МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

«Програмний модуль покращення різкості зображень за допомогою нейронних мереж»

Галузь знань **12 «Інформаційні технології»**
Спеціальність **122 «Комп'ютерні науки»**
Освітня програма «**Комп'ютерні науки»**
Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН- 41
 Бураков Максим Вадимович

 Керівник:_____Мінаєва Ю.І.____
(прізвище та ініціали)
Кандидат технічних наук, доцент

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від « 06» _____червня_____ 2022р.
зав. кафедри_____доц. Іларіонов О.Є.

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.
_____

"___"_____2022 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Бураков Максим Вадимович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи): Програмний модуль покращення різкості зображень за допомогою нейронних мереж
   затверджена протоколом засідання кафедри від «23» грудня 2021 року. № 4

2. Термін здачі студентом закінченого проекту (роботи) « » _____ 2022 року

3. Вихідні дані до проекту (роботи): Веб застосунок для покращення різкості зображень за допомогою нейронних мереж

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
   Аналіз предметної області, вибір архітектури, розробка нейронної мережі, розробка клієнтської, серверної частини програмного забезпечення, тестування та створення макету застосунку.

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій):
   1) Актуальність теми (1 слайд);
   2) Мета, предмет та об'єкт дослідження (1 слайд);
   3) Проектування програмного модулю покращення різкості зображень за допомогою НМ (7 слайдів);
   4) Розробка програмного модулю(3 слайди);
   5) Тестування(3 слайди);
   6) Висновки по роботі(1 слайд).

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			

3			

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ _____ / Мінаєва Ю.І. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Бураков М.В. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

ор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
	Дослідження джерел та збір інформації для випускної кваліфікаційної роботи	15.02.2022 – 22.02.2021	
	Аналіз предметної області	23.02.22 – 07.03.22	
	Вибір методів рішення задачі	08.03.22 – 15.04.22	
	Проектування та розробка програмного модулю покращення різкості зображень за допомогою нейронних мереж	16.04.2022 – 10.05.2021	
	Тестування програмного модулю покращення різкості зображень за допомогою нейронних мереж	11.05.2022 – 17.05.2022	
	Оформлення пояснювальної записки, підготовка презентаційних матеріалів	18.05.2022– 29.05.2022	

Студент-дипломник _____ / Бураков М.В. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Мінаєва Ю.І. /
(підпис) (ПІБ)

АНОТАЦІЯ

Бураков Максим Вадимович виконав випускну кваліфікаційну роботу на тему «Програмний модуль покращення різкості зображень за допомогою нейронних мереж».

Дана випускна кваліфікаційна робота фокусується на дослідженні сучасних технологій в сфері обробки зображень з використанням технологій штучного інтелекту, аналізу існуючих рішень, та розробці системи, яка проводить покращення різкості зображень.

Ключові слова: штучний інтелект, нейронна мережа, глибинні нейронні мережі, генеративні нейронні мережі, покращення різкості зображень.

ANNOTATION

The degree project «Software module to improve image sharpness using neural networks» has been completed by Maksym Burakov.

This paper focuses on current artificial intelligence image processing technologies research, analyzing existing solutions and practical implementation.

Key words: artificial intelligence, neural network, deep neural networks, generative neural networks, image sharpness improvement.

Зміст

Перелік умовних позначень та скорочень

НМ – нейронна мережа

ШНМ – штучна нейронна мережа

ГЗМ - Генеративні змагальні мережі

ВСТУП

В світі цифрових технологій дуже багато даних та фотографій. Більшість фотографій проходить велику кількість обробок та конвертацій з одного формату в іншій, що в свою чергу погіршує якість фотографій. Також проблему с якістю фото торкнулося більшості компаній, які розробляють смартфони, бо покращувати матрицю та інші компоненти фотокамери стало дуже складно. На допомогу приходять нейромережі та спеціальні нейронні модулі, які дуже швидко обробляють і покращують фото одразу після знімку.

Застосування нейронних мереж дозволить значно пришскорити покращення зображень, а також значно спростити процес покращення фотографій для звичайних користувачів.

Основна мета роботи полягає в розробці системи, яка проводить покращення різкості зображення за допомогою нейронних мереж.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРОГРАМНОГО МОДУЛЯ ПОКРАЩЕННЯ ЯКОСТІ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

1.1 Аналіз предметної області

Нейронні мережі є одним із напрямків у розробці систем штучного інтелекту. Ідея полягає в тому, щоб якомога точніше змоделювати роботу нервової системи людини, а саме її здатність вчитися та виправляти помилки. У цьому головна особливість будь-якої нейронної мережі - вона здатна самостійно вчитися і діяти на основі попереднього досвіду, з кожним разом роблячи все менше помилок.

Нейромережа імітує як діяльність, а й структуру нервової системи людини. Така мережа складається з великої кількості окремих обчислювальних елементів («нейронів»). Найчастіше кожен «нейрон» належить до певному шару мережі. Вхідні дані послідовно проходять обробку всіх шарах мережі. Параметри кожного "нейрона" можуть змінюватися залежно від результатів, отриманих на попередніх наборах

Задачі які можуть виконувати НМ:

1. класифікація. Система визначає, чи відповідає аналізований об'єкт заданим параметрам, і відносить його до тієї чи іншої групи; Система має кінцеву кількість відповідей: чи є на фотографії собака, чи є на зображені людське обличчя. Результат подається в 0 або 1, а потім декодується в слова;
2. регресія. регресія відрізняються від класифікації тим, що результат може бути дійсними числами і відсутнє обмеження [0;1]. Может використовуватися для очікуваних результатів магазину на наступний місяць, вартість цінних паперів або ціна квартири;
3. кластеризація. Це завдання розбиття безлічі об'єктів на групи, які називаються кластерами. Усередині кожної групи повинні виявитися «схожі» об'єкти, а об'єкти різних групи мають бути якомога відміннішими;

4. розпізнавання зображень. Розпізнавання зображень – підкатегорія комп’ютерного зору та штучного інтелекту, являє собою набір методів для виявлення та аналізу зображень для автоматизації певних завдань. Ця технологія здатна ідентифікувати місця, людей, об’єкти та багато інших типів елементів у зображенні та роботи висновки аналізуючи їх. Може використовуватися для інтелектуальних систем відеоспостереження, заснована на розпізнаванні зображень, яка здатна повідомляти про будь-яку незвичайну поведінку або ситуацію на автостоянках[11].

Існує 3 типи навчання нейронних мереж:

1. навчання з вчителем. Для кожного вхідного вектору існує значення вихідного вектору. З учителем машина вчиться набагато краще й швидше, тому для вирішення практичних завдань такі алгоритми використовують частіше. До алгоритмів навчання з учителем належать такі типи задач, як регресія і класифікація[10];

2. навчання без вчителя. Алгоритм має тільки вхідні значення, а далі намагається виконати поставлену задачу без допомоги. Великі корпорації, які мають величезний потік користувачів, можуть використовувати для розмітки своїх клієнтів. Наприклад, Google і його anticaptcha: знаходячи автобуси на фото, не тільки підтверджуєте, що «користувач не робот», а й навчаєте нейронну мережу тому, який вигляд має автобус. У випадках, коли розмітка даних неможлива, вдаються до методів навчання без учителя. До таких алгоритмів належать задачі кластеризації, зменшення розмірності і пошуку правил[10];

3. навчання з підкріпленням. Навчання з підкріпленням менше схоже на попередні види, бо нагадує швидше той штучний інтелект, яким нас намагаються вразити у фантастичних фільмах. Такі алгоритми використовують не там, де потрібно проаналізувати дані, а там, де потрібно вижити у реальному середовищі.

Середовищем може бути що завгодно: як реальний світ, так і симуляція, і навіть комп'ютерна гра. Наприклад, існують роботи, які навчилися грати в Dota2

просто поринув у среду, або автопілот Tesla, який у симуляції вчиться не збивати пішоходів[10].

Архітектури нейромереж:

1. Багатошаровий персептрон (MLP) є доповненням до нейронної мережі прямого зв’язку. Він складається з трьох типів шарів — вхідного шару, вихідного шару та прихованого шару. Вхідний шар отримує вхідний сигнал для обробки. Вихідний шар виконує такі функції, як прогнозування та класифікація. Довільна кількість прихованих шарів, які розміщуються між вхідним і вихідним шарами, є справжньою обчислювальною машиною MLP. Подібно до мережі прямого зв'язку в MLP, дані переходять у прямому напрямку від рівня входу до рівня виведення. Нейрони в MLP тренуються за допомогою алгоритму навчання зворотного поширення. MLP розроблені для апроксимації будь-якої безперервної функції і можуть вирішувати задачі, які не є лінійно роздільними. Основними випадками використання MLP є класифікація шаблонів, розпізнавання, передбачення та апроксимація. [12].

2. Згорткова нейронна мережа (ConvNet/CNN) — це алгоритм глибокого навчання, який приймає вхідне зображення, призначає важливість різним аспектам/об’єктам зображення та має можливість відрізнити один об’єкт від від іншого. Необхідна попередня обробка для ConvNet є набагато нижчою в порівнянні з іншими алгоритмами класифікації. Хоча в примітивних методах фільтри розробляються вручну, з достатньою підготовкою, ConvNets мають можливість вивчати ці фільтри/характеристики[13].

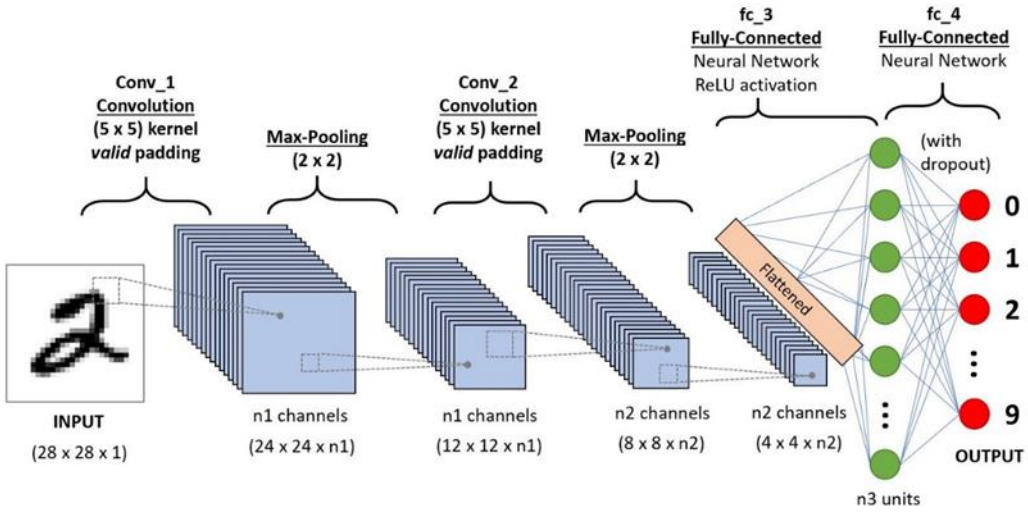

Рис. 1.1 Архітектура CNN

3. Рекурентна нейронна мережа (RNN) — це ще один тип ШНМ, яка використовується для послідовних даних або часових рядів. Основні практичні напрямки використання НМ, такі як мовний переклад, обробка природної мови (NLP), розпізнавання мовлення та підписання зображень; вони включені в популярні програми, такі як розумний помічник Siri, голосовий пошук і Google Translate. Подібно до прямих і згорткових нейронних мереж (CNN), рекуррентні нейронні мережі використовують навчальні дані для навчання У той час як традиційні глибокі нейронні мережі припускають, що входи та виходи незалежні один від одного, вихід рекурентних нейронних мереж залежить від попередніх елементів у послідовності. Хоча майбутні події також можуть бути корисними для визначення результату даної послідовності, односпрямовані рекурентні нейронні мережі не можуть врахувати ці події у своїх прогнозах.

4. Мережі довгострокової пам’яті, які зазвичай називають просто «LSTM» – це особливий тип RNN, здатний вивчати довгострокові залежності. Вони були представлені Hochreiter & Schmidhuber (1997), і були вдосконалені та популяризовані багатьма людьми в наступних роботах. Вони надзвичайно добре працюють над широким спектром проблем і зараз широко використовуються. LSTM явно розроблені, щоб уникнути проблеми довгострокової залежності. Запам’ятовування інформації протягом тривалого періоду часу – це практично їхня поведінка за замовчуванням, а не те, що їм важко вивчити! Усі рекуррентні

нейронні мережі мають вигляд ланцюга повторюваних модулів нейронної мережі. У стандартних RNN цей повторюваний модуль матиме дуже просту структуру, наприклад, один шар tanh[18].

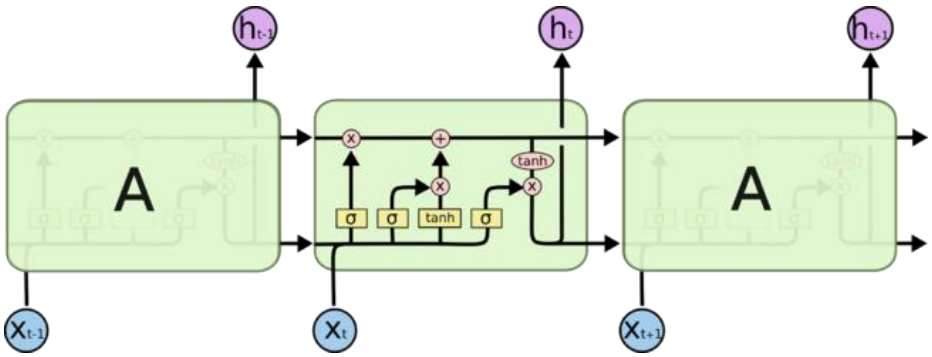

Рис. 1.2 Архітектура LSTM

5. Стаття під назвою «Attention Is All You Need», опублікована в 2017 році, представляє архітектуру кодера-декодера, засновану на шарах уваги, що називається трансформатором. Однією з основних відмінностей є те, що вхідна послідовність може передаватися паралельно, щоб GPU можна було ефективно використовувати, а також можна було збільшити швидкість навчання. І він заснований на багатоголовому шарі уваги, проблема зникаючого градієнта також подолана з великим відривом. Стаття базується на застосуванні трансформатора на NMT (Neural Machine Translator) [15].

