під задачу збільшення роздільної здатності термальних зображень. Створено програмне забезпечення, яке збільшує роздільну здатність термального зображення за допомогою штучних нейронних мереж.

Висновки. Розроблено програмне забезпечення для збільшення роздільної здатності термальних зображень, яке автоматизує та покращує процес обробки термальних зображень.

Галузь використання – термографія в галузях виробництва, медицини, будівництва.

ABSTRACT

Explanatory note to the master's final qualifying work: 88 pages, 55 figures, 5 tables, 3 appendixes, 51 sources.

ARTIFICIAL NEURAL NETWORKS, DEEP LEARNING, IMAGE PROCESSING, SUPER RESOLUTION, THERMOGRAPHY.

Object of study is process of thermal images super resolution.

Subject of study is thermal image super resolution methods using artificial neural networks.

The purpose of the work is software implementation of thermal image super resolution methods using artificial neural networks.

Materials, methods and tools: Python programming language, deep learning framework PyTorch, server with Nvidia A10G graphics card in AWS cloud service.

Results. Selected and trained models to increase the resolution of thermal images. Created and trained own modification of the existing model for the task of thermal images super resolution. Created software that increases the resolution of thermal images using artificial neural networks.

Conclusions. Software for thermal image super resolution was created. It automates and improves process of processing thermal images.

The field of use is thermography in manufacturing, healthcare, construction.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Основні поняття	10
1.2 Аналіз існуючих методів.....	15
1.3 Порівняння методів ЗРЗ	40
1.4 Постановка задачі до роботи	41
1.5 Висновки за розділом 1	43
2 Розробка архітектури мереж і програми.....	44
2.1 Вибір мови програмування	44
2.2 Вибір фреймворку глибинного навчання	46
2.3 Вибір додаткових програмних бібліотек	47
2.4 Вибір датасету	48
2.5 Обробка даних	49
2.6 Архітектура компонентів програми.....	50
2.7 Архітектура системи.....	53
2.8 Висновки за розділом 2	53
3 Аналіз результатів та реалізація програмного забезпечення	54
3.1 Вибір гіперпараметрів навчання.....	54
3.2 Аналіз результатів навчання мереж	55
3.3 Реалізація компонентів програмного забезпечення	57
3.4 Висновки за розділом 3	59
4 Експлуатація програмного забезпечення.....	60
4.1 Призначення програми	60
4.2 Умови використання програми	60
4.3 Виконання програми.....	60
4.4 Повідомлення користувачу	62

4.5 Висновки за розділом 4	63
Висновки	64
Перелік джерел посилання	65
Додаток А Технічне завдання	70
Додаток Б Текст програми	75
Додаток В Слайди презентації.....	80

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- ГЗМ – генеративно змагальна мережа;
ЗНМ – згорткова нейронна мережа;
ЗРЗ – збільшення роздільної здатності;
ООП – об’єктно-орієнтоване програмування;
ШНМ – штучна нейронна мережа.

Вступ

Термографія отримала популярність в багатьох сферах життя. Оскільки людське око не бачить термальний спектр, а також погано бачить при низькій освітленості, аналіз термальних зображень став невід'ємною частиною медицини, виробництва, будівництва та інших галузей.

Більшість тепловізорів при аналізі температури об'єктів видає зображення з низькою роздільною здатністю, що ускладнює процес аналізу вихідної термограми. Тому проблема збільшення якості термальних зображень є актуальною на сьогодні.

Із розвитком сфери штучного інтелекту і технологій глибинного навчання з'являються нові методи підвищення якості зображень, а саме збільшення роздільної здатності за допомогою штучних нейронних мереж. Такі методи також можуть бути адаптовані під обробку термальних зображень.

Актуальність роботи полягає в дослідженні різних архітектур нейронних мереж для збільшення роздільної здатності термальних зображень.

У випускній кваліфікаційній роботі створюється програма для збільшення роздільної здатності термальних зображень, розроблюються і навчаються штучні нейронні мережі різних видів, а також аналізуються результати і описуються шляхи покращення якості роботи розроблених нейронних мереж.

1 Аналіз предметної області

1.1 Основні поняття

1.1.1 Теплобачення

Інфрачервона термографія (або теплобачення) – це наука, що вивчає отримання та обробку теплової інформації від безконтактних вимірювальних пристроїв. Вона заснована на інфрачервоному випромінюванні, формі електромагнітного випромінювання з більшою довжиною хвилі, ніж у видимого світла. Будь-який об'єкт при температурі вище абсолютного нуля випромінює інфрачервоне випромінювання. Людське око не може бачити цей тип випромінювання. Тому для отримання та обробки цієї інформації потрібні інфрачервоні вимірювальні пристрої [1].

Інфрачервоне (ІЧ) випромінювання – це вид електромагнітного випромінювання, яке знаходиться між видимим світлом та радіохвилями в електромагнітному спектрі. Воно має довжину хвилі від 0,74 мкм (червоний край видимого світла) до 100 мкм (початок короткохвильового радіодіапазону) [2].

На рис. 1.1 зображено схематичне представлення електромагнітного спектру.

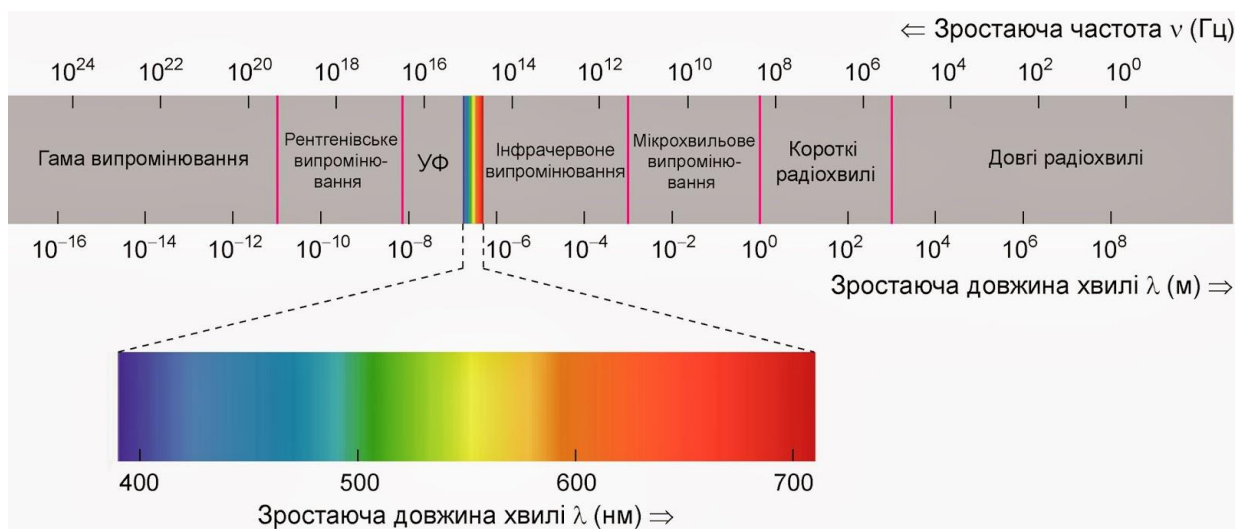


Рисунок 1.1 – Електромагнітний спектр [3]

Кожне тіло випромінює електромагнітні хвилі. Інтенсивність і спектр випромінювання визначаються законом Стефана-Больцмана й залежить від температури тіла. Проте у видимій області спектру тіла починають світитися тільки при високих температурах порядку 500 °С, спочатку це світіння червоне, при підвищенні температури дедалі більше зсувається в бік коротших довжин хвиль і стає білим. В області звичних для людини температур тіла світяться інтенсивно лише в інфрачервоній області спектру. Для детектування цього випромінювання необхідні спеціальні прилади [4].

Тепловізор – це пристрій, який отримує теплове зображення в інфрачервоній області спектру без безпосереднього контакту з обладнанням [5].

Тепловізор працює наступним чином: інфрачервоне випромінювання фокусується за допомогою оптики тепловізора на приймачі випромінювання, який видає сигнал, зазвичай як зміни напруги або електричного опору. Отриманий сигнал реєструється електронікою тепловізійної системи. Сигнал, який дає тепловізор, перетворюється на електронне зображення (термограму), яке відображається на екрані дисплея [5].

Термограма – це зображення об'єкта, оброблене електронікою для відображення на дисплеї таким чином, що різні градації кольору відповідають розподілу інфрачервоного випромінювання поверхнею об'єкта. Таким чином, термографіст може просто побачити термограму, що відповідає тепловому випромінюванню, яке надходить з поверхні об'єкта [5].

Термограму ще називають термальним (тепловим) зображенням або інфрачервоним зображенням.

Оскільки тіла з вищою температурою світяться інтенсивніше, то в інфрачервоному спектрі добре розрізняються теплокровні істоти: люди й тварини. Зображення, проте, не особливо високої якості й потребує досвіду для правильної інтерпретації [4].

На рис. 1.2 зображено приклад термограми фруктів. У класичній колірній схемі теплових зображень чим гарячіше тіло – тим більше воно

наближене до червоного кольору. Холодні об'єкти мають синій або фіолетовий відтінки.

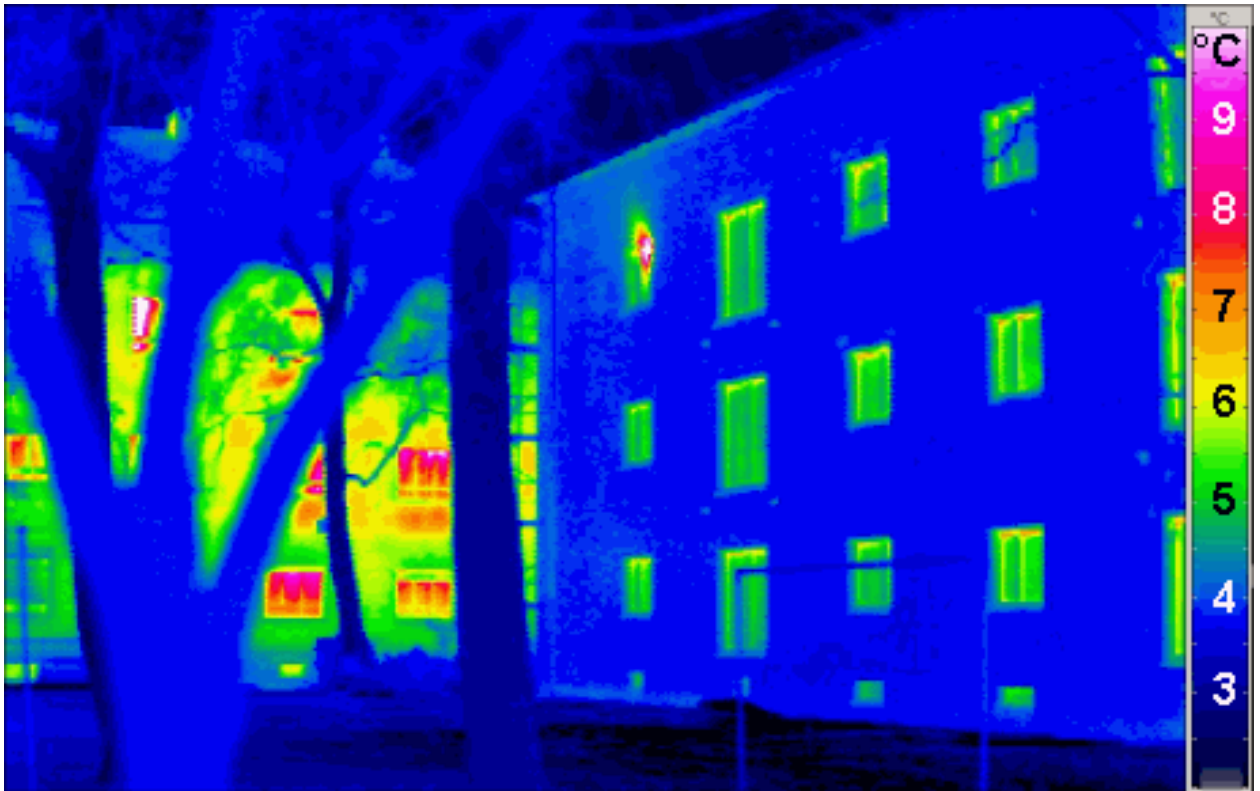


Рисунок 1.2 – Приклад теплового зображення [4]

Інфрачервоне випромінювання широко використовується у промисловості, наукових дослідженнях, медицині та військовій техніці. Реєстрація спектрів випромінювання та поглинання в ІЧ-діапазоні (ІЧ-спектроскопія) використовується при вивченні будови електронної оболонки атомів з метою визначення молекулярної структури, а також для якісного та кількісного аналізу складних сумішей молекулярного складу. Цей метод широко використовується як на виробництві, так і в лабораторії у фізиці та астрономії, біології, медицині, хімії та криміналістиці [2].

Не можна не згадати про таку важливу область, як інфрачервона термографія. Аналіз зображення в ІЧ-променях можна назвати універсальним способом отримання різноманітної інформації про навколишній світ. Запис теплового зображення дозволяє побачити та проаналізувати картину

розподілу температурних полів. Пожежні використовують цей метод для встановлення вогнища та порятунку людей. На теплових зображеннях можна виявити локальні перегріву у різних системах та структурах. У медицині теплосканування дозволяє візуалізувати запальні процеси, поєднуючи в собі абсолютну нешкідливість, швидкість та простоту способу отримання інформації. В даний час будь-яка геологорозвідка не може обійтися без аналізу супутникових знімків в ІЧ-променях [2].

1.1.2 Нейронні мережі

Глибинне навчання — це вид машинного навчання, що наділяє комп'ютери здатністю вчитися на досвіді і розуміти світ у термінах ієрархії концепцій. Ієрархічна організація дає комп'ютеру можливість вчитися складнішим поняттям шляхом побудови їх з більш простих. Граф, що описує цю ієрархію, буде глибоким і містити багато рівнів. Тому такий підхід називається глибинним навчанням [6].

Штучна нейронна мережа (ШНМ) — це математична модель, взаємопов'язана сукупність простих обробних елементів, блоків або вузлів (нейронів), функціональність яких базується на роботі нейронів тварин (зокрема людей). Здатність мережі до обробки зберігається в вагах міжблокового з'єднання, отриманих в результаті процесу адаптації до набору шаблонів навчання [7].

Штучний нейрон — це еквівалент біологічного нейрону, що є вузлом в штучній нейронній мережі. Синапси моделюють одним числом або вагою так, що кожен вхідний сигнал помножується на вагу перед відправкою в тіло нейрона. Тут зважені сигнали сумуються шляхом простого арифметичного складання для отримання активації вузла. Активация потім порівнюється з порогом. Якщо активация перевищує поріг, нейрон виробляє вихід із високим значенням (умовно "1"), інакше він видає нуль [7].

На рис. 1.3 схематично зображено штучний нейрон, а на рис. 1.4 зображено структуру простої ШНМ.

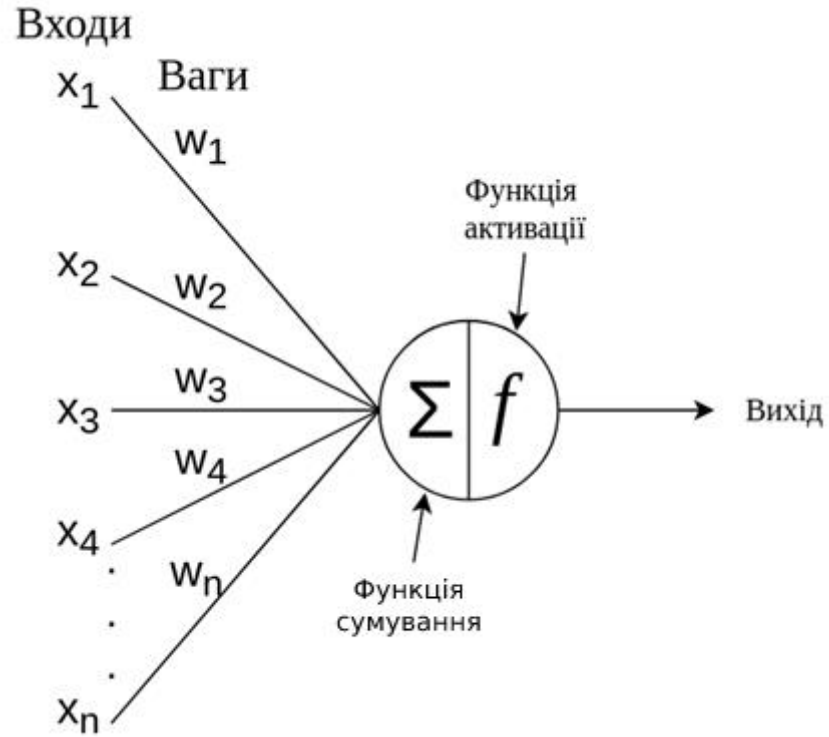


Рисунок 1.3 – Штучний нейрон

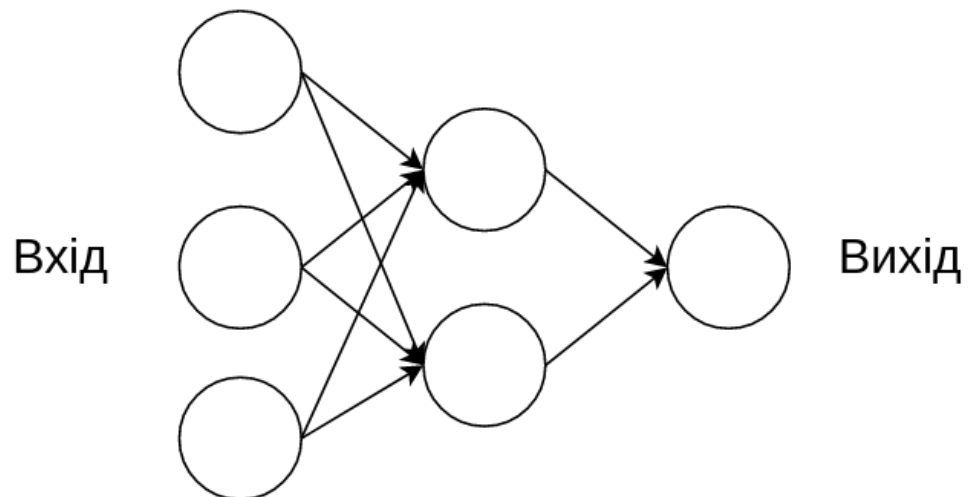


Рисунок 1.4 – Схема ШНМ

ШНМ мають велику область застосування та багато підвидів. Так, зазвичай, багатошарові перцептрони застосовуються для структурованої і неструктурованої інформації (наприклад, прогнозування ціни на дім на основі характеристик дому). Рекурентні нейронні мережі використовують для аналізу послідовностей, наприклад, класифікації або генерації тексту. Згорткові нейронні мережі використовують для вирішення певних задач, пов'язаних із зображеннями, наприклад, класифікація, детекція об'єктів, тощо.

1.1.3 Задача збільшення роздільної здатності

Основна мета задачі збільшення роздільної здатності (ЗРЗ) — створення зображення більш високої роздільної здатності із зображення нижчої роздільної здатності. Зображення з високою роздільною здатністю забезпечує високу щільність пікселів і, таким чином, більше деталей про вихідну сцену. Необхідність у високій якості зображень часто виникає у сфері комп'ютерного зору для підвищення ефективності розпізнавання образів та аналізу зображень [8].

1.2 Аналіз існуючих методів

Існуючі методи можна умовно поділити на чотири категорії: методи із застосуванням класичних алгоритмів, методи із застосуванням згорткових нейронних мереж, методи із застосуванням генеративно змагальних нейронних мереж і методи із застосуванням трансформерів.

1.2.1 Методи інтерполяції

Класичними та найпростішими методами ЗРЗ є методи інтерполяції. Інтерполяція — це метод знаходження невідомих проміжних значень деякої

функції за наявним дискретним набором відомих значень [9]. Ці методи наявні практично в кожному пакеті програмних засобів для обробки зображень. Незважаючи на простоту реалізації та зрозумілість, головних недолік методів інтерполяції – якість вихідного зображення.

1.2.1.1 Метод найближчого сусіда

Метод найближчого сусіда — це найпростіший підхід до інтерполяції. Замість того, щоб обчислювати середнє значення за якимись ваговими критеріями або генерувати проміжне значення на основі складних правил, цей метод просто визначає "найближчий" сусідній піксель і приймає його значення за значення інтенсивності поточного пікселя [10]. На рис. 1.5 зображено принцип роботи методу на прикладі збільшення зображення у три рази.

