МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна магістерська робота

на тему «Розроблення програмного забезпечення
для масштабування растрових зображень
із застосуванням нейронних мереж»

Виконала: студентка групи МгІТ-21
спеціальності
122 Комп'ютерні науки

Шега Ді
Керівник доц. Оксана КОЛИСКО

Рецензент

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Спеціальність **122 Комп'ютерні науки**
Освітня програма **Комп'ютерні науки**

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф.Щербань В.Ю.
"____" _____2023 року

З А В Д А Н Н Я

НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТЦІ

Шега Ді

1. Тема роботи Розроблення програмного забезпечення для масштабування растрових зображень із застосуванням нейронних мереж, науковий керівник роботи Колиско Оксана Зенонівна, затверджені наказом вищого навчального закладу від "_12_"вересня_2023 року №_210-уч_

2. Строк подання студентом роботи _1 листопада 2023р__.

3. Вихідні дані до роботи: *Розробки кафедри комп'ютерних наук; рекомендована література, додатки.*

4. Зміст дипломної роботи : *ВСТУП; РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ; РОЗДІЛ 2 ПРОЕКТУВАННЯ; РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ; ВИСНОВКИ; СПИСОК ЛІТЕРАТУРИ; ДОДАТОК А ОКРЕМІ ФРАГМЕНТИ ПРОГРАМНОГО КОДУ; ДОДАТОК Б ПРЕЗЕНТАЦІЯ.*

5. Перелік графічного матеріалу*: презентація на* _____ *слайдах*

6. Консультанти розділів дипломної магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ1	Колиско О.З., доцент, к.т.н.		
Розділ 2	Колиско О.З., доцент, к.т.н.		
Розділ 3	Колиско О.З., доцент, к.т.н.		

7. Дата видачі завдання серпень 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання	
			Студ.	*Керівн.*
1	Вступ	*15.09.2023*		
2	Розділ 1. Постановка задачі	*20.09.2023*		
3	Розділ 2 Проектування	*30.09.2023*		
4	Розділ 3. Програмна реалізація	*10.10.2023*		
5	Висновки	*25.10.2023*		
6	Оформлення дипломної магістерської роботи (чистовий варіант)	*1.11.2023*		
7	Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту)	*4.11.2023*		
8	Перевірка дипломної магістерської роботи на наявність текстових співпадінь та помилок (за 10 днів до захисту)	*6.11.2023*		
9	Подання дипломної магістерської роботи у відділ магістратури для перевірки виконання додатку до індивідуального навчального плану (за 10 днів до захисту)	*8.11.2023*		
10	Подання дипломної магістерської роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	*10.11.2023*		

Студентка _____ Шега ДІ

Науковий керівник роботи _____ Оксана КОЛИСКО

Директор НМЦУПФ _____ Олена ГРИГОРЕВСЬКА

АНОТАЦІЯ

Метою кваліфікаційної роботи є аналіз існуючих методів обробки зображень для їх покращення за допомогою згорткової нейронної мережі (SRCNN), а також розробка програмного додатку, створеного для демонстрації роботи обраної нейромережі.

Обробка графічних файлів - сфера, яка інтенсивно розвивається в сучасному світі комп'ютерних технологій. З'являються нові засоби та алгоритми обробки зображень. На даний момент багато редакторів, створених для персональних комп'ютерів, можуть виконувати дуже складні графічні функції. Однак не всі з них використовують нейронні мережі, які надають можливість високого рівня якості обробки зображень. Тому варто спробувати застосувати алгоритми згортальних нейронних мереж, які зможуть ефективно виконувати обробку зображень на новому рівні

Випускна кваліфікаційна роботи займає___ сторінок, включає ___ малюнків, ___ таблиць, ____ додатка і складається з 3 розділів.

В першому розділі зроблено аналіз предметної області машинного навчання і описано принцип роботи згортальної нейронної мережі. У другому розділі описано архітектуру згортальної нейронної мережі SRCNN, та її деяку модифікацію. У третьому розділі описана реалізація додатка для демонстрації роботи нейронної мережі. Метою цієї роботи є створення нейромережевого програмного модуля для збільшення растрових зображень з мінімальною втратою в якості, та демонстрація можливостей цього модуля за допомогою веб-інтерфейсу.

*Ключові слова: нейронні мережі, SRCNN мережі, WEB-застосунок***.**

ABSTRACT

The purpose of the qualification work is to analyze the existing methods of image processing for their improvement using a convolutional neural network (SRCNN), as well as to develop a software application created to demonstrate the work of the selected neural network.

Processing of graphic files is an area that is intensively developing in the modern world of computer technologies. New image processing tools and algorithms are emerging. At the moment, many editors created for personal computers can perform very complex graphic functions. However, not all of them use neural networks that enable high-quality image processing. Therefore, it is worth trying to apply algorithms of convolutional neural networks, which can effectively perform image processing at a new level

The graduation thesis takes ___ pages, includes ___ figures, ___ tables, ____ appendix and consists of 3 sections.

In the first chapter, an analysis of the subject area of machine learning is made and the principle of operation of a convolutional neural network is described. In the second section, the architecture of the SRCNN convolutional neural network is described, and some of its modifications are proposed. The third section describes the implementation of the application for demonstrating the operation of the neural network. The purpose of this work is to create a neural network software module for enlarging bitmap images with minimal quality loss, as well as to demonstrate the use of this module using a web interface.

Keywords: neural networks, SRCNN networks, WEB-application.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ГНМ (deep neural network, DNN) – глибока нейронна мережа

ЗНМ – згорткова нейронна мережа

ШНМ (штучна нейронна мережа) – обчислювальна нелінейна модель, на основі котрій є нейронна структура мозга. Вона здатна навчатися виконанню задач класифікації, передбачення, прийняття рішень та візуалізації.

LSTM (Long Short-Term Memory) – мережа довгої короткострокової пам'яті

ВСТУП

Цифрова обробка та розпізнавання зображень – один з напрямів наукових досліджень, що інтенсивно розвивається. Багато галузей техніки, що мають відношення до отримання, обробки, зберігання та передачі інформації, в значній мірі орієнтуються на розвиток систем, в котрих інформація має характер зображень.

