

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр
(ступінь вищої освіти)

на тему ПРОГРАМНІ ЗАСОБИ ЗБІЛЬШЕННЯ РОЗДІЛЬНОЇ
ЗДАТНОСТІ ТЕПЛОВИХ ЗОБРАЖЕНЬ НА ОСНОВІ
ЗОБРАЖЕННЯ БІЛЬШОЇ РОЗДІЛЬНОЇ ЗДАТНОСТІ
У ВИДИМОМУ СПЕКТРІ
SOFTWARE TOOLS TO INCREASE RESOLUTION QUALITY
OF THERMAL IMAGES BASED ON HIGHER RESOLUTION
IMAGES IN THE VISIBLE SPECTRUM

Виконав: студент(ка) 4 курсу, групи КНТ-139
Спеціальності 121 Інженерія програмного
забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ЧІЧУК К.О.
(прізвище та ініціали)

Керівник СУББОТІН С.О.
(прізвище та ініціали)

Рецензент КОРОТКИЙ О.В.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

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.

Сергій СУББОТІН

«_____» _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЧІЧУКА Костянтина Олексійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмні засоби збільшення роздільної здатності теплових зображень на основі зображення більшої роздільної здатності у видимому спектрі. Software Tools to Increase Resolution Quality of Thermal Images Based on Higher Resolution Images in the Visible Spectrum

керівник проєкту (роботи) СУББОТІН Сергій Олександрович, д.т.н., професор,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “04” квітня 2023 року №87

2. Строк подання студентом проєкту (роботи) 07 червня 2023 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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	СУББОТІН С.О., професор		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання «04» квітня 2023 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.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Костянтин ЧІЧУК
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Сергій СУББОТІН
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
114 с., 20 рис., 6 табл., 2 дод., 73 джерел.

СУПЕРМАСШТАБУВАННЯ, МАШИННЕ НАВЧАННЯ,
ІНФРАЧЕРВОНІ ЗОБРАЖЕННЯ, ПЕРЕНЕСЕННЯ ОЗНАК, НЕЙРОННІ
МЕРЕЖІ, PYTHON, PYTORCH

Об'єкт дослідження – програмні засоби для супер масштабування інфрачервоних зображень з допомогою електрооптичного зразка.

Предмет дослідження – методи та технології розробки нейронних мереж для супер масштабування інфрачервоних зображень.

Мета роботи – дослідити, проаналізувати та реалізувати методи супер масштабування для побудови модулярних нейронних мереж у якості основи для моделей супер масштабування.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, технології супер масштабування, мова програмування Python, фреймворк PyTorch, середовище розробки Google Colab.

Результати. Створено програмне забезпечення, що реалізує метод супер масштабування за зразком для побудови модулярної нейронної мережі для супер масштабування.

Висновки. Розроблено програмний продукт для побудови та навчання модулярної нейронної мережі для супер масштабування за зразком.

Галузь використання – системи супер масштабування зображень для технічного, побутового, медичного та військового застосування.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 114 pages, 20 figures, 6 tables, 2 appendixes, 73 sources.

SUPER RESOLUTION, MACHINE LEARNING, INFRARED IMAGES, FEATURE TRANSFER, NEURAL NETWORKS, PYTHON, PYTORCH

Object of study is a software tools for super resolution of infrared images using an electro-optical sample.

Subject of study is the methods and technologies for developing neural networks for super resolution of infrared images.

The purpose of the study is to investigate, analyse and implement super resolution methods for building modular neural networks as a basis for super resolution models.

Materials, methods and tools: object-oriented programming, super resolution technologies, Python programming language, PyTorch framework, Google Colab development environment.

Results. The software that implements the super scaling method for building a modular neural network for super scaling was created.

Conclusion. A software product has been developed for building and training a modular neural network for super resolution by reference.

The field of use is super scaling systems for technical, household, medical and military applications.

ЗМІСТ

С.

Перелік скорочень та умовних позначок	10
Вступ.....	11
1 Аналіз проблемної області та постановка завдань дослідження	13
1.1 Супер масштабування.....	13
1.1.1 Визначення проблемної області супер розширення. Сучасний стан області.....	13
1.1.2 Існуючі методи супер розширення.....	13
1.1.2.1 Традиційні методи.....	13
1.1.2.2 Методи глибокого навчання	14
1.2 Особливості інфрачервоних зображень.....	15
1.2.1 Відмінності функції деградації.....	15
1.2.2 Апаратні проблеми інфрачервоної зйомки.....	16
1.2.2.1 Труднощі, пов'язані з лінзами.....	17
1.2.2.2 Труднощі, пов'язані з сенсорами	17
1.2.2.3 Труднощі, пов'язані з процесорами.....	18
1.3 Сфери використання	19
1.3.1 Медичні та біохімічно інженерні завдання	20
1.3.1.1 Виявлення амілоїдних згустків.....	20
1.3.1.2 Виявлення вірусів.....	21
1.3.1.3 Вивчення процесів в клітинах.....	21
1.3.1.4 Аналіз структури білків.....	21
1.3.2 Задачі бачення	22
1.3.2.1 Мультиспектральні узгодження.....	22
1.3.2.2 Перетворення електрооптичних зображень на інфрачервоні зображення.....	23
1.3.2.3 Термальне виявлення.....	23
1.3.2.4 Розпізнавання облич	24
1.3.3 Інженерні задачі	25

1.3.3.1 Обчислення глибини	25
1.3.3.2 Дистанційне зондування.....	26
1.3.3.3 Контроль якості харчових продуктів	26
1.3.3.4 Управління сільським господарством.....	27
1.3.3.5 Зореутворення.....	28
1.3.4 Методи масштабування	29
1.3.4.1 Масштабування на основі інтерполяції	29
1.3.4.1.1 Інтерполяція найближчого сусіда.	29
1.3.4.1.2 Метод білінійної інтерполяції.....	30
1.3.4.1.3 Бікубічна інтерполяція.....	31
1.3.4.1.4 Масштабування на основі навчання.....	31
1.3.4.1.5 Транспонований шар згортки	33
1.3.4.1.6 Субпіксельна згортка.....	33
1.3.4.1.7 Масштабування з мета-навчанням	35
1.4 Постановка завдання.....	39
1.5 Висновки за розділом 1.....	41
2 Вибір і обґрунтування інструментів і архітектури	43
2.1 Опис основних вимог до програми	43
2.2 Вибір мови програмування	44
2.2.1 Python.....	44
2.2.2 C++.....	45
2.2.3 MATLAB.....	47
2.2.4 Java.....	48
2.2.5 R	48
2.3 Вибір фреймворку	51
2.3.1 TensorFlow.....	51
2.3.2 Pytorch.....	52
2.3.3 Keras.....	53
2.3.4 MXNet.....	54
2.4 Вибір середі розробки.....	57

2.4.1 Jupiter Notebook	57
2.4.2 PyCharm.....	57
2.4.3 VS Code	58
2.4.4 Google Colab.....	59
2.5 Підсумки за розділом 2.....	61
3 Розробка програмного забезпечення.....	61
3.1 Основні рішення щодо реалізації	61
3.2 Висновки за розділом 3.....	64
4 Експлуатація, тестування та експериментальне дослідження програми	65
4.1 Вхідні дані для тестування	65
4.2 Метрики оцінювання	66
4.1.1 PSNR.....	66
4.1.2 SSIM.....	68
4.3 Результати	69
4.3.1 Результати переносу стилю.....	67
4.3.2 Результати супермасштабування.....	68
4.3.3 Результати тестування	68
Висновки	70
Список літератури	Error! Bookmark not defined.
Додаток А Слайди презентації.....	80
Додаток Б Код програми	Error! Bookmark not defined.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

CNN - convolutional neural network;

GAN - generative adversarial network;

HR - high resolution;

IDN - interactive deep neural network;

LR - low resolution;

MSE - Mean Squared Error

PSNR - Peak Signal-to-Noise Ratio

RNN - recurrent neural network;

SR – super resolution;

SRCNN - super resolution convolutional neural network;

SRGAN - super resolution generative adversarial network;

ООП – Об’єктно Орієнтоване Програмування;

ВСТУП

Сьогодні, зі зростаючою потребою у якісних візуальних зображеннях та розвитком комп'ютерної техніки, супер масштабування стало надзвичайно важливим процесом. Воно дозволяє збільшити розмір зображення з мінімальною втратою якості та деталей. Для досягнення цього, застосовуються інструменти, такі як глибоке навчання та нейронні мережі, які обробляють велику кількість даних, використовуючи їх для створення більш детальних та реалістичних зображень.

Задача супер масштабування інфрачервоних зображень полягає в підвищенні роздільної здатності зображення, збільшенні його розміру та покращенні якості. Це може бути досягнуто за допомогою різних методів, включаючи традиційні методи, машинне навчання, глибоке навчання та інші методи. Задача полягає в тому, щоб знайти оптимальний алгоритм для підвищення роздільної здатності зображення, що забезпечує оптимальну якість зображення та зберігає важливі деталі.

Супер масштабування інфрачервоних зображень стає все більш важливим інструментом для багатьох наукових і прикладних досліджень. Це особливо важливо в області медицини, де інфрачервона термографія використовується для діагностики захворювань та контролю за станом хворих. Також супер масштабування інфрачервоних зображень корисне для науки про матеріали, де воно дозволяє вивчати властивості і дефекти кристалів, композитів та інших матеріалів. У космічних дослідженнях супер масштабування інфрачервоних зображень використовується для покращення результатів нових, та особливо старих телескопів інфрачервоного спектру, що в свою чергу дає більше інформації про процеси формування, розвитку планет, зірок та всесвіту в цілому. В розвідці та безпеці, воно може бути корисним для виявлення людей та тварин за допомогою їх теплового випромінювання.

У багатьох сферах використовуються камери відеоспостереження з інфрачервоним зображенням. Супер масштабування інфрачервоних зображень

допоможе аналізувати ці зображення з більшою точністю та здатністю розпізнавати деталі.

Однією з основних проблем, що стоїть перед супер масштабуванням інфрачервоних зображень, є забезпечення високої точності ідентифікації об'єктів на зображенні. Це стає особливо складним завданням через те, що інфрачервоне випромінювання має низьку роздільну здатність порівняно з візуальним спектром. Тому для досягнення високої точності ідентифікації необхідно використовувати спеціалізовані алгоритми обробки зображень, які здатні ефективно вирішувати проблему розмивання деталей об'єктів на інфрачервоних зображеннях.

Крім того, для забезпечення високої точності ідентифікації на інфрачервоних зображеннях необхідно мати достатньо велику базу даних для тренування та тестування алгоритмів. Тому важливо забезпечити якість та репрезентативність цієї бази даних, щоб забезпечити високу точність та надійність системи ідентифікації об'єктів на інфрачервоних зображеннях.

Перенесення текстури - це метод, що дозволяє перенести текстуру одного зображення на інше зображення, зберігаючи основні характеристики першого зображення. Цей метод може бути застосований для покращення якості зображень, отриманих з інфрачервоних джерел.

Застосування переносу текстури може допомогти вирішити проблему недостатньої якості зображень, отриманих з інфрачервоних джерел. Завдяки цьому методу можна отримати більш якісні зображення, зберігаючи при цьому важливі особливості вихідного зображення. Також цей метод може бути корисним для зменшення розміру даних, отриманих з інфрачервоних джерел, що сприятиме поліпшенню ефективності і швидкості обробки даних.

Мета роботи – дослідження та програмна реалізація методів та засобів супер масштабування інфрачервоних зображень за допомогою інформації з зображення електрооптичного спектру.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

У першому розділі буде проведений детальний аналіз проблемної області, пов'язаної з суперрозширенням та особливостями інфрачервоних зображень, а також з використанням машинного навчання для розв'язання сучасних проблем. В процесі роботи будуть досліджені основні терміни, такі як супермасштабування, інтерполяція, машинне навчання та перенесення текстур.

1.1 Супер масштабування

1.1.1 Визначення проблемної області супер розширення. Сучасний стан області

Підвищення роздільної здатності зображень (SR) - це процес відновлення зображень високої роздільної здатності (HR) з зображень низької роздільною здатності (LR), який є важливим класом технік обробки зображень в області комп'ютерного зору та обробки зображень.

SR технології мають широкий спектр застосувань у реальному світі у різних галузях: медична візуалізація [1], фармацевтична промисловість [2], спостереження за клітинами [3], мультиспектральні зіставлення [4], виявленні цілей [5], розпізнавання облич [6].

Застосування SR технологій також допомагає у інженерних завданнях, SR використовується для автоматизованих транспортних засобів [7], контролю якості продуктів харчування [8], дистанційного зондування [9], управління сільськогосподарською діяльністю [10], моделей рельєфу [11], управління водними ресурсами [12], вивчення поверхні землі [13], вивчення формування зірок [14].

1.1.2 Існуючі методи супер розширення

Технологія відновлення зображень високої роздільності (SR) передбачає отримання HR зображень шляхом обробки сигналів зі зниженою роздільною здатністю, з одного або декількох кадрів. Існуючі методи відновлення зображень високої роздільності можна розділити на дві категорії: традиційні методи та методи, засновані на глибинному навчанні.

1.1.2.1 Традиційні методи

Традиційні методи включають методи на основі інтерполяції або реконструкції. Методи супер-розширення зображень на основі інтерполяції мають переваги простого процесу та швидкої швидкості реконструкції, але точність реконструкції низька. Методи супер-розширення на основі реконструкції наближають низько розмірне зображення до високо розмірного зображення за допомогою неперервної ітерації, та реконструйоване зображення має високу точність.

Ця проблема загалом є дуже складною і неоднозначною, оскільки існує безліч високо-роздільних зображень, які відповідають одному низько-роздільному зображенню. У науковій літературі було запропоновано різноманітні методи, які розрізняються за підходом та математичними моделями, що використовуються. Наприклад, існують методи, які базуються на передбаченні пікселів високо-роздільних зображень з низько-роздільних зображень [15]. Також є методи, які використовують ребра зображення, щоб підвищити роздільну здатність [16]. Деякі методи базуються на статистичних моделях [17], в той час як інші використовують патчі зображень для відновлення високо-роздільних зображень [18]. Крім того, розріджена репрезентація також використовується для зменшення складності задачі відновлення високо-роздільних зображень [19].

1.1.2.2 Методи глибокого навчання

Зі стрімким розвитком технік глибокого навчання в останні роки, моделі глибокого навчання для SR активно досліджуються та часто досягають найкращої продуктивності на різних бенчмарках SR.

Однією з ключових відмінностей між методами глибокого навчання є тип архітектури мережі. Наприклад, деякі моделі використовують тільки згорткову нейронну мережу (Convolutional Neural Network - CNN), тоді як інші використовують більш складні архітектури, такі як залишкові нейронні мережі (Residual Neural Network - RNN) або щільні нейронні мережі (Dense Neural Network - DNN).

Також, різні методи можуть використовувати різні принципи та стратегії навчання, такі як навчання з вчителем (supervised learning), навчання без учителя (unsupervised learning), навчання з підкріпленням (reinforcement learning) тощо.

Для вирішення завдань SR було застосовано різноманітні методи глибокого навчання, починаючи з ранніх методів на основі згорткових нейронних мереж (наприклад, SRCNN [20]) та закінчуючи останніми перспективними підходами SR, що використовують Генеративні змагальні мережі (GAN) [21] (наприклад, SRGAN [21]). Загалом, родина алгоритмів SR, що використовують техніки глибокого навчання, відрізняється між собою за наступними основними аспектами: різними типами архітектур мереж [23]-[24], різними типами функцій втрат [25]-[26], різними типами принципів та стратегій навчання [25]-[27] тощо.

SRCNN навчала тришарову згорткову нейронну мережу для вивчення відображувального відношення між LR та HR зображеннями. Пізніші роботи також успішно використовували глибші мережі. Модель мережі навчається за допомогою середньоквадратичної помилки (MSE) між SR та HR зображеннями, але результат має тенденцію до розмиття та відсутності високочастотного текстури, що не дає хорошого візуального ефекту. Тому концепція сприйняття втрати [28] поступово вводиться в поліпшений метод, де втрати передаються з

простору пікселів в простір високорівневих ознак. Ледіг [22] та інші ввели генеративна змагальні мережі (Gans), в яких використовуються дві мережі: генератор, що створює високоякісні зображення, та дискримінатор, який намагається відрізнити згенеровані зображення від оригіналів. Така підхід дозволяє отримати більш візуально задовольняючі результати, включаючи більш детальну текстуру та більш точні контури. Саджаді [25] інтегрував ідеї синтезу текстури та перенесення стилю [29] і запропонував втрату відповідності текстури, щоб покращити інформацію про текстуру у зображенні високої роздільної здатності.

Застосування глибинного навчання до задачі SR має деякі переваги, оскільки зазвичай забезпечується краща точність та більш реалістичні результати порівняно з традиційними методами. Крім того, з використанням глибинного навчання можна автоматизувати процес вибору відповідного фільтра для кожного вхідного зображення, що є складною задачею для традиційних методів. Оскільки глибинні моделі можуть навчатись на великих обсягах даних, вони зазвичай виявляються більш ефективними у вирішенні задачі SR на великих масштабах.

1.2 Особливості інфрачервоних зображень

Технологія інфрачервоного зображення має високу стійкість до перешкод, що дозволяє використовувати її для здійснення спостережень у будь-яку погоду. Це широко використовується в таких галузях, як дистанційне виявлення, військова розвідка, охорона та інших галузях.

1.2.1 Відмінності функції деградації

Проте варто зазначити, що процес деградації в інфрачервоних зображеннях є складнішим. Головні причини полягають у наступному: по-перше, ядро розмиття, що використовується при відновленні зображень

видимого спектру, є відносно простим. Зазвичай, це гаусівський шум та стиснення JPEG [29]. Однак ці ядра розмиття недостатньо стійкі для інфрачервоних зображень, що відображають складні умови реального світу. Це пов'язано з обмеженнями оптики та електроніки, що використовуються в інфрачервоних системах зображення. Іншими словами, шум, який виникає, є різним. Крім того, багато камер ІЧ знаходяться в дикому середовищі, що ще більше ускладнює процес деградації зображення. З іншого боку, зображення ІЧ мають простий колір, незначні градієнти та перекриття інформації між високими та низькими частотами [29]. Ці особливості, відмінні від зображень видимого спектру, можна вважати унікальними паттернами в інфрачервоних зображеннях. Отже, необхідно запропонувати підходи до відновлення суперроздільності, які відповідають властивостям інфрачервоних зображень.

1.2.2 Апаратні проблеми інфрачервоної зйомки

Найбільш прямий спосіб покращити просторову роздільну здатність інфрачервоного зображення - збільшити розмір і продуктивність інфрачервоної детекторної матриці. Однак цей метод певною мірою обмежений матеріалом детектора і технологічним рівнем, а вартість його реалізації є відносно високою.

У класичній системі інфрачервоного зображення включаються наступні основні компоненти: лінзи, сенсори, процесори та інші [30]-[31]. Приклад такої системи зображена на рисунку 1.1

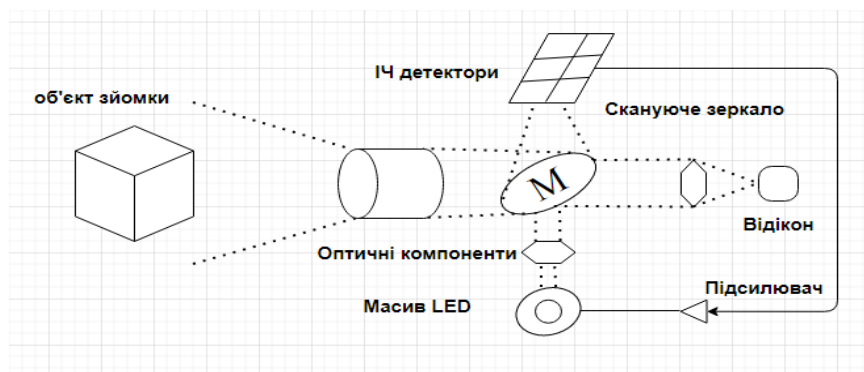


Рисунок 1.1 – Принцип роботи FLIR систем

На лінійному масиві ІЧ-детекторів об'єкт створює зображення. Після підсилення вихідних даних кожного детектора, відповідні світлодіоди (LED) вмикаються. Після відбивання від зворотної сторони скануючого дзеркала, цей масив LED передається на відео трубку (відікон). З свого боку, це живить типовий дисплей з катодно-променевою трубкою. Вся система перетворює скановане інфрачервоне зображення в цифрове представлення, яке можна переглядати на моніторі [31]. На сьогоднішній день, найпопулярніші системи інфрачервоного зображення використовують твердотільні детектори. Покращення якості ІЧ-зображення можна досягти, підвищуючи продуктивність іміджингових систем.

Згідно з поточними дослідженнями, деякі ключові компоненти в системі зображення: об'єктиви, сенсори та процесори, мають визначальний вплив на покращення якості ІЧ зображень.

1.2.2.1 Труднощі, пов'язані з лінзами

Загалом, щоб отримати кращу якість зображення, нам потрібно більше світла, щоб захопити його системою візуалізації [31]-[32]. Але більше світла означає більший об'єктив, що створює дві основні проблеми:

По-перше, більш складні прилади, що вимагаються, збільшать витрати і труднощі в усьому промисловому дизайні [32].