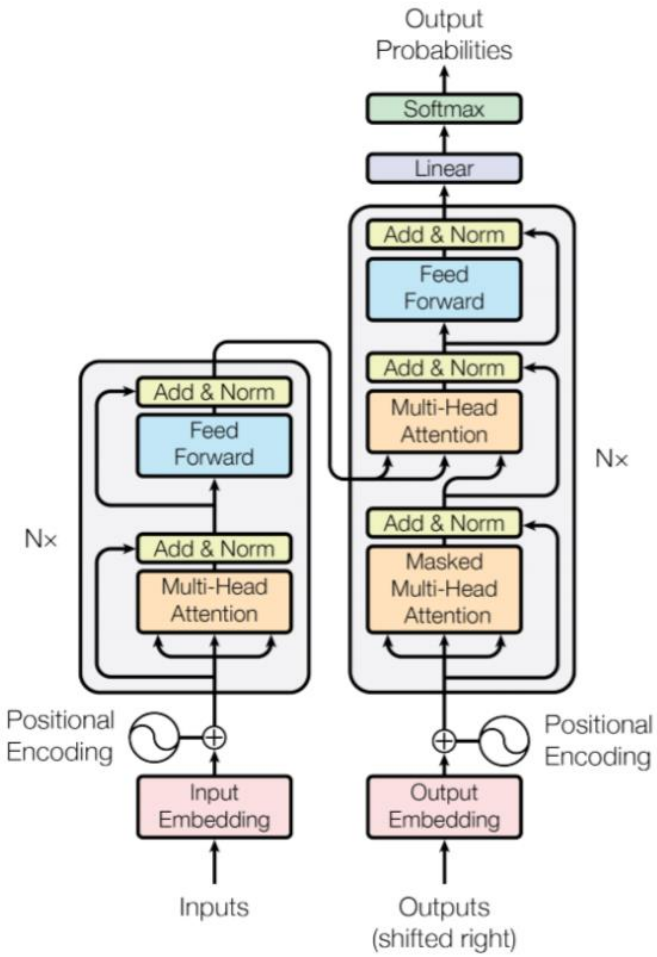

Рис. 1.3 Архітектура НМ трансформерів

6. GAN – це розумний спосіб навчання генеративної моделі, формуючи проблему як проблему навчання під наглядом за допомогою двох підмоделей: моделі генератора, яку розробники навчають для створення нових прикладів, і моделі дискримінатора, яка намагається класифікувати приклади як реальні (з домену) або підробка (згенерована). Дві моделі навчаються разом у змагальній грі з нульовою сумою, поки модель дискримінатора не буде обдурена приблизно в половині часу, що означає, що модель генератора генерує правдоподібні приклади. GAN є захоплюючою та швидко змінною сферою, яка забезпечує обіцянку генеративних моделей у своїй здатності генерувати реалістичні приклади в ряді проблемних областей, особливо в завданнях перекладу зображення в зображення, таких як зміну фотографій літа на зиму або дня до ночі, а також при створенні фотореалістичних фотографій об’єктів, сцен і людей, які навіть люди не можуть розпізнати[16].

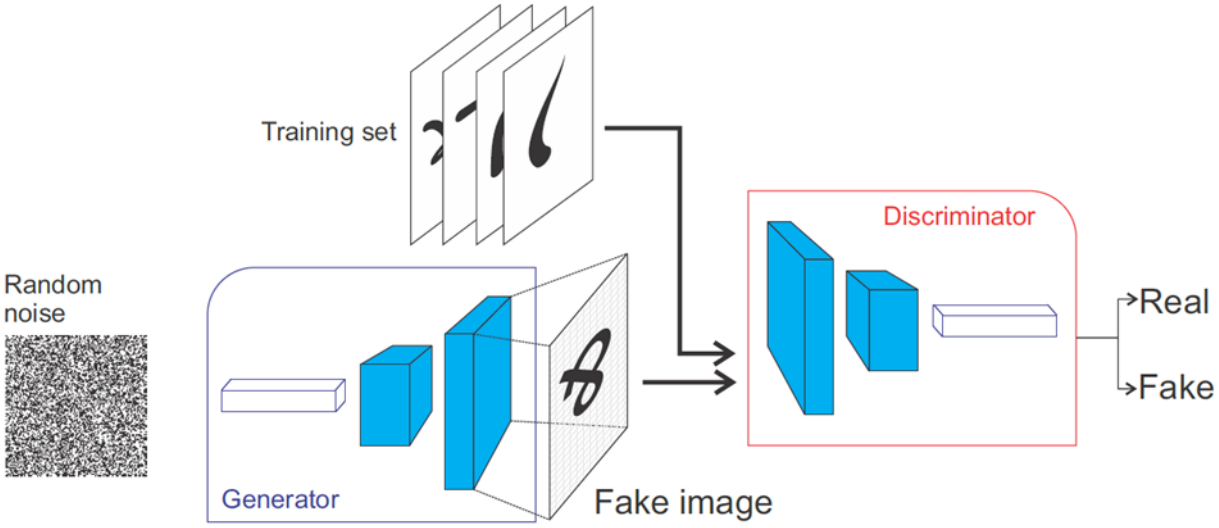

Рис. 1.4 Архітектура GAN

Отже, краще всього для покращення якості підходять генеративні мережі.

Людство майже в кожну сферу інтегрувало нейронні мережі, а саме:

- прогноз погоди;
- розпізнавання мови та обробка(Siri, Cortana);
- генерація голосу;
- обробка фото;
- прогнозування майбутніх хвороб людини;
- знаходження вакцин;
- роботи, що автоматично торгують на біржі;
- автоматичні чат боти, а саме обробка та генерація тексту.

Нейромережі стали ще більш популярними, бо з’явилися вже готові мережі, які можна одразу інтегрувати в свої системи і не займатися важким та довгим тренуванням.

1.2. Аналіз сучасного стану покращення зображень за допомогою нейронних мереж

В світі цифрових технологій дуже багато даних та фотографій. Більшість фотографій проходить велику кількість обробок та конвертацій з одного формату

в іншій, що в свою чергу погіршую якість фотографій. Також проблему с якістю фото торкнулося більшості компаній, які розробляють смартфони, бо покращувати матрицю та інші компоненти фотокамери стало дуже складно. На допомогу приходять нейромережі та спеціальні нейронні модулі, які дуже швидко обробляють і покращують фото одразу після знімку.

ESDR - одна з моделей суперроздільної здатності, яка відповідає високорівневій архітектурі, описана в статті Enhanced Deep Residual Networks for Single Image Super-Resolution (EDSR)[14]. Він став переможцем конкурсу на супер роздільну здатність NTIRE 2017. Автори EDSR стверджують, що пакетна нормалізація втрачає інформацію про масштаб зображень і зменшує гнучкість діапазону активацій. Видалення шарів пакетної нормалізації не тільки збільшує продуктивність із супер роздільною здатністю, але й зменшує пам’ять GPU до 40%, щоб можна було навчати значно більші моделі.

Інша модель супер-роздільної здатності є похідною від EDSR і описана в статті Wide Activation for Efficient and Accurate Image Super-Resolution[19], яка стала переможцем у реалістичних змаганнях NTIRE 2018 із супер роздільною здатністю. Модель має зміни в конструкції залишкового блоку, зменшуючи кількість каналів на шляху відображення ідентичності та збільшуючи кількість каналів у кожному залишковому блоці без збільшення загальної кількості параметрів. Автори припускають, що збільшення кількості каналів перед ReLU в залишкових блоках дозволяє більше інформації проходити через функцію активації, що ще більше підвищує продуктивність моделі. Вони також виявили, що реалізація нормалізації ваги додатково полегшує навчання та зближення більш глибоких моделей, щоб вони могли використовувати швидкість навчання, яка на порядок вища в порівнянні з тими, які використовуються в навчанні EDSR. Це не дивно, оскільки видалення шарів нормалізації ваги в EDSR ускладнює навчання більш глибоких моделей. Нормалізація ваг — це просто перепараметризація ваг нейронної мережі, яка відокремлює напрямок векторів ваги від їх величини, що покращує умовність задачі оптимізації та прискорює збіжність.Залежна від даних ініціалізація параметрів рівня нормалізації ваги не

виконується. Це змінило б функції, подібні до пакетної нормалізації, що знизило б продуктивність моделі, як було показано в документі EDSR і підтверджено в документі WDSR. З іншого боку, використання лише нормалізації ваги без ініціалізації, що залежить від даних, призводить до кращої точності більш глибоких моделей WDSR.

В оригінальному дослідженні [1] 2017 року під назвою “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network” порівнювали бікубічну інтерполяцію, SRResNet, SRGAN та оригінальну фотографію. SRGAN показав найкращий результат. В основі останнього лежить архітектура генеративних нейронних мереж. Перший документ [2] з’явився в 2014 році під назвою “Generative Adversarial Nets”. Ідея полягає в тому, що є 2 мережі які змагаються між собою - генератор та дискримінатор. Один генерує зображення і намагається обдурити, а задачі іншого розрізнити фото справжнє чи згенероване. Після деякого часу навчання генератор генерує дуже якісні фото, що навіть дискримінатор не може відрізнити реальне це фото чи ні.

1.3. Визначення профілів зацікавлених сторін

Основна мета дипломної роботи є розробка програмного модуля для покращення зображень.

Зацікавлені стороні:

- компаніїї, які обробляють фото;
- користувачі, яким потрібно відновити якість старої фотографії;
- користувачі, яким потрібно збільшити фото та не втратити якість, або навіть покращити фото;
- компанії, які розробляють ПЗ для камер;
- медицина(представники мед.галузі);
- компанії, які займаються знімками космосу.

Таблиця 1.1. Профілі зацікавлених сторін

Зацікавлена сторона	Плюси	Мінуси	Вимоги
Великі та середні компанії	Покращення фото одразу після знімку	Довге тренування нейромережі	Швидкодія, точність і мінімальна затрата пам’яті
Користувачі	Покращення фото	Довгий час очікування, ціна за фото	Покращене фото

1.4. Огляд існуючих рішень

Zyro.[6] Щоб покращити масштаб зображення без втрати якості, розробники продукту використовують глибокі згорткові нейронні мережі. Можна сказати, що це як мозок, але комп’ютерний. Ці мережі навчаються на великих наборах даних зображень, тому вони вивчають основні властивості всіх видів зображень. Наприклад, якщо комп’ютер бачить зображення цегли з низькою роздільною здатністю стіни, він дізнається, як виглядає текстура цегли, і заповнить деталі цегляної стіни. Таким чином, зображення стає більш реалістичним. Перед машинним навчанням і штучним інтелектом, коли ви хотіли отримати більше зображення, вам потрібно було повторити пікселі 2 або більше разів, щоб збільшити зображення, але в кінцевому підсумку все одно матимете блочний вигляд і розмитим зображенням.

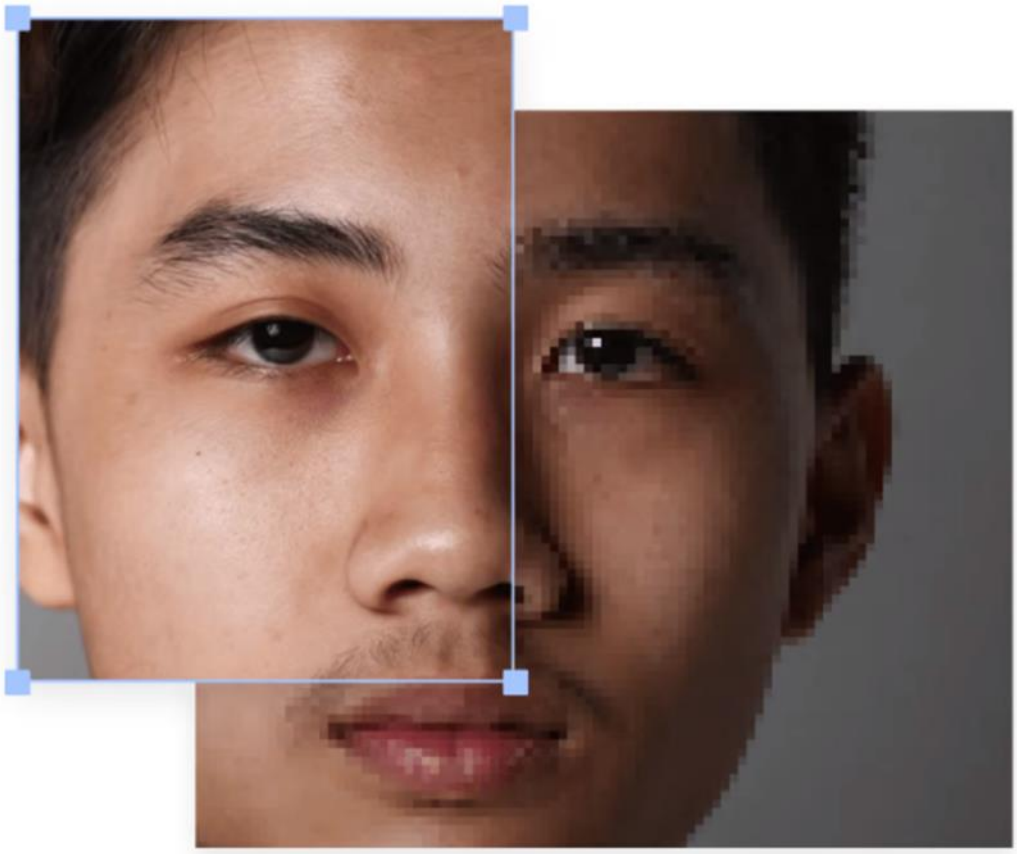

Рис. 1.5 Приклад покращення зображення з веб-сторінки

Let’s Enhance [7] використовує передову технологію Super Resolution на основі глибоких згорткових нейронних мереж. До появи цієї технології неможливо було різко збільшити розмір фото або зображення без втрати якості. Ваш найкращий варіант у Photoshop, який називається Bicubic Interpolation, зробив ваше зображення нечітким і розмитим. Ті з розробників, які займаються математикою, можуть посперечатися – як би не збільшувати роздільну здатність зображення – нової інформації про зображення немає – розробники просто не можете додати якість! Це не вірно у випадку, коли використовується нейронна мережа та AI. Нейронна мережа від Let`s Enhance навчається на величезному наборі даних зображень, тому вона вивчає типові особливості фізичних об’єктів – цеглини стін, волосся та шкіру. Після того, як мережа розпізнає ці функції на завантажених зображеннях і додасть додаткові деталі на основі своїх загальних знань про світ.