На даний момент машинне навчання переживає збільшений інтерес до нейронних мереж з боку розробників, дослідників і власників бізнесу. Це обумовлено тим, що як з'ясувалося, в одному з класичних завдань комп'ютерного зору - завданні класифікації зображень, - згортальна нейронна мережа впоралася набагато краще, ніж усі інші методи, які були розроблені за останні декілька років. Ці факти були описані командою переможців змагання по розпізнаванню зображень ImageNet LSVRC – ще в 2012 в роботі "ImageNet Classification with Deep Convolutional Neural Networks" [2].

Зважаючи на їх успішне використання, інтерес до згортальних нейромереж дуже різко зріс, і за декілька років в задачах класифікації зображень згортальні нейронні мережі добилися точності, порівнянної з точністю, що досягається людиною. Зараз, це один з найпопулярніших, відносно точних і універсальних методів для задач, пов'язаних з розпізнаванням образів на зображенні. Так в роботі "Recent Advances in Convolutional Neural Networks" [3] описані способи застосування згортальних нейронних мереж як інструмент відстежування динамічних об'єктів, визначення стану об'єктів, а також розпізнавання тексту, що доводить застосування надточних нейронних мереж в широкому спектрі завдань, пов'язаних з аналізом і перетворенням зображень.

Сьогодні нейронні мережі набирають все більшу популярність, та все частіше використовуються у різних сферах життя. Особливо часто нейромережі використовуються, коли це стосується задач аналізу даних, кластеризації, розпізнавання об’єктів на зображеннях, виявлення схованих закономірностей, комп’ютерного зору, прогнозування даних. Налічується багато різних архітектур

нейронних мереж, кожна з яких виконує свої функції, має певні алгоритми тренування, та має переваги й недоліки порівняно із іншими архітектурами.

У більшості випадків для зберігання зображень використовується саме растровий формат за рахунок його відносної простоти і великих можливостей. Цифрові растрові зображення являють собою сітку пікселів, що дозволяє зберігати в такому форматі дуже складні візуальні об'єкти

Але крім переваг у растрового формату є істотні обмеження. Так, наприклад, одна з його основних характеристик, це розмір зображення в пікселях по довжині і ширині. Це означає, що при збереженні зображення було зафіксовано обмежену кількість точок, причому при збільшенні цієї кількості прямо пропорціонально збільшується і розмір займаного файлом місця на диску, що іноді є досить істотним нюансом.

Саме ця особливість формату вступає в протиріччя з потребою в масштабуванні зображення без втрати якості, якщо під масштабуванням мається на увазі збільшення розмірів оригінального зображення. Тож питання збільшення якості отриманого результату є актуальним для таких організацій, як, наприклад, різного роду студії, що займаються розробкою цифрового контенту; різні соціальні медіа, на зразок новинних сайтів; і так далі. Слід зауважити, що існуючі алгоритми масштабування не дозволяють досить точно передбачити кольори пікселів, доданих під час масштабування, а іноді деякі графічні деталі зникають або стають менш помітними (контрастними) при збільшенні, через що страждає загальне сприйняття зображення, яке, можливо, було кращим в оригінальному (до збільшення) розмірі.

Таке протиріччя, зважаючи на хвилю популярності штучного інтелекту, перетворюється в задачу застосування сучасних практик машинного навчання для масштабування фотографій і растрових зображень в цілому. Ця проблема на даний час ще не розв'язана повністю, або її рішення не зовсім застосовні в роботі існуючого бізнесу. У статі "A Neural - Network - Based Image Resolution Enhancement Scheme for Image Resizing" [1] наводиться спосіб застосування структури нейронної мережі для вирішення завдання масштабування зображення.

Проте ця робота була проведена ще в 2003 році, а надалі з'явилися і були оптимізовані нові методи машинного навчання, що обумовлює актуальність проблеми. Новий подих розвитку та використанню нейронних мереж дав розвиток згорткових нейронних мереж, особливо що стосується сфери комп’ютерного зору, кластеризації, та аналізу зображень

Об'єктом дослідження у вирішенні цієї проблеми є процес масштабування (збільшення) зображення з мінімальною втратою в якості. Предметом дослідження буде спосіб поєднання класичного методу масштабування і методу машинного навчання.

На підставі сформульованої проблеми і виявлених об'єкту і предмета дослідження метою цієї роботи буде розробка навчаного інструменту (програмного модуля) для масштабування растрових зображень, а також демонстрація його можливостей на прикладі веб-додатка.

РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ

Згорткова нейронна мережа за рахунок застосування спеціальної операції – власне згортки – дозволяє водночас зменшити кількість інформації, що зберігається в пам'яті, за рахунок чого краще справляється з картинками більш високої роздільної здатності, і виділяє опорні ознаки зображення, такі як ребра, контури або грані. Головною особливістю даної архітектури є те, що вона використовує операцію згортки для того, щоб зменшити розмір зображення, виділяючи й розпізнаючи характерні ознаки, в залежності від поставленої задачі. Нейронна мережа поступово та ітеративно проходить по зображенню, виконуючи операцію згортки. Поступово вона отримує все більше даних, які необхідні їй для розпізнавання. На якість розпізнання впливає кількість та різні комбінації шарів згортки всередині її топології. На наступних рівнях обробки з цих ребер і граней можна розпізнати повторювані фрагменти текстур, які далі можуть скластися в фрагменти зображення.

Для досягнення найкращих результатів роботи згорткових нейронних мереж є маса засобів. Зокрема це: гістограми датасетів, представлення зображення у полутоновій формі, розпаралелювання процесів нейронної мережі, додання афінних перетворень до зображень з датасетів, різні ядра згортки та інше. В атестаційній роботі будуть розглянуті проаналізовані найпоширеніші алгоритми обробки зображень, та будуть надані практичні рекомендації по використанню цих алгоритмів, в залежності від існуючих умов й поставленої задачі.

1.1.Методи обробки зображень.

Попередня обробка зображення – процес поліпшення його якості, що ставить за мету отримання на основі оригіналу максимально точного і адаптованого для подальшого автоматичного аналізу зображення.