По-друге, більші оптичні лінзи збільшують розмір детектора [32]. Це призводить до того, що весь розмір стає більшим, а їх поєднання стає більш складним [33]. У швидко рухомих середовищах, на пристрій потрапляє менше світла, що призводить до погіршення якості зображення [32].

1.2.2.2 Труднощі, пов'язані з сенсорами

При захопленні ІЧ-світла сенсор є ключовим компонентом для перетворення сигналу світлової хвилі в цифровий сигнал. Цей компонент також

пов'язаний з двома наступними основними труднощами для покращення якості зображення.

По-перше, існує власний шум сенсора [34], [35]. Це шум, який називається шумом фіксованого шаблону (FPN), який існує в різниці між зміщенням напругами в ланцюгах зчитування стовпчиків необроблених інфрачервоних датчиків, що призводить до сильного смугового шуму, який повільно змінюється в часі [36].

Якщо розглядати цю проблему з точки зору апаратного забезпечення, то повністю вирішити її практично неможливо через фізичні обмеження [37].

По-друге, обмеження розмірів пристроїв ускладнює розвиток сучасних технологій у короткостроковій перспективі [33], [38].

1.2.2.3 Труднощі, пов'язані з процесорами

У багатьох системах ІЧ-зйомки все ще використовуються традиційні мікропроцесори. Тому запуск складних алгоритмів СР на етапі зйомки зображення може бути ускладнений через обмежену обчислювальну потужність самого процесора [39]. Підвищення продуктивності обладнання може швидко виразитися в поліпшенні якості зображення.

Однак, як уже згадувалося вище, ми стикаємося з багатьма труднощами, пов'язаними з ключовими компонентами, що використовуються в системах візуалізації ІЧ-камер. В основному це складність процесу виготовлення та фізичні обмеження. Щодо процесу виробництва, то ми очікуємо, що камера буде більш портативною, але для кращого зображення часто потрібні більші об'єктиви. Для процесорів нам потрібні нові технології для покращення обчислювальної потужності, що потребує більше часу та зусиль.

Таким чином, покращення якості зображень за рахунок апаратних засобів буде дорогим з точки зору промислових витрат і зусиль, а кінцеве покращення продуктивності буде обмеженим через непереборні фізичні перешкоди.

Виходячи з вищезазначених міркувань, для покращення якості ІЧ-зображень ми вдаємося до технології подальшої обробки зображень.

1.3 Сфери використання

Техніки відновлення зображень інфрачервоного спектру з над високою роздільною здатністю мають важливі застосування в різних галузях. У цьому розділі будуть розглянуті ключові області, в яких можуть бути застосовані ці техніки. Загалом, ці застосування можна розподілити на три основні групи: медичну та біохімічну інженерію, завдання, пов'язані зі зором, та інші інженерні завдання.

1.3.1 Медичні та біохімічно інженерні завдання

1.3.1.1 Виявлення амілоїдних згустків

Наприклад використовують SR техніки відновлення інфрачервоного зображень для покращення візуалізації CRANAD-2, фторогенного похідного куркуміну, що використовується для виявлення амілоїдних згустків, що беруть участь у нейродегенеративних захворюваннях [40]. Вони успішно досягли роздільної здатності в діапазоні 45-55 нм, використовуючи методи виснаження випромінювання з стимульованою емісією (STED) та одно молекулярну локалізаційну мікроскопію (SMLM), і застосували їх для візуалізації амілоїдних фібрил в ближньому інфрачервоному діапазоні.

Використання SR дозволило отримати значно більш точні та деталізовані зображення, що відкриває нові можливості для розвитку досліджень в цій області.

1.3.1.2 Виявлення вірусів

Розвиток нових технік для виявлення вірусів став критичним через спалахи життєзагрозливих пандемій, таких як COVID-19. Поточний підхід зі збору зразків з назофарингеального ділянки та подальшого визначення РНК за допомогою полімеразної ланцюгової реакції має проблеми з хибно-позитивними результатами та довгим часом діагностики. Супер масштабована мікроскопія, разом з іншими оптичними техніками, такими як спектроскопія зображення, пропонує перспективну альтернативу для виявлення вірусів. У цьому дослідженні було розглянуто стан сучасних досліджень, що використовують супер масштабовану мікроскопію, а також близькоспектральну спектроскопію, інфрачервону спектроскопію Фур'є, Раманівську спектроскопію, флуоресцентні техніки та детекцію на основі поверхневих плазмонних резонансів для виявлення вірусів [41].

Автори зосередилися зокрема на родині вірусів SARS-CoV з акцентом на виявленні коронавірусу. Також розглянули новітню технологію фотоніки для виявлення SARS-CoV-2 та майбутні можливості. Прогрес у галузі оптичних технік, зокрема використання супер масштабованої мікроскопії, надає перспективний підхід до розробки швидких пристроїв на основі фотоніки для виявлення COVID-19.

1.3.1.3 Вивчення процесів в клітинах

Використання мікроскопії зі супер масштабованою здатністю (SR) дозволяє детально вивчити процеси в клітинах. Magalie Bénard та Damien Scharman запропонували посібник для оптимізації підходів до передової мікроскопії, використовуючи експлуатацію часу життя (τ) флуоресцентних барв червоного / близького інфрачервоного діапазону для зменшення експозиції світла [42]. По-перше, вони охарактеризували ключові інструментальні елементи і встановили, що червоно- / близько інфрачервоні лазерні лінії з підводним

зображенням, забезпечують високу передачу сигналів флуоресценції, низьку інтенсивність опромінення та супер масштабованість. Гібридні детектори (HyD-S, HyD-X та HyD-R), які поєднують вакуумні трубки та напівпровідникові мікроелектронні елементи, особливо підходять для підрахунку червоно- / близько інфрачервоних фотонів та розділення τ . По-друге, вони перевірили та порівняли підходи зображення на основі часу життя для конфокальної мікроскопії, мікроскопії життя флуоресценції (FLIM) та наноскопії з стимульованою емісією депресії (STED), з урахуванням червоно- / близько інфрачервоного мультиплексування. Продемонстрували, що комбінація маркування червоно- / близько інфрачервоними фарбниками з життєвими стратегіями на основі τ пропонує нові перспективи для зображення живих клітин шляхом поліпшення збереження зразків за рахунок скорочення часу придбання і зниження експозиції світла, і змогли швидко і гнучко зображувати багатооб'єктні живі клітини H28, використовуючи HyD-ів разом з аналізом фазових діаграм часу життя флуоресценції.

1.3.1.4 Аналіз структури білків

Використання мікроскопії зі збільшеною роздільною здатністю (SR) дозволило збільшити роздільну здатність оптичного зображення, хоча дуже мало флуоресцентних білків підходять для цієї техніки, особливо в дальньо-червоному і близько інфрачервоному діапазонах випромінювання. Однак, Samuel Barnett та Andrew Hitchcock продемонстрували застосовність CrсA, підрозділу фотосинтетичного антенного комплексу у ціанобактеріях, для зображень STORM та SIM [43]. Виготовивши наноарії CrсA з коівалентно закріпленими хромофорами фікоеритробіліном (PEB) або фікоціанобіліном (PCB), автори змогли досягти періодичності та ширини, які відповідали лініям у відтворених зображеннях STORM. Їх відтворення SIM та STORM клітин *Escherichia coli* з міткою CrсA-позначеного цитохрому bd 1 убіхінолоксидази в цитоплазматичній мембрані показали, що CrсA-PEB та CrсA-PCB підходять для супер

масштабованої зйомки в живих клітинах. Крім того, стабільність, легкість виробництва, мала розмір та яскравість CrcA-PEB та CrcA-PCB демонструють потенціал цієї майже не вивченої сім'ї білків як новітніх зондів для мікроскопії зі збільшеною роздільною здатністю за допомогою SR.

1.3.2 Задачі бачення

1.3.2.1 Мультиспектральні узгодження

Багато систем комп'ютерного бачення базуються на точному визначенні відповідностей між парами зображень, але мультиспектральні методи все ще не мають необхідної стійкості та швидкості. Christoph Bodensteiner та Sebastian Bullinger звернули увагу на проблему знаходження точкових відповідностей у мультиспектральній системі зображення [44]. Традиційні методи спрямовані на досягнення інваріантних перетворень рис за допомогою мультимодальних описів, що часто призводить до зниження дискримінації. Замість цього, вони запропонували модель, яка вчиться перетворенням зображення, відображаючи одну модальність зображення на відповідне цільове зображення, з урахуванням даних оригінального спектрального діапазону. Їх підхід інтегровано з потоком сучасних методів відповідностей з використанням зростаючої складності та часу виконання алгоритмів. Підхід було перевірено на різноманітних мультиспектральних наборах даних, включаючи зображення в ближньому інфрачервоному, колірному інфрачервоному та термальному інфрачервоному діапазонах в день і вночі. Метод значно покращує швидкість та стійкість порівняно зі стандартними мультимодальними методами реєстрації, і його можна успішно інтегрувати у наявні системи. Цей підхід може мати безліч застосувань, включаючи об'єднання мультиспектральних датчиків, мультиспектральні системи одометрії, мультиспектральну сегментацію та методи мультиспектрального збільшення роздільної здатності.

1.3.2.2 Перетворення електрооптичних зображень на інфрачервоні зображення

В цій статті In Ho Lee описує застосування методів надвисокої роздільної здатності для перетворення електрооптичних зображень на інфрачервоні зображення, що можуть бути отримані за допомогою тепловізійних камер [45]. Це дозволяє отримувати зображення об'єктів, незалежно від погодних умов, освітлення або кольору середовища. Застосування глибинного навчання, зокрема генеративних змагальних мереж, дозволяє отримувати ІЧ-зображення з високою роздільною здатністю з електрооптичних зображень.

Для навчання нейронної мережі використовувалися дані з ThermalWorld, а також знімки ЕО та ІЧ-зображень, отримані за допомогою тепловізійних інфрачервоних камер. Функція втрат була адаптована для генерації ІЧ-зображень зі збереженням форми об'єкта. Результати показали, що запропонована техніка надвисокої роздільної здатності дозволяє отримувати ІЧ-зображення з високою якістю, а структура нейронної мережі була покращена для досягнення ще більшої роздільної здатності. Також були використані показники ефективності покращення якості зображення для кількісної оцінки результатів.

1.3.2.3 Термальне виявлення

Shuai You запропонував метод виявлення в реальному часі пішоходів за допомогою термічного зображення для методів обчислення країв, який допомагає подолати проблеми слабкої текстури, приховування об'єктів та малих об'єктів у різних умовах освітлення [46]. Також в цьому методі враховано те, що використання великих моделей високої продуктивності може призвести до збільшення часу затримки на пристроях обчислювального краю з обмеженою обчислювальною потужністю. Для поліпшення продуктивності системи виявлення пішоходів використовується мультимасштабна мозаїчна аугментація даних, що дозволяє покращити різноманітність та текстуру об'єктів, тим самим

зменшуючи вплив складних середовищ. Крім того, в мережу вводиться механізм уваги без параметрів для покращення виявлення рис, що не збільшує обчислювальні витрати мережі. Швидкість виявлення збільшується шляхом прискорення багатоканального відео виявлення за допомогою методів квантування та багатопотоковості на методах обчислення країв. Порівняльні експерименти, проведені на самостійно побудованому наборі даних YDTIP та трьох загальнодоступних наборах даних, показують, що запропонований метод перевершує інші методи, підкреслюючи його ефективність та потенціал для застосування у реальному світі.

1.3.2.4 Розпізнавання облич

Maciej Szankin розглядає проблеми розпізнавання осіб на знімках термальної зйомки, особливо з урахуванням питань безпеки, пов'язаних з віддаленим діагностуванням та системами моніторингу розумних будинків [5]. Хоча техніки обробки зображень можуть надавати детальну інформацію про впізнані об'єкти, людей та дії, різкі кути та чіткі зміни значень пікселів між областями, які часто використовуються для розпізнавання людей, не присутні на термальних зображеннях. Крім того, визнання людей за термографією є складнішим через розмитість та гладку репрезентацію ознак та обмежену кількість доступних даних для навчання моделей. Зниження роздільної здатності даних додатково ускладнює цю проблему, зробивши ознаки ще менш помітними.

Попередні результати показують, що модель SR може підвищити точність на 6,5% для даних, які у 8 разів менші за оригінальні зображення, що дозволяє точно розпізнати навіть 40 добровольців, використовуючи лише 4 зображення на людину в якості еталонного вбудовування. Цей підхід можна використовувати для створення початкових профілів у режимі реального часу, що є додатковою перевагою при налаштуванні рішення в нових умовах.

1.3.3 Інженерні задачі

1.3.3.1 Обчислення глибини

У контексті штучного інтелекту та технологій автономного водіння, сприйняття сцени стає все більш важливим. Непідготовлені методи глибинного навчання показали певний рівень стійкості та точності в складних сценах, але мають дефекти в складних середовищах. У цій статті Jiafeng Shen та Kaiwei Wang запропонували непідготовлену монокулярну мережу оцінки глибини, яка має на меті досягнення високої швидкодії та точності, одночасно удосконалюючи результати глибини, за допомогою зображень, перетворених між різними модальностями [6].

Мережа складається з кодувально-декодувальної мережі, щоб створювати багатомасштабну щільну карту глибини, з суб-піксельним згортковим шаром, який використовується для отримання глибини високої роздільної здатності шляхом заміни гілок піднесення до верхнього рівня. За допомогою включення інфрачервоного та RGB зображень, перехресна модальна оцінка глибини покращує продуктивність порівняно з використанням лише RGB зображень. Режим навчання передбачає перенесення обох зображень до однієї модальності, а потім здійснення глибинного оцінювання з високою роздільною здатністю для кожної пари стереокамер.

Попередні результати показують, що запропонована мережа оцінки глибини з системою крос-модального злиття досягає кращої продуктивності на загальнодоступних наборах даних і мультимодальних даних, зібраних датчиком стереозображення. Зокрема, експеримент підтверджує, що мережа підвищує точність оцінки глибини за рахунок включення іншої модальної інформації, демонструючи потенціал моделі надвисокої роздільної здатності для підвищення продуктивності систем автономного водіння.

1.3.3.2 Дистанційне зондування

В наш час, за допомогою дистанційного зондування, компактні сенсори можуть бути закріплені на платформах безпілотних літальних апаратів (БПЛА), що дозволяє ефективно моніторити та картографувати рельєф на великих територіях. У цьому дослідженні використовуються компактні сенсори дистанційного зондування з видимого спектра (400 нм - 700 нм), ближнього-інфрачервоного спектра (700 нм - 1300 нм) та інфрачервоного термографічного спектра (1300 нм - 15000 нм), щоб контролювати стабільність скельного схилу [7]. Вегетаційні індекси, такі як нормалізований індекс відмінності вегетації (NDVI), водні індекси, зокрема нормалізований індекс відмінності води (NDWI), та радіометрична температура, отримана за допомогою термографічних зображень, використовуються для аналізу стійкості різних типів земель. NDVI використовується для аналізу здоров'я рослинності, де більш здорова рослинність свідчить про сильну кореневу систему, яка допомагає стабілізувати схил. Тим часом, NDWI може виявити наявність води на поверхні скельного схилу, а вищий індекс NDWI вказує на наявність води, що може зробити схил менш стійким. Шляхом отримання теплових даних поверхні скельного схилу за допомогою термографії можливо проаналізувати шкоду, заподіяну погодними умовами. Загалом, ця техніка з використанням ДЗ сенсорів, яка є ефективною та недорогою, є важливим для використання в якості початкового індикатора для дослідження руйнування скельного схилу на основі низької вартості методу і швидкого отримання інформації про дані.

1.3.3.3 Контроль якості харчових продуктів

Сьогодні, для успішності на конкурентному ринку оливкової олії, необхідно гарантувати найвищу якість екстра-віргін оливкової олії. Для досягнення цього, необхідні високі стандарти якості фруктів. У останні роки, для задоволення цієї потреби, було розроблено системи комп'ютерного бачення для

автоматичної інспекції фруктів. Пошкодження шкіри, забруднення та зовнішній колір є основними ознаками, які були обрані для інспекції. Однак, немає відомих звітів, що стосуються оцінки внутрішньої якості оливок. Diego M. Martínez Gila досліджував застосування мультиспектральних зображень (від 600 до 975 нм) для розрізнення оливкових фруктів за їхньою твердістю [9]. Для досягнення цієї мети, експерт класифікував 58 оливок як тверді або м'які, а потім виміряв їх пенетрометром. Після цього було отримано 58 мультиспектральних зображень, і було випадково вибрано 8352 пікселів для вилучення ознак. Вектор ознак кожного пікселя складався з 25 значень поглинання інфрачервоного випромінювання. Ці дані були введені в три алгоритми навчання з учителем: найвний байєсівський класифікатор, багатошаровий перцептрон та варіація машини опорних векторів. Найкращий результат було отримано за допомогою багатошарового перцептрона, з точністю 95% для класифікації твердості. Крім того, три алгоритми були піддані аналізу головних компонент (PCA), який підтвердив, що MLP є найкращою моделлю, і покращив його результати. Згідно з другою головною компонентою, найбільшу вагу мають частоти 675 нм і 855 нм. Загальні результати підтверджують зручність використання мультиспектральної візуалізації для перевірки твердості оливок, перевагою якої є те, що вона є неінвазивною технологією.

1.3.3.4 Управління сільським господарством

Індекс площі листя (LAI) - це параметр, що використовується для прогнозування врожайності, а безпілотні літальні апарати (БПЛА) пропонують новий метод для моніторингу LAI. У цьому дослідженні було використано БПЛА з фіксованим крилом, обладнаний мультиспектральними камерами для дистанційного зондування [8]. Були проведені польові експерименти на двох сортах арахісу з різною густотою посадки для оцінки LAI з мультиспектральних зображень та розробки моделі прогнозування LAI з високою точністю. Було використано вісім індексів рослинності (VIs), і були розроблені моделі простої

регресії та штучної нейронної мережі (BPN) для LAI та спектральних VIs. Емпірична модель була калібрована для оцінки LAI арахісу, і найкраща модель була вибрана на основі коефіцієнта детермінації та середньоквадратичної помилки. Червоний (660 нм) та близький до інфрачервоного (790 нм) діапазони ефективно прогнозували LAI арахісу, який збільшувався з густотою посадки. Модель множинної регресії мала вищу точність прогнозування, ніж моделі простої лінійної регресії, і Modified Red-Edge Simple Ratio Index (MSR), Ratio Vegetation Index (RVI) та Normalized Difference Vegetation Index (NDVI) мали більш високі кореляції з LAI, ніж інші індекси. Комбінована модель BPN VI була більш точною, ніж одиночна модель BPN VI, а точність моделі BPN була вищою. Щільність посадки впливала на LAI арахісу, а індекси рослинності на основі відбивної здатності були корисними для прогнозування LAI.

1.3.3.5 Зореутворення

В останні роки були розроблені нові техніки зображення з фокусом на поліпшенні точності навіть у суперроздільних режимах, що можуть значно покращити інтерферометричні спостереження протопланетарних дисків. Команда Masayuki Yamaguchi вперше використала розріджену модель (SpM) для обробки наборів даних, знятих з допомогою Atacama Large Millimeter/submillimeter Array (ALMA) [10]. Два набори даних, взяті незалежно з різними конфігураціями масиву на Band 7 (330 ГГц), були спрямовані на протопланетний диск навколо HD 142527: один з коротко базовою конфігурацією (~430 м), а інший з довго базовою конфігурацією (~1570 м). Роздільна здатність зображень, відтворених з двох наборів даних, відрізняється в 3 рази. Автори підтвердили, що відомі раніше структури диску з'являються на зображеннях, отриманих як з SpM, так і з CLEAN при стандартному розмірі променевої діаграми. Зображення, відтворене з коротко базового набору даних за допомогою SpM, відповідає зображенню, отриманому з довго базового набору даних за допомогою CLEAN, що дає змогу отримати супер масштабоване

зображення, з якого можна відтворити структуру, що менша за розмір променевої діаграми. Ці результати демонструють, що інтенсивна розробка техніки зображення SpM корисна для зображення з ALMA.

1.3.4 Методи масштабування

Для вдосконалення якості зображень можна використовувати методи масштабування, які збільшують розмір зображення з використанням різних підходів.

У цьому розділі ми детально розглянемо різні методи масштабування зображень, включаючи інтерполяційні та навчальні підходи. Ми також розглянемо переваги та недоліки методів, щоб вибрати найкращий метод.

1.3.4.1 Масштабування на основі інтерполяції

Інтерполяція зображень, використовується для зміни розміру цифрових зображень і є невід'ємною частиною програм, пов'язаних з обробкою зображень. Традиційні методи інтерполяції, такі як інтерполяція найближчого сусіда, білінійна та бікубічна інтерполяція, редискретизація Сінка та Ланчоса, є простими в реалізації та широко використовуються для цієї мети. Незважаючи на їх простоту, деякі з цих методів все ще застосовуються в сучасних моделях супер масштабування на основі згорткових нейронних мереж (CNN), оскільки вони можуть забезпечити ефективну інтерпретацію та досить високу якість зображень.