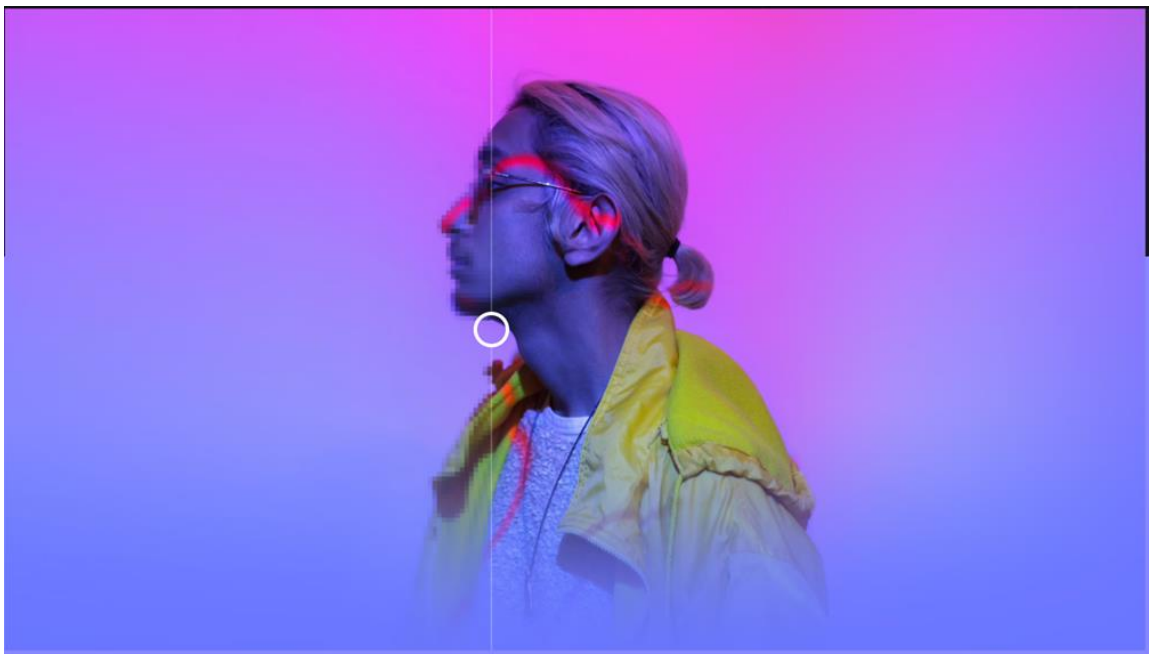

Рис . 1.6 Приклад покращення зображення з веб-сторінки

Topazlabs[5]**.** Регулярна обробка зображень «фільтрує» фото чи відео за допомогою складних математичних операцій. Це часто видаляє деталі та посилює шум/артефакти. ШІ принципово відрізняється: при правильному використанні він може покращити справжню якість зображення

На сайті Topazlabs є алгоритм роботи:

1. Розробка архітектури ШІ. Спочатку вчимо систему «як вчитися». Команда дослідників постійно експериментує з новими та вдосконаленими методами, натхненними останніми розробками в області ШІ
2. Далі розробники збирають та вводять мільйони точок даних, щоб допомогти системі зрозуміти, що означає «якість зображення». Це навчання зазвичай займає тижні або місяці.
3. Щоб забезпечити дуже гарні результати якості зображення, команда покладається як на об’єктивні показники якості, так і на суб’єктивну оцінку реальними фотографами.

Крім зображень вони вміють покращувати якість відео. В покращення відео входить:

- збільшення масштабу. Перетворення з SD в HD або HD 4k;

- зниження шуму. Видалить видимий шум із зображення, зберігаючи деталі, щоб очистити відео;
- різкість. Підвищує різкість без додавання артефактів.

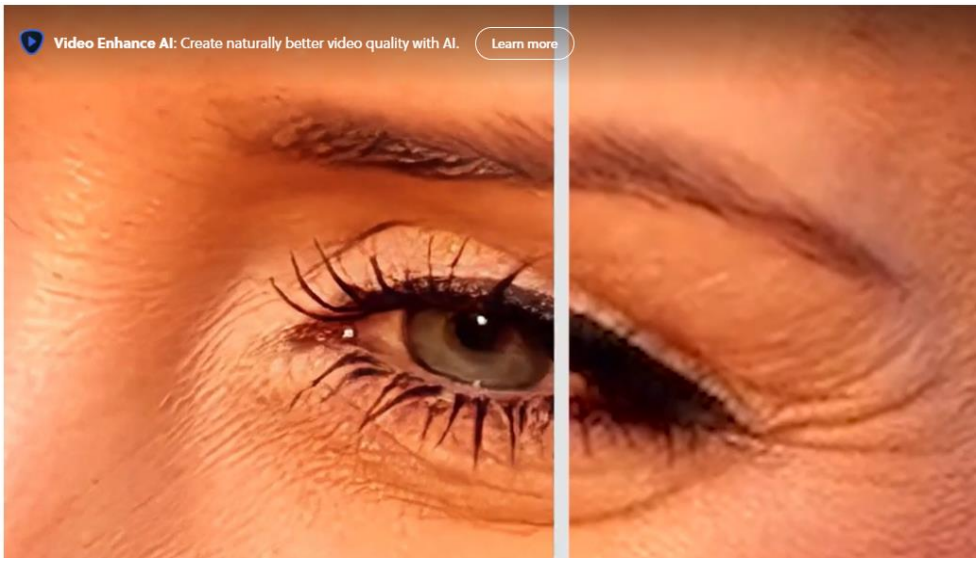

Рис 1.7 Приклад покращення відео з веб-сторінки

Усі ці компанії використовують схожий алгоритм тренування та нейромережі. Для індивідуальності фахівці видозмінюють підхід тренування та види НМ, але сама база залишається.

1.5. Постановка задачі

Об’єктом дослідження є засоби відновлення зображення

Буде розроблено веб-застосунок, який буде покращувати завантажене фото користувачем.

Предмет дослідження - методи відновлення /покращення якості зображень за допомогою інтелектуальних інформаційних технологій, зокрема нейронних мереж.

Метою дипломної роботи є покращення зображень з низькою якістю з використанням програмного застосунку; розробка застосунку для обробки вхідних зображень, які завантажив користувач сервісу.

Функціональні вимоги:

- простий UI для завантаження/порівняння зображень;
- алгоритм повинен тільки покращувати зображення;
- можливість завантажити зображення передбачена;
- відображення завантаженого зображення.

Нефункціональні вимоги:

- конфіденційність зображень;
- програмний продукт повинен працювати в кожному браузері;
- надійність;
- швидкість опрацювання зображення – приблизно від 20-40 секунд;
- обмеження(формат зображень– png та jpeg).

В процесі буде реалізовано нейромережу для покращення фото. Тренування нейромережі буде проводитися на великої кількості даних для кращих результатів. Для збільшення даних буде використовуватися аугментація даних:

- поворот зображення;
- зміщення зображення;
- перегортання зображення.

Для тренування мережі підійде будь-який датасет с гарною чіткістю фотографій.

Далі розглянемо детальніше діаграму процесу покращення зображення (рис 1.8).

На рисунку 1.8 зображено процес, який базується на покращення зображень. Складається із таких під процесів як:

1. запуск веб-сторінки;
2. вибір зображення;
3. перевірка зображення;
4. покращення зображення;
5. відправлення фото на сайт;
6. виведення помилки на экран.

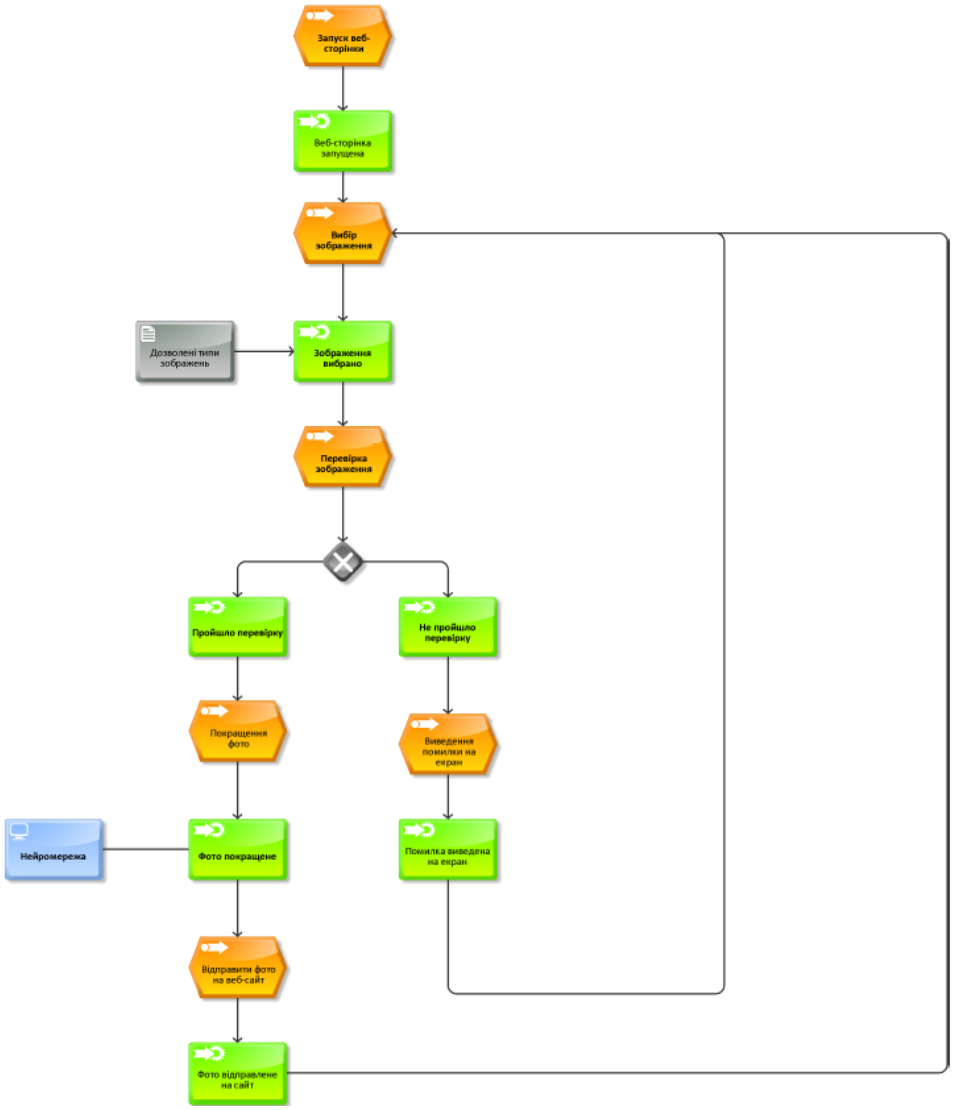

Рис 1.8 Узагальнена діаграма процесу покращення зображення

На наступному рисунку 1.9 зображено IDEF0 діаграму

Вхідна інформація: Будь-яке зображення користувача

Контролюючими та обмежуючими впливами виступають протокол передачі даних і закон про захист інтелектуальних даних

Виконавець: Користувач

Вихідна інформація: Покращене зображення, яке користувач може порівняти і скачати

Рис 1.9 IDEF0 діаграма

На наступному рисунку наведено VAD діаграма процесів покращення різкості зображень

Рис 1.10 VAD діаграма процесу покращення різкості зображень

Висновки до першого розділу

В ході виконання першого розділу було проведено аналітичний огляд кваліфікаційної роботи. Було проаналізовано область застосування майбутнього програмного модуля, визначено зацікавлені сторони, мету і вимоги до системи. Також було проаналізовано існуючі рішення та результати, що було досягнуто різними компаніями.

Було розроблено VAD, IDEF0-діаграми та узагальнену діаграму процесу розпізнавання зображення.

РОЗДІЛ 2. РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОГО МОДУЛЯ ПОКРАЩЕННЯ РІЗКОСТІ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

2.1 Архітектура програмного модуля

Програмний модуль для покращення різкості зображень буде включати в себе основні модулі на рис 2.1, такі як:

- модуль для покращення різкості зображень. Включає в себе код нейронної мережі, архітекутри та нетренованих ваг;
- модуль для взаємодії з користувачем. Включає в себе веб-сторінку для взаємодії з програмним забезпеченням;
- модуль для роботи з нейронною мережею. Включає в себе підготовлену архітектуру НМ для обробки зображень с застосунку
- модуль для обробки вхідних і вихідних зображень.Включає в себе функції для перетворення зображення в формат для нейронної мережі і навпаки.