Серед дефектів цифрового зображення можна виділити наступні види:

- цифровий шум;
- кольорові дефекти (недостатні або надлишкові яскравість і контраст, неправильний колірний тон);
- розмитість (расфокусировка).

Методи попередньої обробки зображень залежать від завдань досліджень і можуть включати наступні види робіт:

- фільтрація зашумленних зображень;
- корекція яскравості і контрасту.

Цифровий шум зображення – дефект зображення, внесений фотосенсором чи електронікою пристроїв, які їх використовують. Для його нейтралізації використовують такі методи:- лінійне усереднення точок по сусідах; - розмиття по Гаусу; - медіанна фільтрація; - морфологічні перетворення.

Лінійне усереднення точок по сусідах - найпростіший вид алгоритмів видалення шуму. Основна ідея їх в тому щоб брати середнє арифметичне значень точок в деякому околі в якості нового значення нашої точки. Фізично така фільтрація реалізується за допомогою обходу пікселів зображення матрицею згортки, що має такий вигляд (рисунок 1.1)

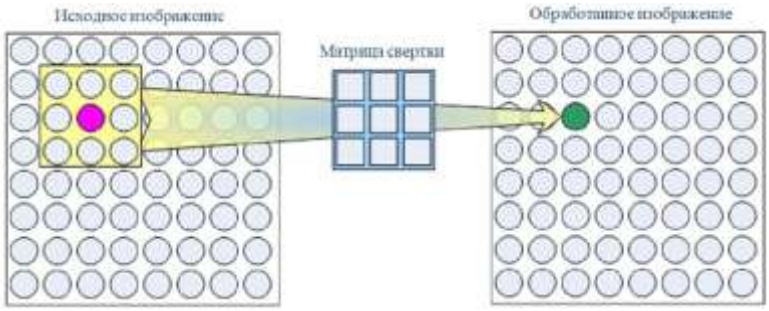

Рисунок 1.1 – Приклад роботи методу фільтрації

Розмиття за Гаусом (різновид лінійного згортання) реалізується за допомогою обходу пікселів зображення матрицею згортки, що має такий вигляд:

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Рисунок 1.2 – Матриця згортки

Матриця 5 × 5 заповнюється за нормальним (Гауссовим законом). Нижче приведена та ж матриця, де коефіцієнти вже є нормованими, так що div для цієї матриці дорівнює одному.

0,000789	0,006581	0,013347	0,006581	0,000789
0,006581	0,054901	0,111345	0,054901	0,006581
0,013347	0,111345	0,225821	0,111345	0,013347
0,006581	0,054901	0,111345	0,054901	0,006581
0,000789	0,006581	0,013347	0,006581	0,000789

Від розміру матриці залежить величина розмиття.

На рисунку 1.3. у верхнього лівого пікселя не існує ≪сусідів≫ зліва і зверху, отже, тут не треба множити коефіцієнти матриці. Для вирішення цієї проблеми потрібне створення проміжного зображення. Ідея в тому, щоб створювати тимчасове зображення з розмірами:

width + 2 • gap / 2, height + 2 • gap / 2,

де width і height - ширина і висота фільтрованого зображення;

gap - розмірність матриці згортки.

Рисунок 1.3 – Візуалізація матриці на зображенні

В центр зображення копіюється вхідна картинка, а краї заповнюються крайніми пікселями зображення (рисунок 1.4).

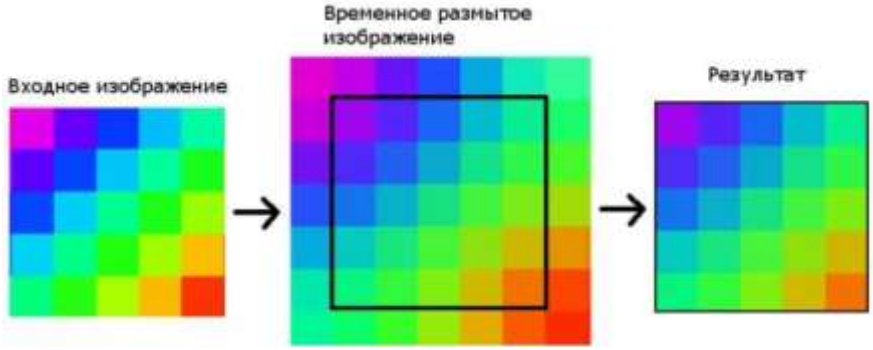

Рисунок 1.4 – Схематичне зображення роботи

1.2. Машинне навчання і згорткові мережі

Згоркові нейронні мережі, які тут пропонується застосувати в якості інструменту для масштабування зображень, є одним з видів нейронних мереж, які у свою чергу є одним з методів машинного навчання.

Саме по собі поняття машинне навчання можна трактувати різними образами. Формальний опис процесу навчання дав Том Мітчел: "Кажуть, що комп'ютерна програма навчається на основі досвіду E по відношенню до деякого класу завдань T і мірі якості P, якщо якість рішення завдань з T, виміряна на основі P, покращується з набуттям досвіду E". [5]. Це визначення більше відноситься до індуктивного (статистичного) навчання, тоді коли існує і дедуктивне, яке активно застосовується в експертних системах. Ми під навчанням і конкретно машинним навчанням розглядаємо саме індуктивне навчання. Це пов'язано з тим, що метою роботи є розробка методу масштабування, який зможе підлаштуватися під заданий тип зображень в процесі навчання, тобто сам виробить потрібні правила.

1.2.1. Навчання по прецедентах

У рамках термінів машинного навчання, можна представити рішення будь-якої поставленої задачі, як значення y_i цільової функції y* при заданому об'єкті x_i. Оскільки навчання проводиться відразу на масиві об'єктів X і результатів Y, то весь процес рішення задачі можна представити, як $y^*: X \to Y$. За визначенням пара, де y при заданому аргументі x є прецедентом, що в загальній їх (цих пар) сукупності представляє навчальну вибірку [6]. Так само в літературі можна зустріти поняття "Навчання на прикладах", яке по суті є тим же самим, що і навчання по прецедентах. Найбільш поширеним способом опису прецедентів є опис за допомогою ознак. Фіксується сукупність з N показників, виміряних в усіх прецедентів. Якщо усі N показників числові, то признакові описи є числовими векторами розмірності N .