1.3.4.1.1 Інтерполяція найближчого сусіда.

Інтерполяція найближчих сусідів [47] - це метод інтерполяції, який використовується для збільшення розміру зображення шляхом додавання нових пікселів між існуючими. В основі цього методу лежить ідея вибрати значення

найближчого пікселя для кожної нової позиції на зображенні. Це означає, що нові пікселі будуть мати ту саму інтенсивність, що і найближчий піксель на початковому зображенні.

Алгоритм працює наступним чином: спочатку визначається новий розмір зображення, який повинен бути отриманий. Далі для кожної нової позиції на зображенні знаходиться найближчий піксель і його значення копіюється на нову позицію. Цей процес повторюється для кожної нової позиції на зображенні, доки не буде створено нове зображення з більшим розміром.

Інтерполяція найближчих сусідів є досить простим методом, який швидко працює і легко реалізується. Проте, він має деякі недоліки. Один з них полягає в тому, що Інтерполяція найближчих сусідів може призводити до появи блочних артефактів на відтвореному зображенні, які проявляються в тому, що зображення має вид кубів з різними кольорами в кожному кубі. Крім того, при збільшенні масштабу зображення, метод може приводити до втрати деталей і розмиття границь між об'єктами. Однак, цей метод може бути корисним у деяких випадках, коли швидкість більш важлива, ніж точність результату.

1.3.4.1.2 Метод білінійної інтерполяції

Метод білінійної інтерполяції (BLI) [48] є одним з методів інтерполяції зображень, який використовують для збільшення розміру зображення. Цей метод працює наступним чином: спочатку виконується лінійна інтерполяція по одній з осей зображення, а потім виконується інтерполяція по іншій осі. В результаті, для кожного нового пікселя на вихідному зображенні, BLI використовує чотири найближчих пікселі з вихідного зображення, щоб отримати значення для нового пікселя.

Крім того, BLI забезпечує квадратичну інтерполяцію з розміром рецептивного поля 2×2 , що дозволяє показувати значно кращу продуктивність, ніж метод інтерполяції найближчих сусідів, зберігаючи при цьому відносно високу швидкість. Білінійна інтерполяція дозволяє отримувати більш гладкі

результати з меншою кількістю артефактів, що дуже важливо для багатьох задач обробки зображень, таких як розпізнавання облич та зменшення шуму зображень.

Загалом, BLI є доволі ефективним методом для збільшення розміру зображення та зменшення його артефактів, проте він може бути менш ефективним для деяких завдань, де потрібна висока якість результуючого зображення.

1.3.4.1.3 Бікубічна інтерполяція

Бікубічна інтерполяція (BCI) [59] - це метод інтерполяції, що використовується для оцінки значень функції в точках, розташованих між відомими точками з відомими значеннями функції. Цей метод використовує кубічні функції для опису поведінки функції в околі кожної відомої точки.

Для застосування BCI спочатку необхідно вибрати відомі точки, які відповідають деяким вхідним даним. За допомогою цих точок будується кубічна поверхня, яка апроксимує значення функції в околі кожної точки. Ця поверхня складається з кубічних поліномів, що містять коефіцієнти, які потрібно визначити.

Для визначення коефіцієнтів BCI використовується система лінійних рівнянь, яка базується на властивостях кубічних функцій. Ці коефіцієнти визначаються з врахуванням значень функції в околі відомих точок та їх похідних.

Після визначення коефіцієнтів кубічних поліномів, що описують поведінку функції в околі кожної відомої точки, можна використовувати ці поліноми для визначення значень функції в будь-якій точці в межах діапазону, обмеженого відомими точками.

1.3.4.1.4 Масштабування на основі навчання

Масштабування на основі навчання - це метод збільшення розміру зображення, що використовується в галузі обробки зображень та машинного навчання. Він полягає у використанні нейронних мереж для відтворення відсутніх деталей в зображенні з використанням статистичних залежностей між пікселями.

У відмінну від традиційних методів збільшення розміру зображення, таких як інтерполяція, масштабування на основі навчання здатний відтворювати складні структури та деталі в зображенні з високою точністю, забезпечуючи більш якісний результат.

Основна ідея масштабування на основі навчання полягає в тому, що нейронна мережа навчається зображати зв'язок між вхідним та вихідним зображенням, використовуючи статистичні залежності між пікселями. На вході нейронна мережа отримує зменшене зображення, а на виході - збільшене зображення. Під час тренування, мережа коригує ваги своїх нейронів, щоб зменшити різницю між вихідним та очікуваним зображенням. Після тренування, мережа може використовуватись для збільшення розміру будь-якого зображення з високою якістю та деталізацією. Схеми варіантів шарів розгортки наведено на рисунку 1.2

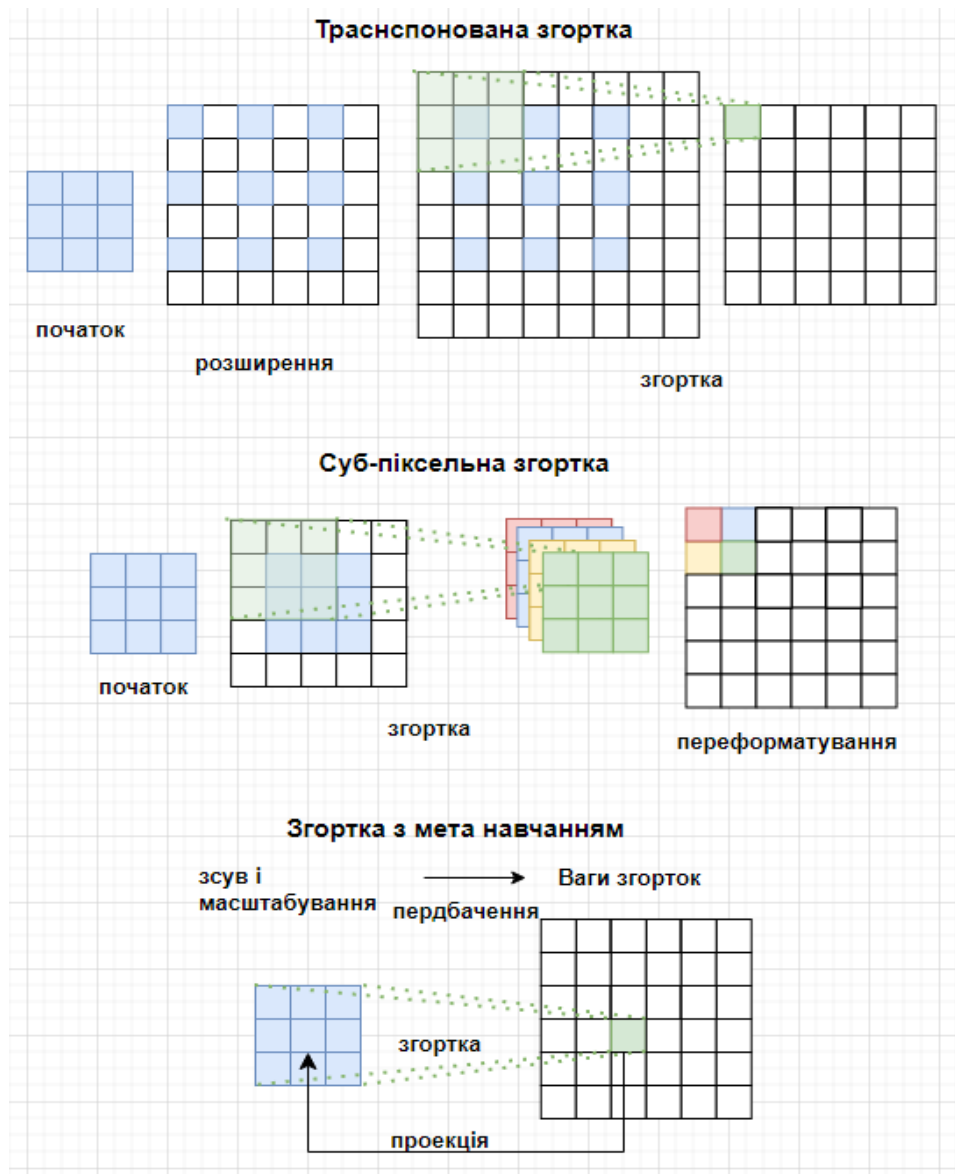


Рисунок 1.2 – Схеми варіантів шарів розгортки

1.3.4.1.5 Транспонований шар згортки

Транспонований шар згортки [50], також відомий як шар розгортки, є операцією в глибокому навчанні, яка спрямована на зворотне перетворення, виконане звичайним шаром згортки. Шар приймає на вхід зображення низької роздільної здатності та видає зображення високої роздільної здатності, вставляючи нулі та виконуючи операцію згортки. Це дозволяє шару збільшувати роздільну здатність зображення та збільшувати розмір вхідних даних шляхом підвищення роздільної здатності.

Наприклад, 2-кратне збільшення зображення з ядром розміром 3×3 досягається шляхом спочатку збільшення розміру зображення в два рази, а потім виконання операції згортки з ядром розміром 3×3 , з кроком 1 та додаванням заповнення 1. Цей шар може використовуватись як шар масштабування в моделях супер масштабування, оскільки він зберігає зв'язкові шаблони, які сумісні з звичайними шарами згортки.

Проте транспоновані шари згортки можуть призводити до "нерівномірного накладання" на кожній осі, що призводить до відображення шахової дошки з різною інтенсивністю, що може погіршити результати супер масштабування. Тому для уникнення цих проблем використовуються деякі техніки, такі як субпіксельна згортка або масштабування з мета-навчанням.

1.3.4.1.6 Субпіксельна згортка

Субпіксельний шар [51] - це ще один тип навчального шару збільшення роздільної здатності з використанням глибокого навчання. Він збільшує роздільну здатність, застосовуючи згортку, щоб створити кілька каналів, а потім перетворюючи їх. У цьому шарі спочатку застосовується згортка для отримання виведень із кількістю каналів, що дорівнює s^2 разів кількості вхідних каналів, де s - це коефіцієнт масштабування. Потім застосовується операція перетворення форми, щоб створити вивід розміром $h * w * s^2 * c$. де c - кількість каналів.

Субпіксельний шар має більший поле рецептивності порівняно з транспонованим шаром згортки, що забезпечує більше контекстуальної інформації для генерації реалістичних деталей. Однак, це може призвести до артефактів біля меж різних блоків через нерівномірний розподіл поля рецептивності та спільний обсяг рецептивного поля блоків. Крім того, незалежне передбачення сусідніх пікселів в блокових областях може призвести до негладких результатів.

Щоб вирішити ці проблеми, Gao та ін. запропонували PixelTCL, який замінює незалежне передбачення пікселів на між залежне послідовне передбачення, що призводить до більш гладких та послідовних результатів.

1.3.4.1.7 Масштабування з мета-навчанням

Шар масштабування з мета-навчанням [52] є новим підходом, який має на меті вирішити недоцільність попередніх методів, які вимагають передбачення масштабних коефіцієнтів та тренування різних модулів для кожного коефіцієнта. Цей модуль призначений для забезпечення супер-роздільної здатності з довільними масштабними коефіцієнтами за допомогою мета-навчання.

Підхід проектує кожну цільову позицію на зображенні високої роздільності (HR) на невеликий патч на картах ознак низької роздільності (LR), передбачає ваги згортки на основі зсувів проекції та масштабного коефіцієнта за допомогою густих шарів та виконує згортку. Цей процес дозволяє модулю безперервно збільшувати зображення з довільними масштабними коефіцієнтами за допомогою однієї моделі.

Модуль масштабування з мета-навчанням навчається на великій кількості даних, яка включає кілька масштабних коефіцієнтів. Це тренування дозволяє модулю досягнути порівнянної або навіть кращої продуктивності, ніж методи з фіксованими масштабними коефіцієнтами. Під час інференсу цей метод передбачає ваги для кожного цільового пікселя, що займає лише близько 1% часу, необхідного для видобутку ознак. Однак, через те, що цей метод передбачає велику кількість ваг згортки для кожного цільового пікселя на основі кількох незалежних значень, він може давати нестабільні та менш ефективні результати при роботі з більшими масштабами.

Порівняння підходів масштабування наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння підходів масштабування

Навчально-орієнтоване масштабування	Інтерполяційне масштабування
Генерує нові дані: Навчання на основі збільшення генерує нові точки даних, що призводить до поліпшення якості зображення та більш точного представлення.	Використовує існуючі дані: Метод збільшення на основі інтерполяції використовує існуючі точки даних для створення нових, які можуть бути не завжди такими точними або представницькими.
Більш складний: Методи навчання на основі збільшення зазвичай включають більш складні алгоритми та вимагають більше обчислювальних ресурсів, що робить їх більш обчислювально витратними.	Менш складний: Методи на основі інтерполяції зазвичай є простішими та швидшими, потребують менше обчислювальних ресурсів.
Адаптивність: Методи навчання на основі збільшення можуть адаптуватися до конкретних даних та навчитися генерувати більш точне представлення.	Обмежена адаптивність: Методи на основі інтерполяції обмежені наявними точками даних та не можуть генерувати нові дані.
Зменшення артефактів: Методи навчання на основі збільшення можуть допомогти зменшити артефакти, такі як шум та спотворення, під час процесу збільшення.	Обмежене зменшення артефактів: Методи на основі інтерполяції можуть бути менш ефективними у зменшенні артефактів.

Продовження таблиці 1.1

Навчально-орієнтоване масштабування	Інтерполяційне масштабування
Краще для великих збільшень: методи навчання на основі генерації даних зазвичай краще підходять для збільшення зображень на великі масштаби, оскільки вони можуть генерувати нові точки даних та краще захоплювати нюанси оригінального зображення.	Краще для малих збільшень: методи інтерполяції на основі використання існуючих даних можуть бути краще підходять для збільшення на менші масштаби, оскільки вони можуть ефективно інтерполювати між існуючими точками даних.

Навчально-орієнтоване масштабування є вигіднішим для використання, оскільки воно дозволяє генерувати нові дані на основі навчального набору даних, що призводить до поліпшення якості зображення та більш точного представлення. Хоча вони і включають більш складні алгоритми та вимагають більше обчислювальних ресурсів, але це компенсується більш точним та реалістичним результатом.

Загалом, навчально-орієнтоване масштабування може бути більш ефективним для застосування в сучасних системах комп'ютерного зору та обробки зображень, особливо у випадках, коли необхідно отримати високоякісні та реалістичні зображення.

Порівняння шарів розгортки наведено у таблиці 1.2.

Таблиця 1.2 – Порівняння шарів розгортки

	Транспонований шар згортки	Субпіксельний шар	Модуль мета - масштабування
Контекстуальна інформація	Обмежена	Більше	Більше
Артефакти шахової дошки	Можливі	Відсутні	Відсутні
Гладкість виходів	Не гарантована	Можлива	Можлива
Стабільність прогнозів	Добра	Добра	Менш стабільна
Ефективність при більших масштабах зображення	Не ефективний	Ефективний	Менш ефективний

На основі порівняння, субпіксельний шар має найбільші переваги та найменші недоліки, що робить його найкращим варіантом серед трьох. Він забезпечує більше контекстної інформації, уникне артефактів шахової дошки та блочних регіонів, і може виробляти гладкі виходи. Крім того, він є ефективним, гнучким та стабільним у прогнозуванні. Хоча масштабування на мета-навчанні і має великий потенціал, це все ще дуже новий підхід котрий потребує додаткових досліджень, що зможуть компенсувати його поточні недоліки.

1.4 Постановка завдання

У порівнянні з інфрачервоними зображеннями низької роздільної здатності (LR), інфрачервоні зображення високої роздільної здатності (HR) містять більше високочастотних деталей і надають багатшу інформацію про сцену. Через обмеження апаратних пристроїв роздільна здатність необроблених інфрачервоних зображень, як правило, низька, а особливості текстури не такі очевидні, як на видимих зображеннях.

Через існування різноманітних шляхів деградації, реконструкція з надвисокою роздільною здатністю для ІЧ-зображень все ще вважається погано поставленою проблемою [30]. Але в цілому завдання виглядає наступним чином (1.1):

$$I_{LR} = D(I_{HR}; \delta) \quad (1.1)$$

де D позначає функцію деградації;

I_{HR} - ІЧ-зображення високої роздільної здатності;

I_{LR} - ІЧ-зображення низької роздільної здатності;

δ - параметри процесу деградації.

У загальному випадку процес деградації включає наступні фактори: зниження дискретизації, шум і стиснення, як показано в наступному рівнянні (1.2). У різних реальних умовах важливість цих факторів буде різною.

$$D(I_{HR}; \delta) = (I_{HR} \otimes k) \downarrow_d; [k, \downarrow_d] \subset \delta \quad (1.2)$$

де $I_{HR} \otimes k$ - згортка між ядром розмиття k та HR зображенням I_{HR} ;

k - шум і тиск;

$\downarrow_d$ - коефіцієнт зменшення вибірки, наприклад, $4\times$ та $8\times$.

Коротко цільову функцію реконструкції ГЧ-зображень з надвисокою роздільною здатністю можна описати рівнянням (1.3)

$$\hat{\theta} = \arg_{\theta} \min L(I_{HR}, I_{SR}) + \lambda \Phi(\theta) \quad (1.3)$$

де L - функція втрат між HR зображенням I_{HR} та SR зображенням I_{SR} ;

$\Phi(\theta)$ - член регуляризації;

λ - параметр покарання;

Початкові роботи відносилися до супер масштабування інфрачервоних зображень як до СМ звичайних зображень і просто адаптували модель СМ видимих зображень (наприклад, методи на основі згорткових нейронних мереж (CNN) [14], методи на основі генеративних змагальних мереж (GAN) [22] і модель трансформера [53]) до нової задачі.

На відміну від методу використання лише однокадрових LR-зображень як вхідних даних, суть методу реконструкції надвисокої роздільної здатності на основі референсного зображення (ReFSR) полягає у введенні додаткових зображень, які допомагають у процесі реконструкції надвисокої роздільної здатності.

Як правило, референсне зображення повинно мати текстуру та вміст, подібні до LR-зображення. У статті [54] для того, щоб забезпечити відповідність двох вхідних зображень, для зіставлення блоків використовується градієнтна характеристика блоку вибірки під референсним HR-зображенням, а для генерації зображення використовується ковзне середнє. Цей метод не повністю використовує високочастотну інформацію при зіставленні, а також не використовує попереднє зображення високої роздільної здатності при генерації реконструйованого зображення, тому він не може вивчити складні деталі відображення між блоками LR і HR зображень.

Чжен [55] та ін. запропонували модель ReFSR CrossNet на основі глибокого навчання, яка витягує різномасштабні ознаки з LR і референсних зображень, а потім вирівнює LR і ознаки референсних зображень у просторі для

синтезу вихідного HR-зображення. Чжан [56] та ін. запропонували зіставлення різномасштабних ознак у просторі ознак замість зіставлення у вихідному піксельному просторі. Шао запропонував застосувати метод глибокого навчання до інфрачервоного поля, зменшити канал передачі даних і змодельовати структуру відповідно до характеристик інфрачервоного зображення, що дозволило досягти гарного візуального ефекту.

Однак, через високу вартість отримання інфрачервоних зображень, опубліковано відносно мало великомасштабних наборів інфрачервоних даних. Крім того, ефект реконструкції з надвисокою роздільною здатністю на основі лише одного кадру є обмеженим.

Для вирішення вищезазначених проблем пропонується алгоритм реконструкції з надвисокою роздільною здатністю на основі псевдо перенесених ознак (PT2SR: pseudo feature transfer super reconstruction). Видиме зображення з високою роздільною здатністю, яке легко отримати, перетворюється на псевдоінфрачервоне зображення з високою роздільною здатністю (PIR) за допомогою фреймворку перетворення зображення в зображення [57] (CycleGAN). PIR і LR зображення багаторівнево зіставляються в просторі карти ознак для перенесення ознак, пов'язаних із семантикою LR інфрачервоного зображення, з псевдоінфрачервоного зображення на SR зображення, ознаки на псевдоінфрачервоному зображенні використовуються для компенсації відсутніх ознак на LR інфрачервоному зображенні. Оригінальне LR-зображення і відповідна карта обмінюваних ознак використовуються в запропонованій моделі перенесення ознак для поступової реконструкції зображення надвисокої роздільної здатності від високого рівня до низького.

Внесок цього методу полягає в наступному:

- для генерації псевдоінфрачервоних зображень з набору видимих зображень високої роздільної здатності використано систему навчання з перенесенням зображення в зображення;

- у цьому методі пропонується мережа для відновлення інфрачервоних зображень на основі псевдоперенесених ознак, яка в основному складається з

модуля зіставлення ознак і модуля перенесення ознак. Модуль зіставлення ознак призначений для обчислення подібності карт ознак з різними шарами між псевдоінфрачервоними та інфрачервоними зображеннями низької роздільної здатності. Модуль перенесення характеристик відповідає за інтеграцію нової міграції карти особливостей в остаточне зображення надвисокої роздільної здатності.