Рис 2.1. Схема архітектури програмного модуля покращення зображень

2.2 Математична модель SRGAN генеративної змагальної мережі

Концепція SRGAN є однією з перших методик, яка дозволяє моделі досягти збільшення зображення майже в 2,4,6 та 8 разів. Ідея створення зображення з високою роздільною здатністю з зображень низькою роздільною здатністю є надзвичайно складним завданням. Раніше CNN використовували для створення зображень з високою роздільною здатністю, які тренуються швидше і досягають високої точності. Однак у деяких випадках вони не в змозі відновити дрібні деталі і часто створюють розмиті зображення. Архітектура SRGAN вирішує більшість цих проблем для створення високоякісних, найсучасніших зображень.

Більшість алгоритмів, які мають справу з супер-роздільністю, використовують середню квадратичну помилку(MSE) між отриманим зображенням з високою роздільною здатністю та основною істинністю конкретного зображення. Цей метод виявляється зручним, оскільки мінімізація середньоквадратичної помилки автоматично максимізує пікове відношення сигнал/шум (PSNR). PSNR є одним із найпоширеніших термінів, який використовується для оцінки зображень із надвисокою роздільною здатністю. Однак ці терміни більше орієнтовані на пошук особливостей кожного окремого пікселя, а не на більш візуально сприйнятливі атрибути, такі як висока деталізація текстури конкретного зображення.

Отже, у роботі щодо створення фото реалістичного єдиного зображення супер-роздільної здатності за допомогою генеративної змагальної мережі пропонується помилка, яка налаштована на боротьбу з більш орієнтованими на сприйняття ознаками за допомогою нещодавно введеної помилки, яка називається втратою сприйняття. VGG Loss — це тип втрати вмісту, представлений у системі Perceptual Loss for Real-Time Style Transfer і Super-Resolution і Super-Resolution. Втрата сприйняття є поєднанням втрат протиборства та втрати вмісту. Формулювання цієї втрати можна інтерпретувати наступним тлумаченням

$$l^{SR} = \underbrace{\underbrace{l_{\mathrm{X}}^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$

Рис. 2.2 Функція втрат

Ця функція втрат є краще ніж середньоквадратична втрата помилки, тому що розробники не піклуються про піксельне порівняння зображень. Тому що більше всього турбує покращення якості зображень. Отже, використовуючи цю функцію втрат у моделі SRGAN, програмісти можуть досягти більш бажаних результатів.

Архітектура генератора в основному є повністю згортковою моделлю SRRESNET, яка використовується для створення високоякісних зображень надвисокої роздільної здатності. Додавання моделі дискримінатора, яка діє як класифікатор зображень, створено для того, щоб гарантувати, що загальна архітектура налаштовується відповідно до якості зображень, і отримані зображення є набагато більш оптимальними. SRGAN модель створює правдоподібно зображення з високою якістю сприйняття.

Архітектура генератора (рис 2.3) повністю відповідає SRRESNET моделі складається з входу низької роздільної здатності, який пропускається через початковий згортковий шар із 9×9 ядер і 64 карт ознак, за якими слідує параметричний шар ReLU. Помітно, що вся архітектура генератора використовує Parametric ReLU як основну функцію активації. Причина вибору Parametric ReLU полягає в тому, що це одна з найкращих нелінійних функцій для цієї конкретної задачі зіставлення зображень з низькою роздільною здатністю до зображень з високою роздільною здатністю.

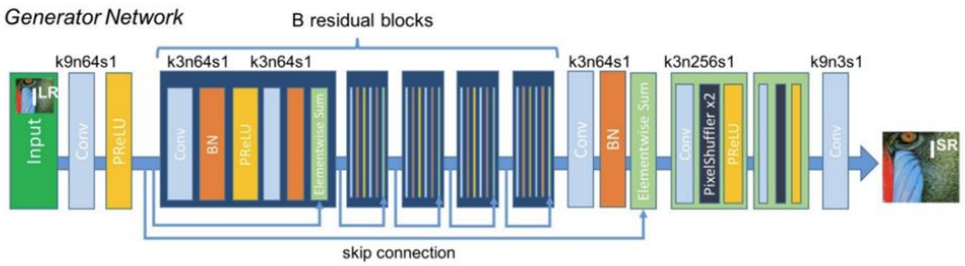

Рис 2.3 Архітектура генератора

Функція активації, як-от ReLU, також може виконувати наступне завдання, але є проблеми, які можуть виникнути через концепцію мертвих нейронів, коли значення, менші за нуль, відображаються безпосередньо на нуль. Альтернативним варіантом є Leaky ReLU, де значення, менші за нуль, зіставляються з числом, встановленим користувачем. Однак у випадку параметричного ReLU розробники можуть дозволити нейронній мережі вибирати найкраще значення самостійно, і, отже, є кращим у цьому сценарії.

$$ReLU(x) = \max(0, x), \tag{2.1}$$

$$LeakyRELU(x) = \begin{cases} x, & x \geq 0 \\ \text{негативний нахил} * x, & x < 0 \end{cases} \tag{2.2}$$

Наступний рівень повністю згорткової моделі SRRESNET із прямим зв’язком використовує багато залишкових блоків. Кожен із блоків містить згортковий шар з ядра 3×3 і 64 карти ознак, за якими слідує шар нормалізації, параметрична функція активації ReLU, інший згортковий шар із нормалізацією та остаточний поелементний метод суми. Метод поелементної суми використовує вихід з прямим зв’язком разом із виходом з’єднання пропуску для надання кінцевого результату.

Ключовий аспект, на який слід звернути увагу щодо наступної архітектури нейронної мережі, полягає в тому, що кожен із згорткових шарів використовує подібне заповнення, щоб розмір наступних входів і виходів не змінювався. На відміну від інших повністю згорткових мереж, таких як архітектура U-Net, часто

використовують шари об’єднання для зменшення розміру зображення. Однак для моделі SRGAN не потрібно наведене вище, оскільки розмір зображення не потрібно зменшувати. Натомість прагнемо досягти протилежного.

Після побудови залишкових блоків будується решта моделі генератора, як показано на зображенні вище. Використовуючи зміну місцями пікселів у цій архітектурі моделі генератора після 4-кратного підвищення дискретизації згорткового шару для отримання зображень із кращою роздільною здатністю. Заміна пікселів бере значення з розміру каналу та вставляють їх у висоту та ширину. У цьому випадку і висота, і ширина множаться на два, а канал ділиться на два.

Архітектура дискримінатора(рис 2.4) створена для підтримки типової процедури GAN. І генератор, і дискримінатор конкурують один з одним, і вони обидва вдосконалюються одночасно. У той час як мережа дискримінатора намагається знайти підроблені зображення, генератор намагається створити реалістичні зображення, щоб він міг уникнути виявлення з боку дискримінатора. Подібна робота і у випадку SRGAN, де генеративна модель G з метою обдурити диференційований дискримінатор D, який навчений відрізняти зображення з супер роздільною здатністю від реальних зображень.

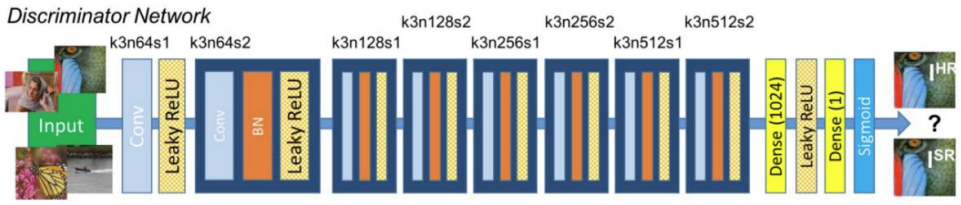

Рис 2.4 Архітектура дискримінатора

Таким чином, архітектура дискримінатора, показана на зображенні вище(рис 2.4), працює, щоб розрізняти зображення з надвисокою роздільною здатністю та реальні зображення. Сконструйована модель дискримінатора спрямована на вирішення змагальної задачі min-max. Загальну ідею формулювання цього рівняння можна інтерпретувати так:

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR} \sim p_{train}(I^{HR})} \left[\log D_{\theta_D}(I^{HR})\right] + \mathbb{E}_{I^{LR} \sim p_G(I^{LR})} \left[\log (1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))\right] \quad (2.3)$$

Створена архітектура дискримінатора є досить інтуїтивно зрозумілою та легкою для розуміння. Використовуємо початковий згортковий шар, за яким слідує функція активації Leaky ReLU. Значення негативного фактору для Leaky ReLU для цієї структури встановлено на 0,2. Далі у нас є багато повторюваних блоків згорткових шарів, за якими слідує шар пакетної нормалізації та функція активації Leaky ReLU. Коли є п’ять із цих повторюваних блоків, маємо щільні шари, за якими слідує функція активації сигмовидної форми для виконання дії класифікації. Початковий розмір згортки становить 64 x 64, який множиться на 2 після двох повних блоків кожен, поки не досягнемо 8-кратного коефіцієнта збільшення 512 x 512. Ця модель дискримінатора допомагає генератору вчитися більш ефективно та отримувати кращі результати.

2.3 Функціональний аналіз

Отримання завантаженого зображення:

- перевірка розширення файлу(після завантаження на сервер файл проходить перевірку на розширення за допомогою python функції);
- встановлення фіксованого розміру зображення на сторінці(Це зроблене для кращого користувацького досвіду);

Обробка зображення:

- Перетворення зображення в математичний формат(за допомогою бібліотеки tensorflow перетворюємо зображення в зрозумілий формат для НМ)
    - Завантаження в нейронну мережу(Після перетворення завантажуємо зображення в нейронну мережу);

Збереження покращеного зображення:

- Завантаження в папку(Після завантаження зображення зберігається в локальній теці)
    - Відправлення на веб сторінку(після відпрацювання нейронної мережі зображення відправляється за допомогою бекенд частини на фронтенд)

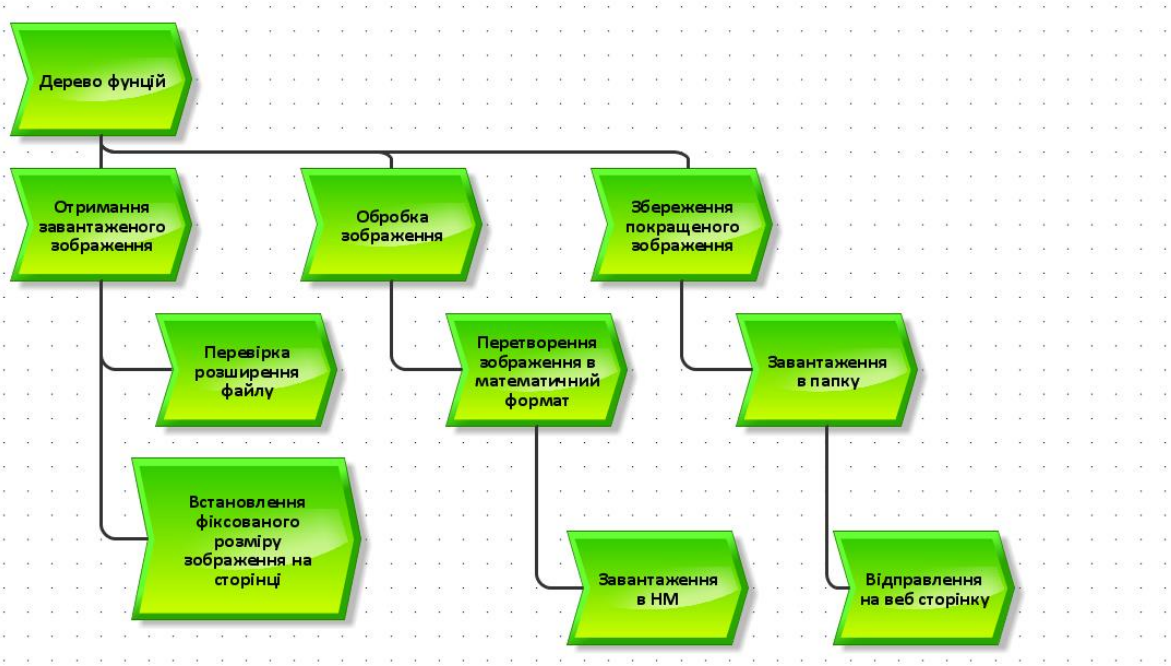

Рис. 2.5 Дерево функцій

2.4 Вибір середовища розробки ПМ покращення зображень

Для початку тренування буде відбуватися на MSI GS63VR.

Характеристики:

- процесор Intel i7;
- відеокарта Nvidia GEFORCE GTX 1060;
- відеопам’ять 6GB GDDR5;
- оперативна пам’ять 16 GB.

Якщо тренування мереж буде займати дуже багато часу, то альтернативою будуть виступати хмарні застосунки: Google Colab та Google cloud

Для веб-застосунку буде застосовуватися мова програмування Python із використанням веб-фреймворку Flask.

Flask — це легка структура веб-додатків WSGI. Він розроблений для того, щоб розпочати роботу швидко та легко, з можливістю масштабування до складних програм. Вона починалася як проста обгортка навколо Werkzeug і Jinja і стала однією з найпопулярніших фреймворків веб-додатків Python [23].

WSGI — стандарт взаємодії між програмою Python, що виконується на стороні сервера, і самим веб-сервером, наприклад Apache.

Flask пропонує пропозиції, але не накладає жодних залежностей чи макета проекту. Розробник має вибрати інструменти та бібліотеки, які вони хочуть використовувати. Спільнота надає багато розширень, які полегшують додавання нових функцій[23].

Для навчання нейромереж буде використовуватися Tensoflow з Keras.