Але можуть зустрічатися і складніші випадки, наприклад, коли прецеденти описуються зображеннями, тобто послідовністю пікселів однакової розмірності. У рамках цієї роботи додаток треба навчати на сукупності пар оригінальних растрових зображень у високій роздільній здатності та їх зменшених в 2 рази

копіях. Для вирішення задачі навчання по прецедентах в першу чергу фіксується модель відновлюваної залежності. Потім вводиться функціонал якості, значення якого покаже, наскільки добре модель описує спостережувані дані. Алгоритм навчання шукає такий набір параметрів моделі, при якому функціонал якості на заданій навчальній вибірці набуває оптимального значення.

Розрізнюють когнітивний і біонічний підходи, де перший представлений як математична модель для ефективного вирішення вузького спектру завдань (з можливістю математичного обгрунтування результату), а другий - спроба відтворити фізичну модель процесів мислення. Штучні нейронні мережі є представником саме біонічного підходу, оскільки моделюють роботу біологічного аналога - мозку живої істоти.

Штучна нейронна мережа, за визначенням, є розподіленим паралельним процесором, що складається з елементарних одиниць обробки інформації, накопичує експериментальні знання [8]. Штучні нейронні мережі можуть змінювати свою поведінку залежно від зовнішнього середовища. Саме цей чинник забезпечує той інтерес, який вони викликають. Після пред'явлення вхідних сигналів (можливо, разом з необхідними виходами) вони самостійно настроюються, щоб забезпечувати необхідну реакцію.

Відгук мережі після навчання може бути до деякої міри нечутливий до невеликих змін вхідних сигналів. Така здатність побачити образ крізь шум і спотворення життєво необхідна для розпізнавання образів у реальному світі. Вона дозволяє подолати вимогу до строгої точності, яка пред'являється комп'ютером, і відкриває шлях до системи, яка може мати справу з недосконалим світом, в якому ми живемо. Важливо відмітити, що штучна нейронна мережа робить узагальнення автоматично завдяки своїй структурі, а не за допомогою використання "людського інтелекту" у формі спеціально написаних комп'ютерних програм.

Деякі з штучних нейронних мереж мають здатність визначати суть з вхідних сигналів. Наприклад, мережа може бути навчена на послідовності спотворених версій якого-небудь об'єкту. Після відповідного навчання пред'явлення такого спотвореного прикладу приведе до того, що мережа породить об'єкт з

правильними параметрами. Цю властивість можна інтерпретувати як здатність виокремлювати ідеальні усереднені прототипи із спотворених даних.

1.2.2. Структура нейронних мереж і їх особливості

Є багато різних видів штучних нейронних мереж, що обумовлено різноманітністю завдань. Будь-яка нейронна мережа складається з декількох основних компонентів: з штучних нейронів - елементів обробки, які мають структуру трьох пов'язаних один з одним шарів : вхідним, що складається з одного або кількох шарів, прихованим і вихідним (рисунок 1.1).

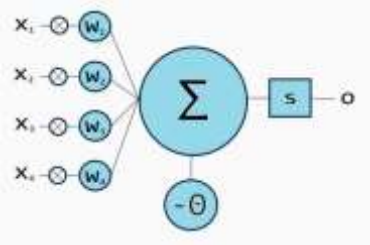

Рисунок 1.1 – Штучна нейронна мережа

Вхідний шар - на вхід подають значення параметрів об’єкта складається з вхідних нейронів, які передають інформацію в приховані шари. Прихований шар, який може бути не один у своєму роді, а також може бути не лише одного виду (наприклад, рекурентний, повнопов'язаний, згортальний і так далі). Вихідний шар, де кожному виходу надається певне значення, яке було визначеним під час аналізу параметрів об'єкта нейронною мережею. Зазвичай останній шар використовують для розмітки класів, тим самим вирішуючи стандартну задачу класифікації.

Кожен нейрон має входи з вагами - синапсами, функцію активації, визначальну вихідну інформацію при заданій вхідній, і один вихід. Синапси - регульовані параметри, що конвертують нейронну мережу в параметризрвану систему.

Навчання (або тренування) – процес оптимізації ваг, в якому мінімізується помилка передбачення, і мережа досягає необхідного рівня точності. Найбільш використовуваний метод для визначення внеску в помилку від кожного нейрона – зворотнє поширення помилки, за допомогою якого обчислюють градієнт. Це одна з модифікацій методу градієнтного спуску.

За допомогою додаткових прихованих шарів можна зробити систему більш гнучкою і потужною. ШНМ з багатьма прихованими шарами називаються глибокими нейронними мережами; вони створюють складні нелінійні зв'язки.

Згорткова нейронна мережа

Згорткова нейронна мережа (Convolutional neural network, CNN) містить один або більше об'єднаних або поєднаних згортальних шарів. CNN використовує варіацію багатошарового перцептрона. Згорткові шари використовують операцію згортки для вхідних даних і передають результат в наступний шар (рисунок 1.2). Ця операція дозволяє мережі бути глибше з меншою кількістю параметрів.

Згорткові мережі показують гарні результати застосовно картинок і мови. У статті Convolutional Neural Networks for Sentence Classification автор описує процес і результати завдань класифікації тексту за допомогою згорткових нейронних мереж. У роботі представлена модель на основі word2vec, яка проводить експерименти, тестується на декількох бенчмарках і демонструє блискучі результати.

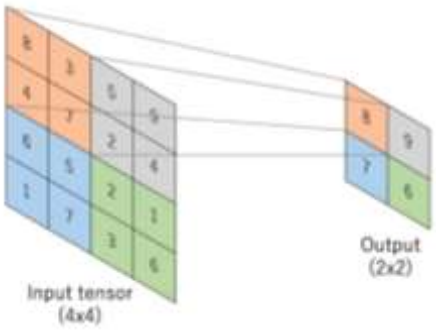

Рисунок 1.2 – Згорткова нейронна мережа

Рекурсивна нейронна мережа

Рекурсивна нейронна мережа – глибока нейронна мережа, сформована при застосуванні одних і тих же наборів ваг рекурсивно над структурою, щоб зробити скалярне або структуроване пророкування над вхідною структурою змінного розміру через активацію структури в топологічному порядку (рисунок 1.3). У простій архітектурі нелінійність, така як тангенціальна функція активації і матриця ваг використовуються для об'єднання вузлів в батьківські об'єкти.