1.5 Висновки за розділом 1

Під час роботи над першим розділом була вивчена проблематика супермасштабування інфрачервоних зображень на основі електрооптичного зображення. В результаті проведеного аналізу встановлено, що одним з найбільш актуальних підходів до супермасштабування є використання аналітичних моделей, заснованих на машинному навчанні, зокрема штучних нейронних мереж. Особливо перспективним виявився підхід, який використовує навчання з використанням субпіксельних згорток та переносу ознак для отримання псевдоінфрачервоних зображень.

2 ВИБІР І ОБҐРУНТУВАННЯ ІНСТРУМЕНТІВ І АРХІТЕКТУРИ

2.1 Опис основних вимог до програми

Об'єктом розробки є ПЗ для супер масштабування інфрачервоних зображень за зразком з використанням нейронної мережі. Метою розробки є створення ефективного і точного програмного забезпечення, яке здатне автоматично масштабувати інфрачервоні зображення за зразком з високою роздільною здатністю та деталізацією.

Розробка ПЗ являє собою реалізацію основних системних модулів. Перший модуль є частиною, що приймає вхідні зображення, та передоброблює. Другий модуль має перенести інформацію з HR зображення до LR. Та третій модуль, котрий відповідає за масштабування зображення.

Функціональні вимоги до розроблюваного ПЗ формують такий перелік:

- масштабування інфрачервоних зображень: ПЗ повинно забезпечувати здатність масштабувати інфрачервоні зображення з високою точністю та зберігати основні деталі і структуру зображення;
- збереження контексту: ПЗ повинно забезпечувати збереження контексту зображення при масштабуванні, щоб уникнути втрати важливих особливостей та деталей, які є важливими для інфрачервоних зображень;
- висока швидкість обробки: ПЗ повинно працювати ефективно і забезпечувати швидку обробку інфрачервоних зображень, щоб користувач міг отримувати результати в реальному часі або з мінімальним затримкою;
- автоматичне виявлення параметрів: ПЗ повинно мати вбудовані алгоритми або механізми, які автоматично виявляють оптимальні параметри для масштабування інфрачервоних зображень, забезпечуючи найкращу якість зображення без необхідності вручну налаштовувати параметри;
- візуалізація результатів: ПЗ повинно мати можливість відображати результати масштабування інфрачервоних зображень, щоб користувач міг оцінити якість та ефективність процесу масштабування.

2.2 Вибір мови програмування

Вибір мови програмування для розробки систем машинного навчання важливий з точки зору продуктивності, швидкодії, зручності розробки, розширюваності та доступності бібліотек та інструментів машинного навчання. Вибір мови впливає на доступність бібліотек, продуктивність, зручність розробки та швидкодію. Він також залежить від спільноти розробників, підтримки, інтеграції та розширюваності. Найкращий вибір мови залежить від конкретних вимог проекту, знань розробників та наявних ресурсів.

2.2.1 Python

Python [58] - це високорівнева мова програмування, яка має важливі переваги для розробки нейронних мереж:

- популярність: Python є однією з найпопулярніших мов програмування у світі. Його популярність пояснюється зрозумілим і простим синтаксисом, що сприяє швидкому навчанню та розробці. Велика спільнота розробників Python забезпечує доступ до різноманітних ресурсів, документації та підтримки;

- легкість використання: Python відомий своєю простотою і легкістю використання. Його зрозумілий синтаксис дозволяє швидко розробляти, тестувати та відлагоджувати нейронні мережі. Багато вбудованих функцій та стандартних бібліотек полегшують завдання з обробки даних та візуалізації;

- швидкодія: В хоча Python не є найшвидшою мовою програмування, вона пропонує широкий вибір оптимізованих бібліотек, таких як NumPy, TensorFlow [59] та PyTorch [60], які реалізують векторизацію та оптимізовані обчислення. Це дозволяє отримати високу швидкодію при обробці великих обсягів даних та тренуванні складних нейронних мереж;

- розширюваність: Python є високорівневою мовою програмування, яка дозволяє легко інтегрувати низькорівневі бібліотеки, написані на C, C++ [61] або

Fortran. Це розширює можливості мови, дозволяючи використовувати оптимізовані алгоритми та обчислення на мовах з низьким рівнем абстракції;

- спільнота: Python має велику та активну спільноту розробників, що підтримується великою кількістю форумів, блогів, пакетних менеджерів та інших ресурсів. Це створює сприятливу середу для обміну досвідом, підтримки та спільної роботи над проектами з нейронними мережами;

- гнучкість: Python - це мова програмування, яка дозволяє реалізувати різноманітні архітектури нейронних мереж та експериментувати з різними методами навчання. Він пропонує широкий вибір фреймворків і бібліотек, що підтримують різні типи нейронних мереж, включаючи згорткові мережі, рекурентні мережі та глибокі навчання;

- наявність бібліотек: Python має велику кількість бібліотек, спеціалізованих на розробці нейронних мереж. TensorFlow, PyTorch, Keras, Theano та інші бібліотеки надають потужні функціональні можливості для тренування, валідації та використання нейронних мереж у різних задачах машинного навчання.

Ці параметри роблять Python першим вибором для багатьох дослідників, розробників та практиків машинного навчання при роботі з нейронними мережами.

2.2.2 C++

C++ [61] - це потужна мова програмування, яка також має важливі переваги для розробки нейронних мереж:

- популярність: C++ є однією з найпопулярніших мов програмування у світі. Вона широко використовується в багатьох галузях, включаючи машинне навчання та комп'ютерний наук. Це означає, що вона має велику спільноту розробників, яка підтримується багатими ресурсами та документацією;

- легкість використання: C++ є мовою з низьким рівнем абстракції, що означає, що розробник має повний контроль над роботою з пам'яттю та

обчисленнями. Використання C++ для розробки нейронних мереж вимагає більшого рівня експертизи, але це дозволяє досягти високої продуктивності та ефективності;

- швидкодія: C++ відомий своєю високою швидкістю. Мова надає прямий доступ до апаратних ресурсів та можливість оптимізувати код для максимальної продуктивності. Це робить C++ привабливим вибором для розробки нейронних мереж, особливо для завдань, де швидкодія є критичною, наприклад, в реальному часі аналізу даних або обробці великих обсягів інформації;

- розширюваність: C++ дозволяє легко інтегрувати код, написаний на інших мовах програмування, зокрема з низькорівневими бібліотеками та фреймворками. Це дає можливість використовувати оптимізовані алгоритми та реалізувати високошвидкісні розрахунки, що дозволяє ефективно розробляти складні нейронні мережі;

- спільнота: В C++ існує активна та велика спільнота розробників, що займається машинним навчанням та нейронними мережами. Це означає, що ви зможете знайти підтримку, ресурси, бібліотеки та інші інструменти, необхідні для розробки нейронних мереж у C++. Також існують платформи, які забезпечують готові фреймворки та бібліотеки, що полегшують роботу з нейронними мережами у C++;

- гнучкість: C++ дозволяє створювати власні структури даних, алгоритми та функції, що надає розробникам велику гнучкість у виборі та налаштуванні архітектури нейронних мереж. Ви можете розробляти власні класи та модулі, що відповідають вашим потребам та специфікаціям проекту;

Ці параметри роблять C++ привабливим вибором для розробки нейронних мереж, особливо в ситуаціях, коли важлива швидкодія, гнучкість та висока продуктивність. Однак, використання C++ для розробки нейронних мереж може бути складнішим порівняно з іншими мовами програмування, і вимагати більшої експертизи з мови та оптимізації коду.

2.2.3 MATLAB

MATLAB [62] - це мова програмування та середовище розробки, яке надає багато переваг для розробки нейронних мереж:

- популярність: MATLAB є однією з найпопулярніших мов програмування у галузі наук про дані та машинного навчання. Вона має широке застосування у вузах, дослідницьких лабораторіях та промисловості. Це означає, що вона має велику спільноту розробників, активне фахове співтовариство та багато доступних ресурсів;

- легкість використання: MATLAB пропонує простий та інтуїтивно зрозумілий синтаксис програмування, який дозволяє швидко реалізовувати нейронні мережі. Вона має багатий набір вбудованих функцій та інструментів, що полегшують створення, тренування та експериментування з нейронними мережами;

- швидкодія: MATLAB відомий своєю високою продуктивністю та швидкодією обчислень. Вона має підтримку оптимізованих функцій та бібліотек, що дозволяють виконувати обчислення з великою швидкістю. MATLAB також має підтримку обчислення на графічних процесорах (GPU), що дозволяє прискорити обробку даних та тренування нейронних мереж;

- розширюваність: MATLAB дозволяє інтегрувати код, написаний на C++, Python та інших мовах програмування. Це дозволяє використовувати оптимізовані бібліотеки та фреймворки для роботи з нейронними мережами, які розроблені для інших мов. MATLAB також надає можливість експортувати натреновані моделі для використання в інших мовах;

- спільнота: MATLAB має велику та активну спільноту розробників, яка працює в галузі машинного навчання та нейронних мереж. Це означає, що ви можете отримати підтримку, поради та розв'язки проблем від інших розробників. Також існує багато відкритих джерел, навчальних матеріалів та прикладів коду, які допоможуть вам у вивченні та розробці нейронних мереж у MATLAB;

- гнучкість: MATLAB дозволяє легко експериментувати з різними архітектурами нейронних мереж, шарами та параметрами. Вона надає зручні інструменти для візуалізації даних, аналізу результатів та налаштування моделей. Ви можете швидко створювати прототипи, тестувати різні ідеї та вдосконалювати свої нейронні мережі у MATLAB;

- наявність бібліотек: MATLAB має велику кількість вбудованих бібліотек для машинного навчання та нейронних мереж. Наприклад, Neural Network Toolbox надає розширені функції для створення та тренування нейронних мереж. Також доступні бібліотеки для обробки зображень, роботи з даними та інші завдання, пов'язані з нейронними мережами.

Загалом, MATLAB є потужним інструментом для розробки нейронних мереж. Вона поєднує легкість використання, продуктивність, широкий спектр функцій та підтримку громадськості. Це робить її популярним вибором серед дослідників та розробників у галузі машинного навчання та нейронних мереж.

2.2.4 Java

Java [63] - це мова програмування та платформа, яка також може бути використана для розробки нейронних мереж.

- популярність: Java є однією з найпопулярніших мов програмування у світі. Вона використовується в широкому спектрі додатків та проектів, що забезпечує наявність великої спільноти розробників та розширену документацію. Хоча частина що займається саме машинним навчанням менша;

- легкість використання: Java має простий синтаксис та багато інструментів, що полегшують розробку нейронних мереж. Вона має велику кількість бібліотек та фреймворків, які допомагають у створенні та тренуванні моделей нейронних мереж;

- швидкодія: Java відома своєю високою продуктивністю та швидкодією. Вона використовується в промислових системах та великих масштабних проектах, що вимагають ефективної обробки даних. Java також має підтримку

для оптимізованих бібліотек, що дозволяють виконувати обчислення з високою швидкістю;

- розширюваність: Java має велику кількість бібліотек та фреймворків для машинного навчання та нейронних мереж. Наприклад, бібліотека Deeplearning4j надає розширені функції для створення та тренування нейронних мереж. Java також підтримує використання бібліотек та фреймворків, розроблених для інших мов програмування;

- спільнота: Java має велику та активну спільноту розробників, яка активно працює у галузі машинного навчання та нейронних мереж. Це означає, що можна знайти підтримку, документацію, приклади коду та рішення проблем у спільноті;

- гнучкість: Java дозволяє розробляти нейронні мережі для різних платформ та систем. Вона підтримує розробку кросплатформових додатків, що дозволяє запускати нейронні мережі на різних операційних системах та пристроях;

- наявність бібліотек: Java має багато бібліотек для машинного навчання та нейронних мереж. Наприклад, Deeplearning4j, DL4J, JOONE та Neuroph - це деякі з популярних бібліотек, які допомагають у розробці нейронних мереж у Java.

Узагальнюючи, Java може бути потужним інструментом для розробки нейронних мереж, завдяки наявності багатофункціональних бібліотек, підтримці великої спільноти розробників та можливості розширення функціональності за допомогою сторонніх бібліотек та фреймворків.

2.2.5 R

R [64] - мова програмування та середовище аналізу даних, яке також може бути використане для розробки нейронних мереж. Ось деякі параметри, які слід враховувати:

– популярність: R є однією з найпопулярніших мов програмування для аналізу даних та статистики. Вона має широке застосування у наукових дослідженнях, академічній громадськості та промисловості. У R також існує активна спільнота розробників, яка постійно розширює функціональність та надає підтримку;

– легкість використання: R має простий та зрозумілий синтаксис, що дозволяє швидко створювати та експериментувати з нейронними мережами. Вона має багато вбудованих функцій та пакетів, які спрощують роботу з даними та моделями нейронних мереж;

– швидкодія: Хоча R може мати деякі обмеження щодо швидкості порівняно з іншими мовами програмування, вона пропонує оптимізовані пакети, такі як TensorFlow та Keras, що забезпечують ефективність обчислень нейронних мереж;

– розширюваність: У R існує велика кількість пакетів та бібліотек для нейронних мереж. Наприклад, 'tensorflow', 'keras', 'neuralnet' та 'caret' - це деякі популярні пакети, які надають потужні інструменти для створення та тренування нейронних мереж. R також підтримує інтеграцію з іншими мовами програмування, що розширює його можливості;

– спільнота: R має відносно велику та активну спільноту розробників та користувачів. Це означає, що ви можете знайти підтримку, документацію та приклади коду в Інтернеті. Також існують форуми та спеціалізовані конференції, де ви можете обмінюватися досвідом та задавати питання;

– гнучкість: R дає вам багато гнучких інструментів для налаштування та експериментування з нейронними мережами. Ви можете легко налаштувати параметри моделі, провести крос-валідацію та здійснити інші маніпуляції з даними;

– наявність бібліотек: R має велику кількість бібліотек для роботи з нейронними мережами. Ви можете використовувати популярні пакети, такі як 'tensorflow', 'keras', 'neuralnet' та інші, які надають розширені можливості для створення та навчання нейронних мереж у R.

Загалом, R є потужною мовою програмування для розробки нейронних мереж. Вона поєднує середню популярність, легкість використання, підтримку спільноти, широкий вибір бібліотек та гнучкість, що робить її привабливою для використання в області нейронних мереж та аналізу даних.

Порівняння мов програмування наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння мов програмування

Параметри порівняння	Python	MATLAB	C++/C	Java	R
Популярність	Висока	Висока	Середня	Середня	Середня
Легкість використання	Висока	Середня	Середня	Висока	Середня
Швидкодія	Середня	Середня	Висока	Середня	Середня
Розширюваність	Висока	Середня	Середня	Висока	Висока
Спільнота	Велика	Велика	Середня	Велика	Велика
Гнучкість	Висока	Середня	Середня	Висока	Середня
Наявність бібліотек	Висока	Висока	Середня	Середня	Середня

На підставі порівняння можна зробити висновок, що Python є найкращим вибором для розробки застосунків з використанням машинного навчання. Це пояснюється тим, що він має велику спільноту, широкий вибір бібліотек та фреймворків, добру підтримку та документацію, а також є досить простим у використанні.

2.3 Вибір фреймворку

2.3.1 TensorFlow

TensorFlow [59] - це програмна бібліотека з відкритим вихідним кодом від Google, яка використовується для програмування потоків даних у різних задачах. Це символічна математична бібліотека, яка використовує потік даних і диференційоване програмування для виконання різних завдань, орієнтованих на навчання і висновки глибоких нейронних мереж.

TensorFlow є найвідомішою бібліотекою для глибокого навчання за останні роки, і її в основному використовують науковці, стартапи та великі компанії. Google використовує TensorFlow майже у всіх щоденних продуктах Google, включаючи Gmail, Фото та Пошукову систему Google. TensorFlow був розроблений командою Google Brain і був оприлюднений у 2015 році. TensorFlow доступний на усіх обчислювальних платформах, включаючи мобільні. Завдяки його гнучкості можливо легко розгортати обчислення на різних платформах (CPU, GPU, TPU), від настільних комп'ютерів та серверів до мобільних і периферійних пристроїв.

TensorFlow можна використовувати в широкому спектрі мов програмування, включаючи Python, JavaScript, C++ та Java. TensorFlow слугує базовою платформою та бібліотекою для машинного навчання, а його API використовують Keras [65], щоб дозволити користувачам робити. TensorFlow.js - це бібліотека для машинного навчання на JavaScript, яка дозволяє розробникам розробляти моделі ML на JavaScript і використовувати ML безпосередньо в браузері або в Node.js.

2.3.2 Pytorch

PyTorch [60] - це фреймворк машинного навчання з відкритим вихідним кодом, заснований на мові програмування Python і бібліотеці Torch. Він був розроблений дослідницькою лабораторією штучного інтелекту Facebook (FAIR)

і забезпечує тензорні обчислення, глибоке навчання та будівельні блоки нейронних мереж.

PyTorch розроблений, щоб забезпечити хорошу гнучкість і високу швидкість для реалізації глибоких нейронних мереж. PyTorch відрізняється від інших фреймворків для глибокого навчання тим, що використовує динамічні обчислювальні графи. У той час як статичні обчислювальні графи (як ті, що використовуються в TensorFlow) визначаються до часу виконання, динамічні графи визначаються "на льоту" за допомогою прямих обчислень. PyTorch є пітонічним за своєю природою, що означає, що він слідує стилю кодування, який використовує унікальні можливості Python для написання читабельного коду.

Python також популярний завдяки використанню динамічних графіків обчислень. Це дозволяє розробникам, науковцям та налагоджувальникам нейронних мереж запускати та тестувати частину коду в режимі реального часу. PyTorch надає дві високорівневі функції: Тензорні обчислення (як NumPy) з сильним прискоренням за допомогою графічних процесорів (GPU) та глибокий блок нейронних мереж.

PyTorch часто порівнюють з TensorFlow, фреймворком для глибокого машинного навчання, розробленим Google. Хоча інтерфейс Python більш відшліфований і є основним напрямком розробки, PyTorch також має інтерфейс C++. PyTorch використовується для таких додатків, як комп'ютерний зір та обробка природної мови. PyTorch - це безкоштовне програмне забезпечення з відкритим вихідним кодом, випущене під модифікованою ліцензією BSD.

2.3.3 Keras

Keras [65] - це високорівневий нейромережевий API з відкритим вихідним кодом, написаний на мові Python, здатний працювати на платформах машинного навчання, таких як TensorFlow, Theano та CNTK. Він був розроблений з акцентом на забезпечення швидкого експериментування, що дозволяє дослідникам якомога швидше перейти від ідеї до результату.

Keras розроблений таким чином, щоб бути зручним у використанні, модульним і розширюваним, з акцентом на сучасне глибоке навчання. Він надає основні абстракції та будівельні блоки для розробки та доставки рішень машинного навчання з високою продуктивністю.

Keras - це високорівневий API платформи TensorFlow, який надає високопродуктивний інтерфейс для вирішення завдань машинного навчання. Основними структурами даних Keras є шари та моделі. Найпростішим типом моделі є Sequential модель - лінійний стек шарів. Для більш складних архітектур слід використовувати функціональний API Keras. Keras підтримує інші поширені утиліти, такі як відсікання, пакетна нормалізація та об'єднання.

Keras дозволяє користувачам перемикатися між різними бекендами і підтримує обчислення нейронних мереж у декількох бекендах. Keras підтримує фреймворки TensorFlow, Theano, PlaidML, MXNet [66] та CNTK. З цих п'яти фреймворків TensorFlow прийняв Keras як свій офіційний високорівневий API. Keras вбудований в TensorFlow і може бути використаний для швидкого виконання глибокого навчання, оскільки він надає вбудовані модулі для всіх нейромережових обчислень. Keras плавно працює як на CPU, так і на GPU, підтримує майже всі моделі нейронних мереж і є модульним за своєю природою, що робить його виразним, гнучким і придатним для інноваційних досліджень.

2.3.4 MXNet

Apache MXNet [66] - це програмний фреймворк для глибокого навчання з відкритим кодом, який використовується для навчання та розгортання глибоких нейронних мереж. Він масштабується, що дозволяє швидко навчати моделі, підтримує гнучку модель програмування і безліч мов програмування, включаючи C++, Python, Java, Julia, MATLAB, JavaScript, Go і R.

MXNet - це потужний програмний фреймворк для глибокого навчання з відкритим вихідним кодом, який допомагає розробникам створювати, тренувати і розгортати моделі глибокого навчання. MXNet - це гнучкий і масштабований

фреймворк глибокого навчання, який підтримує багато моделей і мов глибокого навчання. Це фреймворк загального призначення, який надає гібридний інтерфейс, що плавно переходить між імперативним режимом `Glueon eager` та символічним режимом, забезпечуючи гнучкість та швидкість.

MXNet можна розподіляти в динамічній хмарній інфраструктурі за допомогою розподіленого сервера параметрів, а з декількома графічними або центральними процесорами фреймворк наближається до лінійного масштабу. MXNet є обчислювально ефективним, масштабованим, портативним і швидким, а також забезпечує процвітаючу екосистему інструментів і бібліотек, які розширюють MXNet і дозволяють використовувати його в комп'ютерному зорі, НЛП, часових рядах і багато чому іншому.