TensorFlow — це наскрізна платформа з відкритим кодом для машинного навчання. Він має всеосяжну, гнучку екосистему інструментів, бібліотек і ресурсів спільноти, яка дозволяє дослідникам використовувати найсучасніші технології машинного навчання, а розробникам легко створювати й розгортати програми, що працюють на ML. Отже, TensorFlow - це дуже потужна і зріла бібліотека глибокого навчання з потужними можливостями візуалізації та кількома опціями для розробки моделей високого рівня. Він має готові варіанти розгортання та підтримку мобільних платформ[10]. TensorFlow є хорошим варіантом, якщо для розробника підходять такі пункти:

- розробка моделей для production;
- розробляйте моделі, які потрібно розгорнути на мобільних платформах;
- потрібна хороша підтримка спільноти та вичерпна документація;
- хочете мати багато навчальних ресурсів;
- якщо потрібно використовувати Tensorboard (візуалізація даних);

- необхідність використання широкомасштабної розподіленої моделі навчання.

Конкурентом Tensorflow є PyTorch, але цей фреймворк є дуже молодим.

Цей фреймворк використовують для:

- досліджень;
- швидка розробка коду, для перевірки концепції;
- якщо хочете отримати кращий досвіду розробки та відлагодження коду.

На Github TensorFlow має в 2 раза більше зірок чим Pytorch. В дипломній роботі використати TensorFlow.

Keras - це високорівневий API TensorFlow 2: доступний, високопродуктивний інтерфейс для вирішення проблем машинного навчання з акцентом на сучасне глибоке навчання. Він надає основні абстракції та будівельні блоки для розробки та доставки рішень машинного навчання з високою швидкістю ітерацій[9].

Keras дає можливість інженерам та дослідникам повною мірою скористатися можливостями масштабованості та кросплатформенності TensorFlow 2: Розробники можуть запускати Keras на TPU або на великих кластерах графічних процесорів, а також можете експортувати свої моделі Keras для запуску у браузері або на мобільному пристрої. пристрій[9].

Jinja2 — сучасна мова шаблонів для розробників Python. Він був зроблений за шаблоном Django. Він використовується для створення HTML, XML або інших форматів розмітки, які повертаються користувачеві через HTTP-запит[22].

Для тренування буде використовуватися MIRFLICKR. Датасет має 25 000 зображень.

Python – це інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Python має дуже простий, легкий у засвоєнні синтаксис, який підкреслює читабельність і знижує витрати на можливу підтримку програмного коду. Однією з переваг інтерпретатора є те, що

можна почати інтерактивний сеанс з інтерпретатором і ввести код Python прямо в нього, щоб побачити, що він робить. Це дієвий спосіб спробувати ідеї коду[20].

Оскільки Python дуже легко читати, розробники можуть зосередитися на створенні рішень замість розшифровки природи мови програмування. Завдяки динамічному введенню Python присвоює типи даних під час виконання програми. Таким чином, розробнику не доведеться турбуватися про оголошення змінних або вказівок типів даних під час кодування[21].

Порівнюючи з іншими мовами програмування високого рівня, такими як Java, програмісти можуть вибрати Python через його потужні функції інтеграції, які роблять мову програмування кращим вибором для створення корпоративних програмних програм. Python має величезну колекцію бібліотек. Розробнику не потрібно залежати від зовнішніх бібліотек, оскільки програміст має більш ніж достатньо функцій, які знадобляться для виконання проекту[21].

Незважаючи на широку розробку програмного забезпечення, можливо, немає жодного аспекту, над яким можна працювати без підтримки з великої бібліотеки Python.Окрім більш поширених проектів розробки веб та мобільного програмного забезпечення, Python також має бібліотеки, які можуть допомогти вам із машинним навчанням, моделюванням AI (штучного інтелекту), розробкою відеоігор тощо.Багато складних процесів спрощуються завдяки вбудованій стандартній бібліотеці Python, тому, мабуть, немає нічого, що можна собі уявити, що б не можна було б втілити в життя за допомогою Python — навіть без використання зовнішньої бібліотеки. Якщо потрібно буде використовувати зовнішню бібліотеку, то можна використати менеджер пакетів Python (pip) та індекс Python package (PyPi).В індексі пакетів Python є понад 200 000 пакетів, з якими можна працювати, і можна імпортувати ці пакети за допомогою менеджера пакетів Python. Python безкоштовний, з відкритим вихідним кодом і має велику спільноту людей. Python є улюбленим для багатьох розробників програмного забезпечення, тому що вони мають одну з найбільш підтримуваних спільнот у світі[21].

Google Colaboratory – безкоштовне середовище, щоб писати код у jupyter notebook. Сервіс надає доступ до графічних процесорів GPU та TPU. Завдяки цим потужностям можна досліджувати та тренувати нейронні мережі різної складності. Всі розрахунки на GPU/TPU є безкоштовними, але через 6-12 годин середовище перестає працювати і код перестає далі виконуватися. До Goole Colab дуже можна дуже легко під'єднати Google Drive, щоб скачувати або завантажувати фотографії для тренування НМ[24].

GCP(Google Cloud Platform) . GCP є постачальником хмарних обчислень таких як Amazon Web Services (AWS), Microsoft Azure та Oracle Cloud. Завдяки GCP та іншим хмарним постачальникам клієнти можуть отримати доступ до комп'ютерних ресурсів, розміщених у центрах обробки даних Google по всьому світу, безкоштовно або за плату за використання. GCP пропонує набір обчислювальних послуг, щоб робити все: від управління витратами GCP до керування даними від доставки застосунків в хмару до тренування штучного інтелекту та нейронних мереж. Самий популярний безкоштовний сервіс для тренування нейронних мереж є Google Colab, який базується на Google Cloud і тому досі Google Cloud є лідером серед розробкою нейронних мереж [25].

2.5 Макет інтерфейсу застосунку

Сайт буде складатися з 2 сторінок. На першій сторінці спочатку потрібно буде завантажи файл за допомогою кнопки "Choose file", а потім відправити на backend частину за допомогою кнопки "Load". Після цих дій потрібно натиснути "Upgrade" і завантажене зображення відправиться в НМ для покращення різкості.

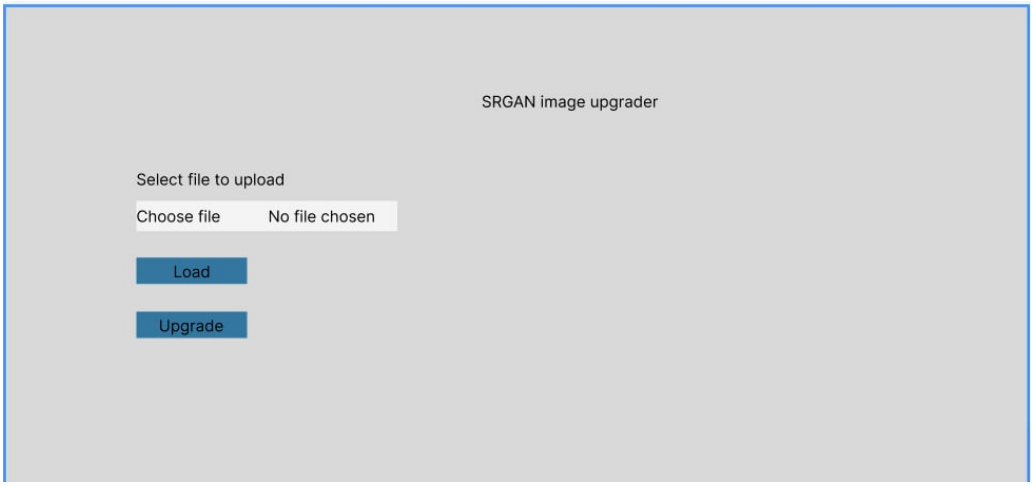

Рис. 2.7 Макет першої сторінку сайту

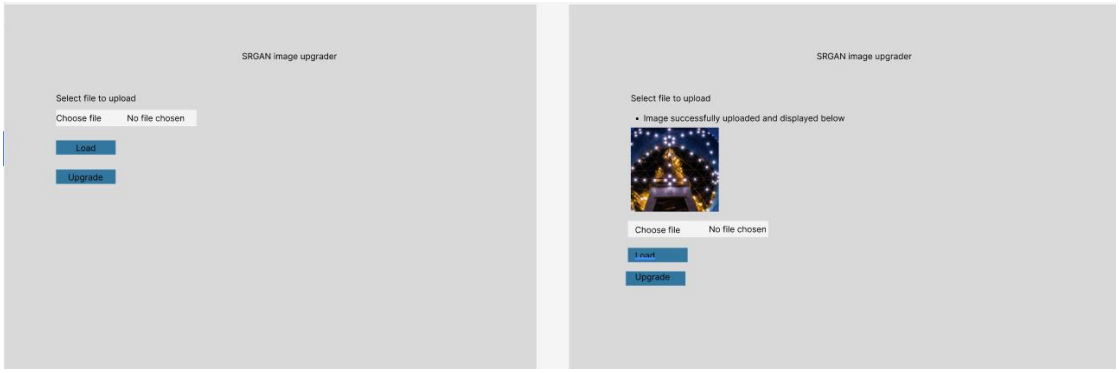

Рис. 2.8 Макет сторінки після завантаження зображення

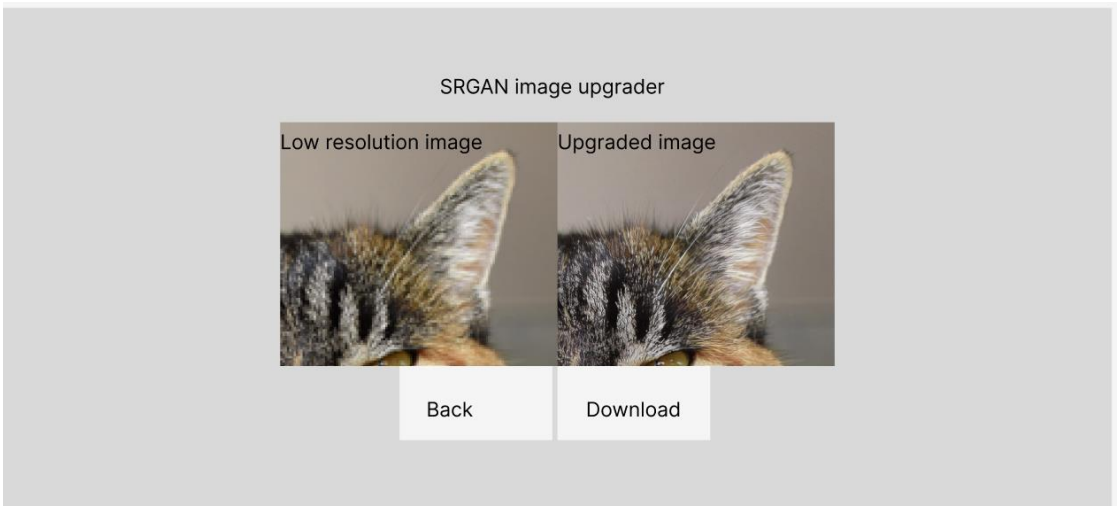

Рис. 2.9 Макет другої сторінку сайту

На рисунку вище можна побачити і порівняти результати роботи після натиснення кнопки “Upgrade”. Покращену картинку можна завантажити, або

Висновки до другого розділу

Отже, в результаті виконання другого розділу дипломної робити було проведено функціональний аналіз та побудовано архітектури програмного модуля.

Розроблено фізичну архітектуру системи в якій описано повністю структуру проекту.Створено макет веб сторінки. В наступному розділі буде втілено розроблений макет, написання документації для користувача, провдення тестування застосунку.

РОЗДІЛ 3. ТЕХНОЛОГІЧНІ ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО МОДУЛЯ

3.1 Структура програмного забезпечення

Розглянемо структурну схему програмних модулів, яка наведена на рис. 3.1.

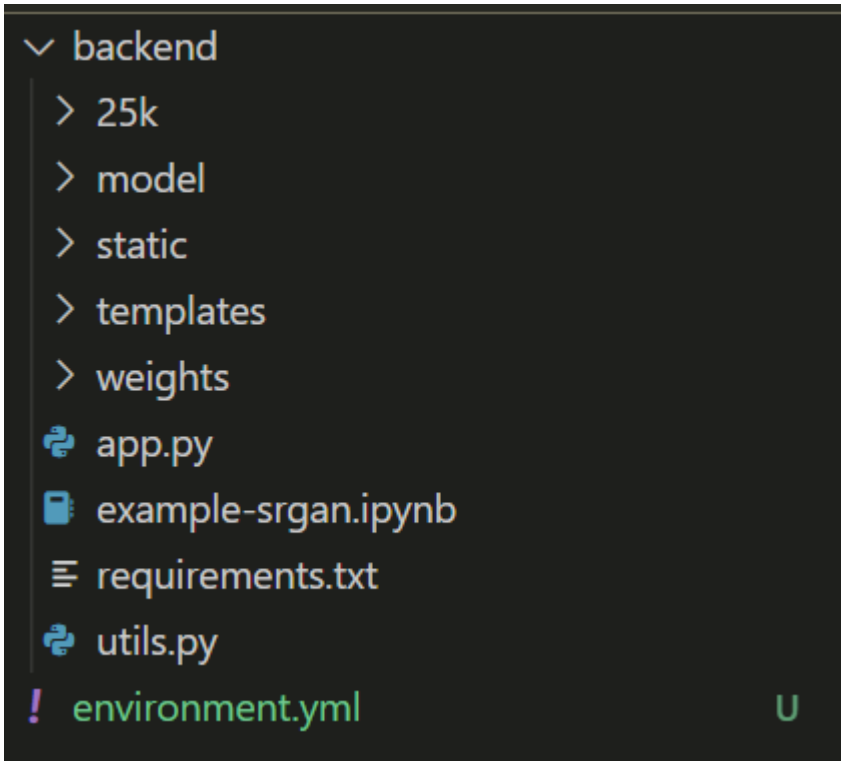

Рис. 3.1 Структура програмного забезпечення

Система складається із наступних об’єктів:

app.py–backend частина застосунку.