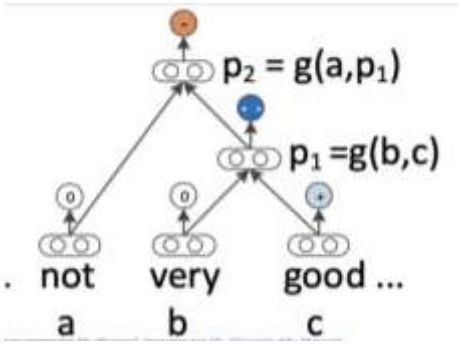

Рисунок 1.3 – Рекурсивна нейронна мережа

Рекурентна нейронна мережа

Рекурентна нейронна мережа, на відміну від прямої нейронної мережі, є варіантом рекурсивної ШНМ, в якій зв'язки між нейронами – спрямовані цикли. Тут вихідна інформація залежить не тільки від поточного входу, але також від станів нейрона на попередньому кроці. Така пам'ять дозволяє користувачам вирішувати завдання нейролінгвістичного програмування: розпізнання рукописного тексту або мови.

Мережа довгої короткостроковій пам'яті

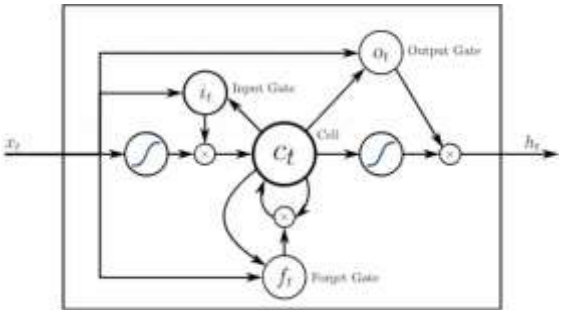

Рисунок 1.4 – Мережа довгої короткостроковій пам'яті з вхідним, вихідним і гейтом забування

Мережа довгої короткострокової пам'яті (LSTM) – різновид рекуррентної нейромережі, створена для більш точного моделювання часових послідовностей і їх довгострокових залежностей, ніж традиційна рекуррентна мережа. LSTM-мережа не використовує функцію активації в рекурентних компонентах, збережені значення не модифікуються, а градієнт не прагне зникнути під час тренування (рисунок 1.4).

Sequence-to-sequence модель

Sequence-to-sequence моделі складаються з двох рекурентних мереж: кодувальника, який обробляє вхідні дані, і декодера, який здійснює висновок.

Sequence-to-Sequence моделі часто використовуються в питально-відповідних системах, чат-ботах і машинному перекладі. У Paraphrase Detection Using Recursive Autoencoder представлена нова рекурсивна архітектура автошифрувальника, в якій представлення – вектора в n-вимірному семантичному просторі, де фрази зі схожими значенням близькі одина до одної.

Неглибокі (shallow) нейронні мережі

Неглибокі моделі, як і глибокі нейронні мережі, теж популярні і корисні інструменти. Наприклад, word2vec - група неглибоких двошарових моделей, яка використовується для створення векторних представлень слів (word embeddings).

Часто для завдань класифікації і розпізнавання графічних образів раніше використовувалися класичні нейромережеві структури, такі як багатошарові персептрони і радіально-базисні мережі. Але є зауваження до цих структур [10]:

– Через велику кількість вхідних параметрів (по одному на кожен піксель зображення) різко зростає розмір нейронної мережі, роблячи процес її навчання вкрай довгим і малоефективним на невеликих вибірках.

– Задля підвищення точності результату рекомендується застосовувати декілька видів нейронних мереж, що в цілому збільшує складність системи, а також підвищує потужностні і часові витрати.

– Модель дуже різко реагує на будь-які геометричні зміни початкового об'єкту, даючи абсолютно невірний результат, наприклад, при зміні перспективи аналізованого об'єкту.

– При перетворенні зображення у векторний вид, втрачається топологія зображення, тобто взаємозв'язок між окремими її частинами.

Найбільш відповідними для роботи із зображеннями визначено два види. Найпростіший - який одночасно є універсальним для великого спектру завдань, це повнопов'язана двошарова штучна нейронна мережа. Також є згортальна нейронна мережа (Convolutional Neural Network, CNN), яка відмінно справляється із завданнями аналізу графічних об'єктів [2, 11]. Тому для роботи з перетвореннями зображень краще вибрати згортальну нейронну мережу.

1.2.3.Топології згортальних нейронних мереж

Є кілька основних різновидів архітектури згортальних нейронних мереж. Перша успішна реалізація згортальної нейронної мережі - мережа LeNet5 розроблена в 1990-х роках [14]. У ній кожен згортальний шар чергується з шаром субдискретизації, що зменшує розмірність при пошуку абстракцій. Ця мережа завершується декількома повнопов'язаними шарами, які, зв'язують абстракції з класами. У цій топології 60 тис. параметрів, вона є порівняно невеликою і тому доступна до будь-яких модифікацій.

Десь в 2012 була розроблена топологія AlexNet [2], Вона створювалася для конкурсу ImageNet і популяризувала згортальні нейронні мережі (і нейронні мережі в цілому) в задачах класифікації. Вона містить близько 60 млн. параметрів, а це збільшує час на навчання мережі. Ця мережа по своїй структурі багато в чому аналогічна LeNet, але AlexNet в рази перевищує попередника в об'ємах. Це можна пояснити тим, що значно збільшилися потужності обчислювальних процесорів, а також збільшилася кількість відкритих даних, за допомогою яких можна навчати нейронну мережу.