MXNet використовується такими компаніями, як Amazon, і є обчислювально ефективним, масштабованим, портативним і швидким фреймворком, який забезпечує гнучку та ефективну бібліотеку для глибокого навчання.

MXNet має обмеження, зокрема, відносно невелику спільноту користувачів, які оновлюють її, що обмежує їхню здатність виправляти помилки, покращувати вміст і додавати нові функції. MXNet не така популярна, як TensorFlow і PyTorch, хоча її активно використовують такі компанії, як Amazon.

Порівняння фреймворків наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння фреймворків

Параметр	TensorFlow	PyTorch	Keras	MXNet
Популярність	Висока	Висока	Висока	Висока
Легкість використання	Середня	Висока	Висока	Середня
Швидкодія	Висока	Висока	Середня	Висока
Розширюваність	Висока	Висока	Середня	Висока
Ком'юніті	Велика	Велика	Велика	Середнє
Гнучкість	Середня	Висока	Середня	Висока

За результатами порівняння можна зробити висновок, що PyTorch є найкращим вибором серед фреймворків для написання нейронних мереж. Він має високу популярність та широку ком'юніті, що забезпечує підтримку та розвиток фреймворку. PyTorch також відрізняється високою легкістю використання, що дозволяє швидко створювати та експериментувати з моделями. Він має високу швидкодію та розширюваність, що дозволяє працювати з великими обсягами даних та різними моделями. Наявність багатой бібліотеки ресурсів та зручний інтерфейс для навчання нейронних мереж робить PyTorch найкращим вибором для розробки машинного навчання з використанням Python.

2.4 Вибір середи розробки

2.4.1 Jupiter Notebook

Jupyter Notebook [67] - це популярна платформа з відкритим вихідним кодом для візуалізації, очищення та перетворення даних, а також машинного навчання (ML). Вона надає інтерактивне обчислювальне середовище для розробки додатків для науки про дані. Блокноти Jupyter поєднують програмний код, результати обчислень і пояснювальний текст в одному документі. Блокноти дозволяють редагувати та виконувати код у браузері, а також відображати результати обчислень.

Блокнот зберігається з розширенням `.ipynb`. Jupyter підтримує понад 40 мов програмування, включаючи Python, R, Julia та Scala.

Загалом, Jupyter Notebook - це універсальний інструмент для машинного навчання, який дозволяє користувачам налаштовувати та організовувати робочі процеси в науці про дані, наукових обчисленнях та машинному навчанні.

2.4.2 PyCharm

PyCharm [68] - це популярне інтегроване середовище розробки (IDE) для Python, яке широко використовується в машинному навчанні. Воно пропонує ряд функцій, які роблять його потужним інструментом для розробки та розгортання моделей машинного навчання.

PyCharm надає інтелектуальний редактор коду, який підтримує підсвічування синтаксису, завершення коду та підсвічування помилок. Він також має ряд інструментів для аналізу коду, які можуть допомогти вам виявити потенційні проблеми у коді. PyCharm дозволяє запускати ваш код безпосередньо з IDE. Він також підтримує запуск коду у віртуальному середовищі, що може бути корисним для управління залежностями і забезпечення узгодженої роботи вашого коду в різних середовищах.

PyCharm має вбудовану підтримку Jupyter Notebook, що дозволяє створювати і запускати блокноти Jupyter безпосередньо з IDE. Це може бути корисно для дослідження даних і створення прототипів моделей машинного навчання.

PyCharm має ряд функцій для спільної роботи, які дозволяють легко працювати з іншими над проектами машинного навчання. Він підтримує системи контролю версій, такі як Git, і має вбудовані інструменти для перегляду коду та парного програмування.

Загалом, PyCharm - це популярний інструмент для машинного навчання, який пропонує широкий спектр функцій для підтримки розробки та розгортання моделей машинного навчання. Підтримка бібліотек машинного навчання, Jupyter Notebook та можливості співпраці роблять його популярним серед експертів з машинного навчання.

2.4.3 VS Code

Visual Studio Code (VS Code) [69] - це безкоштовний редактор коду з відкритим вихідним кодом, який широко використовується в машинному навчанні. VS Code надає розумний редактор коду, який підтримує підсвічування синтаксису, завершення коду та підсвічування помилок. Він також має ряд інструментів для аналізу коду, які можуть допомогти вам виявити потенційні проблеми у вашому коді.

VS Code має вбудовану підтримку Jupyter Notebook, що дозволяє створювати і запускати блокноти Jupyter безпосередньо з редактора. Це може бути корисно для дослідження даних і створення прототипів моделей машинного навчання.

VS Code має величезну колекцію розширень, спеціально розроблених для задач машинного навчання. Ці розширення забезпечують підтримку популярних фреймворків машинного навчання, таких як TensorFlow, PyTorch та scikit-learn. Вони пропонують такі функції, як фрагменти коду, візуалізація моделей та

інтеграція з інструментами дослідження та попередньої обробки даних. Наявність цих розширень робить VS Code потужним інструментом для розробки машинного навчання.

Загалом, VS Code - це потужний інструмент для машинного навчання, який пропонує широкий спектр функцій для підтримки розробки та розгортання моделей машинного навчання. Підтримка бібліотек машинного навчання, Jupyter Notebook, спільної роботи та розширень робить його популярним серед експертів з машинного навчання.

2.4.4 Google Colab

Google Colab [70] - це хмарне середовище Jupyter-блокнотів, яке дозволяє користувачам писати та виконувати код на Python через браузер. Це популярний інструмент для задач машинного навчання, який пропонує цілий ряд функцій для підтримки цих задач.

Google Colab - це хмарний інструмент, що означає, що користувачі можуть розпочати кодування моделей машинного навчання за допомогою браузера. Він безкоштовний з обмеженими ресурсами, але користувачі не повинні розраховувати на те, що зможуть зберігати свої моделі штучного інтелекту або машинного навчання в безкоштовній інфраструктурі Colab нескінченно довго.

Інтерфейс, схожий на Jupyter Notebook. Colab є онлайн-представництвом Jupyter Notebook і дозволяє користувачам писати і виконувати довільні програмні коди на Python просто за допомогою веб-браузера. Він також підтримує розширений текст, зображення, HTML, LaTeX тощо.

Google Colab надає доступ до апаратних прискорювачів, таких як GPU і TPU, які можна використовувати для прискорення завдань машинного навчання. Google Colab дозволяє користувачам легко ділитися своїми ноутбуками з колегами або друзями, що дозволяє їм співпрацювати над проектами машинного навчання. Блокноти Colab зберігаються в обліковому записі користувача на

Google Диску, що полегшує доступ до них і спільне використання на різних пристроях.

Загалом, Google Colab - це потужний інструмент для машинного навчання, який пропонує широкий спектр функцій для підтримки розробки та розгортання моделей машинного навчання. Його хмарний інтерфейс, інтерфейс, схожий на Jupyter Notebook, апаратні прискорювачі, функції співпраці та інтеграція з Google Drive роблять його популярним вибором серед експертів з машинного навчання.

Порівняння середовищ розробки наведено у таблиці 2.3.

Таблиця 2.3 – Порівняння середовищ розробки

	Jupyter Notebook	PyCharm	VS Code	Google Colab
Інтерфейс користувача	Веб-інтерфейс	Локальний запуск	Локальний запуск	Веб-інтерфейс
Інтерактивні віджети	так	так	ні	так
Підтримка IntelliSense	ні	так	так	так
Відладка	обмежена	так	так	так
підтримка хмарних технологій	обмежена	так	ні	так
Розгортання моделей	обмежено	так	ні	обмежено
Обчислювальні ресурси	хмарні	локальні	локальні	хмарні

За результатами порівняння, кожне з середовищ має свої переваги та недоліки. Jupyter Notebook відзначається легкістю використання, великою розширюваністю та потужною візуалізацією даних, але може бути трохи повільним у виконанні обчислень. PyCharm має широкий функціонал, добру

підтримку мови Python та інтеграцію з різними фреймворками. VS Code також надає розширення та настройки, що дозволяють розробникам адаптувати середовище до власних потреб. Але ключова перевага Google Colab заключається в можливості безкоштовного використання хмарних ресурсів.

2.5 Підсумки за розділом 2

Під час розробки цього розділу була проведена проектування рівневої архітектури та структури майбутнього програмного забезпечення. Було визначено структуру рівнів та модулів для майбутнього додатку. Після детального порівняння різних мов програмування було прийнято рішення використовувати Python, який має низький поріг входження і широкий набір інструментів для розробки. Для фреймворку машинного навчання було обрано PyTorch, враховуючи його популярність та потужні можливості. В якості середовища розробки було вибрано Google Colab, яке забезпечує зручну та масштабовану робочу платформу для виконання проекту.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Основні рішення щодо реалізації

В результаті аналізу предметної області, описаного в попередніх розділах було виділено ряд сутностей: “CycleGANModel”, “ImagePool”, “GANLoss”, “UnetGenerator”, “UnetSkipConnectionBlock”, “NLayerDiscriminator”, “PT2SR”, “ConvolutionalBlock”, “SubPixelConvolutionalBlock”, “DenseResidualBlock”, “GramLoss”.

Всі згадані елементи були втілені у вигляді окремих класів. Це означає, що розроблена програма відповідає концепції об'єктно-орієнтованого програмування (ООП), де всі складові є взаємодіючими об'єктами, кожен з яких є екземпляром конкретного класу.

Структуру класів програми, їхні методи та атрибути, представлено на рис. 3.1.

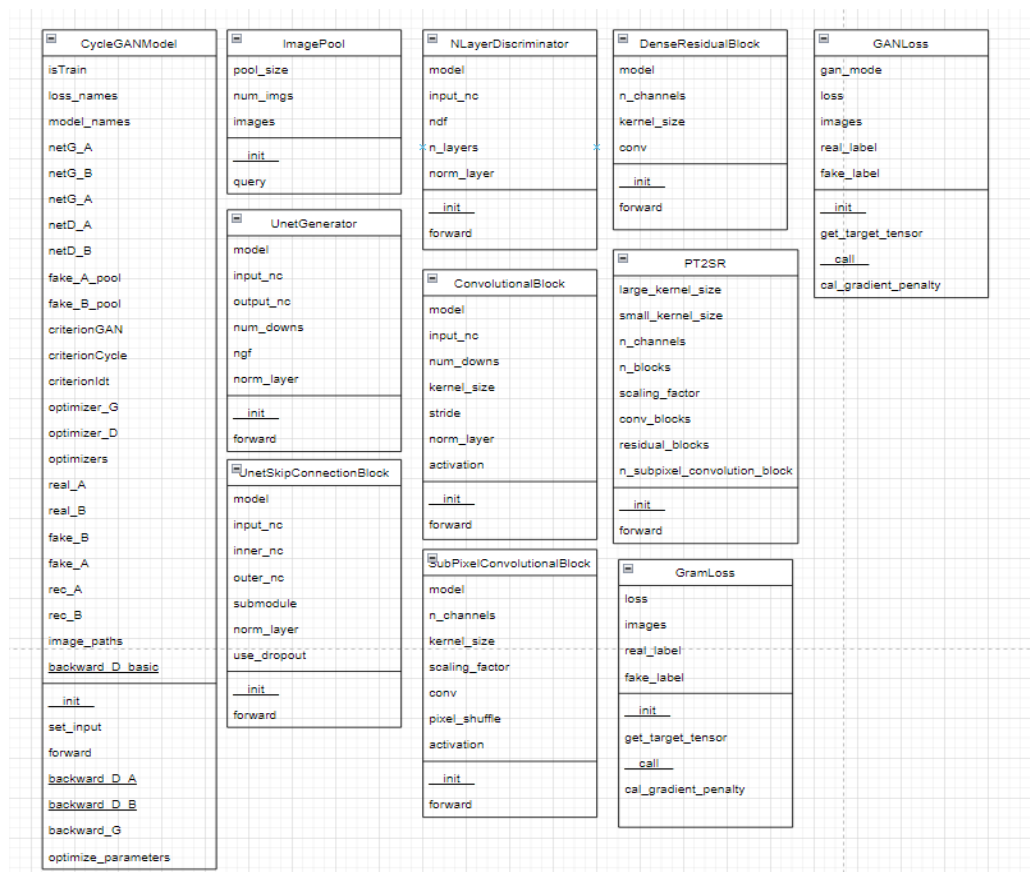


Рисунок 3.1 – Структура класів програми

На рисунку 3.2 представлено схему алгоритму програми.

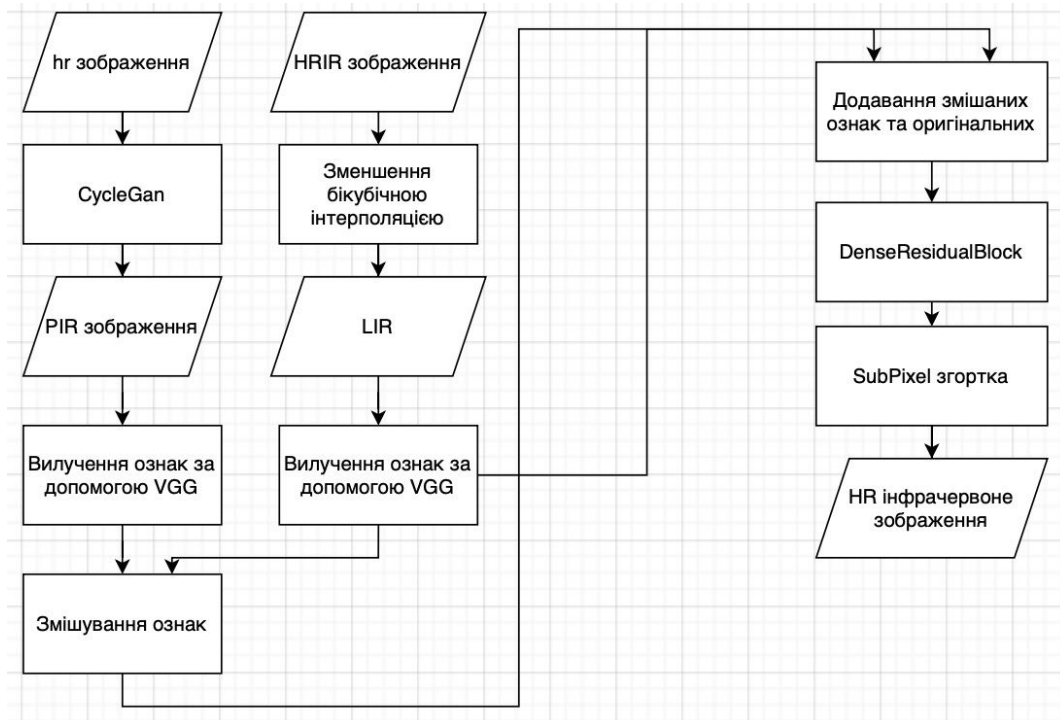


Рисунок 3.2 – Схема алгоритму програми

Спочатку високороздільне електрооптичне зображення поступає до класу CycleGANModel. Цей клас є реалізацією нейронної мережі для зміни стилю зображення, в нашому випадку перетворення з електрооптичного до псевдоінфрачервоного задля наступної передачі високочастотної інформації до масштабованого інфрачервоного зображення.

Клас CycleGANModel спирається в своїй роботі на наступні класи: “ImagePool”, “GANLoss”, “UnetGenerator”, “UnetSkipConnectionBlock”, “NLayerDiscriminator”

ImagePool це допоміжний клас котрий являє собою буфер раніше згенерованих зображень. Завдяки ньому є змога оновлювати дискримінатори використовуючи історію згенерованих зображень, а не ті що створюються останніми генераторами.

Через обмеженість рахувальних можливостей було використано Unet генератор, а не ResNet. Клас UnetGenerator спирається в свою чергу на клас

UnetSkipConnectionBlock, що є його підмодулем з пропускним зв'язком. NLayerDiscriminator є класом дискримінатора.

Низькороздільне інфрачервоне зображення, або як у випадку наявності в датасеті тільки високороздільних інфрачервоних зображень, котрі зменшені за допомогою бікубічної інтерполяції подаються на два напрямки: змішення ознак з псевдоінфрачервоним зображенням, та на подальше конкатування.

За допомогою предтренуваної на датасеті “ImageNet” VGG, щоб послабити вплив контрасту і яскравості при вимірюванні подібності зображень і посилити інформацію про структуру і текстуру, порівнюється схожість у глибокому просторі ознак, що відрізняється від традиційного методу використання вихідних пікселів. Після заміни частини ознак вони конкатануються з оригінальними ознаками інфрачервоного зображення.

На цьому етапі “PT2SR” спирається на наступні класи: “ConvolutionalBlock”, “SubPixelConvolutionalBlock”, “DenseResidualBlock”, “GramLoss”.

Об'єднані ознаки потрапляють до послідовності DenseResidualBlock, котрі складаються з декількох ConvolutionalBlock виходи котрих конкатануються між собою для отримання більшої кількості інформації. Між блоками використовується пропускний зв'язок, щоб на великій глибині не відбувалося затухання градієнту.

Далі отриманий тензор знову конкатанується з оригінальними ознаками, та подається в SubPixelConvolutionalBlock де і відбувається масштабування зображення.

В якості функції втрат використовується GramLoss [71], формула (3.1). Для того, щоб зберегти просторову структуру зображень низької роздільної здатності, покращити візуальну якість відновлених зображень та використати інформацію про особливості псевдоінфрачервоних зображень, функція втрат, що використовується в цій моделі, поєднує втрати при реконструкції l_{mse} , втрати при сприйнятті l_{per} та втрати текстури l_{tex} . l_{mse} - це функція втрат, заснована на середньоквадратичній похибці, яка відображає середньоквадратичну похибку

між оригінальним зображенням і реконструйованим зображенням з надвисокою роздільною здатністю.

$$l_{tex} = \sum_L \sigma_L ||Gram((\phi_L(I^{SR}) * S_L) - Gram(M_L * S_L))||_2^2 \quad (3.1)$$

де L - L -й рівень обміну текстурами;

σ_L - коефіцієнт нормалізації;

$Gram(*)$ - матриця Грама (3.2);

$\phi_L(*)$ - карти особливостей L -го рівня;

S_L - вагова матриця

$$Gram(\alpha_1, \alpha_2, \dots, \alpha_k) = \begin{pmatrix} (\alpha_1, \alpha_1) & \cdots & (\alpha_1, \alpha_k) \\ \vdots & \ddots & \vdots \\ (\alpha_k, \alpha_1) & \cdots & (\alpha_k, \alpha_k) \end{pmatrix} \quad (3.2)$$

Де (α_i, α_k) - скалярний добуток векторів α_i та α_k .

Вона отримує менше "покарання" під час передачі ознак. Таким чином, мережа може адаптивно переносити відповідну інформацію про ознаки з псевдоінфрачервоного зображення на реконструйоване зображення.

3.2 Висновки за розділом 3

Було розроблено програмне забезпечення для побудови та навчання систем з метою супермасштабування інфрачервоних зображень. Для реалізації відповідних функціональних вимог були обрані програмні механізми у вигляді методів та функцій. Представлена архітектура програмного забезпечення підтвердила успішні результати масштабування та розгортання на персональних комп'ютерах.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

У попередніх розділах були розглянуті питання розробки програмного забезпечення, яке забезпечує супермасштабування інфрачервоних зображень на основі електрооптичних зображень високої роздільної здатності.

Цей розділ буде присвячений опису процесу тестування розробленого програмного забезпечення. Будуть визначені метрики, що використовуються для оцінки точності класифікації. Окремо будуть проаналізовані отримані результати тестування.

4.1 Вхідні дані для тестування

У експериментальній частині для навчання мережевої моделі CycleGAN було використано набір даних LLVIP [72], який містить багатоспектральні пішохідні зображення. Для навчання моделі алгоритму PT2SR використовувався набір даних FLIR ADAS [73]. Обидва набори даних містять видимі та інфрачервоні зображення вуличних сцен з об'єктами, такими як пішоходи, автомобілі, дерева, будівлі та інші.

Алгоритм PT2SR випадковим чином вибирав 6053 видимі та інфрачервоні зображення з набору даних FLIR ADAS. Потім він використовував попередньо навчену модель CycleGAN для генерації відповідних псевдоінфрачервоних зображень високої якості з видимих зображень. З інфрачервоних зображень вирізалися блоки розміром 160×160 , які потім зменшувалися для отримання інфрачервоних зображень низької роздільної здатності (LR). Псевдоінфрачервоні зображення також вирізалися відповідно до відповідних блоків зображень. У результаті було зібрано 18 159 парних блоків зображень, які стали навчальною вибіркою. Інфрачервоні зображення високої роздільної здатності (HR) у парних зображеннях використовувалися лише для

генерації інфрачервоних зображень низької роздільної здатності (LR) і не брали участі в навчанні моделі. Цей навчальний набір даних отримав назву "PFLIR".

Подібність між інфрачервоними та псевдоінфрачервоними зображеннями низької роздільної здатності може впливати на результати реконструкції з високою роздільною здатністю. Тому, для оцінки алгоритму, було важливо мати псевдоінфрачервоні зображення з різним ступенем подібності до LR-зображень, які використовуються для реконструкції з високою роздільною здатністю. Для оцінки здатності навчальної моделі до узагальнення було проведено тестування на наборі даних, FLIR ADAS та мультиспектральному тестовому наборі LLVIP.