функція upload_image перевіряє розширення завантаженого зображення і потім відображає завантаження на веб-сторінці;

flask функції для роутінгу(/ та /upgrade);

функція save_upgraded_image завантажує натреновані ваги НМ та архітектуру. Потім обробляє вхідне зображення та зберігає покращене зображення.

environment.yml – файл для налаштування середовища розробки;

model – папка в якій є код нейронної мережі для тренування та для виконання в веб застосунку, який написаний на Keras;

static – в цій папці тимчасово зберігаються завантажені зображення з веб сторінки;

templates – папка з html файли для frontend частини;

weights – натреновані параметри нейронної мережі;

utils.py – додаткові функції для збереження та завантаження зображень;

25k – ця папка містить датасет з 2 типами зображень – low resolution та high resolution.

3.2 Тренування, встановлення та запуск розробленої системи

3.2.1 Тренування нейронної мережі

Перші тренування відбувалися за допомогою Jupyter Notebook в Goole Colab. Відеокарту використовував Tesla T4 на 15 GB(рис 3.2). Весь процес розробки є в даному ноутбуці, від тренування ГЗМ до повного завантаження датасету в пам’ять.

```
!nvidia-smi

Wed Jun 15 17:02:31 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   35C    P8     9W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

3.2 Характеристика відеокарти в Google colab

Для тестування НМ перших 10 епох було вирішено перевірити на тестових даних(рис3.3). Зліва вхідне зображення, а по середині згенероване зображення.

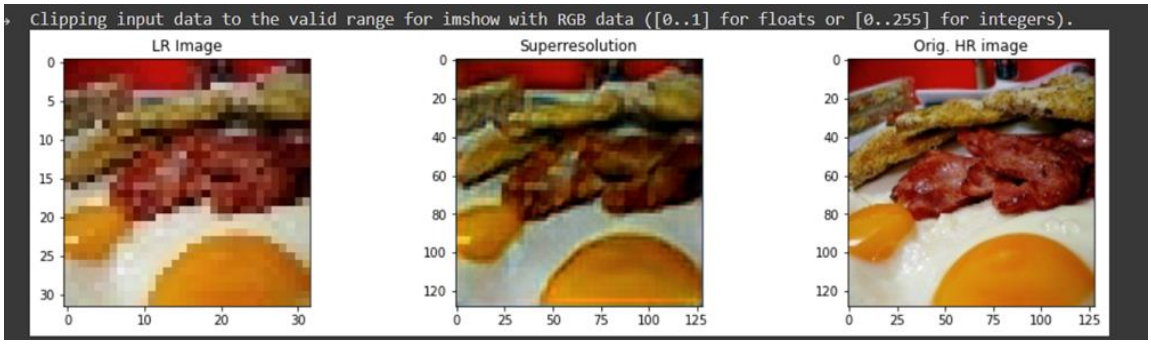

Рис 3.3 Тестові дані

Нейронна мережа потребує дуже довгого навчання і тому було вирішено запустити НМ в Google Cloud Platform(GCP). Відеокарта теж Tesla T4, але вже з більшою кількістю пам’яті.

```
bmaximxx@instance-3:~$ nvidia-smi
Wed Jun 15 17:13:06 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   59C    P0    29W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Рис 3.4 Характеристика відеокарти в GCP

Весь код було перенесено з Jupyter Notebook в python файли для кращої взаємодії з віддаленим сервером.

```
bmaximxx@instance-3:~$ ls -la
total 2997340
drwxr-xr-x 12 bmaximxx bmaximxx       4096 Jun 14 22:12 .
drwxr-xr-x  4 root     root           4096 Jun 14 16:36 ..
-rw-------  1 bmaximxx bmaximxx       1243 Jun 15 16:44 .bash_history
-rw-r--r--  1 bmaximxx bmaximxx        220 Apr 18  2019 .bash_logout
-rw-r--r--  1 bmaximxx bmaximxx       3526 Apr 18  2019 .bashrc
drwxr-xr-x  4 bmaximxx bmaximxx       4096 Jun 14 16:45 .cache
drwxr-xr-x  5 bmaximxx bmaximxx       4096 Jun 14 17:54 .config
drwxr-xr-x  2 bmaximxx bmaximxx       4096 Jun 14 16:38 .docker
drwx------  3 bmaximxx bmaximxx       4096 Jun 14 16:36 .gnupg
drwxr-xr-x  3 bmaximxx bmaximxx       4096 Jun 14 16:50 .keras
drwxr-xr-x  3 bmaximxx bmaximxx       4096 Jun 14 16:44 .local
drwx------  3 bmaximxx bmaximxx       4096 Jun 14 16:50 .nv
-rw-r--r--  1 bmaximxx bmaximxx        807 Apr 18  2019 .profile
drwx------  2 bmaximxx bmaximxx       4096 Jun 15 17:13 .ssh
drwxr-xr-x  5 bmaximxx bmaximxx       4096 Jun 14 16:45 25k
-rw-r--r--  1 bmaximxx bmaximxx 3069184257 Jul  7  2016 mirflickr25k.zip
-rw-r--r--  1 bmaximxx bmaximxx        937 Jun 15 06:51 out.txt
-rw-r--r--  1 bmaximxx bmaximxx         59 Jun 14 20:25 output.txt
-rw-r--r--  1 bmaximxx bmaximxx        486 Jun 14 16:44 prepare_data.py
-rw-r--r--  1 bmaximxx bmaximxx          0 Jun 14 20:57 res.txt
-rw-r--r--  1 bmaximxx bmaximxx       7337 Jun 14 22:12 srgan.py
drwxr-xr-x  2 bmaximxx bmaximxx       4096 Jun 15 06:51 weights
```

Рис 3.5 Структура linux директорії

У разі припинення online зв’язку з сервером було розроблено весь консольний вивід в файл.

```
bmaximxx@instance-3:~$ cat out.txt
TEST----------------
im9998.jpg
im9998.jpg
im1.jpg
im1.jpg
epoch: 1 g_loss: 182.77387957481102 d_loss: [0.38594938 0.94962963]
epoch: 2 g_loss: 333.3642334414765 d_loss: [0.04146019 0.99481481]
epoch: 3 g_loss: 78.95387925624847 d_loss: [0.04201497 0.99811111]
epoch: 4 g_loss: 57.117681348429784 d_loss: [0.05197186 0.99918519]
epoch: 5 g_loss: 54.284711702788314 d_loss: [0.04686601 0.99922222]
epoch: 6 g_loss: 46.79159854608112 d_loss: [0.01129541 0.9997037 ]
epoch: 7 g_loss: 55.63580034521774 d_loss: [0.03867595 0.99955556]
epoch: 8 g_loss: 44.86675009395458 d_loss: [0.0622442  0.99918519]
epoch: 9 g_loss: 52.53497142487985 d_loss: [0.04484167 0.99940741]
epoch: 10 g_loss: 40.53281338956621 d_loss: [0.05720443 0.99911111]
epoch: 11 g_loss: 51.32739052326591 d_loss: [0.05500227 0.99937037]
epoch: 12 g_loss: 698.3113197011594 d_loss: [0.04392963 0.99951852]
epoch: 13 g_loss: 53.86076948658625 d_loss: [0.0046465  0.99992593]
```

Рис 3.6 Навчання НМ.

Ваги кожної епохи можна знайти в папці weights/.

```
bmaximxx@instance-3:~/weights$ ls -la
total 108428
drwxr-xr-x  2 bmaximxx bmaximxx    4096 Jun 15 06:51 .
drwxr-xr-x 12 bmaximxx bmaximxx    4096 Jun 14 22:12 ..
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 14 22:51 gen_e_1.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 04:57 gen_e_10.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 05:37 gen_e_11.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 06:16 gen_e_12.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 06:51 gen_e_13.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 14 23:31 gen_e_2.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 00:12 gen_e_3.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 00:52 gen_e_4.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 01:33 gen_e_5.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 02:15 gen_e_6.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 02:56 gen_e_7.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 03:37 gen_e_8.h5
-rw-r--r--  1 bmaximxx bmaximxx 8538888 Jun 15 04:18 gen_e_9.h5
```

Рис 3.7 Натреновані ваги НМ

3.2.2 Встановлення програмного забезпечення

Для встановлення допоміжних пакетів потрібно запустити спеціальний пакетний мереджер pip і встановити за допомогою команди pip install -r requirements.txt(Рис 3.8)

```
D:\4course\2_PART\diplom\programming\super-resolution\backend>pip install -r requirements.txt
```

Рис 3.8 Встановлення допоміжних пакетів

Після цієї команди менеджер встановить всі потрібні пакети.