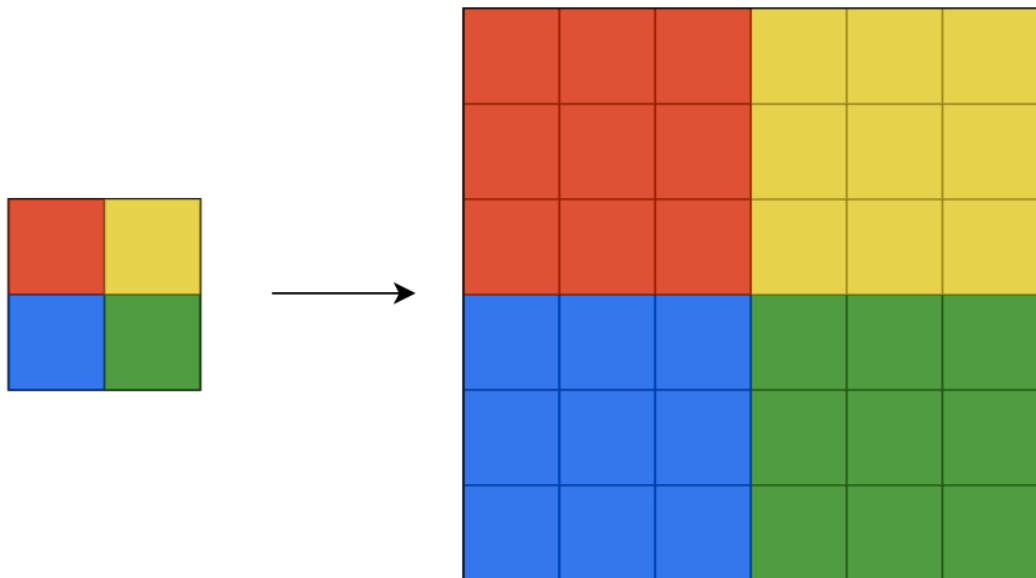


Рисунок 1.5 – Принцип роботи методу найближчого сусіда

1.2.1.2 Білінійна інтерполяція

При білінійній інтерполяції розглядається найближче сусідство 2×2 відомих значень пікселів, що оточують невідомий піксель. Потім береться середньозважене значення цих 4 пікселів, щоб отримати остаточне інтерпольоване значення. В результаті виходять набагато гладкіші зображення, ніж при використанні метода найближчого сусіда [11].

Функція білінійної інтерполяції має наступний вигляд (1.1):

$$F(x, y) = b_1 + b_2 \cdot x + b_3 \cdot y + b_4 \cdot x \cdot y, \quad (1.1)$$

де x, y – координати пікселя, значення якого необхідно розрахувати;

b_1, b_2, b_3, b_4 – коефіцієнти рівняння [12].

У випадку, коли відомі значення функції у точках, що є вершинами одиничного квадрату з координатами $(0, 0)$, $(0, 1)$, $(1, 0)$, $(1, 1)$ формула білінійної інтерполяції спрощується до наступного вигляду [12] (1.2):

$$f(x, y) \approx F(x, y) = f(0,0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy \quad (1.2)$$

1.2.1.3 Бікубічна інтерполяція

Бікубічний метод перевершує білінійний, розглядаючи найближче сусідство 4×4 відомих пікселів — лише 16 пікселів. Оскільки вони знаходяться на різних відстанях від невідомого пікселя, ближчі пікселі мають більшу вагу при розрахунку. Бікубічний метод дає помітно чіткіші зображення, ніж два попередні методи, і, можливо, є ідеальним поєднанням часу обробки та якості результату. З цієї причини він є стандартом для

багатьох програм редагування зображень (включаючи Adobe Photoshop), драйверів принтерів та інтерполяції у фотокамері [11].

Формула бікубічної інтерполяції має наступний вигляд (1.3):

$$f(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j, \quad (1.3)$$

де x, y — координати пікселя, значення якого необхідно розрахувати,

x^i, y^j — координати пікселів у сусідстві,

a_{ij} — коефіцієнти рівняння [13].

1.2.2 Метод комбінації зображень

Основна ідея методів з комбінацією зображень (image stacking, multi-frame) полягає в використанні декількох зображень об'єкта, зроблених з трохи інших ракурсів, для створення нового зображення з більш високою роздільною здатністю. Багатокадрова технологія ЗРЗ використовує субпіксельні зрушення (subpixel shift) між декількома зображеннями однієї і тієї ж сцени з низькою роздільною здатністю. Таким чином, створюється зображення покращеної роздільної здатності, поєднуючи інформацію з усіх зображень низької роздільної здатності, і створені зображення більш високої роздільної здатності краще описують сцену [14]. Така технологія зазвичай використовується в цифрових камерах. Крім того, за умови, що окремі знімки зроблені з однаковою чутливістю, підсумкове комбіноване зображення міститиме менше шуму, ніж окремий знімок, що можна розглядати як ефект усереднення [15].

Метод комбінації зображень також використовується для ЗРЗ термальних зображень. Наприклад, тепловізори компанії Testo обладнано технологією ЗРЗ з використанням декількох термальних зображень [16].

Технологія використовує природні рухи людини в цілях швидкого створення серії послідовних знімків. Для отримання зображення використовується спеціальний алгоритм і відповідні розрахунки. Як результат – у 4 рази збільшення роздільної здатності інфрачервоного зображення [16].

На рисунку 1.6 зображено процес збільшення зображення, запропонований компанією Testo.

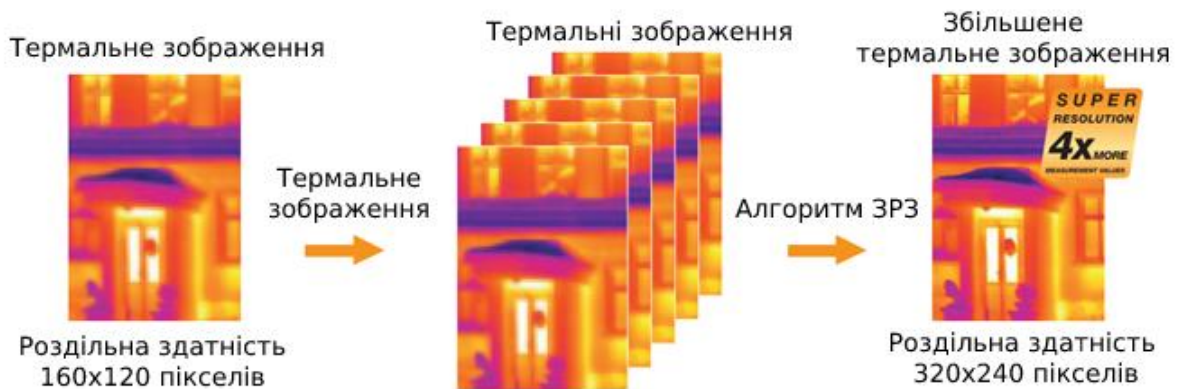


Рисунок 1.6 – Процес збільшення зображення у тепловізорах Testo [15]

У якості безпосереднього алгоритму ЗРЗ можуть використовуватись як класичні алгоритми перетворення, так і згорткові нейронні мережі.

1.2.3 Згорткові нейронні мережі

Задача ЗРЗ також може бути вирішена за допомогою згорткових нейронних мереж (ЗНМ).

ЗНМ широко використовуються для задачі аналізу та обробки зображень, наприклад, класифікація, детекція об'єктів, сегментація, колоризація, тощо. Архітектура ЗНМ була натхнена будовою зорової кори тварин. Вчені показали, що багато нейронів у зоровій корі мають невелике локальне рецептивне поле, тобто вони реагують тільки на візуальні стимули, розташовані в обмеженій області зорового поля. Рецептивні поля різних

нейронів можуть перекриватися, разом вони охоплюють все зорове поле [17]. Більше того, вони показали, що деякі нейрони реагують лише на зображення горизонтальних ліній, а інші – тільки на лінії з різною орієнтацією. Вчені також помітили, що у деяких нейронів рецептивне поле більше, і вони реагують на складніші візерунки, які є комбінаціями більш низького рівня візерунків. Ці спостереження призвели до ідеї, що нейрони вищого рівня засновані на виходах сусідніх нейронів нижчого рівня [17].

ЗНМ мають декілька видів шарів нейронів, що характерні лише для цього виду ШНМ:

- згортковий шар;
- агрегуючий шар;
- шар розгортки;
- шар роз'єднання;
- шар перемішування пікселів (Pixel Shuffle).

Згортковий шар є основним будівельним блоком ЗНМ. Параметри шару складаються з набору фільтрів для навчання (або ядер), які мають невеличке рецептивне поле, але простягаються на всю глибину вхідної ємності. Протягом прямого проходу кожен фільтр здійснює згортку за шириною та висотою вхідної ємності, обчислюючи скалярний добуток даних фільтру та входу, і формуючи 2-вимірну карту збудження цього фільтру [18].

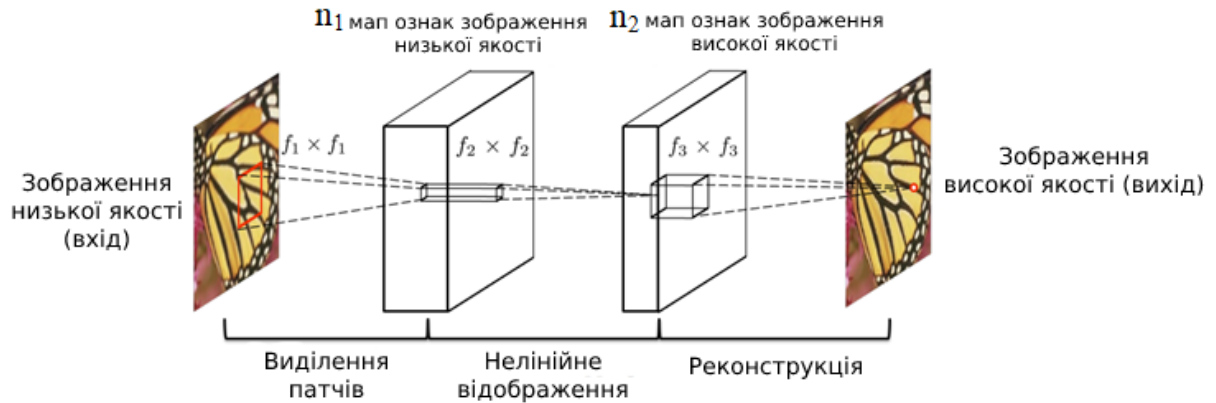
Агрегуючий шар використовує агрегуючі функції над вхідним зображенням. Існує декілька нелінійних функцій для реалізації агрегування, серед яких найпоширенішою є максимізаційне агрегування. Воно розділяє вхідне зображення на набір прямокутників без перекриттів, і для кожної такої підобласті виводить її максимум. Ідея полягає в тому, що точне положення ознаки не так важливе, як її грубе положення відносно інших ознак [18].

Шар розгортки працює обернено до згорткового шару, розширюючи мапу ознак. Фільтри шару розгортки також адаптуються в процесі навчання моделі [19].

Шар роз'єднання працює обернено до агрегуючого шару, розширюючи мапу ознак та заповнюючи пусті комірки значенням найближчого пікселя (за методом найближчого сусіда) [19].

Pixel Shuffle використовується для імплементації субпіксельної згортки [20]. Зокрема, ця операція переставляє елементи в тензорі розмірності $(*, C \times r^2, H, W)$ в тензор розмірністю $(*, C, H \times r, W \times r)$, де C – кількість каналів, H — висота, W — ширина, r – коефіцієнт масштабування.

Вперше ідея використання ЗНМ для задачі ЗРЗ замість класичних методів була використана у методі [21]. У запропонованому методі [21] зображення спочатку збільшується за допомогою бікубічної інтерполяції, потім проходить обробку через ЗНМ, що складається з трьох функцій: виділення патчів зображення, нелінійне відображення та реконструкція. Виділення патчів включає в себе виділення частин зображення та представлення їх у вигляді багатовимірного вектору, що являють собою певний набір ознак. Нелінійне відображення трансформує кожен багатовимірний вектор у інший багатовимірний вектор. На етапі реконструкції представлення патчів зображення агрегуються для генерації вихідного збільшеного зображення. На рис. 1.7 зображено запропоновану архітектуру ЗНМ.



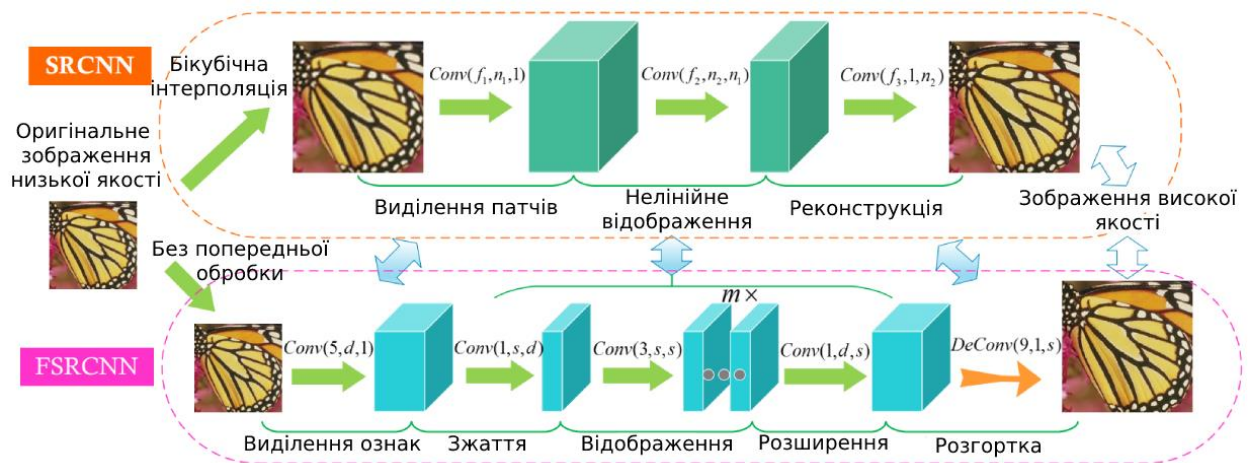
Умовні позначки:

n_1, n_2 – кількість фільтрів;

f_1, f_2, f_3 – розмір фільтрів.

Рисунок 1.7 – Запропонована архітектура ЗНМ [21]

У методі [22] був запропонований метод на основі методу [21], що дозволяє прискорити процес ЗРЗ. На відміну від методу [21], автори не збільшують зображення перед входом у нейронну мережу, а працюють із оригінальним зменшеним зображенням. Крім того, розмір фільтру згорткового шару був зменшений до п'яти. Потім, кількість фільтрів зменшується, що дозволяє зменшити навантаження під час тренування. На етапі нелінійного відображення використовуються декілька згорткових шарів замість одного. На наступному етапі відбувається збільшення кількості фільтрів, що дозволяє підвищити якість обробки зображення. Останнім кроком є шар розгортки, що видає збільшене зображення. Автори прийшли до висновку, що краще використовувати канал Y з колірної моделі $YCbCr$. Також автори показали, що для тренування такої ЗНМ достатньо досить невеликої навчальної вибірки. На рис. 1.8 зображено запропоновану архітектуру ЗНМ.



Умовні позначки:

n_1, n_2, s, d – кількість фільтрів;

f_1, f_2 – розмір фільтрів;

m – кількість блоків.

Рисунок 1.8 – Запропонована архітектура ЗНМ [22]

У методі [23] була запропонована ідея використання глибоких ЗНМ для задачі відновлення зображень, частиною якої є підзадача ЗРЗ. Автори використовували архітектуру згорткового автокодувальника з пропусковими зв'язками. На рис. 1.9 зображено запропоновану архітектуру ЗНМ.

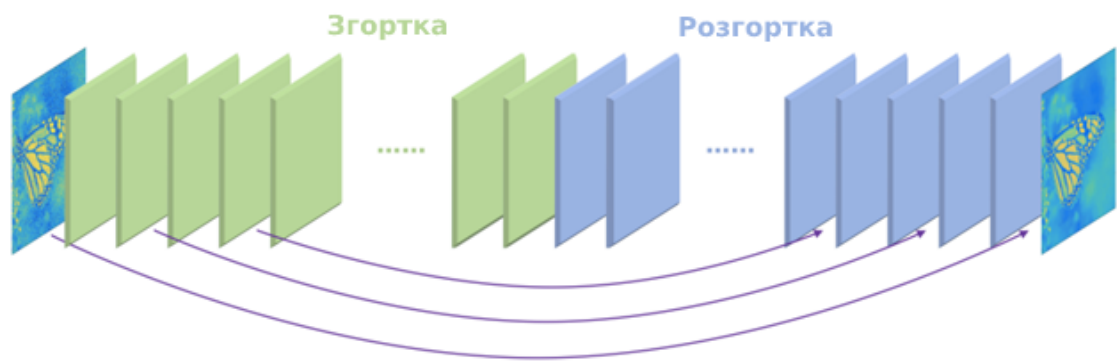
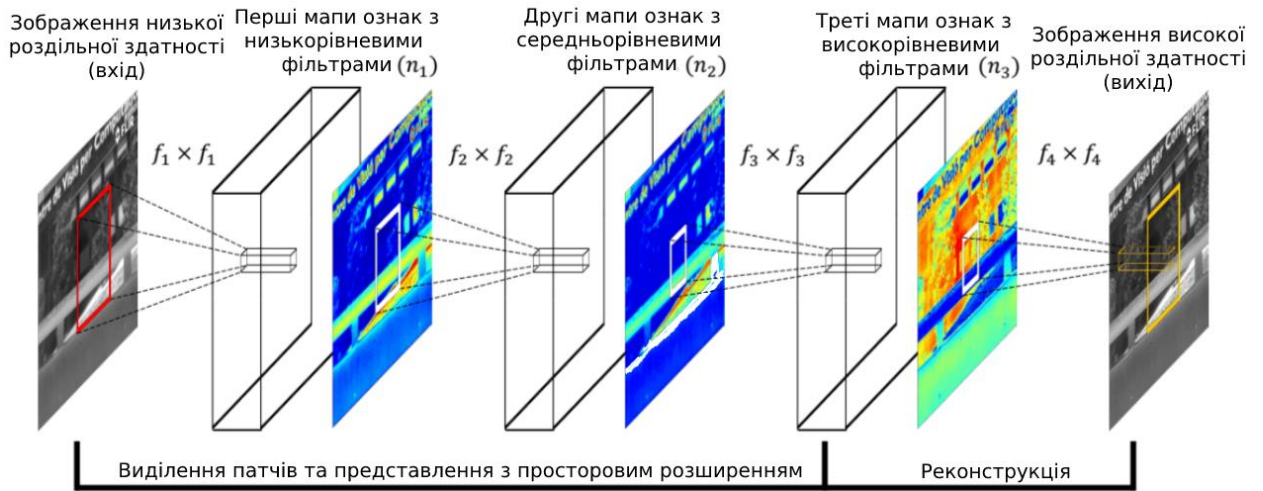


Рисунок 1.9 – Запропонована архітектура ЗНМ [23]

Ідеї метода [21] також знайшли своє застосування для задачі ЗРЗ на термальних зображеннях [24]. Автори дослідили використання датасетів з інших доменів для тренування моделі. Як показали результати, модель, що натренована на збільшення звичайних RGB зображень краще збільшує

термальні зображення, ніж модель, що натренована на термальних зображеннях. На рис. 1.10 зображено запропоновану архітектуру ЗНМ.



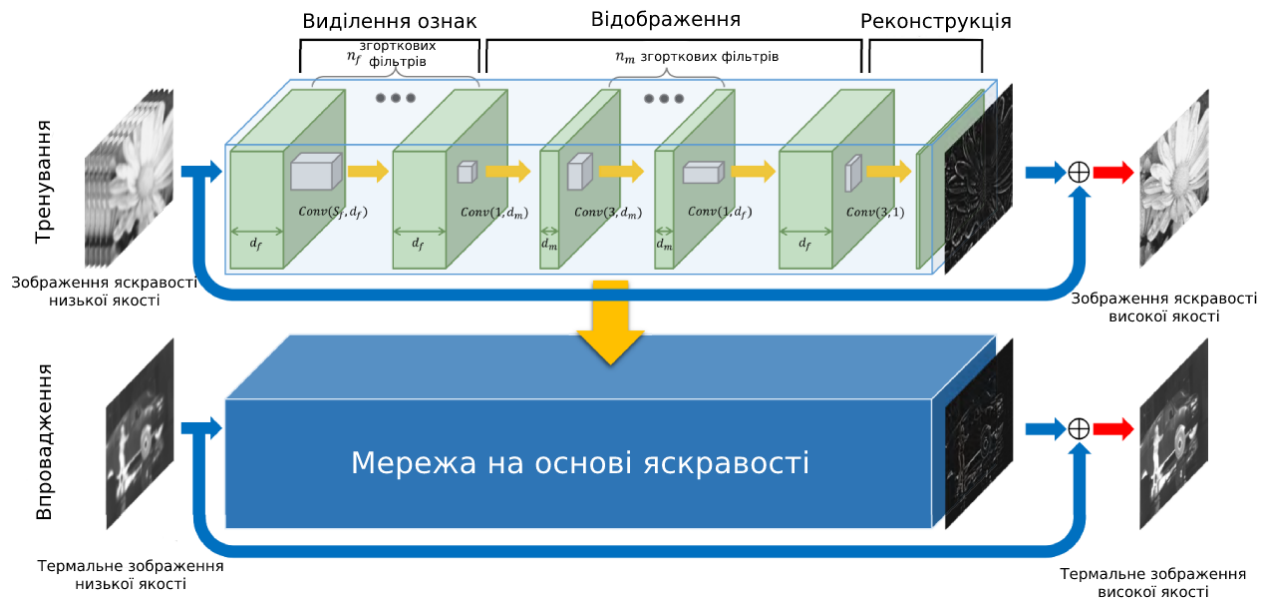
Умовні позначки:

n_1, n_2, n_3 – кількість фільтрів;

f_1, f_2, f_3, f_4 – розмір фільтрів.