4.2 Метрики оцінювання

Метрики в машинному навчанні і обробці даних використовуються для кількісної оцінки моделей, алгоритмів та систем. Вони дозволяють оцінити рівень виконання, ефективність та якість роботи системи, що дозволяє робити об'єктивні порівняння між різними підходами.

Основна мета використання метрик полягає в тому, щоб мати кількісні показники, які допомагають зрозуміти, наскільки добре модель або алгоритм вирішує конкретну задачу. Оцінка на основі метрик допомагає визначити потенційні проблеми, покращити результати та зробити вибір між різними підходами.

4.1.1 PSNR

PSNR (Peak Signal-to-Noise Ratio) [71] - це метрика, яка використовується для вимірювання якості реконструйованого зображення порівняно з оригінальним зображенням. Вона широко застосовується в області обробки зображень та компресії даних.

PSNR вимірює ступінь спотворення або шуму в реконструйованому зображенні, порівнюючи його з оригінальним зображенням. Вищі значення PSNR вказують на кращу якість реконструкції.

Формула для обчислення PSNR виглядає наступним чином (4.1):

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \quad (4.1)$$

Де MAX - максимальне можливе значення пікселя в зображенні (наприклад, 255 для 8-бітного зображення);

MSE - середньоквадратична помилка;

Середньоквадратична помилка представляє собою середнє значення квадратів різниць між відповідними пікселями оригінального та реконструйованого зображень.

PSNR виражається в одиницях децибел (dB) і позначає ступінь різниці між сигналом (оригінальним зображенням) та шумом (реконструйованим зображенням). Вище значення PSNR вказує на меншу помилку та кращу якість реконструкції, оскільки воно означає, що сигнал (оригінальне зображення) перевищує шум (реконструйоване зображення) в більшій мірі.

4.1.2 SSIM

SSIM (Structural Similarity Index) [71] є метрикою, яка використовується для вимірювання структурної схожості між двома зображеннями. Вона оцінює, наскільки добре зберігаються структурні деталі, колір та контраст при порівнянні зображень.

SSIM зазвичай застосовується у сферах обробки зображень, комп'ютерного зору та оцінки якості зображень. Вона дозволяє отримати числову оцінку подібності між двома зображеннями, де значення SSIM

знаходиться в діапазоні від 0 до 1. Більш високе значення SSIM вказує на більшу схожість між зображеннями.

Формула SSIM має наступний вигляд (4.2):

$$SSIM(x, y) = \frac{(\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (4.2)$$

де x і y - порівнювані зображення;

μ_x і μ_y - середні значення пікселів для x та y відповідно

σ_x і σ_y - стандартні відхилення пікселів для x та y відповідно

σ_{xy} - коваріація пікселів між x та y

c_1 і c_2 - сталі, які використовуються для стабільності поділу

Формула розраховує структурну схожість на основі порівняння середніх значень, дисперсій та коваріації пікселів у зображеннях. Чим ближче значення SSIM до 1, тим більша схожість між зображеннями.

SSIM дозволяє виявляти втрату якості, що може виникнути при стисканні, шуму, артефактів або інших артефактах обробки зображень. Використовуючи SSIM, можна порівнювати різні алгоритми чи методи обробки та вибирати той, який зберігає більшу структурну схожість із вихідним зображенням.

4.3 Результати

4.3.1 Результати переносу стилю

Результат переносу стилю наведено на рисунку 4.1.



Рисунок 4.1 – Результат переносу стилю

4.3.2 Результати супермасштабування

Результати супермасштабування наведені на рисунку 4.2.

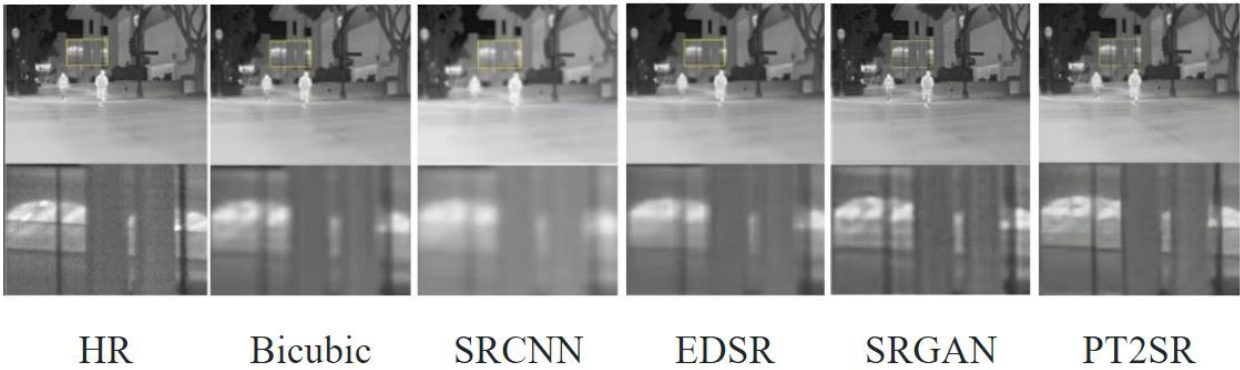


Рисунок 4.2 – Результати супермасштабування

4.3.3 Результати тестування

Порівняння результатів тестування із іншими підходами наведено в таблиці 4.1.

Таблиця 4.1 – Загальні результати тестування

Набір даних	Показники	Bicubic	SRCNN	EDSR	SRGAN	PT2SR
PFLIR	PSNR/SSIM	26.59/0.753	22.69/0.763	33.04/0.926	30.34/0.846	34.23/0.939
	Час (сек)	0.032	0.66	0.71	1.03	1.28
LLVIP	PSNR/SSIM	28.31/0.762	24.58/0.870	29.81/0.831	27.58/0.877	29.79/0.901
	Час (сек)	0.061	0.94	1.07	1.76	2.25

ВИСНОВКИ

Під час виконання дипломної роботи, було реалізовано метод супермасштабування інфрачервоних зображень за зразком електрооптичного зображення високої роздільної здатності.

Для досягнення поставлених цілей були вирішені наступні завдання:

- ознайомлення та аналіз проблематики супермасштабування інфрачервоних зображень, включаючи вивчення мови програмування Python, фреймворку машинного навчання PyTorch та середовищ розробки для цих мов та фреймворку;

- Розроблення програмного забезпечення для реалізації та тестування запропонованого методу;

- запропонований метод використовує сучасні підходи до супермасштабування та базується на архітектурі моделей. Розроблений програмний комплекс має практичне застосування та готовий до використання;

Під час розробки програми та аналізу розробленого програмного комплексу виникли ідеї щодо його вдосконалення, такі як оптимізація навчального набору даних та використання метанавчання. Ці нові можливості можуть розширити область вирішення практичних задач методом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Isaac J. S. Super resolution techniques for medical image processing [Text] / Jithin Saji Isaac, Ramesh Kulkarni // international conference on technologies for sustainable development (ICTSD) : Materials from international conference, 4–6 February 2015 : proceedings. – Mumbai, India : IEEE, 2015. – P. 1-6.
2. Offroy, M. Increasing the spatial resolution of near infrared chemical images (NIR-CI): the super-resolution paradigm applied to pharmaceutical products [Text] / M. Offroy, Y. Roggo, L. Duponchel // Chemometrics and intelligent laboratory systems. – 2012. – Vol. 117. – P. 183–188.
3. Optimization of advanced live-cell imaging through red/near-infrared dye labeling and fluorescence lifetime-based strategies [Text] / [M. Bénard, D. Schapman, C. Chamot et al.] // International journal of molecular sciences. – 2021. – Vol. 22, № 20. – P. 11092.
4. Bodensteiner C. Multispectral matching using conditional generative appearance modeling [Text] / C. Bodensteiner, S. Bullinger, M. Arens // 2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS) : Materials from international conference, 27–30 November 2018 : proceedings. – Auckland, New Zealand : IEEE, 2018. – P. 1-6.
5. A thermal infrared pedestrian-detection method for edge computing devices [Text] / [S. You, Y. Ji, S. Liu et al.] // Sensors. – 2022. – Vol. 22, № 17. – P. 6710.
6. Szankin M. Influence of thermal imagery resolution on accuracy of deep learning based face recognition [Text] / M. Szankin, A. Kwasniewska, J. Ruminski // 2019 12th international conference on human system interaction (HIS) : Materials from international conference, 25–27 June 2019 : proceedings. – Richmond, VA, USA. : IEEE, 2019. – P. 1-6.
7. A depth estimation framework based on unsupervised learning and cross-modal translation [Text] / [J. Shen, K. Wang, K. Yang et al.] // Target and background signatures V, Strasbourg, France, 9–12 September 2019 : proceedings. – Strasbourg, France : SPIE, 2019. – P. 1-6.

8. The advantage of multispectral images in fruit quality control for extra virgin olive oil production [Text] / [D. M. M. Gila, J. P. N. Soto, S. S. Martínez et al.] // Food analytical methods. – 2021. – Vol. 15, № 17. – P. 75-84.
9. Rock slope monitoring using drone based multispectral and thermal images [Text] / [M. L. M. Yaacob, A. W. Rasib, A. S. A. Rashid et al.] // IOP conference series: earth and environmental science. – 2020. – Vol. 540. – P. 12024.
10. Estimation of peanut leaf area index from unmanned aerial vehicle multispectral images [Text] / [H. Qi, B. Zhu, Z. Wu et al.] // Sensors. – 2020. – Vol. 20, № 23. – P. 6732.
11. Lin C.-H. A novel convolutional neural network architecture of multispectral remote sensing images for automatic material classification [Text] / C.-H. Lin, T.-Y. Wang // Signal processing: image communication. – 2021. – Vol. 97. – P. 116329.
12. Can the structure similarity of training patches affect the sea surface temperature deep learning super-resolution? [Text] / [B. Ping, Y. Meng, C. Xue et al.] // Remote sensing. – 2021. – Vol. 13, № 18. – P. 3568.
13. Geometry and adjacency effects in urban land surface temperature retrieval from high-spatial-resolution thermal infrared images Text] / [S. Chen, H. Ren, X. Ye et al.] // Remote sensing of environment. – 2021. – Vol. 262. – P. 112518.
14. Super-resolution imaging of the protoplanetary disk HD 142527 using sparse modeling [Text] / [M. Yamaguchi, K. Akiyama, T. Tsukagoshi et al.] // The astrophysical journal. – 2020. – Vol. 895, № 2. – P. 84
15. Duchon C. E. Lanczos filtering in one and two dimensions [Text] / C. E. Duchon // Journal of applied meteorology. – 1979. – Vol. 18, № 8. – P. 1016–1022.
16. Freedman G. Image and video upscaling from local self-examples [Text] / G. Freedman, R. Fattal // ACM transactions on graphics. – 2011. – Vol. 30, № 2. – P. 1–11.
17. Kwang I. K.. Single-Image super-resolution using sparse regression and natural image prior [Text] / K. I. Kwang, Y. Kwon // IEEE transactions on pattern analysis and machine intelligence. – 2010. – Vol. 32, № 6. – P. 1127–1133

18. Glasner D. Super-resolution from a single image [Text] / D. Glasner, S. Bagon, M. Irani // 2009 IEEE 12th international conference on computer vision (ICCV) : Materials from international conference, 2 October 2009 : proceedings. – Kyoto, Japan: IEEE, 2015. – P. 349-356.
19. Image super-resolution via sparse representation [Text] / [J. Yang, J. Wright, T. S. Huang et al.] // IEEE transactions on image processing. – 2010. – Vol. 19, № 11. – P. 2861–2873.
20. Dong C. Learning a deep convolutional network for image super-resolution / C. Dong, C. C. Loy, K. He // Book. – Cham : Springer International Publishing, 2014. – P. 184-199.
21. Generative adversarial networks [Text] / [I. Goodfellow, J. Pouget-Abadie, M. Mirza et al.] // Communications of the ACM. – 2020. – Vol. 63, № 11. – P. 139-144.
22. Photo-Realistic single image super-resolution using a generative adversarial network [Text] / [C. Ledig, L. Theis, F. Huszár et al.] // 2017 IEEE conference on computer vision and pattern recognition (CVPR) : Materials from international conference, 21–26 July 2017: proceedings. – Honolulu, HI, USA: IEEE, 2015. – P. 105-114.
23. MLWAN: multi-scale learning wavelet attention module network for image super resolution [Text] / [J. Ma, X. Han, X. Zhang et al.] // Sensors. – 2022. – Vol. 22, № 23. – P. 9110.
24. Multi-window back-projection residual networks for reconstructing COVID-19 CT super-resolution images [Text] / [D. Qiu, Y. Cheng, X. Wang et al.] // Computer methods and programs in biomedicine. – 2021. – Vol. 200. – P. 105934.
25. Sajjadi M. S. M. EnhanceNet: single image super-resolution through automated texture synthesis [Text] / Sajjadi M. S. M., B. Scholkopf, M. Hirsch // 2017 IEEE international conference on computer vision (ICCV) : Materials from international conference, 22–29 October 2017 : proceedings. – Venice, Italy : IEEE, 2017. – P. 4501-4510.

26. Bulat A. Super-FAN: integrated facial landmark localization and super-resolution of real-world low resolution faces in arbitrary poses with gans [Text] / Bulat A., Tzimiropoulos G. // 2018 IEEE/CVF conference on computer vision and pattern recognition (CVPR) : Materials from international conference, 18–23 June 2018 : proceedings. – Salt Lake City, UT, USA : IEEE, 2018. – P. 109-117.

27. Enhanced deep residual networks for single image super-resolution [Text] / [B. Lim, S. Son, H. Kim et al.] // 2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW) : Materials from international conference, 21–26 July 2017 : proceedings. – Honolulu, HI, USA : IEEE, 2017. – P. 1132-1140.

28. Johnson J. Perceptual losses for real-time style transfer and super-resolution / J. Johnson, A. Alahi, L. Fei-Fei // Book. – Cham : Springer International Publishing, 2016. – P. 694-711.

29. Gao Z. Maritime infrared image super-resolution using cascaded residual network and novel evaluation metric [Text] / Z. Gao, J. Chen // IEEE access. – 2022. – Vol. 10. – P. 17760–17767.

30. Park. S. C. Super-resolution image reconstruction: a technical overview [Text] / S. C. Park, M. K. Park, M. G. Kang // IEEE signal processing magazine. – 2003. – Vol. 20, № 3. – P. 21–36.

31. Milton L. The method of lens design / L. Milton // Book. – Boca Raton : CRC Press, 2007. – P. 1—39.

32. Compressive hyperspectral imaging and super-resolution [Text] / [H. Yuan, F. Yan, X. Chen et al.] // 2018 IEEE 3rd international conference on image, vision and computing (ICIVC) : Materials from international conference, 27–29 June 2018: proceedings. – Chongqing, China: IEEE, 2018. – P. 618-623.

33. Rabal H. J. Dynamic laser speckle and applications / H. J. Rabal, R. A. Braga – Boca Raton : CRC Press, 2009. – P. 282.

34. Mas A. Study of asymmetric or decentered multi-view designs for uncooled infrared imaging applications [Text] / A. Mas, G. Druart, F. D. L. Barrière // Optics express. – 2020. – Vol. 28, № 23. – P. 35216-35230.

35. Wang, S.-P. Stripe noise removal for infrared image by minimizing difference between columns [Text] / S.-P. Wang // Infrared physics & technology. – 2016. – Vol. 77. – P. 58–64.

36. Super resolution infrared camera using single carbon nanotube photodetector [Text] / [L. Chen, Z. Zhou, N. Xi et al.] // 2014 IEEE Sensors : Materials from international conference, 2–5 November 2014: proceedings. – Valencia, Spain : IEEE, 2014. – P. 1038-1041.

37. Near-Infrared super resolution imaging with metallic nanoshell particle chain array [Text] / [W. Kong, P. Cao, X. Zhang et al.] // Plasmonics. – 2013. – Vol. 8, № 2. – P. 835–842.

38. Sun M. A sur-pixel scan method for super-resolution reconstruction [Text] / M. Sun, K. Yu // Optik. – 2013. – Vol. 124, № 24. – P. 6905–6909.

39. Versatile near-infrared super-resolution imaging of amyloid fibrils with the fluorogenic probe CRANAD-2 [Text] / [J. Torra, F. Viela, D. Megías et al.] // Chemistry – A european journal. – 2022. – Vol. 28, № 19. – P. 19232

40. Lukose J. Optical technologies for the detection of viruses like COVID-19: progress and prospects [Text] / J. Lukose, S. Chidangil, S. D. George // Biosensors and bioelectronics. – 2021. – Vol. 178. – P. 113004.

41. A bimolecular fluorescence complementation flow cytometry screen for membrane protein interactions [Text] / [F. Schmitz, J. Glas, R. Neutze et al.] // Scientific reports. – 2021. – Vol. 11, № 1. – P. 19232.

42. Repurposing a photosynthetic antenna protein as a super-resolution microscopy label [Text] / [S. F. H. Barnett, A. Hitchcock, A. K. Mandal et al.] // Scientific reports. – 2017. – Vol. 7, № 1. – P. 16807.

43. Bodensteiner C. Multispectral matching using conditional generative appearance modeling [Text] / C. Bodensteiner, S. Bullinger, M. Arens // 2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS) : Materials from international conference, 27–30 November 2018 : proceedings. – Auckland, New Zealand: IEEE, 2018. – P. 1-6.

44. Lee I.-H. Super-resolution thermal generative adversarial networks for infrared image enhancement [Text] / I.-H. Lee, W.-Y. Chung, C.-G. Park // Journal of institute of control, robotics and systems. – 2022. – Vol. 28, № 2.

45. Image super-resolution using deep convolutional networks [Text] / [C. Dong, C. C. Loy, K. He et al.] // IEEE transactions on pattern analysis and machine intelligence. – 2016. – Vol. 38, № 2. – P. 295–307.

46. Image processing - nearest neighbour interpolation [Electronic resource]. – Mode of access: https://www.giassa.net/?page_id=207

47. Bilinear interpolation [Electronic resource]. – Mode of access: <https://x-engineer.org/bilinear-interpolation>

48. Image processing - generalized bicubic interpolation [Electronic resource]. – Mode of access: https://www.giassa.net/?page_id=371

49. Transposed Convolutional Layer [Electronic resource]. – Mode of access: <https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11>

50. Real-Time single image and video super-resolution using an efficient sub-pixel convolutional neural network [Text] / [W. Shi, J. Caballero, F. Huszár et al.] // 2016 IEEE conference on computer vision and pattern recognition (CVPR) : Materials from international conference, 27–30 June 2016 : proceedings. – Las Vegas, NV, USA : IEEE, 2016. – P. 1874-1883.

51. Meta learning convolution [Electronic resource]. – Mode of access: https://www.academia.edu/80130818/Meta_Learning_with_Temporal_Convolutions

52. SwinIR: image restoration using swin transformer [Electronic resource] / [J. Liang, J. Cao, J. Sun et al.] // 2021 IEEE/CVF international conference on computer vision workshops (ICCVW) : Materials from international conference, 11–17 October 2021 : proceedings. – Montreal, BC, Canada : IEEE, 2021. – P. 1833-1844.

53. Boominathan V. Improving resolution and depth-of-field of light field cameras using a hybrid imaging system [Text] / V. Boominathan, K. Mitra, A. Veeraraghavan // 2014 IEEE international conference on computational

photography (ICCP) : Materials from international conference, 2–4 May 2014 : proceedings. – Santa Clara, CA, USA : IEEE, 2014. – P. 1-10.

54. Dong C. Learning a deep convolutional network for image super-resolution / C. Dong, C. C. Loy, K. He // Book. – Cham : Springer International Publishing, 2014. – P. 184-199.

55. CrossNet: an end-to-end reference-based super resolution network using cross-scale warping [Text] / [H. Zheng, M. Ji, H. Wang et al.]. – Cham : Springer International Publishing, 2018. – P. 87–104.

56. Image super-resolution by neural texture transfer [Text] / [Z. Zhang, Z. Wang, Z. Lin et al.] // 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR) : Materials from international conference, 15–20 June 2019 : proceedings. – Long Beach, CA, USA : IEEE, 2019. – P. 7974-7983.

57. Unpaired image-to-image translation using cycle-consistent adversarial networks [Text] / [J.-Y. Zhu, T. Park, P. Isola et al.] // 2017 IEEE international conference on computer vision (ICCV) : Materials from international conference, 22–29 October 2017 : proceedings. – Venice, Italy : IEEE, 2017. – P. 2242-2251.