VGG - розроблена оксфордськими ученими. Ця топологія є однією з перших згортальних нейронних мереж, що використала невеликі фільтри розмірами 3×3 в кожному згортальному шарі, які були об'єднані в послідовність. Головною перевагою VGG є наявність ефекту від послідовного розташування згортальних шарів. Так, декілька фільтрів 3×3 могли давати аналогічних результат, як від застосування набагато більших рецептивних полів, наприклад, розмірами 5×5 і 7×7. Ці відкриття надалі були застосовані в сучасних топологіях згортальних нейронних мереж. Але такий підхід призводить до збільшення витрат на навчання нейронної мережі, це привело до розвитку нових підходів для оптимізації цієї топології.

Таким підходом стала топологія Network in Network, NiN [15], яка дозволила використати згортальні шари з фільтрами розміром 1×1 для збільшення комбінацій можливих значень на картах властивостей. Тут після кожного згортального шару використовується подібність багатошарового персептрона,

щоб краще зв'язати властивості перед тим, як передати їх на вхід наступному згортальному шару. Ця ідея надалі стала використовуватися в сучасніших глибоких нейромережах. Її надалі розвинула компанія Google, створивши дві схожі топології GoogleLeNet і Inception [16]. Це дозволило оптимізувати процес навчання, зменшивши накладні витрати у вигляді процесорних потужностей і витрати пам'яті, при цьому збільшивши якість результату.

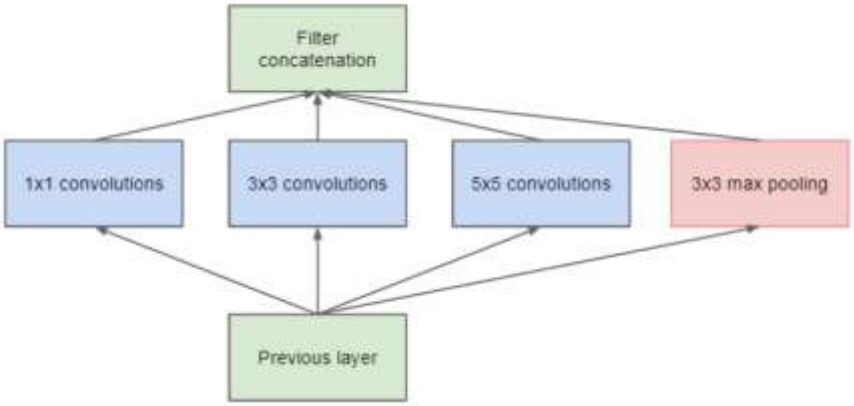

Рисунок 1.5. Модуль Inception.

Розроблений командою Google модуль Inception (малюнок 1.5.) проектувався для паралельної обробки даних шарами, з різною розмірністю рецептивного поля, при цьому використовуючи для зменшення кількості карт спосіб надалі названий "Шийка пляшки" в контексті топологий нейронних мереж". Шийка пляшки полягає в початковому зменшенні кількості властивостей для обробки їх за допомогою звичайної операції згортання. Потім, цей зменшений набір карт властивостей перетворювався за допомогою того ж фільтру одиничної розмірності в новий набір карт властивостей, але вже з первинною розмірністю. Такий спосіб дозволяє, не втрачаючи в якості, скоротити об'єм операцій на етапі виявлення властивостей в звичайному згортальному шарі.

Після аналізу топологій згортальних нейронних мереж, можна припустити, що для задачі перетворення зображення низької якості в зображення підвищеної чіткості, краще використати прогресивніший підхід, а саме "Шийку пляшки" для емуляції роботи багатошарового персептрона і для оптимізації обчислювальних потужностей. Так само розмір фільтру повинен зменшаться у міру просування углиб нейронної мережі, що доведено досвідом розробки топології VGG.

1.3. Алгоритм роботи згортальної нейронної мережі

Цей тип нейронної мережі містить 3 основні види прихованих шарів : згортального, шару субдискретизації (отримання підвибірок) і повнозв'язного. На рисунку 1.6 представлений приклад архітектури згортальної нейронної мережі для класифікації зображень :

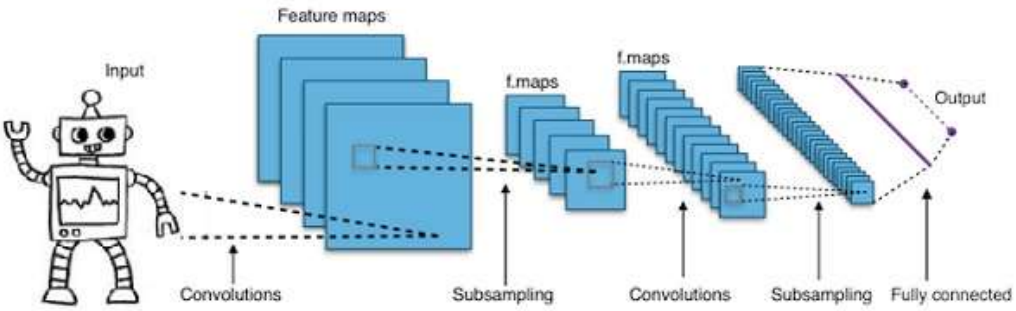

Рисунок1.6. Приклад архітектури CNN

Згортальний шар (Convolution) зазвичай йде відразу після вхідного шару. Кожний наступний нейрон згортального прихованого шару отримує на вхід дані про деякий набір виходів групи нейронів на попередньому шарі. Ця група так само називається рецептивним полем шару [10], яке є вікном (матрицею), як наприклад, на рисунку 1.7:

1	0	1
0	1	0
1	0	1

Рисунок 1.7. Приклад рецептивного вікна

Це вікно прораховує значення виходу для цієї області і пересилає його на вхід наступному нейрону. Це вікно так само називається фільтром, оскільки при множенні області на цю матрицю, активуються ті області (пікселі) які співпали у фільтрі і в матриці. Чим більше збігів області і фільтру, тим більше вірогідність того, що в цій області міститься саме те, що шукає фільтр. Результат поелементного множення двох матриць (області і фільтру) складається і зберігається в якості значення поточного елементу карти властивостей.