Рисунок 1.10 – Запропонована архітектура ЗНМ [24]

У методі [25] автори також досліджували вплив домену датасету на якість тренування і результатів моделі. Автори проводили експерименти у таких колірних моделях, як RGB, HSL, HSI, HSV. За результатами експерименту, модель краще збільшує термальні зображення, якщо була натренована на датасеті у колірній моделі HSV (V – яскравість). Крім того, архітектура моделі також була натхнена дослідженням [21]. На рис. 1.11 зображено запропоновану архітектуру ЗНМ.



Умовні позначки:

n_f, n_m – кількість шарів;

d_f, d_m – кількість фільтрів;

s_f – розмір фільтра;

Рисунок 1.11 – Запропонована архітектура ЗНМ [25]

У методі [26] автори, навпаки, показали, що тренування моделі на датасеті термальних зображень дає кращі результати, ніж тренування на датасеті звичайних RGB зображень. Цей метод використовує ідеї методу ЗРЗ [27], у якому запропоновано використовувати стек мап ознак та підсилювати вихід входним зображенням, що збільшене методом бікубічної інтерполяції. На рис. 1.12 зображено архітектуру методу [26], а на рис 1.13 зображено архітектуру методу [27].

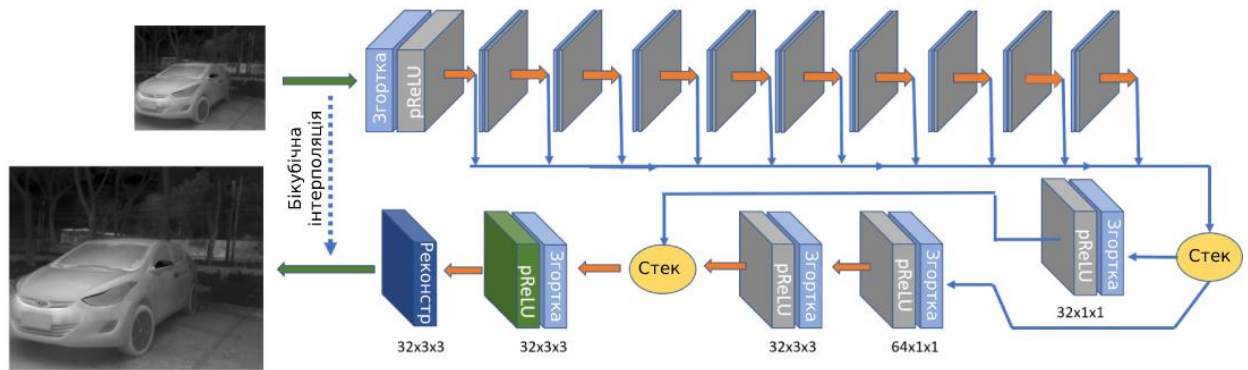


Рисунок 1.12 – Запропонована архітектура ЗНМ для збільшення термальних зображень [26]

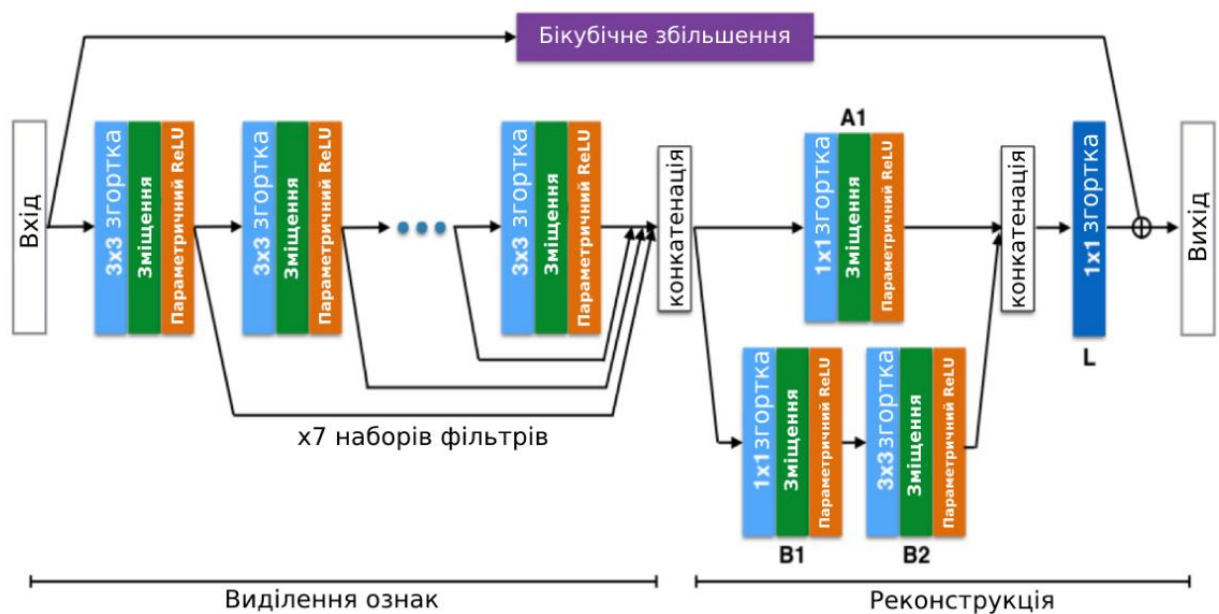


Рисунок 1.13 – Запропонована архітектура ЗНМ [27]

1.2.4 Генеративно змагальні нейронні мережі

Ще один спосіб вирішення задачі ЗРЗ – використання генеративно змагальних мереж (ГЗМ). ГЗМ складається з двох нейронних мереж: генератора та дискримінатора. Генератор бере точку з простору випадкового шуму у якості входу та генерує зображення. Дискримінатор бере на вхід зображення, та намагається визначити, чи є це зображення справжнім або згенерованим. Навчання ГЗМ – це гра з нульовою сумою, у якій генератор

навчається краще обманювати дискримінатор, а дискримінатор навчається краще розрізняти справжні та згенеровані зображення [28].

Роботу ГЗМ схематично зображено на рис. 1.14.

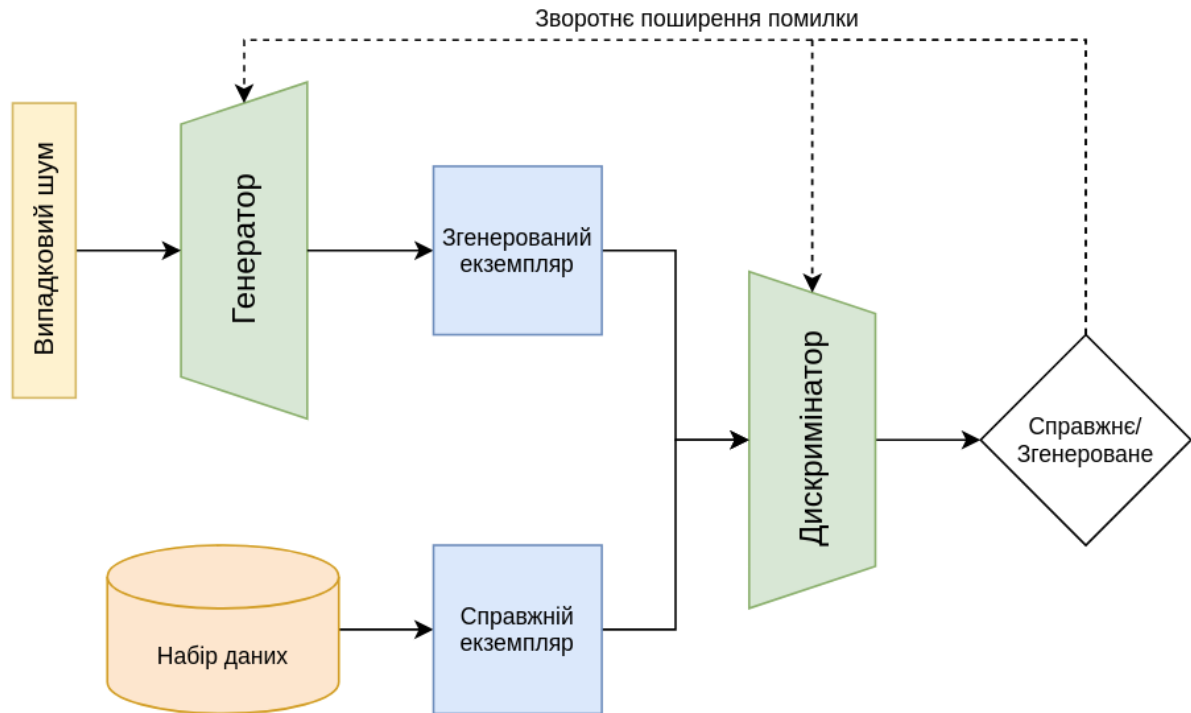


Рисунок 1.14 – Схема роботи ГЗМ

Математично процес навчання ГЗМ можна представити наступним чином. Для того, щоб вивчити розподілення генератора p_g на даних x , спочатку визначається випадковий шум $p_z(z)$, що представляє відображення на простір даних як $G(z; \theta_g)$, де G — це багатошаровий перцептрон (генератор) з параметрами θ_g . Також визначається другий багатошаровий перцептрон (дискримінатор) $D(x; \theta_d)$, що видає єдине число. $D(x)$ — це вірогідність того, що x є реальними даними. Дискримінатор навчається максимізувати вірогідність правильної класифікації як справжніх, так і згенерованих даних. Одночасно, генератор навчається мінімізувати цю вірогідність. Іншими словами, D та G грають в мінімакс гру з цільовою функцією $V(D, G)$ (1.4) [28].

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)} [\log(D(x))] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (1.4)$$

Існують також умовні генеративно змагальні мережі (УГЗМ). Генератор і дискримінатор можуть бути розширеними до умовних моделей, якщо обидві моделі зумовлені деякою інформацією y . Це може бути будь-яка допоміжна інформація, наприклад, клас чи зменшене зображення (для задачі ЗРЗ). Зумовлення може бути виконане за допомогою подання на вхід генератору та дискримінатору y в додатковий вхідний шар [29].

Таким чином, цільова функція приймає наступний вигляд (1.5) [29]:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x|y)} [\log(D(x))] + E_{z \sim p_z(z)} [\log(1 - D(G(z|y)))] \quad (1.5)$$

Схематично це зображено на рис. 1.15.

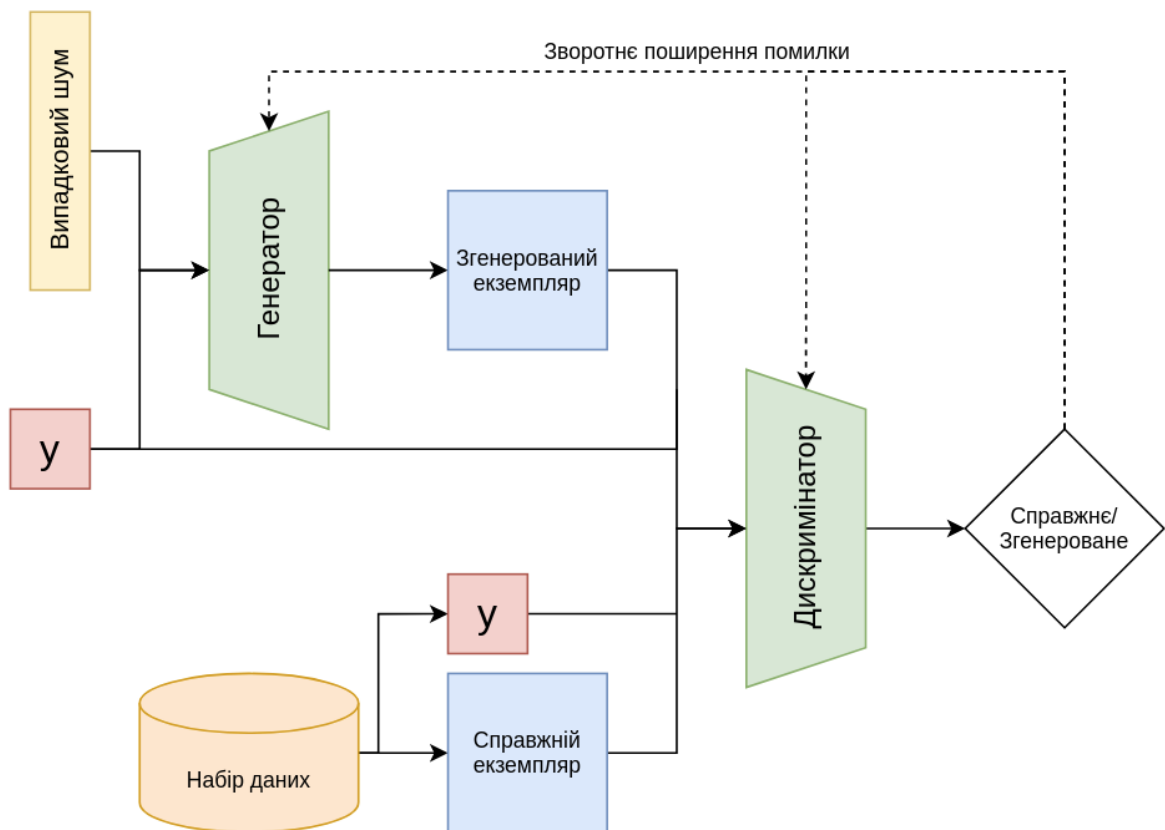
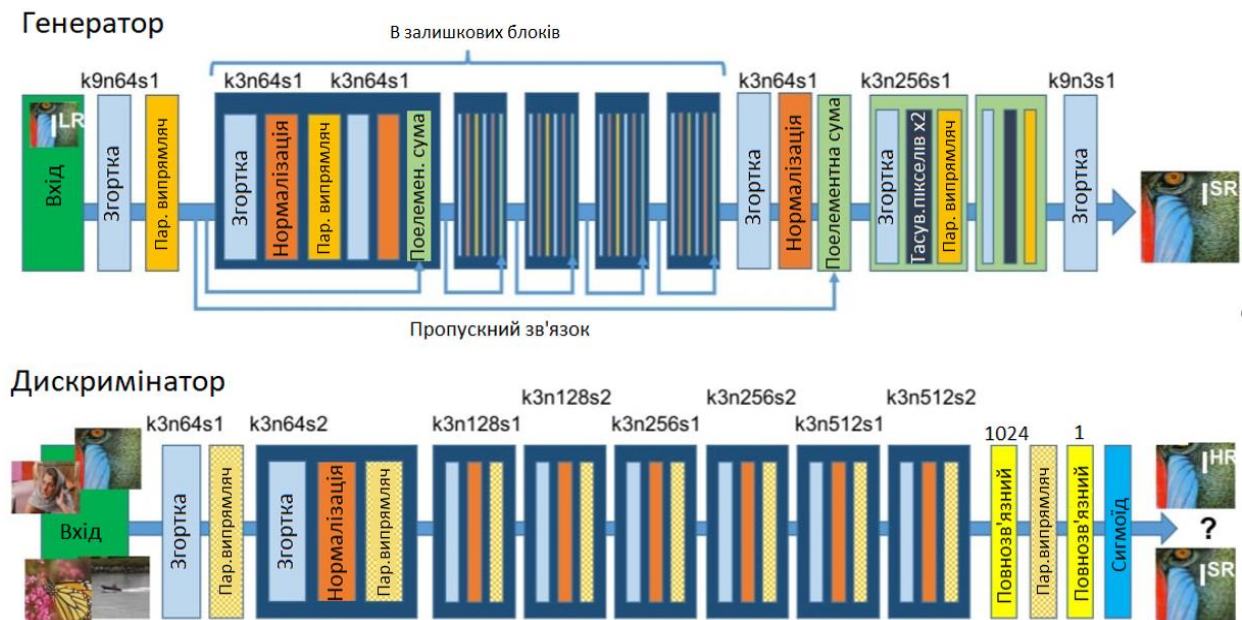


Рисунок 1.15 – Схема роботи УГЗМ

Ідея ЗРЗ за допомогою ГЗМ була представлена у дослідженні [30]. Ця технологія здатна збільшувати зображення у 4 рази. Автори запропонували використовувати перцептивну функцію втрат для вирішення цієї задачі. Перцептивна функція втрат складається зі змагальної функції втрат та функції втрат контенту. Змагальна функція втрат використовує результати дискримінатора, що підштовхує його покращувати свої оцінки. Функція втрат контенту використовує мапи ознак з мережі VGG19 з декількох шарів для оцінки якості згенерованого зображення [30]. Це допомагає генератору покращити свої результати, оскільки мапи з шарів мережі VGG19 містять у собі такі ознаки, як контури, грані, силуети об'єктів, інформація про які дозволяє збільшити якість збільшення зображень.

Крім того, генератор використовує пропускні зв'язки у своїй архітектурі.

На рис. 1.16 зображені архітектури генератора та дискримінатора.



Умовні позначки:

k – розмір ядра фільтру;

n – кількість фільтрів;

s – крок згортки.

Рисунок 1.16 – Архітектура ГЗМ [30]

У статті [31] автори запропонували покращений варіант методу, описаного вище. По-перше, був запропонований спеціальний блок без нормалізації та з багатьма пропускними зв'язками, що називається Residual-in-Residual Dense Block (RRDB). Крім того, було запропоновано використовувати ідеї релятивістської ГЗМ [32], що дозволяє дискримінатору прогнозувати відносну реальність даних замість абсолютного значення. Автори також пропонують використовувати мапи ознак з мережі VGG до функції активації, оскільки ознаки до активації містять більше інформації. На рис. 1.17 зображено архітектуру генератора, а на рис. 1.18 зображено архітектуру блоку RRDB.

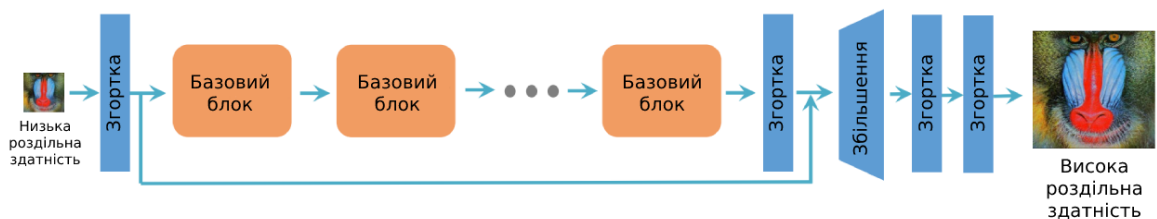
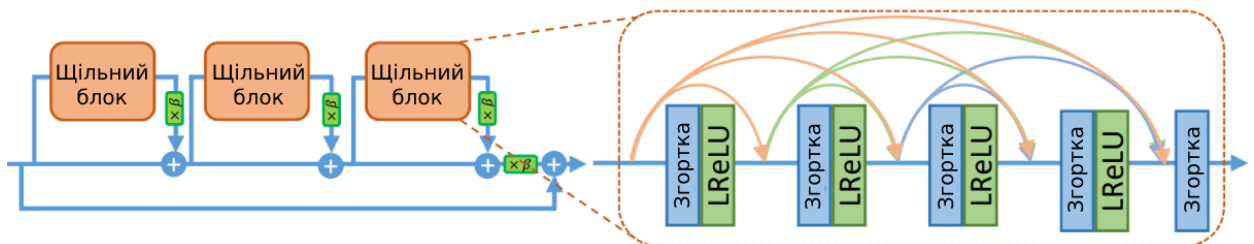


Рисунок 1.17 – Архітектура генератора [31]



Умовні позначки:

β – ваговий коефіцієнт.