58. Python [Electronic resource]. – Mode of access: <https://www.python.org>

59. TensorFlow [Electronic resource]. – Mode of access: <https://www.tensorflow.org>

60. PyTorch [Electronic resource]. – Mode of access: <https://pytorch.org>

61. C++ [Electronic resource]. – Mode of access: <https://isocpp.org>

62. MATLAB [Electronic resource]. – Mode of access: <https://www.mathworks.com/products/matlab.html>

63. JAVA [Electronic resource]. – Mode of access: <https://www.java.com/en/>

64. R [Electronic resource]. – Mode of access: <https://www.r-project.org>

65. Keras [Electronic resource]. – Mode of access: <https://keras.io>

66. MXNet [Electronic resource]. – Mode of access: <https://mxnet.apache.org/versions/1.9.1/>

67. Jupyter Notebook [Electronic resource]. – Mode of access: <https://jupyter.org>

68. PyCharm [Electronic resource]. – Mode of access:
<https://www.jetbrains.com/pycharm/>
69. Visual Studio Code [Electronic resource]. – Mode of access:
<https://code.visualstudio.com>
70. Google Colab [Electronic resource]. – Mode of access:
https://colab.research.google.com/?utm_source=scs-index
71. Zhu S. Super resolution reconstruction method for infrared images based on pseudo transferred features [Text] / S. Zhu, C. Zhou, Y. Wang // Displays. – 2022. – Vol. 74. – P. 102187.
72. LLVIP: A Visible-infrared Paired Dataset for Low-light Vision [Electronic resource]. – Mode of access: <https://bupt-ai-cz.github.io/LLVIP/> .
73. FLIR Thermal Images Dataset [Electronic resource]. – Mode of access:
<https://www.kaggle.com/datasets/deepnewbie/flir-thermal-images-dataset> .

ДОДАТОК А
Слайди презентації

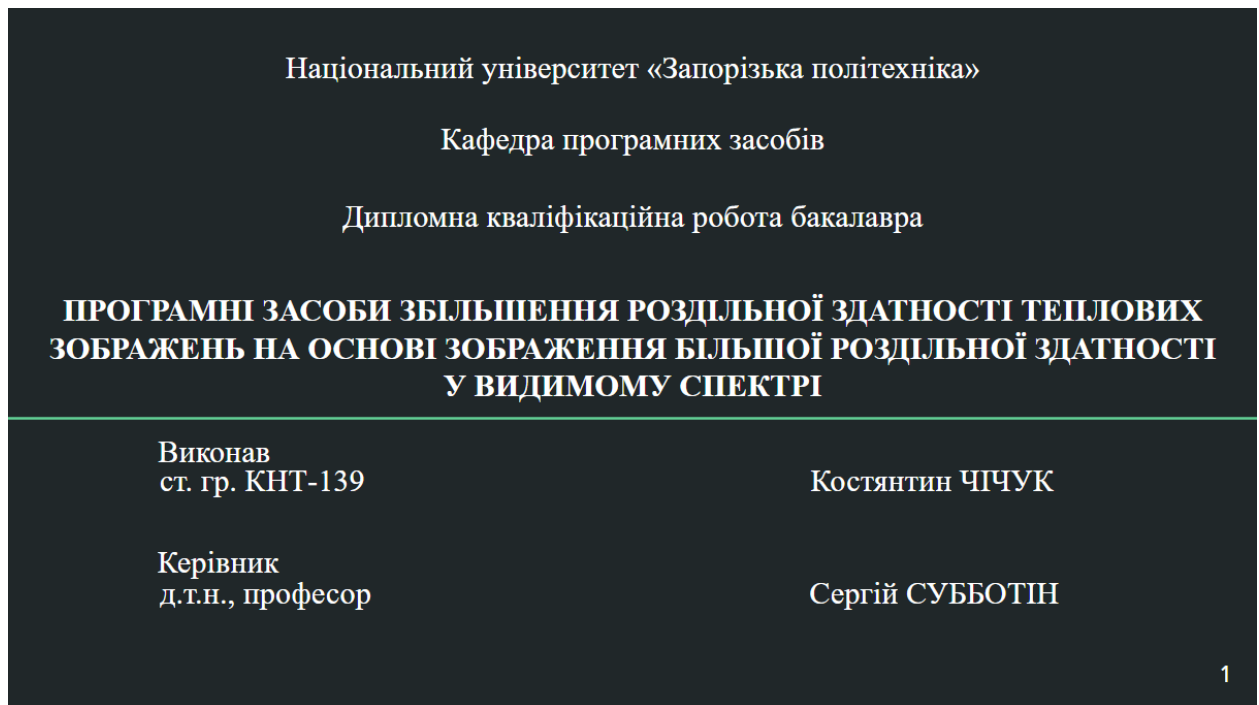


Рисунок А.1 – Слайд вступу

Об'єкт та мета роботи

- Об'єкт розробки – програмні засоби для супер масштабування інфрачервоних зображень з допомогою електрооптичного зразка.
- Мета роботи – дослідити, проаналізувати та реалізувати методи супер масштабування для побудови модулярних нейронних мереж у якості основи для моделей супер масштабування.

Рисунок А.2 – Слайд мети роботи

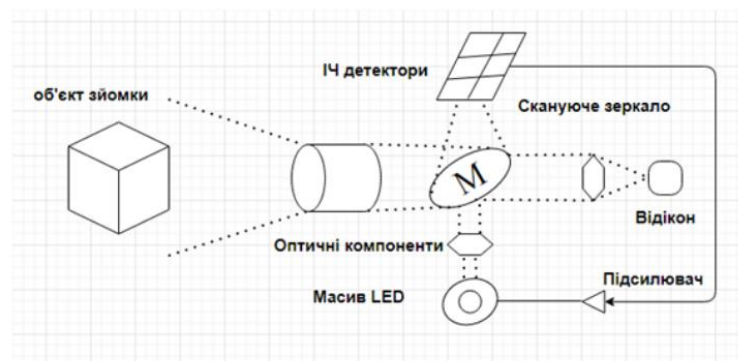
Задачі

- Аналіз предметної області
- Розробка структури системи
- Розробка модулю переносу стилю
- Розробка модулю супермасштабування
- Тестування та рефакторинг розробленого додатка

3

Рисунок А.3 – Слайд задач

Особливості зйомки інфрачервоних зображень



4

Рисунок А.4 – Слайд особливостей зйомки інфрачервоних зображень

Обґрунтування вибору підходу масштабування

Критерій порівняння підходів масштабування	підходи масштабування	
	Навчально-орієнтоване	Інтерполяційне
Джерело даних	Генерує нові дані	Використовує існуючі дані
Складність	Вища	Менша
Адаптивність	Висока	Обмежена
Зменшення артефактів	Можливе	Обмежено

5

Рисунок А.5 – Слайд порівняння підходів масштабування

Обґрунтування вибору шару розгортки

Критерій порівняння шарів розгортки	Шари розгортки		
	Транспонований шар згортки	Субпіксельний шар	Модуль мета-масштабування
Контекстуальна інформація	Обмежена	Більше	Більше
Артефакти шахової дошки	Можливі	Відсутні	Відсутні
Гладкість виходів	Не гарантована	Можлива	Можлива
Стабільність прогнозів	Добра	Добра	Менш стабільна
Ефективність при більших масштабах зображення	Не ефективний	Ефективний	Менш ефективний

6

Рисунок А.6 – Слайд порівняння шарів розгортки

Обґрунтування вибору мови програмування

Критерій порівняння мов програмування	Мови програмування				
	Python	MATLAB	C++/C	Java	R
Популярність	Висока	Висока	Середня	Середня	Середня
Легкість використання	Висока	Середня	Середня	Висока	Середня
Швидкодія	Середня	Середня	Висока	Середня	Середня
Розширюваність	Висока	Середня	Середня	Висока	Висока
Спільнота	Велика	Висока	Середня	Висока	Висока
Гнучкість	Висока	Середня	Середня	Висока	Середня
Наявність бібліотек	Висока	Висока	Середня	Середня	Середня

7

Рисунок А.7 – Слайд порівняння мов програмування

Обґрунтування вибору фреймворку

Критерій порівняння	Фреймворки			
	TensorFlow	PyTorch	Keras	MXNet
Популярність	Висока	Висока	Висока	Висока
Легкість використання	Середня	Висока	Висока	Середня
Швидкодія	Висока	Висока	Середня	Висока
Розширюваність	Висока	Висока	Середня	Висока
Спільнота	Велика	Висока	Висока	Середня
Гнучкість	Середня	Висока	Середня	Висока

8

Рисунок А.8 – Слайд порівняння фреймворків

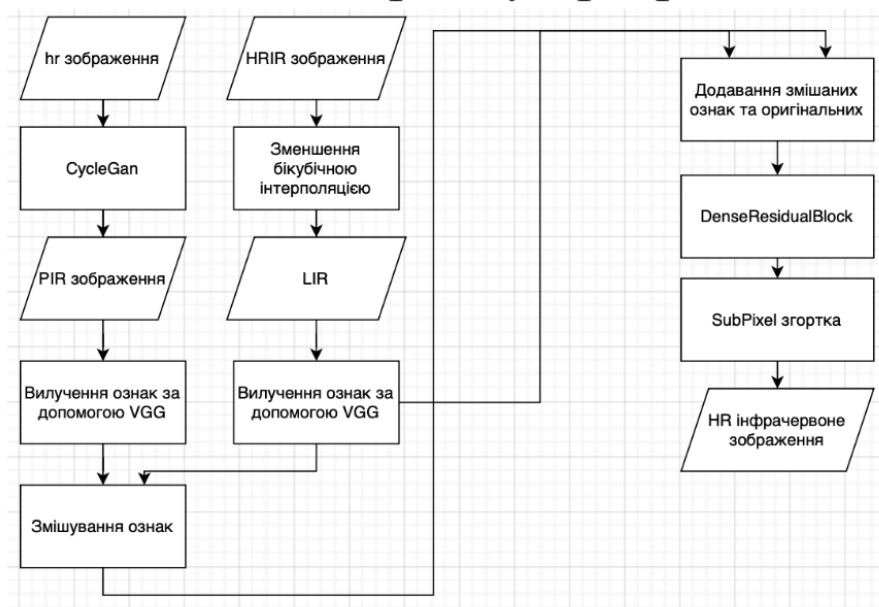
Обґрунтування вибору середовища розробки

Критерій порівняння	Середовища розробки			
	Jupyter Notebook	PyCharm	VS Code	Google Colab
Інтерфейс користувача	Веб-інтерфейс	Локальний запуск	Локальний запуск	Веб-інтерфейс
Інтерактивні віджети	Так	Так	Ні	Так
Підтримка IntelliSense	Ні	Так	Так	Так
Відладка	Обмежена	Так	Так	Так
підтримка хмарних технологій	Обмежена	Так	Ні	Так
Розгортання моделей	Обмежена	Так	Ні	Обмежена
обчислювальні ресурси	Хмарні	Локальні	Локальні	Хмарні

9

Рисунок А.9 – Слайд порівняння середовищ розробки

Схема алгоритму програми



10

Рисунок А.10 – Слайд схема алгоритму роботи програми

Результат переносу стилю



11

Рисунок А.11 – Слайд результату переносу стилю

Порівняння результатів супермасштабування



HR

Bicubic

SRCNN

EDSR

SRGAN

PT2SR

12

Рисунок А.12 – Слайд результатів супермасштабування

Результати тестування

Набір даних	Критерій порівняння	Моделі				
		Bicubic	SRCNN	EDSR	SRGAN	PT2SR
PFLIR	PSNR/SSIM	26.59/ 0.753	22.69/ 0.763	33.04/ 0.926	30.34/ 0.846	34.23/ 0.939
	Час (сек)	0.032	0.66	0.71	1.03	1.28
LLVIP	PSNR/SSIM	28.31/ 0.762	24.58/ 0.870	29.81/ 0.831	27.58/ 0.877	29.79/ 0.901
	Час (сек)	0.061	0.94	1.07	1.76	2.25

13

Рисунок А.13 – Слайд результату тестування

Висновки

Проведено дослідження предметної області, окреслено базові поняття, що стосуються об'єкта дослідження. Визначено основну суть об'єкта дослідження, його провідну ідею та цілі.

Обрано та обґрунтовано компоненти розробки. Спроектовано структуру системи на основі виявлених сутностей та функціональних потреб.

Реалізовано метод супер масштабування інфрачервоних зображень за зразком електрооптичного зображення високої роздільної здатності.

Розроблений програмний комплекс має практичну спрямованість та готовий до використання.

14

Рисунок А.14 – Слайд висновків

ДОДАТОК Б
Код програми

Б.1 Файл get_data.py

```

from __future__ import print_function
import os
import tarfile
import requests
from warnings import warn
from zipfile import ZipFile
from bs4 import BeautifulSoup
from os.path import abspath, isdir, join, basename

class GetData(object):
    """A Python script for downloading CycleGAN datasets.

    Parameters:
        technique (str) --cyclegan'
        verbose (bool) -- If True, print additional information.

    Examples:
        >>> from util.get_data import GetData
        >>> gd = GetData(technique='cyclegan')
        >>> new_data_path = gd.get(save_path='./datasets') # options will be displayed.
    """

    def __init__(self, technique='cyclegan', verbose=True):
        url_dict = {
            'cyclegan':
            'https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets'
        }
        self.url = url_dict.get(technique.lower())
        self._verbose = verbose

    def _print(self, text):
        if self._verbose:
            print(text)

    @staticmethod

```

```

def _get_options(r):
    soup = BeautifulSoup(r.text, 'lxml')
    options = [h.text for h in soup.find_all('a', href=True)
                if h.text.endswith(('.zip', 'tar.gz'))]
    return options

def _present_options(self):
    r = requests.get(self.url)
    options = self._get_options(r)
    print('Options:\n')
    for i, o in enumerate(options):
        print("{0}: {1}".format(i, o))
    choice = input("\nPlease enter the number of the "
                   "dataset above you wish to download:")
    return options[int(choice)]

def _download_data(self, dataset_url, save_path):
    if not isdir(save_path):
        os.makedirs(save_path)

    base = basename(dataset_url)
    temp_save_path = join(save_path, base)

    with open(temp_save_path, "wb") as f:
        r = requests.get(dataset_url)
        f.write(r.content)

    if base.endswith('.tar.gz'):
        obj = tarfile.open(temp_save_path)
    elif base.endswith('.zip'):
        obj = ZipFile(temp_save_path, 'r')
    else:
        raise ValueError("Unknown File Type: {0}".format(base))

    self._print("Unpacking Data...")
    obj.extractall(save_path)

```

```

obj.close()

os.remove(temp_save_path)

def get(self, save_path, dataset=None):
    """

    Download a dataset.

    Parameters:
        save_path (str) -- A directory to save the data to.
        dataset (str) -- (optional). A specific dataset to download.
                           Note: this must include the file extension.
                           If None, options will be presented for you
                           to choose from.

    Returns:
        save_path_full (str) -- the absolute path to the downloaded data.

    """
    if dataset is None:
        selected_dataset = self._present_options()
    else:
        selected_dataset = dataset

    save_path_full = join(save_path, selected_dataset.split('.')[0])

    if isdir(save_path_full):
        warn("\n'{0}' already exists. Voiding Download.".format(
            save_path_full))
    else:
        self._print('Downloading Data...')
        url = "{0}/{1}".format(self.url, selected_dataset)
        self._download_data(url, save_path=save_path)

    return abspath(save_path_full)

```

Б.2 Файл util.py

```

"""This module contains simple helper functions """

from __future__ import print_function
import torch
import numpy as np
from PIL import Image
import os


def tensor2im(input_image, imtype=np.uint8):
    """Converts a Tensor array into a numpy image array.

    Parameters:
        input_image (tensor) -- the input image tensor array
        imtype (type)        -- the desired type of the converted numpy array
    """
    if not isinstance(input_image, np.ndarray):
        if isinstance(input_image, torch.Tensor): # get the data from a variable
            image_tensor = input_image.data
        else:
            return input_image

        image_numpy = image_tensor[0].cpu().float().numpy() # convert it into a numpy
array
        if image_numpy.shape[0] == 1: # grayscale to RGB
            image_numpy = np.tile(image_numpy, (3, 1, 1))
            image_numpy = (np.transpose(image_numpy, (1, 2, 0)) + 1) / 2.0 * 255.0 # post-
processing: tranpose and scaling
        else: # if it is a numpy array, do nothing
            image_numpy = input_image
        return image_numpy.astype(imtype)


def diagnose_network(net, name='network'):
    """Calculate and print the mean of average absolute(gradients)

    Parameters:
        net (torch network) -- Torch network
        name (str) -- the name of the network
    """

```

```

"""
mean = 0.0
count = 0
for param in net.parameters():
    if param.grad is not None:
        mean += torch.mean(torch.abs(param.grad.data))
        count += 1
if count > 0:
    mean = mean / count
print(name)
print(mean)

def save_image(image_numpy, image_path, aspect_ratio=1.0):
    """Save a numpy image to the disk

    Parameters:
        image_numpy (numpy array) -- input numpy array
        image_path (str)          -- the path of the image
    """

    image_pil = Image.fromarray(image_numpy)
    h, w, _ = image_numpy.shape

    if aspect_ratio > 1.0:
        image_pil = image_pil.resize((h, int(w * aspect_ratio)), Image.BICUBIC)
    if aspect_ratio < 1.0:
        image_pil = image_pil.resize((int(h / aspect_ratio), w), Image.BICUBIC)
    image_pil.save(image_path)

def print_numpy(x, val=True, shp=False):
    """Print the mean, min, max, median, std, and size of a numpy array

    Parameters:
        val (bool) -- if print the values of the numpy array
        shp (bool) -- if print the shape of the numpy array
    """

```

```

x = x.astype(np.float64)
if shp:
    print('shape,', x.shape)
if val:
    x = x.flatten()
    print('mean = %3.3f, min = %3.3f, max = %3.3f, median = %3.3f, std=%3.3f' % (
        np.mean(x), np.min(x), np.max(x), np.median(x), np.std(x)))

def makedirs(paths):
    """create empty directories if they don't exist

    Parameters:
        paths (str list) -- a list of directory paths
    """
    if isinstance(paths, list) and not isinstance(paths, str):
        for path in paths:
            mkdir(path)
    else:
        mkdir(paths)

def mkdir(path):
    """create a single empty directory if it didn't exist

    Parameters:
        path (str) -- a single directory path
    """
    if not os.path.exists(path):
        os.makedirs(path)

```

Б.3 Файл pt2sr.ipynb

```

from google.colab import drive
drive.mount('/content/drive')

import os

```

```

os.chdir('/content/drive/MyDrive/diplom/pytorch-CycleGAN')

!pip install -r requirements.txt

!python test.py --dataroot /content/drive/MyDrive/diplom/test+ --name vi2ir_2 --model
test --no_dropout

import torch
import torchvision.transforms as transforms
import torchvision.models as models
import torch.nn.functional as F
from PIL import Image

# Load the image
ir_image = Image.open('/content/drive/MyDrive/diplom/test/testA/test.jpg')
pir_image = Image.open('/content/drive/MyDrive/diplom/pytorch-
CycleGAN/results/vi2ir_2/test_latest/images/test_fake.png')

block_size = 64

# Load the pre-trained VGG model
vgg = models.vgg19(pretrained=True)

# Extract features for each block

preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(256),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

pir_input_tensor = preprocess(pir_image)
pir_input_batch = pir_input_tensor.unsqueeze(0)
ir_input_tensor = preprocess(ir_image)
ir_input_batch = ir_input_tensor.unsqueeze(0)

# Extract features using VGG

```

```

model = models.vgg19(weights=True).features

with torch.no_grad():
    features = vgg.features(pir_input_batch)

reshaped_tensor = F.interpolate(features, size=(256, 256), mode='bilinear',
align_corners=False)
reshaped_tensor = reshaped_tensor[:, :1, :, :]
print(reshaped_tensor.shape)

concatenated_tensor = torch.cat((ir_input_batch, reshaped_tensor), dim=1)

print(concatenated_tensor.shape)

#utils block

import torch
import random
import torchvision.transforms.functional as FT

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
imagenet_mean = torch.FloatTensor([0.485, 0.456, 0.406]).unsqueeze(1).unsqueeze(2)
imagenet_std = torch.FloatTensor([0.229, 0.224, 0.225]).unsqueeze(1).unsqueeze(2)
imagenet_mean_cuda = torch.FloatTensor([0.485, 0.456, 0.406]).to(device).unsqueeze(0).unsqueeze(2).unsqueeze(3)
imagenet_std_cuda = torch.FloatTensor([0.229, 0.224, 0.225]).to(device).unsqueeze(0).unsqueeze(2).unsqueeze(3)

def create_data_lists(train_folders, test_folders, min_size, output_folder):
    """
    Create lists for images in the training set and each of the test sets.

    :param train_folders: folders containing the training images; these will be merged
    :param test_folders: folders containing the test images; each test folder will form
its own test set
    :param min_size: minimum width and height of images to be considered
    :param output_folder: save data lists here

```

```

"""
print("\nCreating data lists... this may take some time.\n")
train_images = list()
for d in train_folders:
    for i in os.listdir(d):
        img_path = os.path.join(d, i)
        img = Image.open(img_path, mode='r')
        if img.width >= min_size and img.height >= min_size:
            train_images.append(img_path)
print("There are %d images in the training data.\n" % len(train_images))
with open(os.path.join(output_folder, 'train_images.json'), 'w') as j:
    json.dump(train_images, j)

for d in test_folders:
    test_images = list()
    test_name = d.split("/")[-1]
    for i in os.listdir(d):
        img_path = os.path.join(d, i)
        img = Image.open(img_path, mode='r')
        if img.width >= min_size and img.height >= min_size:
            test_images.append(img_path)
    print("There are %d images in the %s test data.\n" % (len(test_images),
test_name))
    with open(os.path.join(output_folder, test_name + '_test_images.json'), 'w') as
j:
        json.dump(test_images, j)

    print("JSONS containing lists of Train and Test images have been saved to %s\n" %
output_folder)

def convert_image(img, source, target):
    """
    Convert an image from a source format to a target format.
    :param img: image
    :param source: source format, one of 'pil' (PIL image), '[0, 1]' or '[-1, 1]' (pixel
value ranges)
    :param target: target format, one of 'pil' (PIL image), '[0, 255]', '[0, 1]', '[-1,
1]' (pixel value ranges),