3.2.3 Запуск backend частини

Під час розробки програмного модуля було розроблено функціонал, що домогає покращити різкість завантажених користувачем зображень. Для початку потрібно запустити командний рядок cmd. Потім за допомогою команди cd(change directory) перейти в папку backend/ і за допомогою python запустити backend частину.

```
D:\4course\2_PART\diplom\programming\super-resolution\backend>python app.py
2022-06-03 19:12:45.217014: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic
library cudart64_101.dll
Using TensorFlow backend.
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Рис 3.9 Запуск backend частини

Останній рядок підказує куди потрібно зайти. Відкриваємо браузер і переходимо на локальний хост з портом 5000.

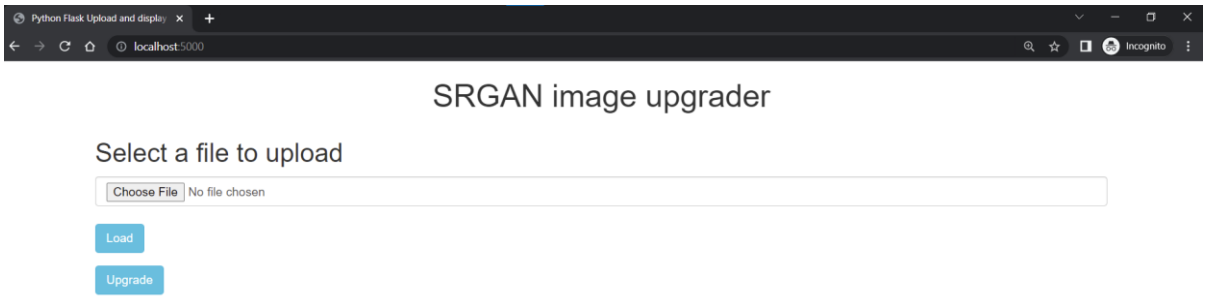

Рис 3.10 Результат переходу на локальний хост

3.3 Інструктивний матеріал користувача для роботи з модулем покращення різкості зображень

1. Натискаємо на choose file і обираємо фотографію

Рис 3.5 Вибір зображення

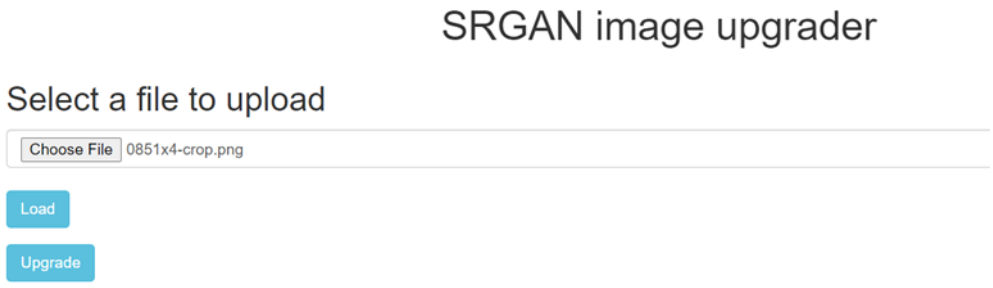

Рис 3.11 Результат вибору файлу

2. За допомогою Load завантажуємо вибрану картинку

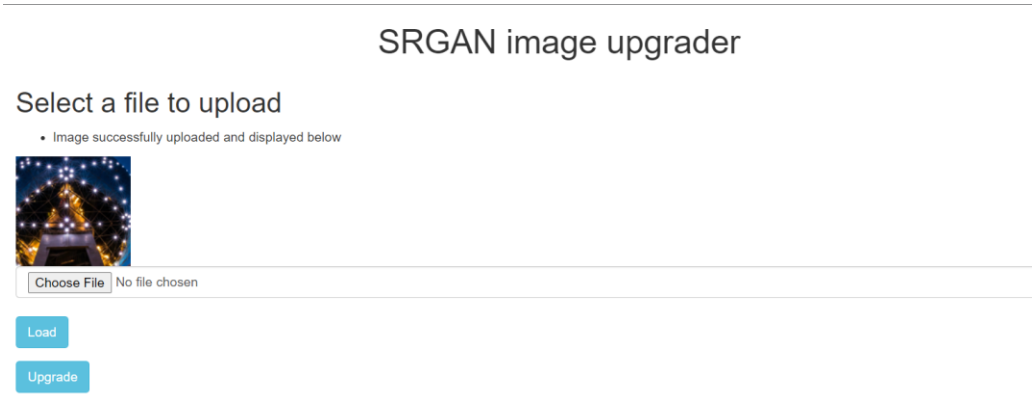

Рис 3.12 Результат завантаження картинки

3. Натискаємо кнопку Upgrade і чекаємо декілька секунд.

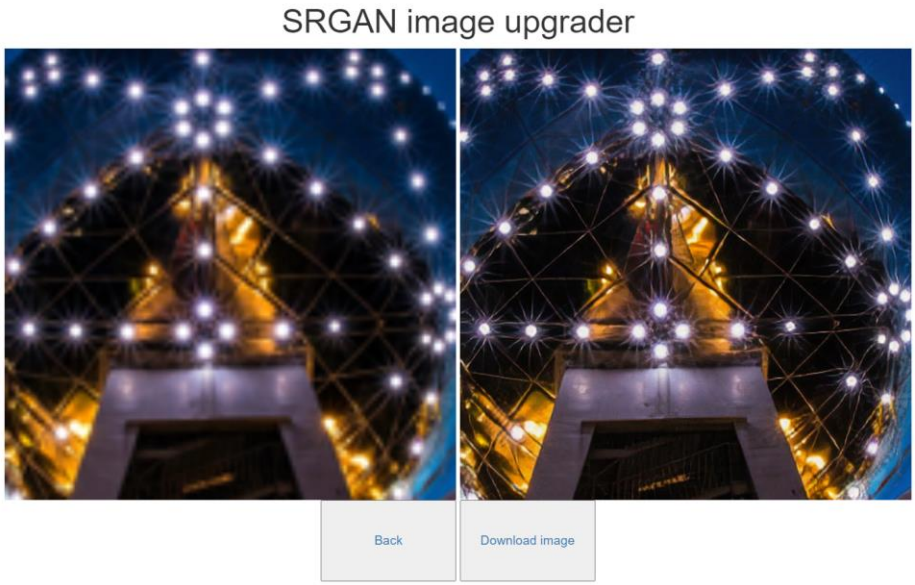

Рис 3.13 Результат покращення зображення

3.4 Тестування розробленого модуля покращення різкості зображень.

Розглянемо основні тестові випадки під час роботи програмного забезпечення:

Завантаження зображення малого розміру 32x32(таблиця 3.1);

Таблиця 3.1 Тестування зображення малого розміру

Назва	Покращення зображення малого розміру
Передумова	Відкрита сторінка застосунку
Кроки	1. Завантажити зображення малого розміру 2. Натиснути на кнопку «Load» 3. Натистути на кнопку «Upgrade»
Очікуваний результат	Покращення різкості зображення

Рис 3.14 Покращення різкості зображення малого розміру

Завантаження зображення середнього розміру 256x265(таблиця 3.2);

Таблиця 3.2 Тестування зображення середнього розміру

Назва	Покращення зображення середнього розміру
Передумова	Відкрита сторінка застосунку
Кроки	1. Завантажити зображення середнього розміру 2. Натиснути на кнопку «Load» 3. Натистути на кнопку «Upgrade»
Очікуваний результат	Покращення різкості зображення

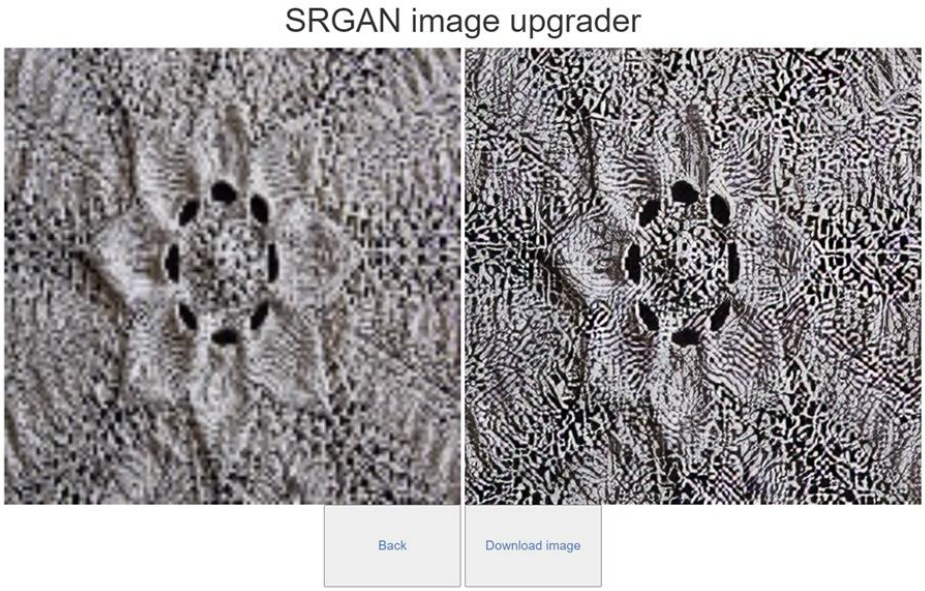

Рис 3.15. Покращення різкості зображення середнього розміру

Завантаження зображення великого розміру 512x512 (таблиця 3.3);

Таблиця 3.3 Тестування зображення великого розміру

Назва	Покращення зображення великого розміру
Передумова	Відкрита сторінка застосунку
Кроки	1. Завантажити зображення великого розміру 2. Натиснути на кнопку «Load» 3. Натистути на кнопку «Upgrade»
Очікуваний результат	Покращення різкості зображення

Результат виконання функції визначення параметрів системи наведено на рис. 3.10.

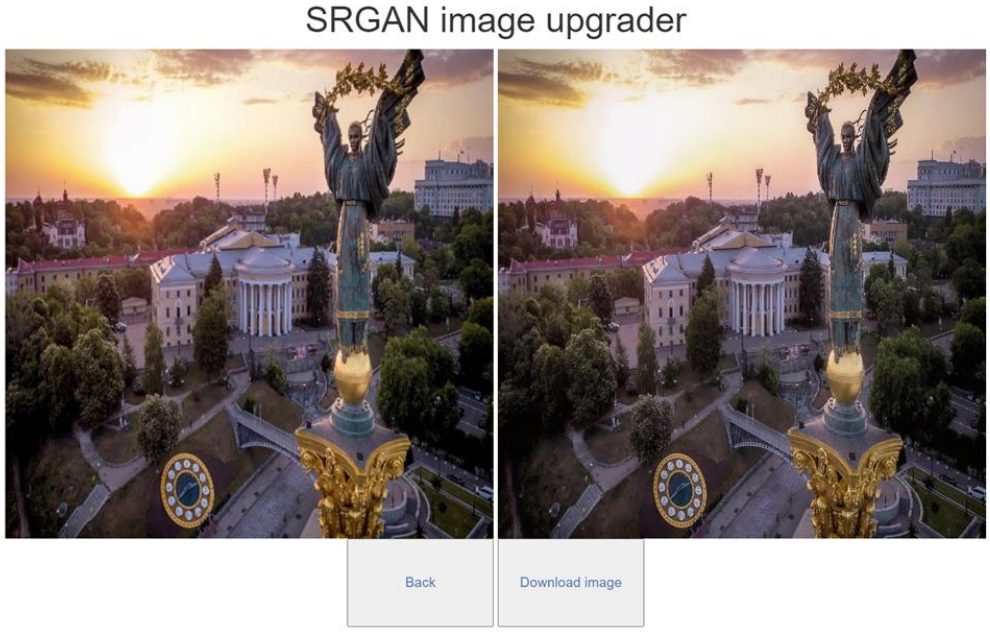

Рис 3.16. Покращення різкості зображення великого розміру

завантаження txt файлу(таблиця 3.4);

Таблиця 3.4 Тестування завантяження txt файлу

Назва	Визначення параметрів системи
Передумова	Відкрита сторінка застосунку
Кроки	1. Завантажити зображення малого розміру 2. Натиснути на кнопку «Load»
Очікуваний результат	Помилка після натискання кнопки «Load»

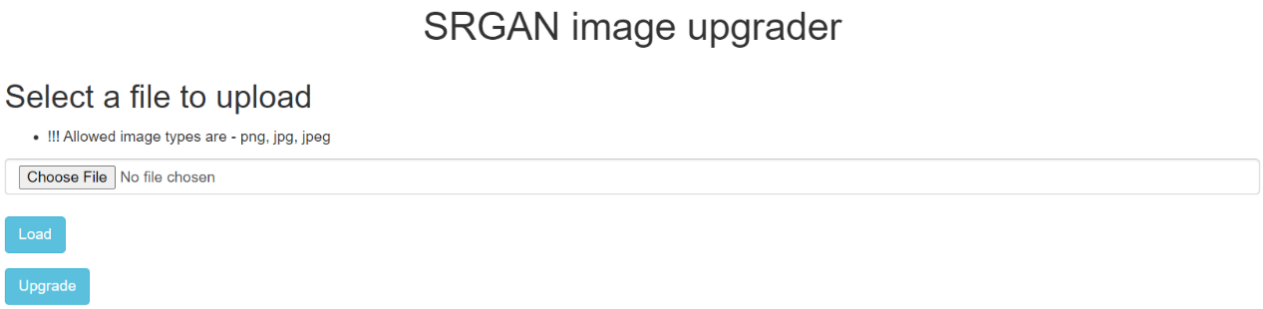

Рис 3.17 Виведення помилки на екран

Висновки до третього розділу

В результаті роботи над третім розділом дипломної роботи було розроблено інструкція для тренування, встановлення та запуск розробленої системи.

Розроблена структура програмного забезпечення була перенесена в хмарне середовище GCP. Всі ваги були збережені в хмарному середовищі та повний цикл тренування відбувався в хмарі.

Було проведено тестування розробленої системи, в рамках якої отримано очікувані результати та розроблена інструкція для користувача.

ВИСНОВКИ

В ході виконання випускної кваліфікаційної роботи, було розроблено програмний модуль покращення різкості зображень за допомогою нейронних мереж

Відповідно до поставленої мети, програмне рішення приймає вхідне зображення та покращує різкість зображення.

Під час виконання даної роботи було виконано аналіз існуючих видів нейронних мереж, досліджено технології, які використовуються для покращення різкості зображення, визначено область застосування, постановку задачі, побудовано програмне рішення, проведено тестування.

Розроблене рішення може бути впроваджено в комерційну діяльність після після узгодження з власниками датасетів, або створити власний датасет і натренувати ще раз нейронну мережу.

Список використаної літератури

1. Офіційний сайт arxiv [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1609.04802.pdf
2. Офіційний сайт arxiv [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1406.2661.pdf
3. Офіційний сайт medium [Електронний ресурс] – Режим доступу до ресурсу:https://sh-tsang.medium.com/review-srgan-srresnet-photo-realistic-super-resolution-gan-super-resolution-96a6fa19490
4. Офіційний сайт towardsdatascience [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/making-deep-neural-networks-paint-to-understand-how-they-work-4be0901582ee
5. Офіційний сайт topazlabs [Електронний ресурс] – Режим доступу до ресурсу:https://www.topazlabs.com/
6. Офіційний сайт zyro [Електронний ресурс] – Режим доступу до ресурсу:https://zyro.com/tools/image-upscaler
7. Офіційний сайт letsenhance [Електронний ресурс] – Режим доступу до ресурсу:https://letsenhance.io
8. Офіційний сайт machinelearningmastery [Електронний ресурс] – Режим доступу до ресурсу:https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/
9. Офіційний сайт keras [Електронний ресурс] – Режим доступу до ресурсу:https://keras.io/
10. Офіційний сайт tensorflow [Електронний ресурс] – Режим доступу до ресурсу:https://www.tensorflow.org/learn
11. Офіційний сайт deepomatic [Електронний ресурс] – Режим доступу до ресурсу:https://deepomatic.com/what-is-image-recognition
12. Офіційний сайт sciencedirect [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron

13. Офіційний сайт towardsdatascience [Електронний ресурс] – Режим доступу до ресурсу:https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

14. Офіційний сайт colah [Електронний ресурс] – Режим доступу до ресурсу:https://colah.github.io/posts/2015-08-Understanding-LSTMs/

15. Офіційний сайт towardsdatascience [Електронний ресурс] – Режим доступу до ресурсу:https://towardsdatascience.com/transformer-neural-network-step-by-step-breakdown-of-the-beast-b3e096dc857f

16. Офіційний сайт machinelearningmastery [Електронний ресурс] – Режим доступу до ресурсу:https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

17. Офіційний сайт paperspace [Електронний ресурс] – Режим доступу до ресурсу:https://blog.paperspace.com/super-resolution-generative-adversarial-networks/

18. Офіційний сайт arxiv [Електронний ресурс] – Режим доступу до ресурсу:https://arxiv.org/abs/1707.02921.

19. Офіційний сайт arxiv [Електронний ресурс] – Режим доступу до ресурсу:https://arxiv.org/abs/1808.08718

20. Офіційний сайт python [Електронний ресурс] – Режим доступу до ресурсу:https://www.python.org/doc/essays/blurb/

21. Офіційний сайт thecode [Електронний ресурс] – Режим доступу до ресурсу:https://thecodest.co/blog/pros-and-cons-of-python

22. Офіційний сайт jinja [Електронний ресурс] – Режим доступу до ресурсу: https://jinja.palletsprojects.com/en/3.1.x/

23. Офіційний сайт flask [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.1.x/

24. Офіційний сайт webagesolutions [Електронний ресурс] – Режим доступу до ресурсу:https://www.webagesolutions.com/blog/learning-the-colab-jupyter-notebook-environment

25. Офіційний сайт eweek [Електронний ресурс] – Режим доступу до ресурсу:https://www.eweek.com/cloud/aws-vs-google-cloud-platform/

/

Додатки

Додаток А:Лістинг коду для тренування НМ в GCP