Рисунок 1.18 – Архітектура блоку RRDB [31]

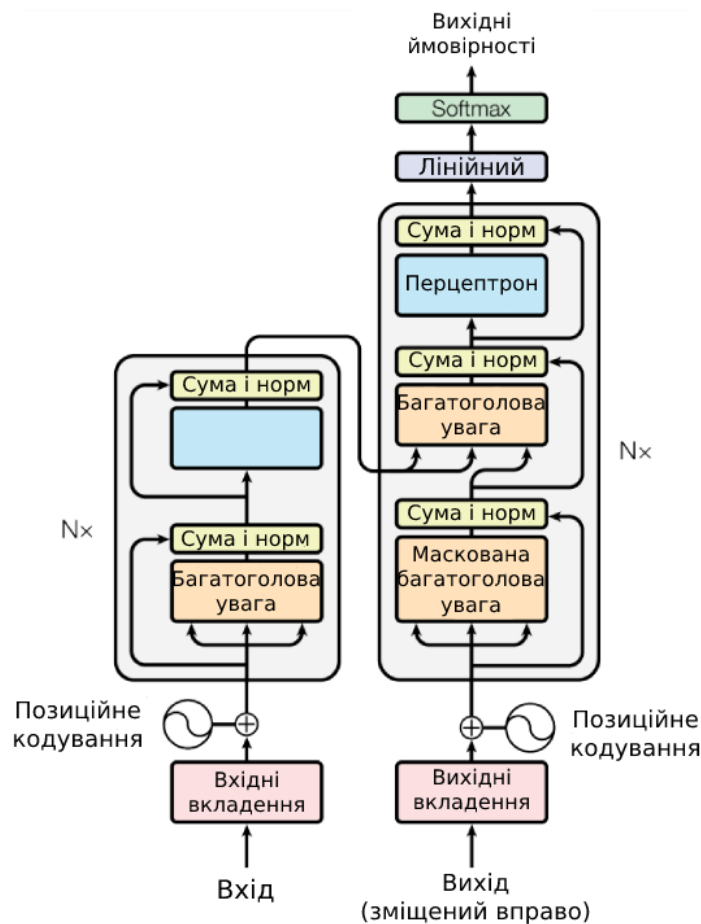
1.2.5 Трансформери

Останнім часом в глибокому навчанні все частіше використовується архітектура нейронної мережі під назвою “трансформер” [33] для вирішення

низки задач, зокрема в сфері обробки природної мови та сфері комп'ютерного зору.

Трансформер складається з кодувальника і декодувальника. Кодувальник відображає вхідну послідовність даних $(x_1, x_2, \dots, x_n)$ в послідовність їхніх представлень $z = (z_1, z_2, \dots, z_n)$. Маючи на вході z , декодувальник генерує вихідну послідовність $(y_1, y_2, \dots, y_n)$ по одному елементу [33].

На рис. 1.19 зображено архітектуру трансформера.



Умовні позначки:

N – кількість блоків.

Рисунок 1.19 – Схема архітектури трансформера [33]

Однією з особливостей даної нейронної мережі є використання так званого механізму уваги [34]. Цей механізм вперше був запропонований для покращення обробки природної мови.

Основна проблема моделей виду “послідовність до послідовності” (наприклад, машинний переклад) полягала в тому, що вони не були здатні якісно оброблювати довгі послідовності, оскільки декодувальником використовувався останній прихований стан, який видавала рекурентна нейронна мережа кодувальника. Крім того, у рекурентних нейронних мережах присутня проблема “затухаючого градієнта”, коли при обробці довгої послідовності поступово втрачається інформації про перші елементи цієї послідовності.

Механізм уваги, навпроти, зберігає та використовує усі приховані стани на кожному кроці. Таким чином, на кожному кроці декодувальник має доступ до усієї вхідної послідовності та шляхом зважування може вибирати значущі елементи вхідної послідовності для формування виходу.

В трансформері використовуються масштабована скалярнодобуткова увага та багатоголова увага [33].

В масштабованій скалярнодобутковій увазі вхід складається з запитів і ключів розміром d_k та значень, розміром d_v . Запити складаються в матрицю Q , ключі та значення складаються в матриці K і V відповідно [33]. Якщо для ключів і запитів використовуються одні й ті самі дані, то такий механізм називають самоувагою.

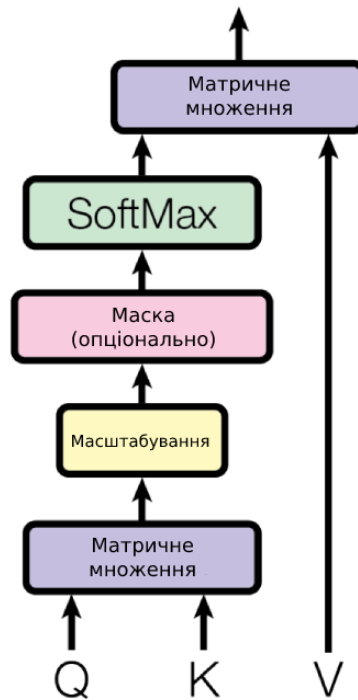
Масштабована скалярнодобуткова увага розраховується наступним чином [33] (1.6):

$$Attention(Q, K, V) = \text{soft max}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (1.6)$$

де Q, V, K – матриці запитів, значень і ключів відповідно;

d_k – розмірність матриць Q, K .

На рис. 1.20 зображено механізм масштабованої скалярнодобуткової уваги.



Умовні позначки:

Q , V , K – матриці запитів, значень і ключів відповідно.

Рисунок 1.20 – Схема масштабованої скалярнодобуткової уваги [33]

У механізмі багатоголової уваги замість того, щоб виконувати одну функцію уваги над ключами, значеннями та запитами, доцільніше лінійно спроектувати запити, ключі та значення h разів з різними навчальними проєкціями. На кожній з таких проєкцій запитів, ключів та значень паралельно розраховується функція уваги. Потім, результати конкатенуються та лінійно проєктуються [33].

Багатоголова увага розраховується наступним чином [33] (1.7):

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O, \quad (1.7)$$

де Q , V , K – матриці запитів, значень і ключів відповідно;

$head$ – одна голова багатоголової уваги;

W^O – матриця вагів розмірністю hd_v на d_{model} ;

d_{model} – довжина вхідного вектора даних;

d_v – розмірність матриці значень V ;

h – кількість голів.

Формула розрахунку однієї голови має наступний вигляд [33] (1.8):

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V), \quad (1.8)$$

де i – індекс голови;

Q, V, K – матриці запитів, значень і ключів відповідно;

W_i^Q – матриця ваг запитів, розмірністю d_{model} на d_k ;

W_i^K – матриця ваг ключів, розмірністю d_{model} на d_k ;

W_i^V – матриця ваг значень, розмірністю d_{model} на d_v ;

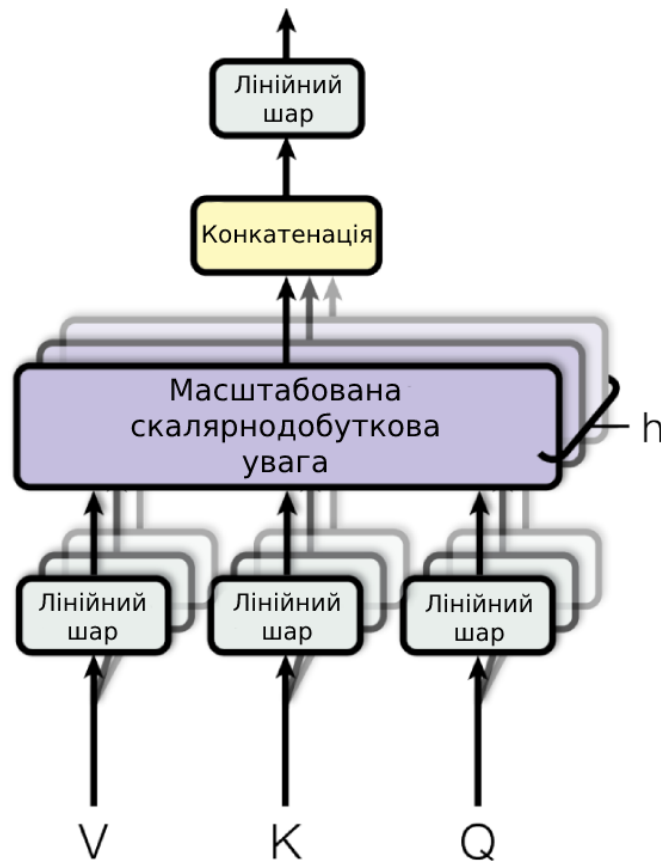
d_{model} – довжина вхідного вектора даних;

d_v – розмірність матриці значень V ;

d_k – розмірність матриці ключів K .

Багатоголова увага дозволяє моделі одночасно оброблювати інформацію з різних представлень.

На рис. 1.21 зображено механізм багатоголової уваги.



Умовні позначки:

Q, V, K – матриці запитів, значень і ключів відповідно;

h – кількість голів.

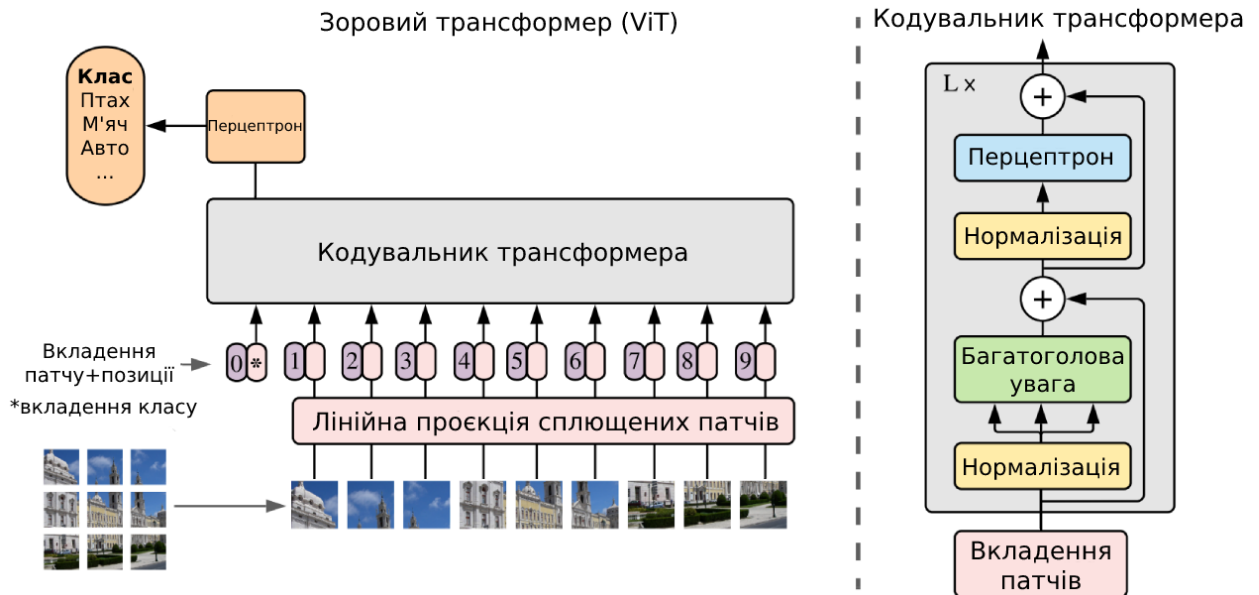
Рисунок 1.21 – Схема механізму багатоголової уваги [33]

Архітектура трансформера була розроблена для вирішення задачі машинного перекладу, але завдяки своїй робастності може використовуватись для обробки зображень.

У дослідженні [35] автори пропонують використовувати трансформер, а саме кодувальник, для класифікації зображень, при цьому з максимальним збереженням оригінальної архітектури. Така архітектура називається зоровий трансформер. Для цього картинка розбивається на декілька патчів (у статті патч – 16×16 пікселів), таким чином на вхід трансформера подається послідовність патчів. Для збереження інформації про розташування патчів до вхідної послідовності додається послідовність закодованої інформації про позицію цих патчів один відносно одного (позиційне кодування). Для того, щоб модель могла навчатися класифікувати зображення, до вхідної

послідовності на початок додається репрезентація класу зображення, що буде навчатися під час тренування моделі. Класифікаційна голова на виході є перцептроном з одним прихованим шаром.

На рис. 1.22 зображено зоровий трансформер.



Умовні позначки:

L – кількість блоків.

Рисунок 1.22 – Зоровий трансформер [35]

Незважаючи на високу якість роботи зорових трансформерів, у них є декілька важливих недоліків. Один з таких недоліків полягає в обробці зображень з великою роздільною здатністю, оскільки зі збільшенням зображення обчислювальна складність росте квадратично. Крім того, архітектура звичайного зорового трансформера погано підходить для вирішення інших задач комп'ютерного зору, наприклад, сегментації, оскільки в цій задачі важливо якісно розрізняти ознаки зображень на різних рівнях обробки.

Для вирішення цієї проблеми було запропоновано вдосконалення архітектури зорового трансформера – Swin трансформер [36] (Shifted window – зміщене вікно), однією з важливих переваг якого є лінійна обчислювальна складність відносно розміру зображення.

Swin трансформер включає в себе декілька важливих концепцій. Перша – це ієрархічна мапа ознак. Модель конструює ієрархічну мапу ознак, починаючи з невеликих патчів і поступово, з кожним шаром, з'єднує сусідні патчі [36]. Це дозволяє ефективно вирішувати задачі, наприклад, сегментацію на зображеннях з великою роздільною здатністю, оскільки в кожній мапі містяться різні за розміром ознаки.

На рис. 1.23 зображено ієрархічну мапу ознак swin трансформера.

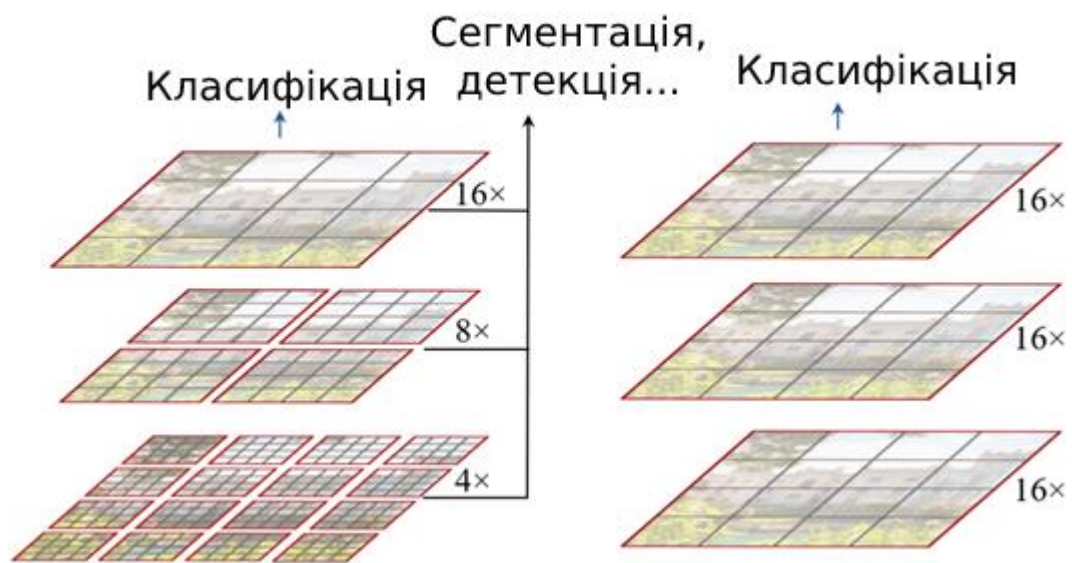
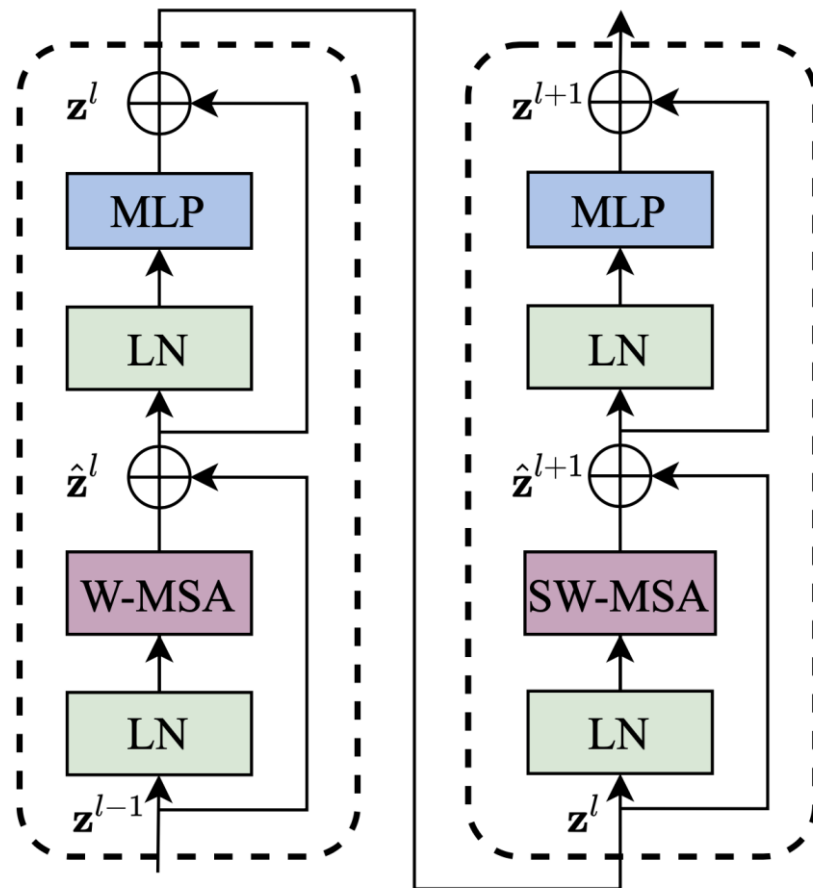


Рисунок 1.23 – Ієрархія мап ознак swin трансформера (зліва) і зорового трансформера (справа) [36]

Ще однією особливістю swin трансформера є використання особливих блоків, що складаються з механізмів віконної багатоголової самоуваги та зсунутої віконної багатоголової самоуваги (рис 1.24).



Умовні позначки:

$z^l, \hat{z}^l$ – вихідні ознаки блока l ;

LN – нормалізація;

MLP – багатошаровий перцептрон;

$W\text{-MSA}$ – віконна багатоголова самоувага;

$SW\text{-MSA}$ – зсунута віконна багатоголова самоувага.

Рисунок 1.24 – блок swin трансформера [36]

У той час, коли механізм звичайної багатоголової самоуваги розраховується глобально, а взаємозв'язки між патчами розраховуються за принципом “кожен з кожним”, механізм віконної багатоголової самоуваги розраховується у локальних неперетинаючихся вікнах, а взаємозв'язки розраховуються між патчами у окремо взятому вікні [36]. Це дозволяє зменшити складність алгоритма уваги з квадратичної до лінійної, оскільки кількість патчів у вікні фіксована.

Однак у віконній багатоголової уваги відсутні зв'язки між вікнами, що може значно погіршити якість моделі. Щоб подолати цей недолік,

використовується зміщена віконна багатоголова увага [36]. Для цього вікна зсуваються в нижній правий кут на відстань, що дорівнює половині розміру вікна. Потім в новоутворених вікнах розраховуються взаємозв'язки між патчами механізмом багатоголової самоуваги. Такий трюк дозволяє розраховувати зв'язки між неперетинаючимися вікнами, що значно впливає на якість вирішення задач комп'ютерного зору.

На рис. 1.25 зображено механізм зсуву вікон.

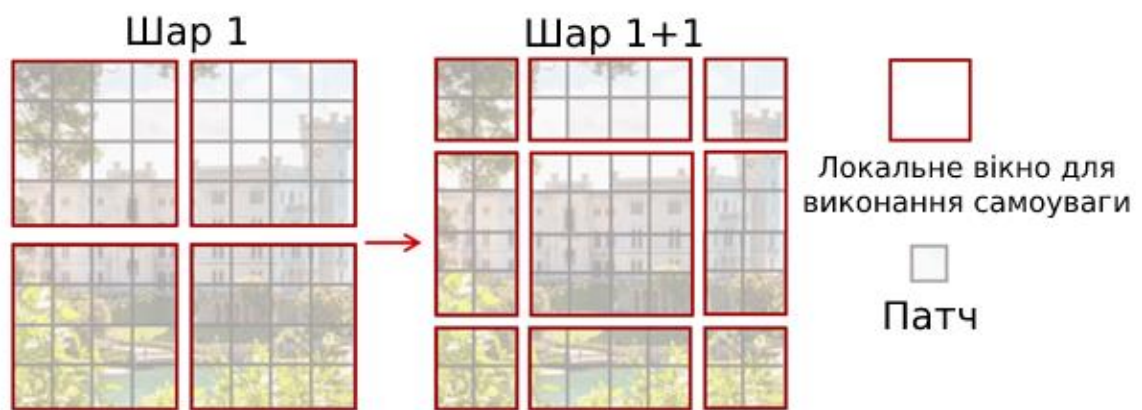
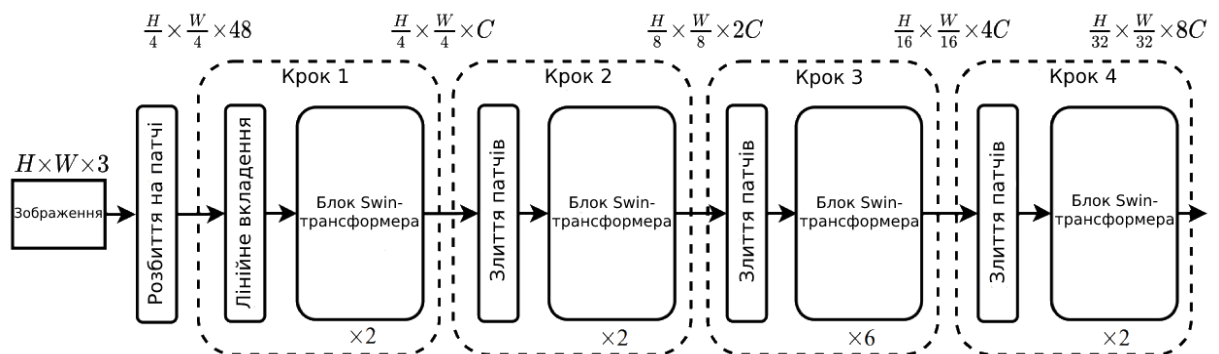


Рисунок 1.25 – Механізм зсуву вікон [36]

На рис. 1.26 зображено архітектуру swin трансформера.