```

```

        'imagenet-norm' (pixel values standardized by imagenet mean and std.),
        'y-channel' (luminance channel Y in the YCbCr color format, used to
calculate PSNR and SSIM)

    :return: converted image
    """
    assert source in {'pil', '[0, 1]', '[-1, 1]'}, "Cannot convert from source format
%s!" % source
    assert target in {'pil', '[0, 255]', '[0, 1]', '[-1, 1]', 'imagenet-norm',
        'y-channel'}, "Cannot convert to target format %s!" % target

    # Convert from source to [0, 1]
    if source == 'pil':
        img = FT.to_tensor(img)

    elif source == '[0, 1]':
        pass # already in [0, 1]

    elif source == '[-1, 1]':
        img = (img + 1.) / 2.

    # Convert from [0, 1] to target
    if target == 'pil':
        img = FT.to_pil_image(img)

    elif target == '[0, 255]':
        img = 255. * img

    elif target == '[0, 1]':
        pass # already in [0, 1]

    elif target == '[-1, 1]':
        img = 2. * img - 1.

    elif target == 'imagenet-norm':
        if img.ndimension() == 3:
            img = (img - imagenet_mean) / imagenet_std
        elif img.ndimension() == 4:

```

```

img = (img - imagenet_mean_cuda) / imagenet_std_cuda

elif target == 'y-channel':
    img = torch.matmul(255. * img.permute(0, 2, 3, 1)[: , 4:-4, 4:-4, :], rgb_weights)
/ 255. + 16.

return img

class ImageTransforms(object):
    """
    Image transformation pipeline.
    """

    def __init__(self, split, crop_size, scaling_factor, lr_img_type, hr_img_type):
        """
        :param split: one of 'train' or 'test'
        :param crop_size: crop size of HR images
        :param scaling_factor: LR images will be downsampled from the HR images by this
factor
        :param lr_img_type: the target format for the LR image; see convert_image() above
for available formats
        :param hr_img_type: the target format for the HR image; see convert_image() above
for available formats
        """
        self.split = split.lower()
        self.crop_size = crop_size
        self.scaling_factor = scaling_factor
        self.lr_img_type = lr_img_type
        self.hr_img_type = hr_img_type

        assert self.split in {'train', 'test'}

    def __call__(self, img):
        """
        :param img: a PIL source image from which the HR image will be cropped, and then
downsampled to create the LR image
        :return: LR and HR images in the specified format
        """

```

```

# Crop
if self.split == 'train':
    # Take a random fixed-size crop of the image, which will serve as the high-
    resolution (HR) image
    left = random.randint(1, img.width - self.crop_size)
    top = random.randint(1, img.height - self.crop_size)
    right = left + self.crop_size
    bottom = top + self.crop_size
    hr_img = img.crop((left, top, right, bottom))
else:
    # Take the largest possible center-crop of it such that its dimensions are
    perfectly divisible by the scaling factor
    x_remainder = img.width % self.scaling_factor
    y_remainder = img.height % self.scaling_factor
    left = x_remainder // 2
    top = y_remainder // 2
    right = left + (img.width - x_remainder)
    bottom = top + (img.height - y_remainder)
    hr_img = img.crop((left, top, right, bottom))

    # Downsize this crop to obtain a low-resolution version of it
    lr_img = hr_img.resize((int(hr_img.width / self.scaling_factor),
int(hr_img.height / self.scaling_factor)),
                           Image.BICUBIC)

# Sanity check
assert hr_img.width == lr_img.width * self.scaling_factor and hr_img.height ==
lr_img.height * self.scaling_factor

# Convert the LR and HR image to the required type
lr_img = convert_image(lr_img, source='pil', target=self.lr_img_type)
hr_img = convert_image(hr_img, source='pil', target=self.hr_img_type)

return lr_img, hr_img

class AverageMeter(object):
    """

```

```

Keeps track of most recent, average, sum, and count of a metric.
"""

def __init__(self):
    self.reset()

def reset(self):
    self.val = 0
    self.avg = 0
    self.sum = 0
    self.count = 0

def update(self, val, n=1):
    self.val = val
    self.sum += val * n
    self.count += n
    self.avg = self.sum / self.count

def clip_gradient(optimizer, grad_clip):
    """
    Clips gradients computed during backpropagation to avoid explosion of gradients.
    :param optimizer: optimizer with the gradients to be clipped
    :param grad_clip: clip value
    """
    for group in optimizer.param_groups:
        for param in group['params']:
            if param.grad is not None:
                param.grad.data.clamp_(-grad_clip, grad_clip)

from torch.utils.data import Dataset
import json
import os
from PIL import Image

create_data_lists(train_folders=['/content/drive/MyDrive/diplom/pt2sr/train',
                                '/content/drive/MyDrive/diplom/pt2sr/val'],

```

```

test_folders=['/content/drive/MyDrive/diplom/pt2sr/test/LLIVP'],
min_size=100,
output_folder='/content/drive/MyDrive/diplom/pt2sr/output')

from torch.utils.data import Dataset
import json
import os
from PIL import Image

class SRDataset(Dataset):
    """
    A PyTorch Dataset to be used by a PyTorch DataLoader.
    """

    def __init__(self, data_folder, split, crop_size, scaling_factor, lr_img_type,
hr_img_type, test_data_name=None):
        """
        :param data_folder: # folder with JSON data files
        :param split: one of 'train' or 'test'
        :param crop_size: crop size of target HR images
        :param scaling_factor: the input LR images will be downsampled from the target
HR images by this factor; the scaling done in the super-resolution
        :param lr_img_type: the format for the LR image supplied to the model; see
convert_image() in utils.py for available formats
        :param hr_img_type: the format for the HR image supplied to the model; see
convert_image() in utils.py for available formats
        :param test_data_name: if this is the 'test' split, which test dataset? (for
example, "Set14")
        """

        self.data_folder = data_folder
        self.split = split.lower()
        self.crop_size = int(crop_size)
        self.scaling_factor = int(scaling_factor)
        self.lr_img_type = lr_img_type
        self.hr_img_type = hr_img_type
        self.test_data_name = test_data_name

```

```

assert self.split in {'train', 'test'}
if self.split == 'test' and self.test_data_name is None:
    raise ValueError("Please provide the name of the test dataset!")
assert lr_img_type in {'[0, 255]', '[0, 1]', '[-1, 1]', 'imagenet-norm'}
assert hr_img_type in {'[0, 255]', '[0, 1]', '[-1, 1]', 'imagenet-norm'}

# If this is a training dataset, then crop dimensions must be perfectly divisible
by the scaling factor

# (If this is a test dataset, images are not cropped to a fixed size, so this
variable isn't used)

if self.split == 'train':
    assert self.crop_size % self.scaling_factor == 0, "Crop dimensions are not
perfectly divisible by scaling factor! This will lead to a mismatch in the dimensions of
the original HR patches and their super-resolved (SR) versions!"

# Read list of image-paths
if self.split == 'train':
    with open(os.path.join(data_folder, 'train_images.json'), 'r') as j:
        self.images = json.load(j)
else:
    with open(os.path.join(data_folder, self.test_data_name +
'_test_images.json'), 'r') as j:
        self.images = json.load(j)

# Select the correct set of transforms
self.transform = ImageTransforms(split=self.split,
                                crop_size=self.crop_size,
                                scaling_factor=self.scaling_factor,
                                lr_img_type=self.lr_img_type,
                                hr_img_type=self.hr_img_type)

def __getitem__(self, i):
    """
    This method is required to be defined for use in the PyTorch DataLoader.
    :param i: index to retrieve
    :return: the 'i'th pair LR and HR images to be fed into the model
    """

```

```

    # Read image
    img = Image.open(self.images[i], mode='r')
    img = img.convert('RGB')
    if img.width <= 96 or img.height <= 96:
        print(self.images[i], img.width, img.height)
    lr_img, hr_img = self.transform(img)

    return lr_img, hr_img

def __len__(self):
    """
    This method is required to be defined for use in the PyTorch DataLoader.
    :return: size of this data (in number of images)
    """
    return len(self.images)

import torch
from torch import nn
import torchvision
import math

class ConvolutionalBlock(nn.Module):
    """
    A convolutional block, comprising convolutional, BN, activation layers.
    """

    def __init__(self, in_channels, out_channels, kernel_size, stride=1, batch_norm=True,
activation=None):
        """
        :param in_channels: number of input channels
        :param out_channels: number of output channels
        :param kernel_size: kernel size
        :param stride: stride
        :param batch_norm: include a BN layer?
        """
        super(ConvolutionalBlock, self).__init__()
        if activation is not None:

```

```

        activation = activation.lower()
        assert activation in {'prelu', 'leakyrelu', 'tanh'}

# A container that will hold the layers in this convolutional block
layers = list()

# A convolutional layer
layers.append(
    nn.Conv2d(in_channels=in_channels, out_channels=out_channels,
              kernel_size=kernel_size, stride=stride, padding=kernel_size // 2))

# A batch normalization (BN) layer, if wanted
if batch_norm is True:
    layers.append(nn.BatchNorm2d(num_features=out_channels))

# An activation layer, if wanted
if activation == 'prelu':
    layers.append(nn.PReLU())
elif activation == 'leakyrelu':
    layers.append(nn.LeakyReLU(0.2))
elif activation == 'tanh':
    layers.append(nn.Tanh())

# Put together the convolutional block as a sequence of the layers in this
container
self.conv_block = nn.Sequential(*layers)

def forward(self, input):
    """
    Forward propagation.
    :param input: input images, a tensor of size (N, in_channels, w, h)
    :return: output images, a tensor of size (N, out_channels, w, h)
    """
    output = self.conv_block(input) # (N, out_channels, w, h)

    return output

```

```

class DenseResidualBlock(nn.Module):
    def __init__(self, kernel_size=3, n_channels=64, grow_rate=4):
        """
        :param kernel_size: kernel size
        :param n_channels: number of input and output channels (same because the input
must be added to the output)
        """
        super(ResidualBlock, self).__init__()

        # The first convolutional block

        self.conv_block1 = ConvolutionalBlock(in_channels=n_channels,
out_channels=n_channels * grow_rate, kernel_size=kernel_size, batch_norm=True,
activation='PReLU')

        self.conv_block2 = ConvolutionalBlock(in_channels=n_channels * grow_rate,
out_channels=n_channels * (grow_rate ** 2), kernel_size=kernel_size, batch_norm=True,
activation='PReLU')

        self.conv_block3 = ConvolutionalBlock(in_channels=n_channels * (grow_rate ** 2),
out_channels=n_channels * (grow_rate ** 3), kernel_size=kernel_size, batch_norm=True,
activation='PReLU')

        # The second convolutional block

        self.conv_block4 = ConvolutionalBlock(in_channels=n_channels * (grow_rate ** 3),
out_channels=n_channels * (grow_rate ** 4), kernel_size=kernel_size, batch_norm=True,
activation=None)

    def forward(self, input):
        """
        Forward propagation.
        :param input: input images, a tensor of size (N, n_channels, w, h)
        :return: output images, a tensor of size (N, n_channels, w, h)
        """
        residual = input # (N, n_channels, w, h)
        output = self.conv_block1(input) # (N, n_channels, w, h)
        output = self.conv_block2(output) # (N, n_channels, w, h)
        output = output + residual # (N, n_channels, w, h)

        residual = input
        features = [input]

```

```

new_features = self.conv_block1(input)
features.append(new_features)
new_features = self.conv_block2(new_features)
features.append(new_features)
new_features = self.conv_block3(new_features)
features.append(new_features)
new_features = self.conv_block4(new_features)
features.append(new_features)

output = torch.cat(features, 1) + residual

print("=====")
print(features.shape)
print(torch.cat(features, 1).shape)
print(output.shape)
print("=====")

return output

class SubPixelConvolutionalBlock(nn.Module):
    """
    A subpixel convolutional block, comprising convolutional, pixel-shuffle, and PReLU
    activation layers.
    """

    def __init__(self, kernel_size=3, n_channels=64, scaling_factor=2):
        """
        :param kernel_size: kernel size of the convolution
        :param n_channels: number of input and output channels
        :param scaling_factor: factor to scale input images by (along both dimensions)
        """
        super(SubPixelConvolutionalBlock, self).__init__()

        # A convolutional layer that increases the number of channels by scaling factor^2,
        # followed by pixel shuffle and PReLU
        self.conv = nn.Conv2d(in_channels=n_channels, out_channels=n_channels *
                               (scaling_factor ** 2), kernel_size=kernel_size, padding=kernel_size // 2)

```

```

        # These additional channels are shuffled to form additional pixels, upscaling
        each dimension by the scaling factor

        self.pixel_shuffle = nn.PixelShuffle(upscale_factor=scaling_factor)

        self.prelu = nn.PReLU()

    def forward(self, input):
        """
        Forward propagation.

        :param input: input images, a tensor of size (N, n_channels, w, h)
        :return: scaled output images, a tensor of size (N, n_channels, w * scaling
        factor, h * scaling factor)
        """

        output = self.conv(input) # (N, n_channels * scaling_factor^2, w, h)

        output = self.pixel_shuffle(output) # (N, n_channels, w * scaling_factor, h *
        scaling_factor)

        output = self.prelu(output) # (N, n_channels, w * scaling_factor, h * scaling
        factor)

        return output

class PT2SR(nn.Module):

    def __init__(self, large_kernel_size=9, small_kernel_size=3, n_channels=64,
n_blocks=16, scaling_factor=4):
        """

        :param large_kernel_size: kernel size of the first and last convolutions which
        transform the inputs and outputs

        :param small_kernel_size: kernel size of all convolutions in-between, i.e. those
        in the residual and subpixel convolutional blocks

        :param n_channels: number of channels in-between, i.e. the input and output
        channels for the residual and subpixel convolutional blocks

        :param n_blocks: number of residual blocks

        :param scaling_factor: factor to scale input images by (along both dimensions)
        in the subpixel convolutional block
        """

        super(PT2SR, self).__init__()

        # Scaling factor must be 2, 4, or 8
        scaling_factor = int(scaling_factor)
        assert scaling_factor in {2, 4, 8}, "The scaling factor must be 2, 4, or 8!"

```

```

    # The first convolutional block

    self.conv_block1 = ConvolutionalBlock(in_channels=3, out_channels=n_channels,
kernel_size=large_kernel_size, batch_norm=False, activation='PReLU')


    # A sequence of n_blocks residual blocks, each containing a skip-connection
across the block

    self.residual_blocks = nn.Sequential(

        *[DenseResidualBlock(kernel_size=small_kernel_size, n_channels=n_channels)
for i in range(n_blocks)])


    # Another convolutional block

    self.conv_block2 = ConvolutionalBlock(in_channels=n_channels,
out_channels=n_channels, kernel_size=small_kernel_size, batch_norm=True,
activation=None)


    # Upscaling is done by sub-pixel convolution, with each such block upscaling by
a factor of 2

    n_subpixel_convolution_blocks = int(math.log2(scaling_factor))

    self.subpixel_convolutional_blocks = nn.Sequential(

        *[SubPixelConvolutionalBlock(kernel_size=small_kernel_size,
n_channels=n_channels, scaling_factor=2) for i
in range(n_subpixel_convolution_blocks)])


    # The last convolutional block

    self.conv_block3 = ConvolutionalBlock(in_channels=n_channels, out_channels=3,
kernel_size=large_kernel_size, batch_norm=False, activation='Tanh')


def forward(self, lr_imgs):
    """
    Forward prop.

    :param lr_imgs: low-resolution input images, a tensor of size (N, 3, w, h)
    :return: super-resolution output images, a tensor of size (N, 3, w * scaling
factor, h * scaling factor)
    """

    output = self.conv_block1(lr_imgs)
    residual = output
    output = self.residual_blocks(output)
    output = self.conv_block2(output)
    output = output + residual

```

```

        output = self.subpixel_convolutional_blocks(output)
        sr_imgs = self.conv_block3(output)

    return sr_imgs

import time
import torch.backends.cudnn as cudnn
import torch
from torch import nn

# Data parameters
data_folder = '/content/drive/MyDrive/diplom/pt2st/output' # folder with JSON data files
crop_size = 96 # crop size of target HR images
scaling_factor = 4 # the scaling factor for the generator; the input LR images will be
downsampled from the target HR images by this factor

# Model parameters
large_kernel_size = 9 # kernel size of the first and last convolutions which transform
the inputs and outputs
small_kernel_size = 3 # kernel size of all convolutions in-between, i.e. those in the
residual and subpixel convolutional blocks
n_channels = 64 # number of channels in-between, i.e. the input and output channels for
the residual and subpixel convolutional blocks
n_blocks = 16 # number of residual blocks

# Learning parameters
checkpoint = None # path to model checkpoint, None if none
batch_size = 16 # batch size
start_epoch = 0 # start at this epoch
iterations = 100 # number of training iterations
workers = 4 # number of workers for loading data in the DataLoader
print_freq = 500 # print training status once every __ batches
lr = 1e-4 # learning rate
grad_clip = None # clip if gradients are exploding

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

cudnn.benchmark = True

```

```

# Initialize model or load checkpoint
if checkpoint is None:
    model = PT2SR(large_kernel_size=large_kernel_size,
                  small_kernel_size=small_kernel_size,
                  n_channels=n_channels, n_blocks=n_blocks,
                  scaling_factor=scaling_factor)
    # Initialize the optimizer
    optimizer = torch.optim.Adam(params=filter(lambda p: p.requires_grad,
                                                model.parameters()),lr=lr)

else:
    checkpoint = torch.load(checkpoint)
    start_epoch = checkpoint['epoch'] + 1
    model = checkpoint['model']
    optimizer = checkpoint['optimizer']

# Move to default device
model = model.to(device)
criterion = nn.MSELoss().to(device)

# Custom dataloaders
train_dataset = SRDataset(data_folder,
                           split='train',
                           crop_size=crop_size,
                           scaling_factor=scaling_factor,
                           lr_img_type='imagenet-norm',
                           hr_img_type='[-1, 1]')

train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
                                           shuffle=True, num_workers=workers,
                                           pin_memory=True) # note that we're passing
the collate function here

def train(train_loader, model, criterion, optimizer, epoch):
    """
    One epoch's training.
    :param train_loader: DataLoader for training data
    :param model: model

```

```

:param criterion: content loss function (Mean Squared-Error loss)
:param optimizer: optimizer
:param epoch: epoch number
"""

model.train() # training mode enables batch normalization

batch_time = AverageMeter() # forward prop. + back prop. time
data_time = AverageMeter() # data loading time
losses = AverageMeter() # loss

start = time.time()

# Batches
for i, (lr_imgs, hr_imgs) in enumerate(train_loader):
    data_time.update(time.time() - start)

    # Move to default device
    lr_imgs = lr_imgs.to(device)
    hr_imgs = hr_imgs.to(device)

    # Forward prop.
    sr_imgs = model(lr_imgs)

    # Loss
    loss = criterion(sr_imgs, hr_imgs)

    # Backward prop.
    optimizer.zero_grad()
    loss.backward()

    # Clip gradients, if necessary
    if grad_clip is not None:
        clip_gradient(optimizer, grad_clip)

    # Update model
    optimizer.step()

```

```

# Keep track of loss
losses.update(loss.item(), lr_imgs.size(0))

# Keep track of batch time
batch_time.update(time.time() - start)

# Reset start time
start = time.time()

# Print status
if i % print_freq == 0:
    print('Epoch: [{0}][{1}/{2}]----'
          'Batch Time {batch_time.val:.3f} ({batch_time.avg:.3f})----'
          'Data Time {data_time.val:.3f} ({data_time.avg:.3f})----'
          'Loss {loss.val:.4f} ({loss.avg:.4f})'.format(epoch, i,
len(train_loader),
                                                    batch_time=batch_time,
                                                    data_time=data_time,
loss=losses))
    del lr_imgs, hr_imgs, sr_imgs # free some memory since their histories may be stored

# Total number of epochs to train for
epochs = int(iterations // len(train_loader) + 1)

print(epochs)
print(len(train_loader) + 1)

# Epochs
for epoch in range(start_epoch, epochs):
    # One epoch's training
    train(train_loader=train_loader,
          model=model,
          criterion=criterion,
          optimizer=optimizer,
          epoch=epoch)

```

```
# Save checkpoint
torch.save({'epoch': epoch,
           'model': model,
           'optimizer': optimizer},
           'checkpoint_pt2sr.pth.tar')

import torch
from PIL import Image, ImageDraw, ImageFont

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

# Model checkpoints
srresnet_checkpoint = "/content/checkpoint_pt2sr.pth.tar"

# Load models
pt2sr = torch.load(srresnet_checkpoint)['model'].to(device)
pt2sr.eval()
```