Цей фільтр переміщається по зображенню на один піксель, формуючи новий елемент карти властивостей і так триває до того моменту, поки фільтр не

обробить останню область, що залишилася, на початковому зображенні. Приклад роботи цього фільтру зображений на рисунку 1.8:

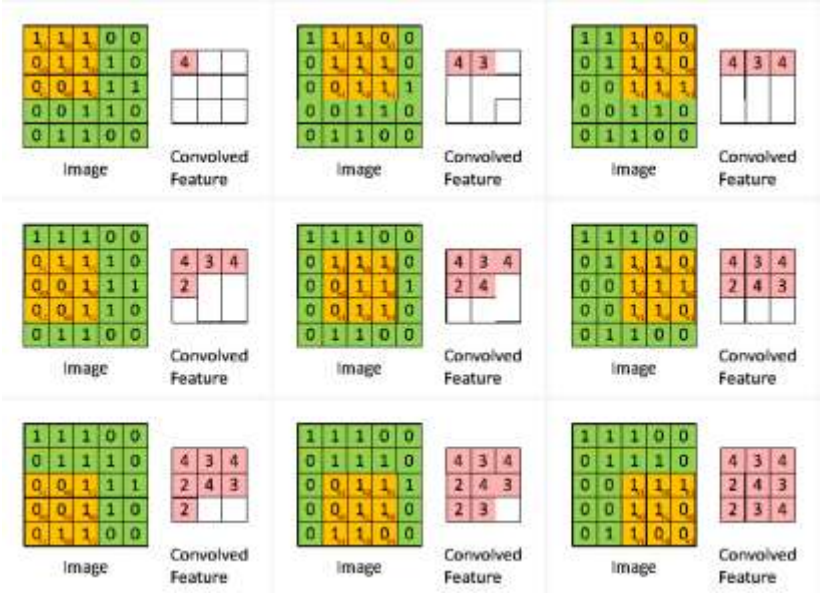

Рисунок 1.8. Приклад роботи конволюціонного фільтра.

Згортальний шар визначається 3 гіперпараметрами, які встановлюються на етапі моделювання архітектури нейронної мережі :

- Глибина, яка дорівнює числу фільтрів згортального шару, оскільки для кожного фільтру формується своя карта властивостей для зображення або матриці.
- Крок, який визначає на яку кількість пікселів переміщатиметься фільтр по початковому зображенню (у прикладі вище він дорівнював 1).
- Параметр неявного заповнення нулями, який визначає наскільки фільтр на крайніх позиціях може "вилізти" за межі зображення, дозволяючи тим обробити більше пікселів зображення, при цьому ігноруючи область, за яку довелося вийти, приймаючи ці пікселі за нулі.

Ця ідея свого часу була взята [12] і адаптована на основі відкриттів в області нейробіології, де було доведено, що різні ділянки зорової кори головного мозку реагують на різні види об'єктів [13].

Таким чином згортальні шари породжують набір карт властивостей, де кількість карт дорівнює кількості фільтрів, на яких був застосований фільтр, і де кожна карта містить узагальнення про місцезнаходження області з певним елементом, на пошук якого був налаштований фільтр.

Шари (Pooling) субдискретизації, використовуються для зменшення розмірності карт властивостей, усереднюючи значення кожної області, що наводиться на карті ознак, тим самим ще більше узагальнюючи зображення, посилюючи ті області, в яких ознака виявилася домінуючою. Ці шари зазвичай бувають 3 типів, залежно від операцій, які застосовують для зменшення матриці : максимізують, усереднюють чи підсумовують. При максимізації, з матриці виділяється найбільше значення і застосовується в якості виходу для цього осередку. На відміну від цього методу, усереднюючий шар знаходить середнє арифметичне значення для цієї області. Підсумовуючий тип виводить суму елементів кожної області. На практиці було з’ясовано, що шар, який максимізує, діє краще, оскільки дозволяє позбавитися від зайвих деталей.

Застосування шару субдискретизації позитивно впливає на стан згортальної нейронної мережі :

– Розмірність входу нейронної мережі зменшується, що робить метод більше керованим.

– Зменшується кількість параметрів і обчислень, що дозволяє контролювати ефект перенавчання.

– Модель стає більше інваріантною до невеликих змін, на зразок графічного шуму, артефактів і трансформацій.

– Оскільки модель стає більше інваріантною до трансформацій, то сам процес отримання об'єкту за допомогою такого фільтру стає інваріантним.

Традиційна згортальна нейронна мережа зазвичай завершується одним або кількома повнопов'язаними шарами (Fully Connected), які приймають на вхід значення карт властивостей а на вихід передають вектор певної довжини, яка, зазвичай дорівнює кількості класів для даної задачі. Тобто цей шар формує вектор з вірогідністю відношення початкового зображення до певних класів. Комбінація повнопов'язаних шарів має вигляд багатошарового персептрона, яка допомагає класифікувати високорівневі ознаки на останніх згортальних шарах субдискретизації.

Цей вид нейронних мереж, як у випадку з більшістю глибоких нейронних мереж, навчається за допомогою алгоритму зворотного поширення помилки. Так само для поліпшення і привнесення нелінійності в роботу нейронної мережі, на виході у згортальних шарів, що субдискретизують або повнопов'язаних, може використовуватися активаційна функція. При задачі класифікації найбільш відповідним варіантом є функція Softmax, оскільки сума результатів цієї функції для вектору аргументів дорівнюватиме 1, тим самим інтерпретуючи кожен результат як вірогідність належності до якого-небудь класу:

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Найбільш популярним вибором для згортального шару є рампова функція випрямленого лінійного елементу (ReLU) або функція-випрямитель (Rectifier), та її варіації. Цей вид нелінійної активаційної функції має наступний вигляд:

$$f(x) = \max(0, x)$$

Результат цієї функції можна представити у вигляді системи, де до 0 по X значення функції дорівнює 0, а після лінійно зростає. Застосування даної активаційної функції привносить до моделі, додаткової нелінійності, оскільки сама операція конволюції є лінійною. У разі застосування цієї функції для матриці, усі її елементи стануть ненегативними. Так само окрім ReLU можна використати будь-яку іншу нелінійну функцію, наприклад, Гіперболічний Тангенс або Сигмоїдальну функцію, але як показує практика, ReLU має кращий ефект у більшості випадків.