Умовні позначки:

H – висота вхідного зображення;

W – ширина вхідного зображення;

C – кількість мап ознак.

Рисунок 1.26 – Архітектура swin трансформера [36]

Ідея використання трансформерів, а саме swin трансформера, була запропонована в дослідженні [37]. Модель (SwinIR) складається з трьох частин: блок вилучення поверхневих ознак, блок вилучення глибоких ознак та блок реконструкції зображення. У свою чергу, блок вилучення глибоких ознак складається з декількох пропусних блоків swin трансформера, кожен з яких має декілька шарів swin трансформера з пропусним зв'язком [37]. Така архітектура дозволила методу досягти найкращих серед інших методів результатів у відновленні зображень на момент виходу дослідження.

На рис. 1.27 зображено архітектуру моделі SwinIR.

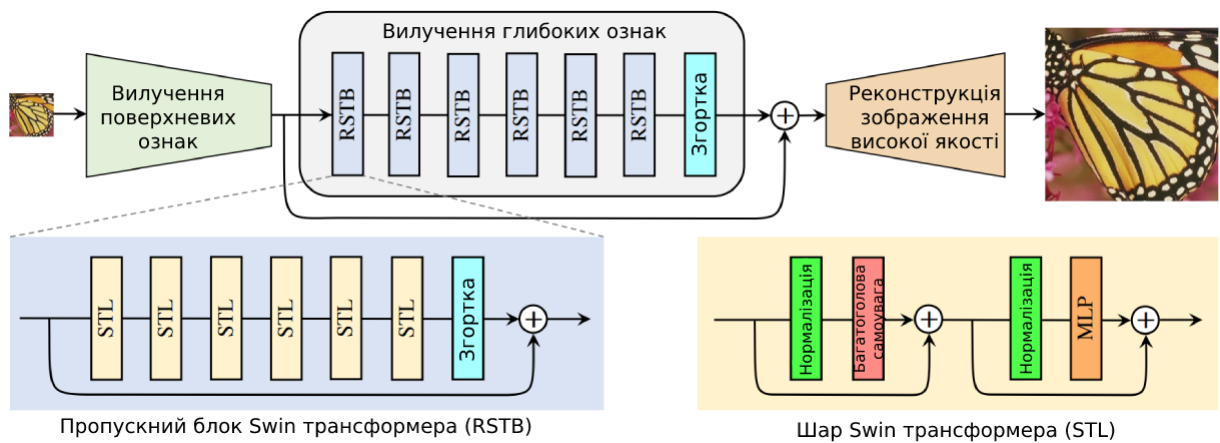


Рисунок 1.27 – Архітектура моделі SwinIR [37]

1.3 Порівняння методів ЗРЗ

Проаналізувавши методи ЗРЗ, можна зробити порівняльну характеристику (табл. 1.1).

Таблиця 1.1 – Порівняння методів ЗРЗ

Критерій / метод	Класичні алгоритми	ЗНМ	ГЗМ	Трансформери
Швидкість роботи	+++	++	++	++
Складність реалізації	+	++	++	+++
Узагальнення	+	++	++	+++
Складність навчання	Не навчаються	++	+++	+++
Якість роботи	+	++	+++	++++

1.4 Постановка задачі до роботи

У роботі розглядається вирішення задачі ЗРЗ термальних зображень з використанням різних методів. Основним етапом є моделювання архітектур нейромереж та їхня реалізація.

Математично сформулювати постановку задачі можна наступним чином. Нехай дано деяке зображення I_{HR} та його зменшену копію I_{LR} . Необхідно розробити модель ЗРЗ m . Збільшене зображення описується наступним чином (1.9):

$$I_{SR} = m(I_{LR}) \quad (1.9)$$

Створені моделі необхідно порівняти, використовуючи деяку метрику L над заданим зображенням та його збільшеною копією. У загальному вигляді це можна записати наступним чином (1.10):

$$v = L(I_{HR}, I_{SR}) \quad (1.10)$$

У якості метрик доцільно використовувати індекс структурної подібності (SSIM) та пікове співвідношення сигналу до шуму (PSNR).

SSIM задається наступним чином (1.11):

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \quad (1.11)$$

де x, y – області зображення розміром $N \times N$;

μ_x – середнє x ;

μ_y – середнє y ;

σ_x – дисперсія x ;

σ_y – дисперсія y ;

c_1, c_2 – змінні [38].

Змінні c_1, c_2 визначаються наступним чином (1.12):

$$c_i = (k_i L)^2, \quad (1.12)$$

де $k_1 = 0.01, k_2 = 0.03$ – константи;

L – динамічний діапазон пікселів [38];

i – індекс змінної (1 чи 2).

PSNR задається наступним чином (1.13):

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right) = 20 \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right), \quad (1.13)$$

де MAX_I – максимальне значення, яке приймається пікселем зображення (255, коли пікселі мають розрядність 8 біт);

MSE – середньоквадратичне відхилення між двома зображеннями (оригінальним та збільшеним) [39].

Метою роботи є програмна реалізація методів ЗРЗ термальних зображень з використанням ШНМ.

Для досягнення поставленої мети необхідно виконати наступні кроки:

- розробити архітектури моделей реалізувати їх;
- створити навчальну та тестову вибірки зображень;
- навчити моделі та проаналізувати результати;
- створити програмне забезпечення для можливості використовувати моделі.

1.5 Висновки за розділом 1

У даному розділі було проаналізовано предметну область, а саме термографію та області її застосування. Також було проаналізовано методи ЗРЗ, серед яких: класичні алгоритми, ЗНМ, ГЗМ. На основі аналізу було наведено порівняльну характеристику. Крім того, було визначено основні критерії оцінки моделей, поставлено задачу до роботи та визначено основні кроки для досягнення поставленої мети.

2 Розробка архітектури мереж і програми

Перед початком реалізації та порівняння методів і розробки програми необхідно визначитись з набором програмних засобів, що будуть використовуватися. До програмних засобів можна віднести мову програмування, фреймворк глибинного навчання, а також додаткові бібліотеки для створення моделей ЗРЗ та набір даних.

2.1 Вибір мови програмування

Перед початком роботи необхідно обрати мову програмування. Сьогодні існує достатня кількість мов програмування, що використовуються в різноманітних сферах. Найбільш популярні в машинному і глибинному навчанні – це Python, R, C++, Java.

Python – це інтерпретована мова програмування високого рівня з динамічною типізацією та підтримкою об'єктно-орієнтованого програмування (ООП) [40]. До переваг мови програмування Python можна віднести простоту розуміння і написання коду, наявність великої кількості різноманітних бібліотек, а також автоматичний контроль пам'яті. Недоліками Python є порівняно низька швидкодія та помилки під час виконання програми, які складно відслідкувати та попередити через динамічну типізацію та інтерпретованість мови.

R – це мова програмування та середовище розробки головним чином для статистичних задач і аналізу даних [41]. Перевагами мови програмування R є наявність великої кількості пакетів і бібліотек для машинного навчання і аналізу даних, а також платформна незалежність. До недоліків можна віднести низьку швидкодію, складність у вивченні мови, а також погане керування пам'яттю.

C++ – це високорівнева статично типізована мова програмування, з підтримкою ООП та інших парадигм програмування [42]. До переваг мови

програмування C++ зазвичай відносять швидкодію, масштабованість та платформну незалежність. Одними з недоліків є складність у вивченні мови і ручне керування пам'яттю.

Java – це високорівнева статично типізована та об'єктно-орієнтована мова програмування, програми якої компілюються у байт-код та інтерпретуються віртуальною машиною Java [43]. Перевагами Java є зрозумілість синтаксису, наявність збирача сміття та платформна незалежність. Недоліками Java є відносно низька швидкодія та погане керування пам'яттю.

Проаналізувавши мови програмування, можна зробити їхню порівняльну характеристику (табл. 2.1).

Таблиця 2.1 – Порівняльна характеристика мов програмування

Критерій	Python	R	C++	Java
Типізація	Динамічна	Динамічна	Статична	Статична
Швидкість	++	++	++++	+++
Простота	+++	+	+	++
Кількість бібліотек	+++	+++	++	++
Підтримка машинного навчання	+++	+++	++	++

Таким чином, для вирішення поставленої задачі було обрано мову програмування Python, оскільки вона проста у використанні та має велику кількість бібліотек для машинного та глибинного навчання, а низька швидкість виконання програм, написаних мовою Python, компенсується тим, що ядра багатьох бібліотек (зокрема бібліотек глибинного навчання) написані мовою програмування C або C++.

2.2 Вибір фреймворку глибокого навчання

Наступним кроком є вибір фреймворку глибокого навчання, оскільки основною задачею роботи є створення ШНМ. До популярних фреймворків належать PyTorch, Tensorflow та Keras.

PyTorch – це фреймворк машинного навчання на базі бібліотеки Torch, що використовується для задач комп’ютерного зору, обробки природньої мови, тощо. Спочатку розроблювалась Meta AI, а тепер є частиною Linux Foundation [44]. Серед переваг PyTorch – велика кількість бібліотек зі збірками популярних моделей глибокого навчання, що значно спрощує процес створення та навчання моделей, а також популярність фреймворка серед наукових досліджень.

Tensorflow – це відкритий фреймворк машинного навчання для вирішення різноманітних задач штучного інтелекту. Розроблюється компанією Google [45]. Перевагою Tensorflow є оптимізованість під створення моделей глибокого навчання виробничого рівня і можливість створювати моделі для різних платформ.

Keras – це відкрита бібліотека машинного навчання, що працює поверх Tensorflow [46]. Keras поєднує в собі переваги Tensorflow, а також є простим у вивченні і створенні прототипів моделей глибокого навчання як для виробництва, так і для досліджень.

Проаналізувавши фреймворки глибокого навчання, можна зробити їхню порівняльну характеристику.

Таблиця 2.2 – Порівняльна характеристика фреймворків глибинного навчання.

Критерій	PyTorch	Tensorflow	Keras
Простота і зрозумілість	+++	++	+++
Гнучкість	+++	+++	++
Спільнота	+++	++	+++
Наукова популярність	+++	++	++
Оновлюваність	+++	++	++

Таким чином, для вирішення задачі було обрано фреймворк глибинного навчання PyTorch, оскільки він простий на інтуїтивно зрозумілий у використанні, а також має велику кількість наукових статей та моделей, написаних за допомогою нього, що спрощує

2.3 Вибір додаткових програмних бібліотек

Під час проведення експериментів з моделями глибинного навчання виникає необхідність контролювати написання коду, щоб він був читабельним та міг редагуватися без зниження надійності і великих витрат часу. Це особливо відчутно при проведенні експериментів з декількома моделями, коли необхідно узагальнювати код аби уникнути великої кількості повторів і помилок, що можуть виникнути під час додання до експериментів нових моделей. Крім того, різні методи і моделі були написані різними дослідниками і командами, що також ускладнює узагальнення коду для проведення експериментів.

Для того, щоб приділити більше уваги саме на проведення експериментів, було обрано зручну бібліотеку BasicSR [47], що написана мовою Python та працює на PyTorch. Ця бібліотека містить в собі базові

моделі ЗРЗ, а також пропонує зручний спосіб проведення експериментів через конфігураційні файли.

2.4 Вибір датасету

Оскільки задача ЗРЗ термальних зображень є досить специфічною, то кількість датасетів у відкритому доступі є досить низькою.

Один з найбільших та найпопулярніших – це датасет FLIR [48]. Цей датасет пропонує анотовані термальні та RGB зображення для розробки систем детекції об'єктів. Датасет містить понад 26000 анотованих фреймів та 15 різних категорій об'єктів. Загалом датасет містить майже 10000 термальних та більш ніж 9000 RGB зображень розміром 640 на 480 пікселів [48].

На рис. 2.1 зображено приклад даних з датасету FLIR.



Рисунок 2.1 – Приклад зображень з датасету FLIR [48]

Найпопулярнішим датасетом для вирішення задачі ЗРЗ RGB зображень є DIV2K [49]. Цей датасет містить 1000 зображень високої роздільної здатності (1500-2000 пікселів на стороні зображення), розділених на навчальну, тестову та валідаційну вибірки. Також датасет містить зменшені зображення в 2, 3, 4 та 8 разів.

На рис. 2.2 зображено приклад даних з датасету DIV2K.



Рисунок 2.2 – Приклад зображень з датасету DIV2K [49]

Проаналізувавши датасети, можна зробити їхню порівняльну характеристику.

Таблиця 2.3 – Порівняння датасетів для задачі ЗРЗ

Критерій	FLIR	DIV2K
Ліцензія	Некомерційне використання	Некомерційне використання
Тип зображень	Термальні, RGB	RGB
Кількість зображень	9,711 термальних та 9,233 RGB	1000 RGB
Розмір зображень	640 на 480 пікселів	Близько 1500-2000 пікселів на одній стороні зображення
Основна задача	Детекція об'єктів	ЗРЗ

2.5 Обробка даних

Для використання датасету FLIR для задачі ЗРЗ термальних зображень було виділено тільки термальні зображення (одноканальні) та створено версію датасету зі зменшеними в 2 рази термальними зображеннями (320 на 240 пікселів).

Для використання датасету DIV2K для задачі ЗРЗ термальних зображень було створено його версію з зображеннями у чорно-білому форматі. Таким чином, зображення мають один канал, значення пікселів якого знаходиться у проміжку від 0 до 255.

2.6 Архітектура компонентів програми

Для проведення експериментів було обрано наступні моделі:

- SRGAN;
- ESRGAN;
- SwinIR;
- ГЗМ SwinIR разом з дискримінатором (SwinIRGAN – запропонований метод);
- окремо генератор від SRGAN (SRResNet);
- окремо генератор від ESRGAN (RRDBNet).

Архітектура обраних моделей дозволяє збільшувати термальні зображення у 2 рази.

На рис. 2.3 зображено архітектуру генератора від SRGAN (SRResNet).

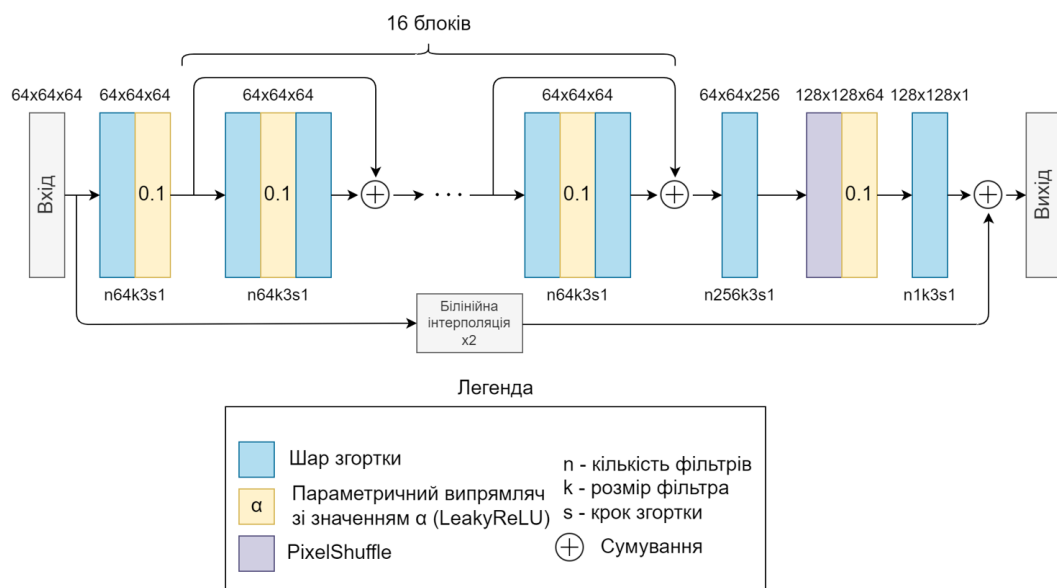


Рисунок 2.3 – Архітектура SRResNet

На рис. 2.4 зображено архітектуру генератора ESRGAN (RRBDNet).

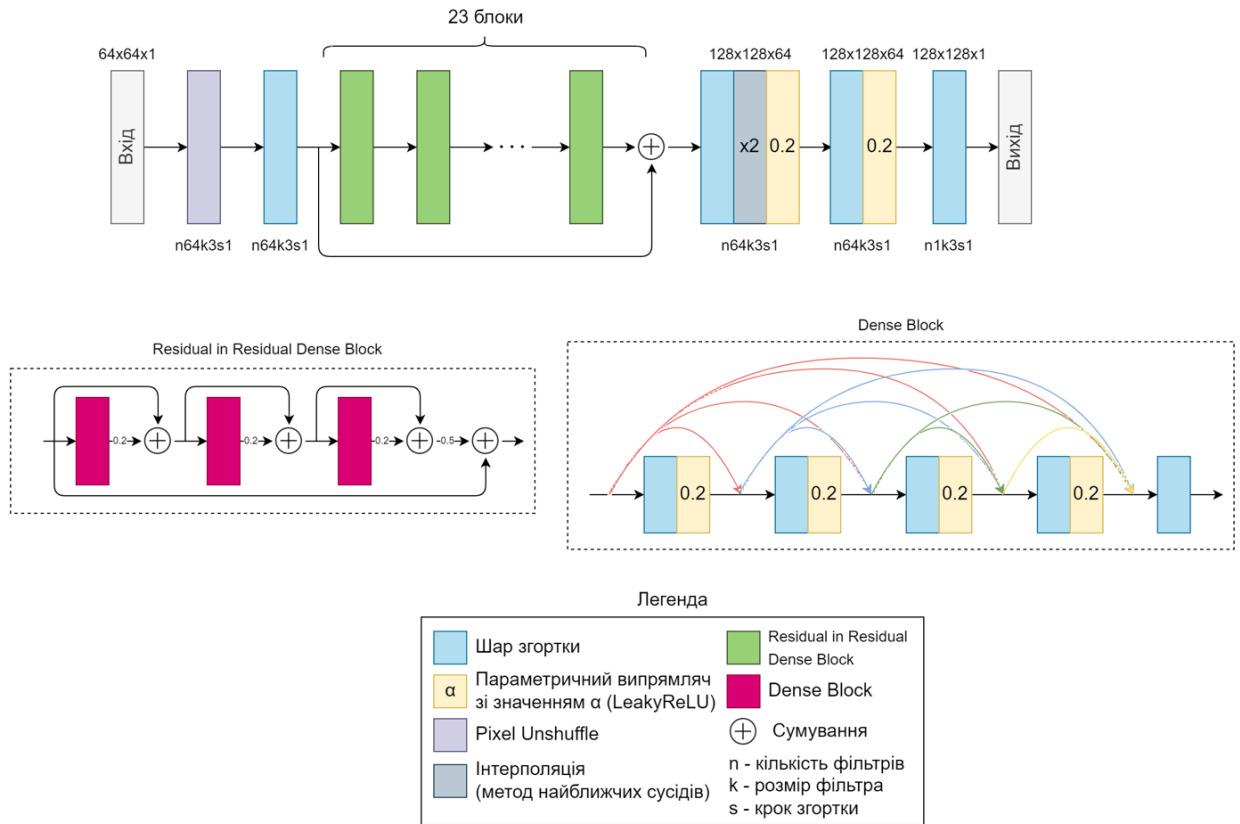


Рисунок 2.4 – Архітектура RRBDNet

На рис 2.5 зображено архітектуру дискримінатора ГЗМ.