```
import cv2
import numpy as np
import os
import matplotlib.pyplot as plt

train_dir = "./25k/"

for img in os.listdir( train_dir + "original_images"):
    img_array = cv2.imread(train_dir + "original_images/" + img)
    print(train_dir + "original_images/" + img)



    img_array = cv2.resize(img_array, (128,128))
    lr_img_array = cv2.resize(img_array,(32,32))
    cv2.imwrite(train_dir+ "hr_images/" + img, img_array)
    cv2.imwrite(train_dir+ "lr_images/"+ img, lr_img_array)

import os
import cv2
import numpy as np
from numpy.random import randint
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tqdm import tqdm
import random
import sys

from keras.models import Sequential
from keras import layers, Model
from keras import Model
from keras.layers import Conv2D, PReLU,BatchNormalization, Flatten
from keras.layers import UpSampling2D, LeakyReLU, Dense, Input, add
from keras.applications.vgg19 import VGG19
from keras.models import load_model

def res_block(ip):

    res_model = Conv2D(64, (3,3), padding = "same")(ip)
```

```
    res_model = BatchNormalization(momentum = 0.5)(res_model)
    res_model = PReLU(shared_axes = [1,2])(res_model)

    res_model = Conv2D(64, (3,3), padding = "same")(res_model)
    res_model = BatchNormalization(momentum = 0.5)(res_model)

    return add([ip,res_model])

def upscale_block(ip):

    up_model = Conv2D(256, (3,3), padding="same")(ip)
    up_model = UpSampling2D( size = 2 )(up_model)
    up_model = PReLU(shared_axes=[1,2])(up_model)

    return up_model

def create_gen(gen_ip, num_res_block):
    layers = Conv2D(64, (9,9), padding="same")(gen_ip)
    layers = PReLU(shared_axes=[1,2])(layers)

    temp = layers

    for i in range(num_res_block):
        layers = res_block(layers)

    layers = Conv2D(64, (3,3), padding="same")(layers)
    layers = BatchNormalization(momentum=0.5)(layers)
    layers = add([layers,temp])

    layers = upscale_block(layers)
    layers = upscale_block(layers)

    op = Conv2D(3, (9,9), padding="same")(layers)

    return Model(inputs=gen_ip, outputs=op)

#Descriminator block that will be used to construct the discriminator
def discriminator_block(ip, filters, strides=1, bn=True):

    disc_model = Conv2D(filters, (3,3), strides = strides, padding="same")(ip)

    if bn:
```

```
        disc_model = BatchNormalization( momentum=0.8 )(disc_model)

    disc_model = LeakyReLU( alpha=0.2 )(disc_model)

    return disc_model


#Descriminartor, as described in the original paper
def create_disc(disc_ip):

    df = 64

    d1 = discriminator_block(disc_ip, df, bn=False)
    d2 = discriminator_block(d1, df, strides=2)
    d3 = discriminator_block(d2, df*2)
    d4 = discriminator_block(d3, df*2, strides=2)
    d5 = discriminator_block(d4, df*4)
    d6 = discriminator_block(d5, df*4, strides=2)
    d7 = discriminator_block(d6, df*8)
    d8 = discriminator_block(d7, df*8, strides=2)

    d8_5 = Flatten()(d8)
    d9 = Dense(df*16)(d8_5)
    d10 = LeakyReLU(alpha=0.2)(d9)
    validity = Dense(1, activation='sigmoid')(d10)

    return Model(disc_ip, validity)

def build_vgg(hr_shape):

    vgg = VGG19(weights="imagenet",include_top=False, input_shape=hr_shape)

    return Model(inputs=vgg.inputs, outputs=vgg.layers[10].output)

def create_comb(gen_model, disc_model, vgg, lr_ip, hr_ip):
    gen_img = gen_model(lr_ip)

    gen_features = vgg(gen_img)

    disc_model.trainable = False
    validity = disc_model(gen_img)
```

```
    return Model(inputs=[lr_ip, hr_ip], outputs=[validity, gen_features])


file_path = 'out.txt'
sys.stdout = open(file_path, "w")


n=15000
lr_list = os.listdir("25k/lr_images")[:n]

lr_list = sorted(lr_list)

lr_images = []
for img in lr_list:
    img_lr = cv2.imread("25k/lr_images/" + img)
    img_lr = cv2.cvtColor(img_lr, cv2.COLOR_BGR2RGB)
    lr_images.append(img_lr)


hr_list = os.listdir("25k/hr_images")[:n]
hr_list = sorted(hr_list)

hr_images = []
for img in hr_list:
    img_hr = cv2.imread("25k/hr_images/" + img)
    img_hr = cv2.cvtColor(img_hr, cv2.COLOR_BGR2RGB)
    hr_images.append(img_hr)

lr_images = np.array(lr_images)
hr_images = np.array(hr_images)

print("TEST----------------")
print(hr_list[-1])
print(lr_list[-1])
print(hr_list[0])
print(lr_list[0])

lr_images = lr_images / 255.
hr_images = hr_images / 255.

lr_train, lr_test, hr_train, hr_test = train_test_split(lr_images, hr_images,
                                                        test_size=0.1, random_state=42)
```

```
hr_shape = (hr_train.shape[1], hr_train.shape[2], hr_train.shape[3])
lr_shape = (lr_train.shape[1], lr_train.shape[2], lr_train.shape[3])

lr_ip = Input(shape=lr_shape)
hr_ip = Input(shape=hr_shape)

generator = create_gen(lr_ip, num_res_block = 16)

discriminator = create_disc(hr_ip)
discriminator.compile(loss="binary_crossentropy", optimizer="adam", metrics=['accuracy'])

vgg = build_vgg((128,128,3))
vgg.trainable = False

gan_model = create_comb(generator, discriminator, vgg, lr_ip, hr_ip)
gan_model.compile(loss=["binary_crossentropy", "mse"], loss_weights=[1e-3, 1], optimizer="adam")

batch_size = 1
train_lr_batches = []
train_hr_batches = []
for it in range(int(hr_train.shape[0] / batch_size)):
    start_idx = it * batch_size
    end_idx = start_idx + batch_size
    train_hr_batches.append(hr_train[start_idx:end_idx])
    train_lr_batches.append(lr_train[start_idx:end_idx])

epochs = 10000
for e in range(epochs):

    fake_label = np.zeros((batch_size, 1))
    real_label = np.ones((batch_size,1))



    g_losses = []
    d_losses = []

    for b in tqdm(range(len(train_hr_batches))):
        lr_imgs = train_lr_batches[b]
        hr_imgs = train_hr_batches[b]

        fake_imgs = generator.predict_on_batch(lr_imgs) #Fake images
```

```
        discriminator.trainable = True
        d_loss_gen = discriminator.train_on_batch(fake_imgs, fake_label)
        d_loss_real = discriminator.train_on_batch(hr_imgs, real_label)

        discriminator.trainable = False

        #Average the discriminator loss, just for reporting purposes.
        d_loss = 0.5 * np.add(d_loss_gen, d_loss_real)

        #Extract VGG features, to be used towards calculating loss
        image_features = vgg.predict(hr_imgs)

        g_loss, _, _ = gan_model.train_on_batch([lr_imgs, hr_imgs], [real_label, image_features])
        d_losses.append(d_loss)
        g_losses.append(g_loss)

    g_losses = np.array(g_losses)
    d_losses = np.array(d_losses)

    g_loss = np.sum(g_losses, axis=0) / len(g_losses)
    d_loss = np.sum(d_losses, axis=0) / len(d_losses)

    print("epoch:", e+1 ,"g_loss:", g_loss, "d_loss:", d_loss)
    if (e+1) % 1 == 0: #Change the frequency for model saving, if needed
        #Save the generator after every n epochs (Usually 10 epochs)
        generator.save("weights/gen_e_"+ str(e+1) +".h5")



file.close()
```

Додаток Б: Лістинг коду застосунку

```
from flask import Flask, flash, request, redirect, url_for, render_template
import os
from werkzeug.utils import secure_filename
from utils import load_image, save_image
from model.pretrained import generator
from model.common import resolve_single

app = Flask(__name__)
```

```
UPLOAD_FOLDER = 'static/uploads/'

app.secret_key = "secret key"
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])
upload_files = []

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/')
def home():
    return render_template('index.html')

@app.route('/upgrade', methods=['POST'])
def upgrade_image():

    save_upgraded_image(f'static/uploads/{upload_files[-1]}', upload_files[-1])

    upgraded_image = "upgraded_"+upload_files[-1]

    return render_template('upgrade.html', filename=upload_files[-1], height=500, width=500,
upgraded=upgraded_image)


def save_upgraded_image(lr_image_path, name):
    weights_dir = 'weights\srgan'
    weights_file = lambda filename: os.path.join(weights_dir, filename)

    gan_generator = generator()

    gan_generator.load_weights(weights_file('gan_generator.h5'))
    lr = load_image(lr_image_path)

    gan_sr = resolve_single(gan_generator, lr)
    print(lr_image_path)
    save_image(f'static/uploads/upgraded_{name}', gan_sr)
```

```
@app.route('/', methods=['POST'])
def upload_image():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(request.url)
    file = request.files['file']
    if file.filename == '':
        flash('No image selected for uploading')
        return redirect(request.url)
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        upload_files.append(filename)
        flash('Image successfully uploaded and displayed below')
        return render_template('index.html', filename=filename)
    else:
        flash('!!! Allowed image types are - png, jpg, jpeg')
        return redirect(request.url)


@app.route('/display/<filename>')
def display_image(filename):
    return redirect(url_for('static', filename='uploads/' + filename), code=301)

if __name__ == "__main__":
    app.run()

<html>
<head>
<title>Python Flask Upload and display image</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
</head>
<body>
<p><h1 align="center">SRGAN image upgrader</h1></p>
<div class="container">
    <div class="row">
        <h2>Select a file to upload</h2>
        <p>
            {% with messages = get_flashed_messages() %}
            {% if messages %}
                <ul>
```

```
                {% for message in messages %}
                <li>{{ message }}</li>
                {% endfor %}
                </ul>
            {% endif %}
            {% endwith %}
        </p>
        {% if filename %}
            <div>
                <img src="{{ url_for('display_image', filename=filename) }}">
            </div>
        {% endif %}
        <form method="post" action="/" enctype="multipart/form-data">
            <dl>
                <p>
                    <input type="file" name="file" class="form-control" autocomplete="off" required>
                </p>
            </dl>
            <p>
                <input type="submit" value="Load" class="btn btn-info">
            </p>
        </form>
        <form method="post" action="/upgrade" enctype="multipart/form-data">
            <p>
                <input type="submit" value="Upgrade" class="btn btn-info">
            </p>
        </form>

    </div>
</div>
</body>
</html>
```

```
from tensorflow.python.keras.layers import Add, BatchNormalization, Conv2D, Dense, Flatten, Input, LeakyReLU, PReLU, Lambda
from tensorflow.python.keras.models import Model
from tensorflow.python.keras.applications.vgg19 import VGG19

from model.common import pixel_shuffle, normalize_01, normalize_m11, denormalize_m11
```

```
def upsample(x_in, num_filters):
    x = Conv2D(num_filters, kernel_size=3, padding='same')(x_in)
    x = Lambda(pixel_shuffle(scale=2))(x)
    return PReLU(shared_axes=[1, 2])(x)


def res_block(x_in, num_filters, momentum=0.8):
    x = Conv2D(num_filters, kernel_size=3, padding='same')(x_in)
    x = BatchNormalization(momentum=momentum)(x)
    x = PReLU(shared_axes=[1, 2])(x)
    x = Conv2D(num_filters, kernel_size=3, padding='same')(x)
    x = BatchNormalization(momentum=momentum)(x)
    x = Add()([x_in, x])
    return x


def generator(num_filters=64, num_res_blocks=16):
    x_in = Input(shape=(None, None, 3))
    x = Lambda(normalize_01)(x_in)

    x = Conv2D(num_filters, kernel_size=9, padding='same')(x)
    x = x_1 = PReLU(shared_axes=[1, 2])(x)

    for _ in range(num_res_blocks):
        x = res_block(x, num_filters)

    x = Conv2D(num_filters, kernel_size=3, padding='same')(x)
    x = BatchNormalization()(x)
    x = Add()([x_1, x])

    x = upsample(x, num_filters * 4)
    x = upsample(x, num_filters * 4)

    x = Conv2D(3, kernel_size=9, padding='same', activation='tanh')(x)
    x = Lambda(denormalize_m11)(x)

    return Model(x_in, x)


#gan_model = sr_resnet
```

```
def discriminator_block(x_in, num_filters, strides=1, batchnorm=True, momentum=0.8):
    x = Conv2D(num_filters, kernel_size=3, strides=strides, padding='same')(x_in)
    if batchnorm:
        x = BatchNormalization(momentum=momentum)(x)
    return LeakyReLU(alpha=0.2)(x)


def discriminator(num_filters=64):
    x_in = Input(shape=(HR_SIZE, HR_SIZE, 3))
    x = Lambda(normalize_m11)(x_in)

    x = discriminator_block(x, num_filters, batchnorm=False)
    x = discriminator_block(x, num_filters, strides=2)

    x = discriminator_block(x, num_filters * 2)
    x = discriminator_block(x, num_filters * 2, strides=2)

    x = discriminator_block(x, num_filters * 4)
    x = discriminator_block(x, num_filters * 4, strides=2)

    x = discriminator_block(x, num_filters * 8)
    x = discriminator_block(x, num_filters * 8, strides=2)

    x = Flatten()(x)

    x = Dense(1024)(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = Dense(1, activation='sigmoid')(x)

    return Model(x_in, x)


def vgg_22():
    return _vgg(5)


def vgg_54():
    return _vgg(20)


def _vgg(output_layer):
    vgg = VGG19(input_shape=(None, None, 3), include_top=False)
```

```
    return Model(vgg.input, vgg.layers[output_layer].output)


<html>
<head>
<title>Upgrade</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
</head>
<body>
<p><h1 align="center">SRGAN image upgrader</h1></p>
<div class="container">
  <div class="row" align="center">
    {% if filename %}
      <div>
        <img src="{{ url_for('display_image', filename=filename) }}" width="500" height="500">
        <img src="{{ url_for('display_image', filename=upgraded) }}" width= {{ width }} height={{ height }} >
      </div>
    {% endif %}
  </div>
</div>
<p align="center">
  <a href="/" ><button class=grey style="height:90px;width:150px">Back</button></a>
  <a href="./static/uploads/{{ upgraded}}" download class="download-btn"><button class=grey style="height:90px;width:150px">Download image</button></a>
</p>
</body>
</html>
```