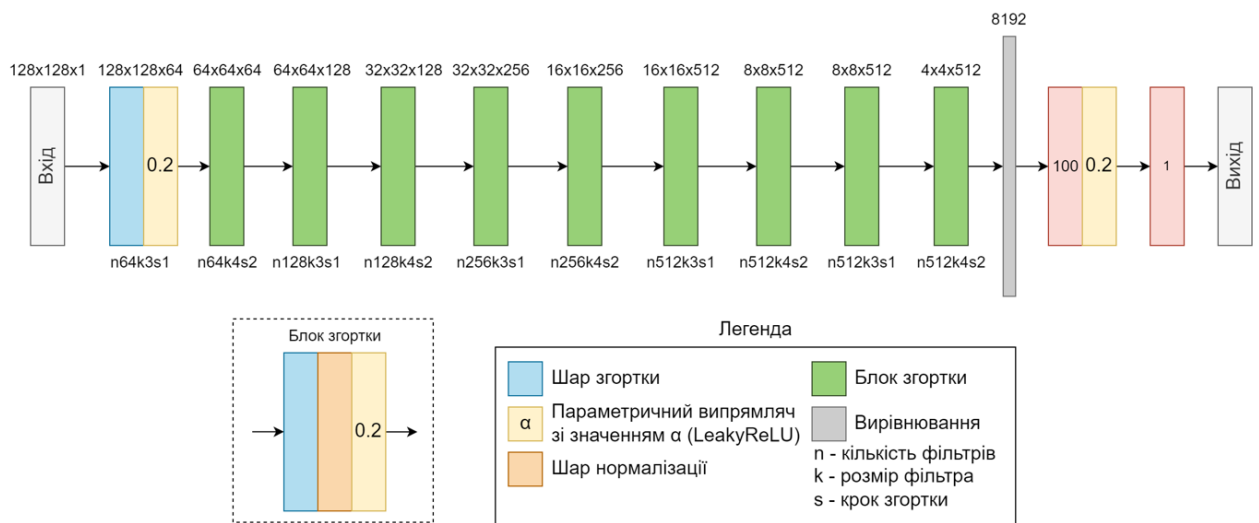


Рисунок 2.5 – Архітектура дискримінатора

На рис. 2.6 зображено архітектуру моделі SwinIR.

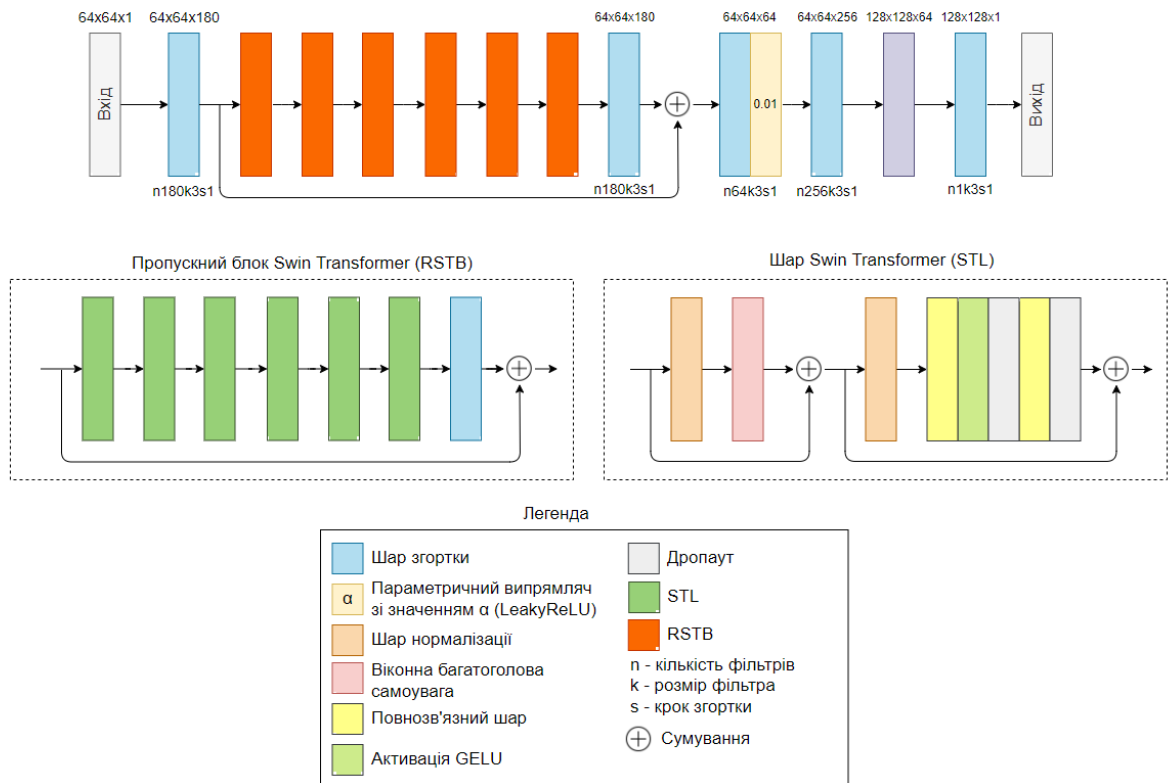


Рисунок 2.6 – Архітектура моделі SwinIR

В моделях використовуються дві функції активації, а саме параметричний випрямляч (Leaky ReLU) та GELU.

Функція параметричного випрямляча має наступний вигляд:

$$f(x) = \begin{cases} 0, & x \leq 0 \\ \alpha x, & x > 0 \end{cases}, \quad (2.1)$$

де $f(x)$ – параметричний випрямляч;

α – параметр випрямляча.

Функція активації GELU має наступний вигляд:

$$GELU = xP(X \leq x) = x\Phi(X), \quad (2.2)$$

де $\Phi(x)$ – функція розподілу ймовірностей стандартного нормального розподілу;

$$X \sim N(0,1) [50].$$

2.7 Архітектура системи

Архітектура розроблюваної системи складається з таких компонентів:

- моделі ЗРЗ, які оброблюють вхідне термальне зображення та видають відповідне збільшене;
- ядро системи, яке конвертує вхідні дані від користувача у формат, який приймають моделі, та вихідні дані від моделей;
- інтерфейс користувача, через який здійснюється зв'язок між ядром і користувачем програми.

2.8 Висновки за розділом 2

У даному розділі було проаналізовано мови програмування та фреймворки глибинного навчання. На основі аналізу було обрано мову програмування Python та фреймворк глибинного навчання PyTorch.

Також було обрано датасет для навчання моделей та додаткові бібліотеки для спрощення проведення експериментів.

Крім того, було обрано моделі для проведення експериментів зі ЗРЗ термальних зображень та запропоновано власну модель, а саме ГЗМ на основі SwinIR.

3 Аналіз результатів та реалізація програмного забезпечення

У даному розділі наведено аналіз результатів навчання моделей, а також описано вибір гіперпараметрів навчання.

3.1 Вибір гіперпараметрів навчання

Для навчання моделей необхідно визначитись з наступними гіперпараметрами моделі, а саме:

- функція втрат;
- крок навчання;
- алгоритм оптимізації;
- кількість епох та/або ітерацій;
- кількість зображень в пакеті (батчі);
- розмір вхідного патчу зображення під час навчання.

В якості алгоритму оптимізації для всіх моделей було обрано Adam [51] з кроком навчання 0,0002.

Кількість ітерацій становить 100000 для всіх моделей.

Кількість зображень в одному батчі – 16 для моделей SwinIR, ESRGAN і RRDBNet, 64 для моделей SRGAN і SRResNet. Такі значення були обрані відповідно до обчислювальної спроможності комп'ютера, на якому проходило навчання моделей.

Усі моделі навчались не на цілих зображеннях, а на квадратних патчах, що були виділені випадковим чином з кожного зображення. Це дозволяє прискорити навчання та змушує нейронну мережу приділяти увагу високочастотним ознакам. Так, для усіх моделей було обрано патч розміром 128 на 128 пікселів, крім класичної моделі SwinIR (96 на 96 пікселів).

Для ЗНМ використовується середня абсолютна помилка в якості функції втрат (3.1).

$$MAE(HR, SR) = \frac{1}{HW} \sum_{i=1}^W \sum_{j=1}^H |HR(i, j) - SR(i, j)|, \quad (3.1)$$

де H, W – висота і ширина зображення;

HR – оригінальне зображення;

SR – збільшене зображення;

i, j – координати пікселя зображення.

Для ГЗМ використовується перцептивна функція втрат для генератора.

3.2 Аналіз результатів навчання мереж

Результати навчання моделей наведено у таблиці 3.1.

Таблиця 3.1 – Результати навчання моделей

Модель	SSIM	PSNR
Метод найближчих сусідів	0,7389	30,3350
Білінійна інтерполяція	0,7557	31,1910
Бікубічна інтерполяція	0,7672	31,5820
SRGAN	0,6617	28,3197
SwinIRGAN	0,6751	29,6528
ESRGAN	0,6853	29,8002
SwinIR (DIV2K)	0,7642	31,4355
RRDBNet	0,7829	32,3622
SRResNet	0,7828	32,3471
SwinIR (FLIR)	0,7833	32,3688

На рис. 3.1 – 3.2 зображено значення метрик SSIM і PSNR на валідаційному наборі даних відповідно.

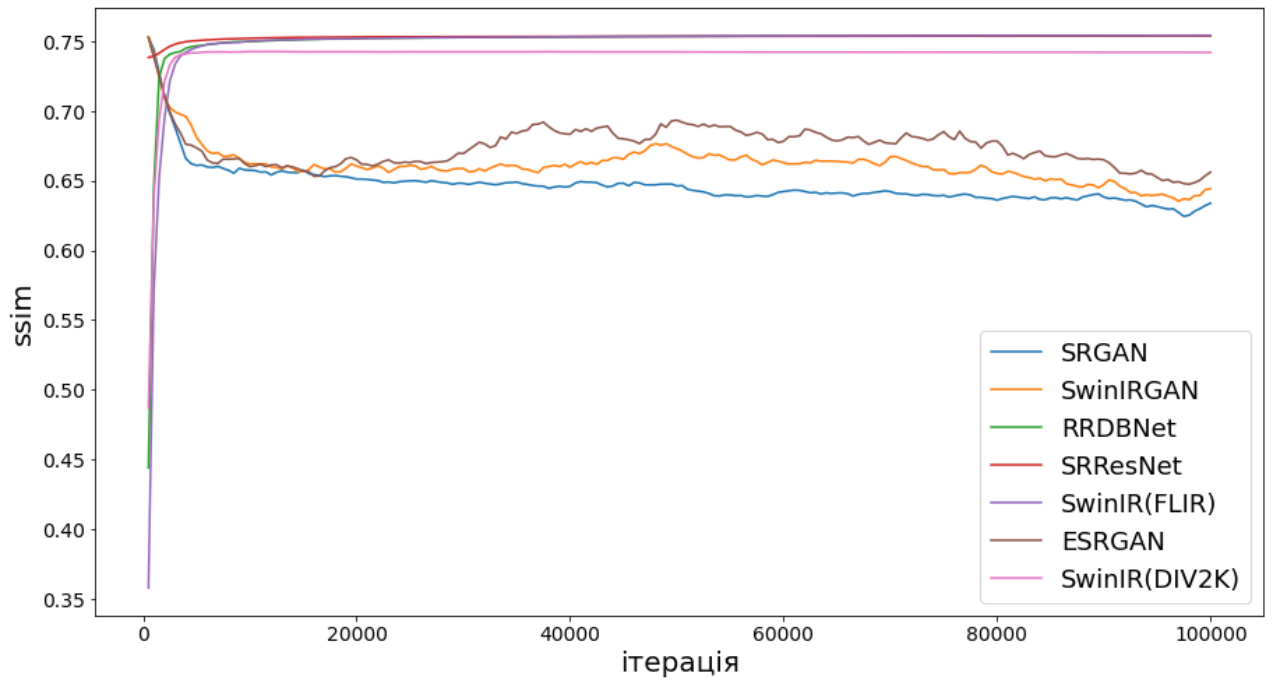


Рисунок 3.1 – Метрика SSIM на валідаційному наборі даних під час навчання

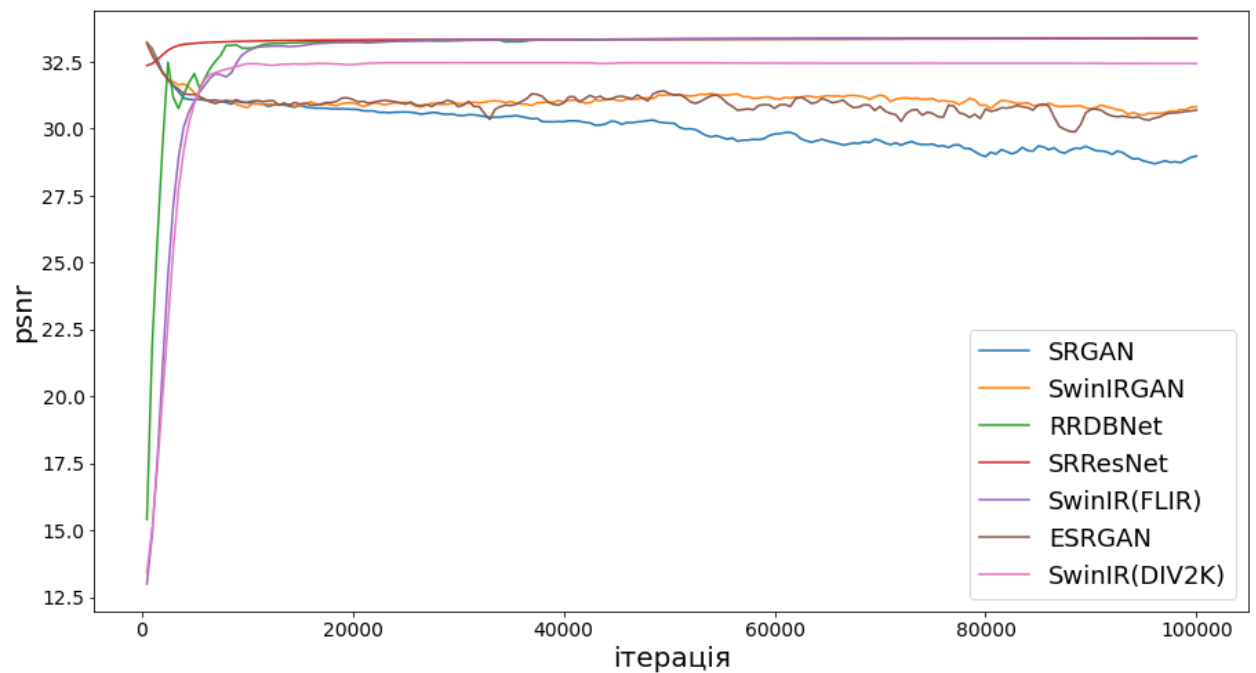


Рисунок 3.2 – Метрика PSNR на валідаційному наборі даних під час навчання

На рис. 3.3 зображено результати роботи моделей на одному із зображень з тестової вибірки FLIR.

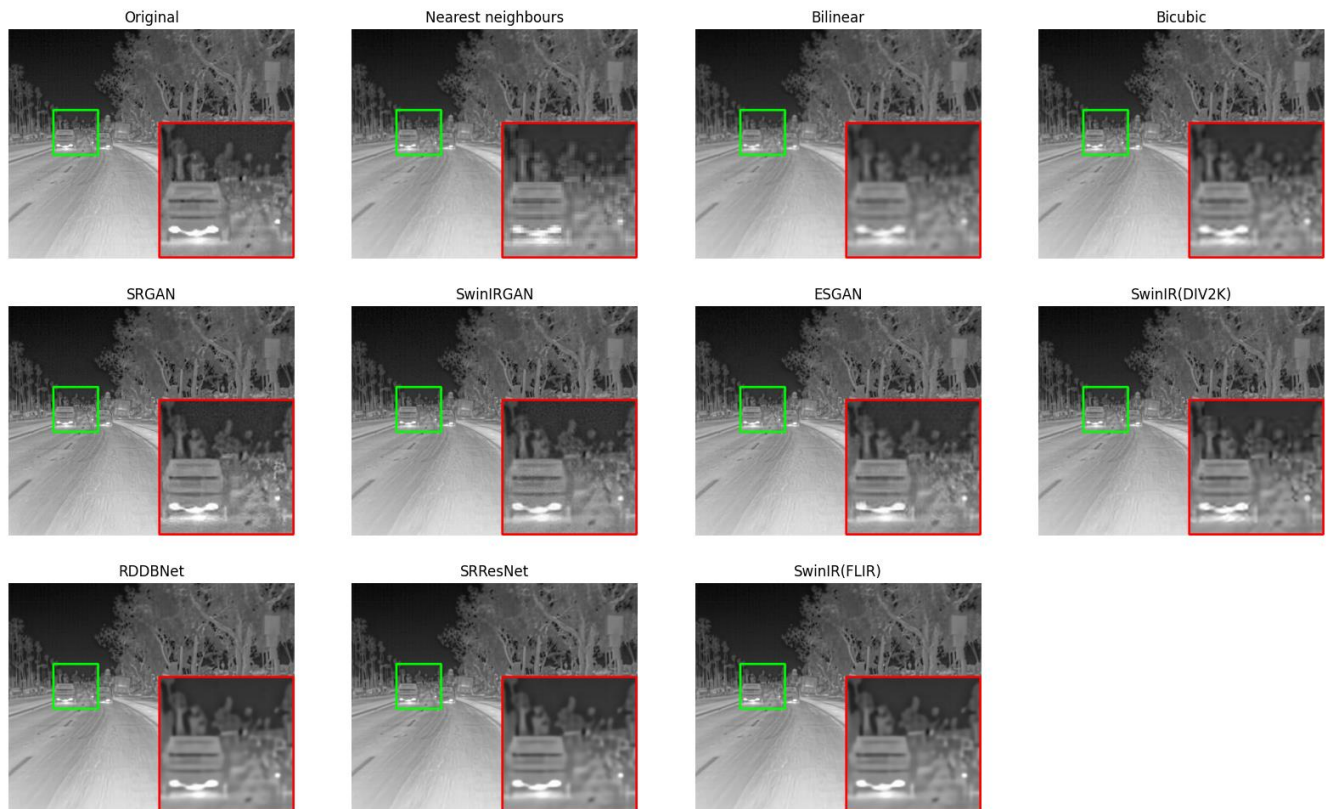


Рисунок 3.3 – Результати роботи моделей (термальне зображення з тестової вибірки FLIR [48])

Серед моделей ЗРЗ найкращі результати має SwinIR, натренована на датасеті FLIR.

Для усіх ГЗМ генератором були обрані відповідні натреновані моделі ЗНМ. Усі ГЗМ мають найнижчі результати, що можна пояснити великою кількістю артефактів на вихідному зображенні, складністю навчання ГЗМ, а також наявністю шуму на вхідних зображеннях.

Крім того, на моделях ЗНМ присутнє згладжування шуму, наявного на оригінальних зображеннях.

3.3 Реалізація компонентів програмного забезпечення

Реалізація системи програмного забезпечення була виконана мовою програмування Python. Під час розробки використовувались фреймворки

Flask (основа програми), PyTorch (завантаження та використання моделей) та бібліотека BasicSR.

На рис. 3.4 зображено взаємодію між основними компонентами програми.

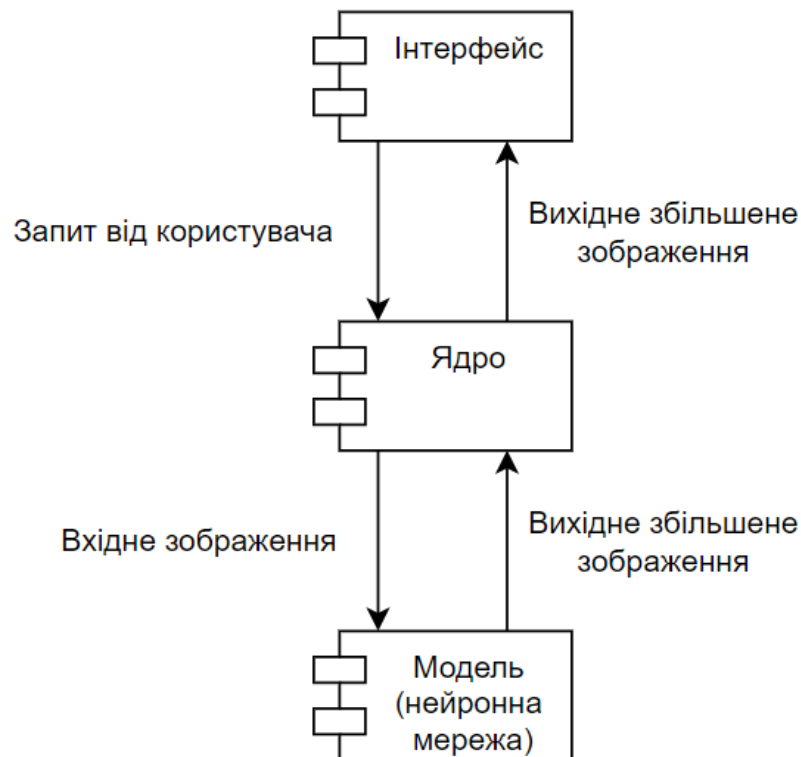


Рисунок 3.4 – Основні компоненти програми

Користувач взаємодіє з системою через графічний інтерфейс. Дані (вхідне зображення), надіслані користувачем на сервер попередньо оброблюються ядром системи та надсилаються до модуля нейронної мережі, де відбувається основна робота по збільшенню зображення. Після цього, ядро системи отримує результат і виводить його користувачу. Користувач, в свою чергу, може завантажити вихідне зображення та/або збільшити інше.

Під час розробки системи програмного забезпечення було створено наступні класи ООП:

- Upscaler: реалізація основної логіки обробки зображення за допомогою нейронної мережі;
- UpscalerAPI: реалізація зв'язку між вхідним запитом від користувача і обробником даних, а також вивід даних користувачу.

3.4 Висновки за розділом 3

У даному розділі було описано вибір гіперпараметрів моделей. Крім того, було розглянуто та проаналізовано результати навчання моделей. Також було описано архітектуру розробленої системи, наведено діаграми і схеми роботи програми.

4 Експлуатація програмного забезпечення

У даному розділі розглядається призначення розробленої програми, а також умови її виконання, алгоритм виконання і повідомлення користувачу.

4.1 Призначення програми

Основним призначенням розробленої програми є збільшення термальних зображень.

4.2 Умови використання програми

До комп'ютера (сервера), на якому буде виконуватись програма, висуваються наступні вимоги:

- 8 ГБ оперативної пам'яті;
- відеокарта з 16 ГБ пам'яті;
- 1 ГБ вільного місця на диску;
- процесор з тактовою частотою 4 ГГц;
- операційна система Linux;
- встановлені бібліотеки PyTorch, BasicSR, Flask;
- встановлені драйвера CUDA;
- наявність файлів створених нейронних мереж на комп'ютері.

4.3 Виконання програми

Перед запуском програми необхідно упевнитись в тому, що комп'ютер (сервер) на якому буде запусчена програма, відповідає мінімальним вимогам з продуктивності.

Запуск програми відбувається через запуск файлу `run.py` у терміналі.

Доступ користувача до програми відбувається через веб-браузер, веб-адресою є ір-адреса сервера, порт 5000.

Після відкриття користувач потрапляє на головну сторінку за стосунку (рис. 4.1).

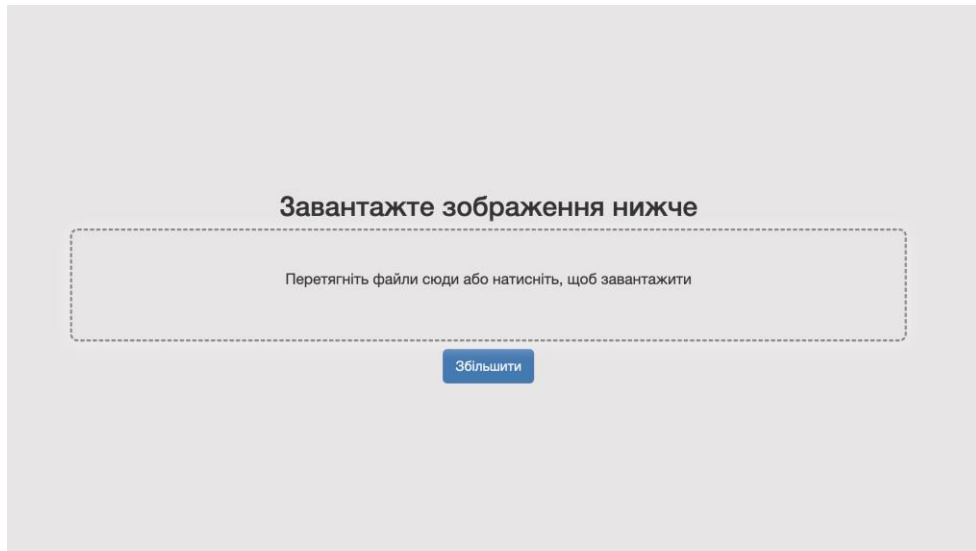


Рисунок 4.1 – Головна сторінка

Потім користувач може завантажити термальне зображення на збільшення (рис. 4.2).

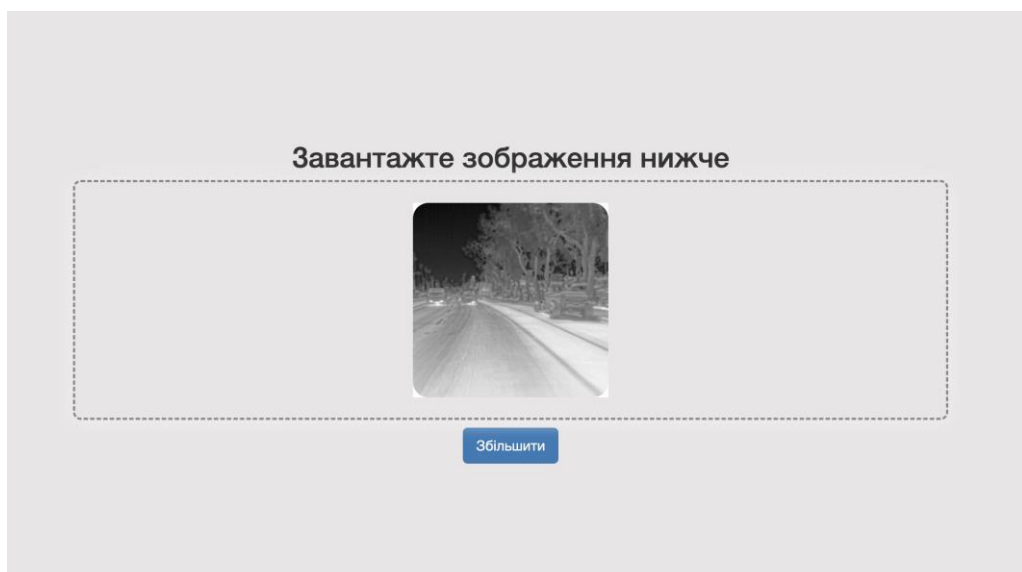


Рисунок 4.2 – Завантажене зображення (термальне зображення з датасету FLIR [48])

Після натискання кнопки «Збільшити» зображення надсилається на сервер, де відбувається обробка. По завершенню обробки відкривається сторінка з результатами збільшення зображення (рис. 4.3), де користувач може порівняти збільшене зображення за допомогою нейронної мережі з класичним алгоритмом. Також користувачу пропонується завантажити збільшене зображення та/або повернутися на головну сторінку.

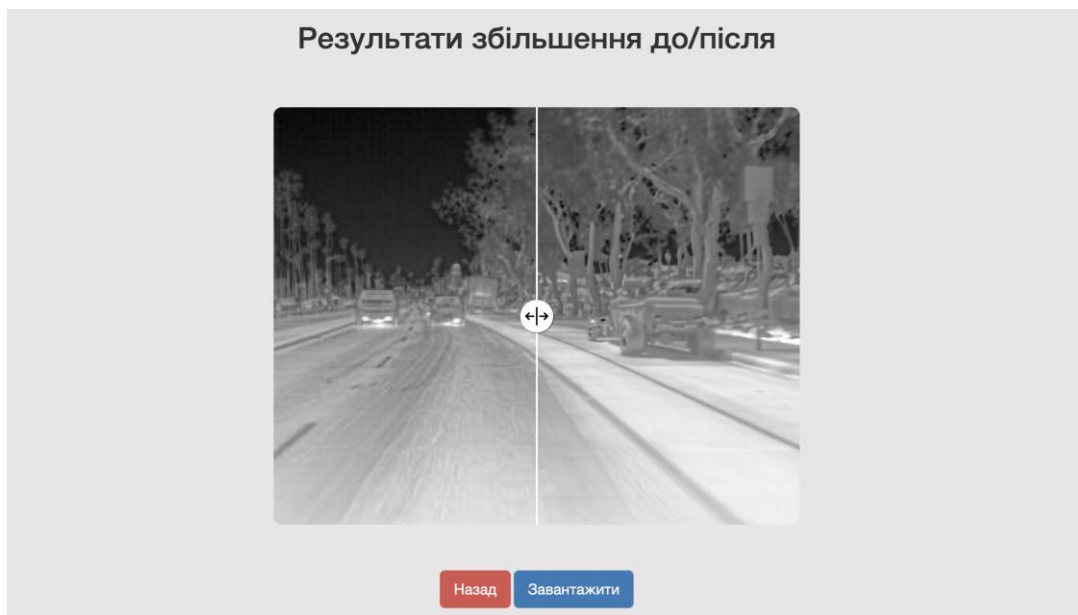


Рисунок 4.3 – Результати збільшення зображення (термальне зображення з датасету FLIR [48])

4.4 Повідомлення користувачу

Під час роботи з програмою можуть виникати помилки, повідомлення про які надсилаються користувачу, а саме:

- «Зображення завелике» – повідомлення про те, що зображення перевищило розмір в 1Мб;
- «Браузер не підтримується» – повідомлення, що браузер користувача не підтримується програмою;
- «Зображення не обрано» – повідомлення про те, що зображення не було завантажено на обробку.

4.5 Висновки за розділом 4

У даному розділі було розглянуто умови використання програми та її призначення, а також процес виконання програми і повідомлення, які можуть надсилатися користувачу під час виникнення помилок у програмі.

Висновки

Під час виконання роботи було проаналізовано предметну область та існуючі дослідження зі сфери збільшення роздільної здатності зображень з використанням нейронних мереж.

На основі проведеного аналізу було обрано та навчено моделі нейронних мереж для вирішення проблеми ЗРЗ термальних зображень, а також запропоновано власну модифікацію існуючої моделі. Крім того, було обрано датасет для тренування та тестування моделей та набір гіперпараметрів.

Базуючись на результатах роботи створених моделей було обрано найкращу модель, а також висунуто припущення щодо чинників, що впливали на якість навчання моделей.

Наукова новизна роботи полягає в адаптації обраних моделей штучних нейронних мереж під задачу ЗРЗ термальних зображень.

Крім того, можна виділити наступні кроки для покращення якості ЗРЗ термальних зображень:

- збільшення розміру датасету та пакету даних однієї ітерації навчання для підвищення узагальнювальної спроможності моделей;
- збір датасету термальних зображень з більшою якістю і меншим відсотком шуму у зображеннях для покращення результатів ЗРЗ;
- вибір інших гіперпараметрів навчання моделей, таких як алгоритм оптимізації, крок навчання і т.п.;
- удосконалення архітектури використаних моделей та їхня модифікація під задачу ЗРЗ термальних зображень.

Було розроблено програмне забезпечення, що дозволяє користувачу збільшувати термальне зображення у 2 рази за допомогою найкращої навченої моделі. Створене програмне забезпечення повністю відповідає вимогам, поставленим у технічному завданні.

Перелік джерел посилання

1. Infrared Thermography for Temperature Measurement and Non-Destructive Testing [Text] / [R. Usamentiaga, P. Venegas, J. Guerediaga et al.] // Sensors. – 2014. – 14(7). – P. 12305–12348.
2. Morozhenko, V. Infrared Radiation [Text] / V. Morozhenko. – Rijeka. : InTech, 2012. – 228 p.
3. Оптичне випромінювання [Електрон. ресурс]. – Режим доступу : http://gutpfusik.blogspot.com/2014/02/blog-post_19.html.
4. Теплобачення [Електрон. ресурс]. – Режим доступу : uk.wikipedia.org/wiki/Теплобачення.
5. Кононюк, А. Е. Зрительное восприятие изображений искусственным интеллектом / А. Е. Кононюк. – Т. 3 (ч. 4) : Основы фундаментальной теории искусственного интеллекта – Киев : Освіта України, 2017. – 480 с.
6. Goodfellow, I. Deep Learning [Text] / I. Goodfellow, Y. Bengio, A. Courville. – Cambridge, Massachusetts : MIT Press, 2016 – 800 p.
7. Gurney, K. An Introduction to Neural Networks [Text] / K. Gurney. – Florida. : CRC Press, 1997. – 248 p.
8. Gaidhani, P. Super-Resolution [Electronic recourse] / P. Gaidhani. – Access mode : https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/Super_Resolution_CVonline.pdf.
9. Інтерполяція [Електрон. ресурс]. – Режим доступу : <https://uk.wikipedia.org/wiki/Інтерполяція>.
10. I – Nearest Neighbour [Electronic resource]. – Access mode : https://www.giassa.net/?page_id=207.
11. Digital Image Interpolation [Electronic resource]. – Access mode : <https://www.cambridgeincolour.com/tutorials/image-interpolation.htm>.

12. Білінійна інтерполяція [Електрон. ресурс]. – Режим доступу : uk.wikipedia.org/wiki/Білінійна_інтерполяція.
13. Bicubic interpolation [Electronic resource]. – Access mode : en.wikipedia.org/wiki/Bicubic_interpolation.
14. Super-resolution imaging [Electronic resource]. – Access mode : https://en.wikipedia.org/wiki/Super-resolution_imaging.
15. Pixel shift [Electronic resource]. – Access mode : https://en.wikipedia.org/wiki/Pixel_shift.
16. High-resolution thermal images in building thermography thanks to Testo SuperResolution. [Electronic resource]. – Access mode : <https://static-int.testo.com/media/66/37/72ea088360f9/TA-TI-SuperResolution-Building-0981-8434.pdf>.
17. Geron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition [Text] / A. Geron. – Sebastopol : O'Reilly Media, Inc., 2019. – 856 p.
18. Згорткова_нейронна_мережа [Електрон. ресурс]. – Режим доступу : uk.wikipedia.org/wiki/Згорткова_нейронна_мережа.
19. Autoencoder: Downsampling and Upsampling [Electronic resource]. – Access mode : <https://kharshit.github.io/blog/2019/02/15/autoencoder-downsampling-and-upsampling>.
20. Shi, W. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network [Electronic resource] / [W. Shi, J. Caballero, F. Huszar et al.] – Access mode : <https://arxiv.org/pdf/1609.05158v2.pdf>.
21. Dong, C. Image Super-Resolution Using Deep Convolutional Networks [Electronic resource] / [C. Dong, C. C. Loy, K. He et al.] – Access mode : <https://arxiv.org/pdf/1501.00092v3.pdf>.
22. Dong, C. Accelerating the Super-Resolution Convolutional Neural Network [Electronic resource] / C. Dong, C. C. Loy, X. Tang. – Access mode : <https://arxiv.org/pdf/1608.00367.pdf>.

23. Mao, X. Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections [Electronic resource] / X. Mao, C. Shen, Y. Yang. – Access mode : <https://arxiv.org/pdf/1606.08921.pdf>.
24. Thermal Image Enhancement using Convolutional Neural Network [Text] / [Y. Choi, N. Kim, S. Hwang et al.] // Intelligent Robots and Systems (IROS) : 2016 IEEE/RSJ International Conference, Daejeon, 09-14 October 2016 : proceedings. – IEEE, 2016. – P. 223-230.
25. Brightness-Based Convolutional Neural Network for Thermal Image Enhancement [Text] / [K. Lee, J. Lee, J. Lee et al.] // IEEE Access. – 2017. - №5. – P. 26867-26879.
26. Rivadeneira, R. E. Thermal Image Super Resolution through Deep Convolutional Neural Network [Electronic resource] / [R. E. Rivadeneira, P. L. Suarez, A. D. Sappa et al.] – Access mode : https://www.researchgate.net/publication/335108778_Thermal_Image_SuperResolution_Through_Deep_Convolutional_Neural_Network.
27. Yamanaka, J. Fast and Accurate Image Super Resolution by Deep CNN with Skip Connection and Network in Network [Electronic resource] / J. Yamanaka, S. Kuwashima, T. Kurita. – Access mode : <https://arxiv.org/pdf/1707.05425.pdf>.
28. 25. Goodfellow, I. J. Generative Adversarial Networks [Electronic resource] / [I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al.] – Access mode : <http://www.arxiv.org/abs/1406.2661>.
29. Mirza, M. Conditional Generative Adversarial Nets [Electronic resource] / M. Mirza, S. Osindero. – Access mode : <https://arxiv.org/pdf/1411.1784.pdf>.
30. Ledig, C. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network [Electronic resource] / [C. Ledig, L. Theis, F. Huszár et al.] – Access mode : <https://arxiv.org/pdf/1609.04802.pdf>.

31. Wang, X. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks [Electronic resource] / [X. Wang, K. Yu, S. Wu et al.] – Access mode : <https://arxiv.org/pdf/1809.00219.pdf>.
32. Jolicœur-Martineau, A. The relativistic discriminator: a key element missing from standard GAN [Electronic resource] / A. Jolicœur-Martineau. – Access mode : <https://arxiv.org/pdf/1807.00734.pdf>.
33. Vaswani, A. Attention is all you need [Electronic resource] / [A. Vaswani, N. Shazeer, N. Parmar et al.] – Access mode : <https://arxiv.org/pdf/1706.03762.pdf>.
34. Bahdanau, D. Neural Machine Translation By Jointly Learning To Align And Translate [Electronic resource] / D. Bahdanau, K. Cho, Y. Bengio. – Access mode : <https://arxiv.org/pdf/1409.0473.pdf>.
35. Dosovitskiy, A. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale [Electronic resource] / [A. Dosovitskiy, L. Beyer, A. Kolesnikov et al.] – Access mode : <https://arxiv.org/pdf/2010.11929.pdf>.
36. Liu, Z. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows [Electronic resource] / [Z. Liu, Y. Lin, Y. Cao et al.] – Access mode : <https://arxiv.org/pdf/2103.14030.pdf>.
37. Liang, J. SwinIR: Image Restoration Using Swin Transformer [Electronic resource] / [J. Liang, J. Cao, G. Sun et al.] – Access mode : <https://arxiv.org/pdf/2108.10257.pdf>.
38. Structural similarity [Electronic resource]. – Access mode : en.wikipedia.org/wiki/Structural_similarity.
39. PSNR [Электрон. ресурс]. – Режим доступа : uk.wikipedia.org/wiki/PSNR.
40. Python [Электрон. ресурс]. – Режим доступа : <https://uk.wikipedia.org/wiki/Python>.
41. What is R? [Electronic resource]. – Access mode : <https://www.r-project.org/about.html>.

42. C++ [Электрон. ресурс]. – Режим доступа : uk.wikipedia.org/wiki/C++.
43. Java [Электрон. ресурс]. – Режим доступа : <https://uk.wikipedia.org/wiki/Java>.
44. PyTorch [Electronic resource]. – Access mode : <https://pytorch.org>.
45. Tensorflow [Electronic resource]. – Access mode : <https://www.tensorflow.org>.
46. Keras [Electronic resource]. – Access mode : <https://keras.io>.
47. BasicSR [Electronic resource]. – Access mode : <https://github.com/XPixelGroup/BasicSR>.
48. FREE Teledyne FLIR Thermal Dataset for Algorithm Training [Electronic resource]. – Access mode : <https://www.flir.eu/oem/adas/adas-dataset-form>.
49. Agustsson, E. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study [Text] / E. Agustsson, R. Timofte // Computer Vision and Pattern Recognition Workshops (CVPRW) : 2017 IEEE Conference, Honolulu, 21-26 July 2017 : proceedings. – IEEE, 2017. – P. 1110-1121.
50. Hendrycks, D. Gaussian Error Linear Units (GELUs) [Electronic resource] / D. Hendrycks, K. Gimpel. – Access mode : <https://arxiv.org/pdf/1606.08415v3.pdf>.
51. Ruder, S. An overview of gradient descent optimization algorithms [Electronic resource] / S. Ruder. – Access mode : <https://arxiv.org/pdf/1609.04747.pdf>.

Додаток А
Технічне завдання

A.1 Вступ

У даному розділі описано область застосування, призначення і вимоги до програми, що розроблюється.

A.1.1 Назва програми

Програма називається «Thermal Image Upscaler».

A 1.2 Призначення та область застосування

Програма призначена для збільшення роздільної здатності термальних зображень у 2 рази.

Програма є веб-сервісом, пропонує зручний інтерфейс і застосовується як обгортка для використання навчених моделей штучних нейронних мереж.

A.2 Підстави для розробки

Підставою для розробки є завдання на диплому роботу на тему «Дослідження та програмна реалізація методів збільшення якості термальних зображень з використанням штучних нейронних мереж».

Актуальність роботи полягає в потребі швидких, автоматизованих та якісних інструментів для збільшення роздільної здатності термальних зображень.

A.3 Призначення розробки

Програма надає можливості збільшення роздільної здатності термальних зображень шляхом використання штучних нейронних мереж.

A.4 Вимоги до програми

В даному розділі описані вимоги до програми.

A.4.1 Вимоги до функціональних характеристик

Програмний продукт повинен забезпечувати виконання наступних функцій:

- користувач повинен мати можливість збільшувати термальні зображення в 2 рази;
- користувач повинен мати можливість завантажувати збільшене термальне зображення;
- програма повинна виконуватись на сервері;
- користувач повинен мати доступ до сервісу через інтернет в будь-який момент часу.

A.4.2 Вимоги до надійності

Програмний продукт призначений для виконання на сервері і має наступні вимоги до надійності:

- забезпечення безперебійної роботи програми;
- захист від помилкових дій користувача.

A.4.3 Вимоги до технічних та програмних засобів

До комп'ютера (сервера), на якому буде виконуватись програма, висуваються наступні вимоги:

- 8 ГБ оперативної пам'яті;
- відеокарта з 16 ГБ пам'яті;

- 1 ГБ вільного місця на диску;
- процесор з тактовою частотою 4 ГГц;
- операційна система Linux;
- встановлені бібліотеки PyTorch, BasicSR, Flask;
- встановлені драйвера CUDA;
- наявність файлів створених нейронних мереж на комп'ютері.

A.5 Вимоги до програмної документації

Склад програмної документації до програми:

- технічне завдання;
- опис розробки програми;
- опис експлуатації програми.

A.6 Стадії і етапи розробки

Розробка програмного продукту має бути проведена в наступні стадії:

- створення та узгодження технічного завдання;
- аналіз методів збільшення роздільної здатності термальних і звичайних зображень;
- пошук і обробка датасету для навчання моделей;
- створення архітектури і навчання моделей штучних нейронних мереж для збільшення роздільної здатності термальних зображень;
- аналіз результатів роботи навчених моделей штучних нейронних мереж;
- створення програмного застосунку для взаємодії користувача зі створеними моделями;
- створення супровідної документації.

A.7 Порядок контролю та приймання

Програмного продукту та документації необхідно пройти наступний порядок контролю та приймання:

- аналіз роботи створених та навчених моделей штучних нейронних мереж;
- тестування програми;
- затвердження працездатності програмного продукту та супровідної документації науковим керівником;
- нормоконтроль документації до програмного продукту.

Додаток Б
Текст програми

Б.1 Код ядра системы

Код файлу run.py

```
import os
from flask import request, render_template, send_file, session, redirect,
flash
from image_upscaler.app import create_app

app = create_app()

@app.route('/', methods=['POST', 'GET'])
def upload():
    error = None
    session['filename'] = None
    session['root_path'] = None
    if request.method == 'POST':
        f = request.files.get('file')
        save_path = os.path.join(app.config['UPLOADED_PATH'], f.filename)
        session['filename'] = f.filename
        session['root_path'] = app.root_path
        f.save(save_path)
    return render_template('index.html', error=error)

@app.route('/check_image', methods=['POST', 'GET'])
def check_image():
    if session['filename'] is None or session['root_path'] is None:
        error = "Зображення не обрано!"
        return render_template('index.html', error=error)
    return redirect('results')

@app.route('/uploads/<path:filename>', methods=['POST', 'GET'])
def download(filename):
    abs_path = os.path.join(session['root_path'], 'static/upscaled',
filename.split('/')[1])
    return send_file(abs_path, as_attachment=True)

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=5000, debug=True, threaded=True)
```

Код файлу app.py

```
from flask import Flask
import os
from flask_dropzone import Dropzone
from image_upscaler.models.upscaler import Upscaler
from image_upscaler.views.upscaler_view import UpscalerAPI
from image_upscaler.utils.utils import load_config

def create_app():
    opt = load_config('image_upscaler/configs/model_config.yaml')
```

```

basedir = os.path.abspath(os.path.dirname(__file__))

app = Flask(__name__)
app.secret_key = ''

app.config.update(
    UPLOADED_PATH=os.path.join(basedir, 'static/uploads'),
    DROPZONE_ALLOWED_FILE_TYPE='image',
    DROPZONE_MAX_FILE_SIZE=1,
    DROPZONE_MAX_FILES=1,
    DROPZONE_DEFAULT_MESSAGE='Перетягніть файли сюди або натисніть, щоб
завантажити'
)

upscaler = Upscaler(opt)
app.add_url_rule("/results", view_func=UpscalerAPI.as_view("upscaler",
upscaler))
dropzone = Dropzone(app)
return app

```

Код файлу utils.py

```

import yaml
from basicsr.utils.options import ordered_yaml

def load_config(config_path: str):
    with open(config_path, mode='r') as f:
        opt = yaml.load(f, Loader=ordered_yaml())[0]

    opt['is_train'] = False
    opt['dist'] = False
    return opt

```

Код файлу upscaler_view.py

```

from flask.views import MethodView
from flask import render_template, session
import os
import cv2

class UpscalerAPI(MethodView):
    def __init__(self, upscaler):
        self.upscaler = upscaler

    def get(self):
        filename = session['filename']
        root_path = session['root_path']
        static_path = 'static'

        sr_image_name = "nn_" + filename
        sr_inter_image_name = "inter_" + filename

        lr_image_path = os.path.join(root_path, static_path, 'uploads',
filename)
        sr_image_path = os.path.join(root_path, static_path, 'upscaled',
sr_image_name)

```

```

        sr_inter_image_path = os.path.join(root_path, static_path,
        'upscaled', sr_inter_image_name)

        sr_image_nn = self.upscaler.upscale_image_nn(lr_image_path)
        sr_image_interpolation = self.upscaler.upscale_image_interpolation(lr_image_path)

        cv2.imwrite(sr_image_path, sr_image_nn)
        cv2.imwrite(sr_inter_image_path, sr_image_interpolation)

        return render_template('results.html',
                               sr_image_nn=os.path.join(static_path,
        'upscaled', sr_image_name),

sr_image_interpolation=os.path.join(static_path, 'upscaled',
sr_inter_image_name))

```

Код файла upscaler.py

```

import numpy as np
import cv2
import torch
from basicsr import build_model
from basicsr.utils import tensor2img

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

def load_image(filename: str) -> np.ndarray:
    image = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    image = np.expand_dims(image, -1)
    return image

def img2tensor(image: np.ndarray) -> torch.Tensor:
    image = torch.from_numpy(image).permute(2, 0, 1)
    image = image.unsqueeze(0)
    return image

class Upscaler:
    def __init__(self, opt):
        self.model = build_model(opt)

    def upscale_image_nn(self, filename: str) -> np.ndarray:
        lr_image = load_image(filename).astype(np.float32) / 255.
        lr_image_tensor = img2tensor(lr_image).to(device)

        self.model.lq = lr_image_tensor
        self.model.test()
        sr_image_nn = tensor2img(self.model.output)
        return sr_image_nn

    def upscale_image_interpolation(self, filename: str) -> np.ndarray:
        lr_image = load_image(filename)
        sr_image_nearest = cv2.resize(lr_image,
                                     (lr_image.shape[1] * 2,
        lr_image.shape[0] * 2),
                                     interpolation=cv2.INTER_NEAREST)
        sr_image_nearest = np.expand_dims(sr_image_nearest, -1)

```

```
return sr_image_nearest
```

Б.2 Код графічного інтерфейсу

Код файлу index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">

  <link                                rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/dropzone/5.7.1/min/dropzone.min.
css">
  <link                                rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <link                                rel="stylesheet"            href="{{ url_for('static',
filename='styles/main.css') }}" type="text/css"/>
  <link rel="icon" href="{{ url_for('static', filename='icon.png') }}" />
  <title>Upscaler</title>
</head>

<body>
<main>
  <div class="container">
    <div class="text-center"><h1>Завантажте зображення нижче</h1></div>
    {{ dropzone.create(action='upload') }}
    <div class="row">
      <div class="col text-center">
        <a class="btn btn-primary btn-lg" href="{{
url_for('check_image') }}">Збільшити</a>
      </div>
    </div>
  </div>
  {{ dropzone.load_js() }}
  {{ dropzone.config(custom_init='this.on("maxfilesexceeded", function (file) {
                                this.removeAllFiles();
                                this.addFile(file);
                                });',
                                custom_options='thumbnailWidth: 250, thumbnailHeight: 250') }}
</main>
</body>
</html>
```

Додаток В

Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дослідження та програмна реалізація методів збільшення якості термальних зображень з використанням штучних нейронних мереж

Виконав:
ст. гр. КНТ-111м

Ан. Є. Діденко

Керівник:
професор, д.т.н.

А. О. Олійник

2022

Рисунок В.1 – Слайд №1

Мета, об'єкт, предмет дослідження

Мета роботи – програмна реалізація методів збільшення роздільної здатності термальних зображень з використанням штучних нейронних мереж.

Об'єкт дослідження – процес збільшення роздільної здатності термальних зображень.

Предмет дослідження – методи збільшення роздільної здатності термальних зображень з використанням штучних нейронних мереж.

2

Рисунок В.2 – Слайд №2

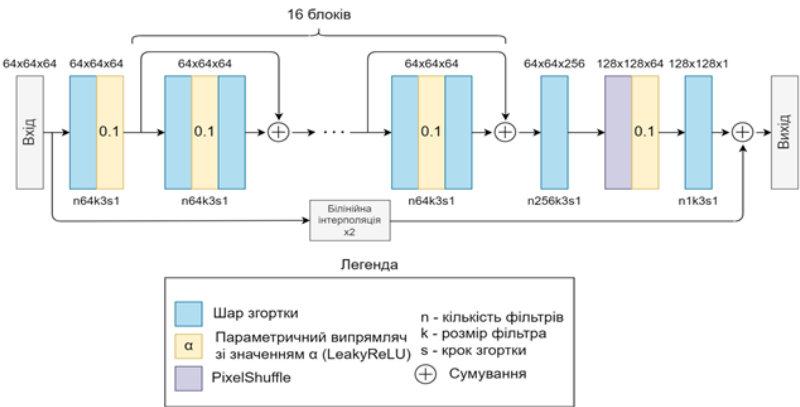
Порівняння існуючих методів

Критерій/ метод	Класичні алгоритми	ЗНМ	ГЗМ	Трансформери
Швидкість роботи	Висока	Помірна	Помірна	Помірна
Складність реалізації	Низька	Помірна	Помірна	Висока
Узагальнення	Низьке	Помірне	Помірне	Високе
Складність навчання	Не навчаються	Помірна	Висока	Висока
Якість роботи	Низька	Помірна	Висока	Висока

3

Рисунок В.3 – Слайд №3

Архітектури штучних нейронних мереж

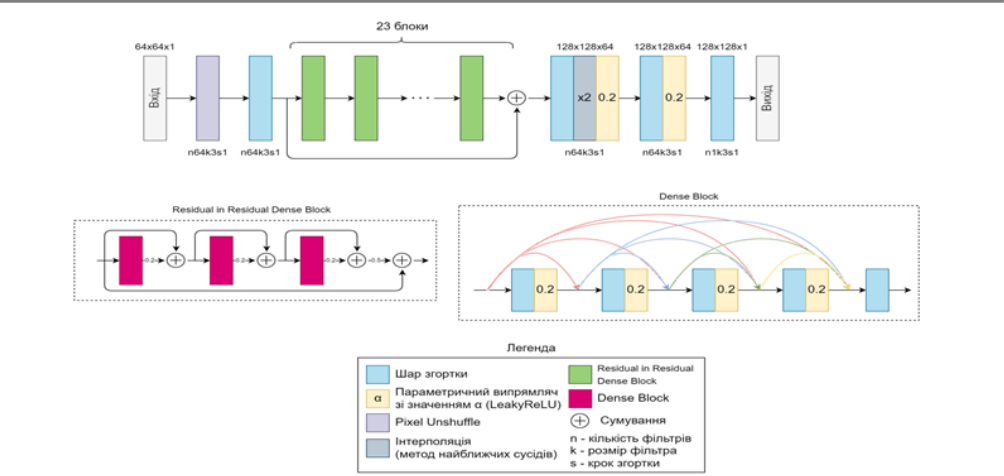


Архітектура SRResNet (генератор моделі SRGAN)

4

Рисунок В.4 – Слайд №4

Архітектури штучних нейронних мереж

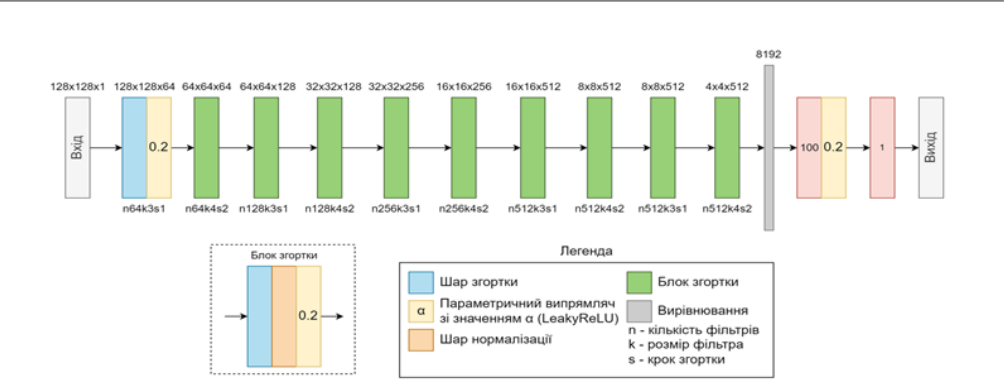


Архітектура RRDBNet (генератор моделі ESRGAN)

5

Рисунок В.5 – Слайд №5

Архітектури штучних нейронних мереж

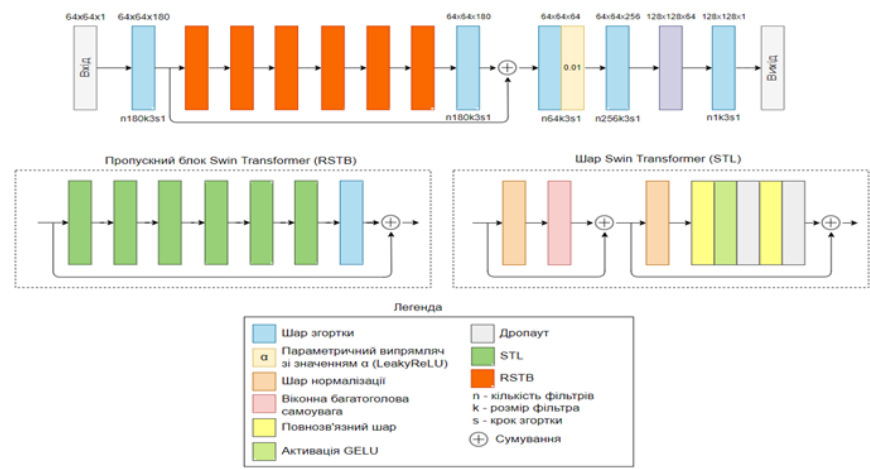


Архітектура дискримінатора ГЗМ

6

Рисунок В.6 – Слайд №6

Архітектури штучних нейронних мереж



Архітектура SwinIR

7

Рисунок В.7 – Слайд №7

Вибір мови програмування

Критерій	Python	R	C++	Java
Типізація	Динамічна	Динамічна	Статична	Статична
Швидкість	++	++	++++	+++
Простота	+++	+	+	++
Кількість бібліотек	+++	+++	++	++
Підтримка машинного навчання	+++	+++	++	++

Обрана мова програмування - Python

8

Рисунок В.8 – Слайд №8

Вибір фреймворку глибинного навчання

Критерій	PyTorch	Tensorflow	Keras
Простота і зрозумілість	+++	++	+++
Гнучкість	+++	+++	++
Спільнота	+++	++	+++
Наукова популярність	+++	++	++
Оновлюваність	+++	++	++

Обраний фреймворк глибинного навчання - PyTorch

9

Рисунок В.9 – Слайд №9

Вибір гіперпараметрів навчання

Параметр	Значення
Датасет	FLIR, чорно-білий DIV2K
Розмір навчальної вибірки	8842 (FLIR), 800 (DIV2K)
Розмір тестової вибірки	1366 (FLIR)
Розмір зображень	640x480 (FLIR), ~1500x1500 (DIV2K)
Розмір пакету даних	16 (SwinIR, ESRGAN і RRDBNet) 64 (SRGAN і SRResNet)
Кількість ітерацій	100000
Метод оптимізації	Adam
Крок навчання	0,0002
Функція втрат	MAE, Perceptual loss

10

Рисунок В.10 – Слайд №10

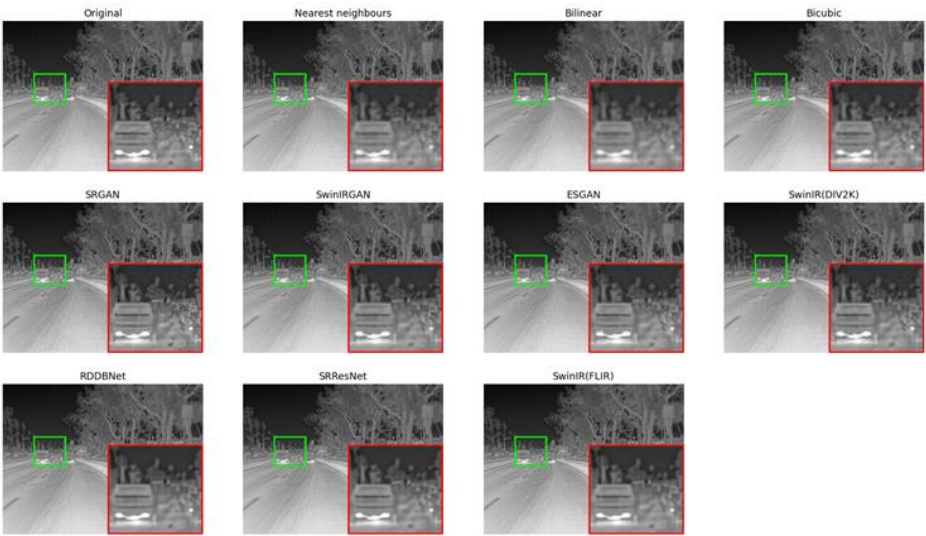
Результати збільшення роздільної здатності

Модель	SSIM	PSNR
Метод найближчих сусідів	0,7389	30,3350
Білінійна інтерполяція	0,7557	31,1910
Бікубічна інтерполяція	0,7672	31,5820
SRGAN	0,6617	28,3197
SwinIRGAN	0,6751	29,6528
ESRGAN	0,6853	29,8002
SwinIR (DIV2K)	0,7642	31,4355
RRDBNet	0,7829	32,3622
SRResNet	0,7828	32,3471
SwinIR (FLIR)	0,7833	32,3688

11

Рисунок В.11 – Слайд №11

Результати збільшення роздільної здатності

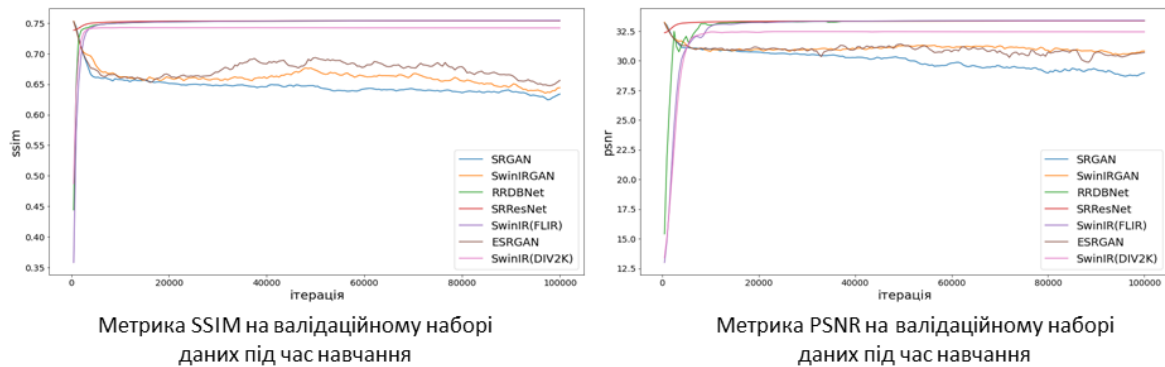


Термальне зображення з датасету FLIR

12

Рисунок В.12 – Слайд №12

Результати збільшення роздільної здатності



13

Рисунок В.13 – Слайд №13

Висновки

- Проаналізовано предметну область та існуючі дослідження зі сфери збільшення роздільної здатності зображень з використанням нейронних мереж
- Обрано та навчено моделі нейронних мереж для вирішення проблеми ЗРЗ термальних зображень, а також запропоновано власну модифікацію існуючої моделі
- Розроблено програмне забезпечення, що дозволяє користувачу збільшувати термальне зображення у 2 рази за допомогою найкращої навченої моделі

14

Рисунок В.14 – Слайд №14

Подальші вдосконалення

- Збільшення розміру датасету та пакету даних однієї ітерації навчання для підвищення узагальнювальної спроможності моделей
- Збір датасету термальних зображень з більшою якістю і меншим відсотком шуму у зображеннях для покращення результатів ЗРЗ
- Вибір інших гіперпараметрів навчання моделей, таких як алгоритм оптимізації, крок навчання і т.п.
- Удосконалення архітектури використаних моделей та їхня модифікація під задачу ЗРЗ термальних зображень

15

Рисунок В.15 – Слайд №15