1.4. Колірні Простори RGB І YCBCR

На вхід згортальної нейронної мережі, якщо завдання полягає в роботі із зображеннями, подається вектор, де кожен піксель представлений певною кількістю вимірів, залежно від колірного простору, в якому представлено зображення. Традиційним колірним простором для графіки є RGB, тобто опис кожного пікселя через інтенсивність червоного, зеленого і синього каналів. Цей колірний простір описує усі ці компоненти, як рівнозначні. Цей формат

переважний для комп'ютерної графіки, але для зображення реальних фотографій цей формат може бути оптимізований, оскільки RGB для фотографій часто є надмірним.

YCbCr є оптимальнішим для цих задач колірним простором, оскільки заснований на економнішому принципі кодування RGB. Сигнали цього простору формуються із застосуванням гамма-корекції початкових сигналів RGB. Y - компонента яскравості, а Cb і Cr є спеціальними кольоророзницевими компонентами для синього і червоного каналів відповідно. Оскільки людське око менш чутливе до перепадів кольору, ніж світла, даний формат дозволяє передати менше інформації про кольороутворюючі компоненти, настроївши належним чином яскравість. Тому цей метод кодування, в цілях оптимізації витрат пам'яті, використовується при нормалізації і масштабуванні зображень.

1.5. Постановка задачі. Технічне завдання.

1. Вступ. Нейромережеве веб-застосування для збільшення растрових зображень (Neural network based web application for upscalling bitmap images).
Програма для розробників ігор, графічних редакторів, дизайнерів, журналістів і фотографів. Призначення розробки. Функціональним призначенням програми є надання користувачеві можливості збільшення розмірів зображення в два рази з мінімальною втратою якості за допомогою згортальної нейронної мережі.
Вимоги до програми або програмного виробу
1.1 Вимоги до функціональних характеристик
Вимоги до складу виконуваних функцій

- Завантаження растрових зображень.
- Збільшення зображень з мінімальною втратою в якості.
- Можливість отримати результат як на відкритій веб-сторінці, так і по прямому посиланню.

1.2 Вимоги до надійності
Надійне (стійке) функціонування програми має бути забезпечене виконанням замовником сукупності організаційно-технічних заходів, перелік яких приведений нижче :

- організацією безперебійного живлення технічних засобів;
- використанням ліцензійного програмного забезпечення;

1.3 Умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

1.4 Вимоги до складу і параметрів технічних засобів

Вимоги для сервера

1) Процесор Intel Xeon E5 - 2667 Haswell з тактовою частотою 3,2 ГГц, може бути виділено 4 і більше ядер.
2) Оперативна пам'ять 8 Гб, не менше.
3) Фізична пам'ять 500 Гб, не менше.

Вимоги для клієнта

1) У користувача на персональному комп'ютері має бути встановлений браузер, рекомендується Chrome версії 55 і вище.

1.5 Вимоги до інформаційної і програмної сумісності

Вимоги до інформаційних структур не пред'являються.

Вимоги до початкових кодів і мов програмування

Початкові коди програмного продукту мають бути реалізовані на мові C# для серверної частини веб-додатка і на мові JavaScript для клієнтської частини. У пріоритеті використання Visual Studio 2017 версій в якості середовища розробки.

Вимоги до захисту інформації і програм не пред'являються.

1.6 Вимоги до маркування і упаковки

Маркування і упаковка не потрібні.

1.7 Вимоги до транспортування і зберігання

Транспортування і зберігання не потрібно.

1.8 Спеціальні вимоги

1) Клієнтська частина має бути спроектована відповідно до правил User Experience Design.

2) Серверна частина має бути спроектована і розроблена відповідно до загальноприйнятих шаблонів проектування (software pattern), якщо це необхідно і можливо в конкретній ситуації.

3) Додаток повинен коректно працювати у браузері Chrome версії 55 і вище.

Стадії і етапи розробки

1.9 Стадії розробки

1) Технічне завдання.

2) Технічний проект.

3) Впровадження.

1.10 Етапи розробки

На стадії "Технічне завдання" має бути виконаний етап розробки, узгодження і затвердження технічного завдання.

На стадії "Технічний проект" мають бути виконані наступні етапи робіт :

1) Проектування архітектури нейронної мережі.

2) Проектування архітектури модуля масштабування.

3) Проектування архітектури веб-додатка.

4) Реалізація модуля масштабування.

5) Реалізація каналів обміну повідомленнями між вузлами додатка.

6) Реалізація графічного інтерфейсу.

7) Реалізація серверної частини веб-додатка.

Висновки по першому розділу

У цьому розділі була розглянуті питання збільшення (масштабування) зображення без втрати якості, а також задачі проектування нейронних мереж для збільшення зображень. За основу буде взято архітектуру Super Resolution Convolutional Neural Network

Виходячи з проведеного аналізу літературних джерел і ресурсів інтернет, список завдань включатиме наступне:

1. Вивчити алгоритм роботи нейронної мережі, розробленої для масштабування зображень.
2. Спроектувати модель нейронної мережі, яку можливо вбудувати в якості окремого модуля, що інкапсулює логіку масштабування зображення.
3. Підібрати оптимальні, з точки зору ефективності роботи нейронної мережі, параметри.
4. Розробити прототип додатка, що дозволяє продемонструвати роботу інтелектуального модуля для збільшення зображення.
5. Відлагодити розроблений програмний додаток з виправленням критичних помилок в роботі програми або нейронної мережі.

РОЗДІЛ 2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ

На даний час вже існують і доволі поширені класичні алгоритми для збільшення зображень (наприклад, білінійна або бікубічна інтерполяція). Проте вони не здатні повною мірою коректно заповнити ту інформацію, яка була втрачена при стисканні або конвертації зображення з одного формату в інший. Так, наприклад, алгоритм JPEG сильно схильний до втрати інформації. Усе відбувається тому, що JPEG є форматом стискання з втратами, і при кожному збереженні зростає кількість математичних усереднень, неточностей, помилок. Два збереження з 90 % стискуванням приблизно еквівалентні одному з 81 % по кількості артефактів. Така сама ситуація склалася і відносно відеоматеріалів. Для економії простору накопичувачів відеоматеріали піддаються стисканню з втратами і зменшенню роздільної здатності. Це призводить до появи і поширення низькоякісного відеоконтенту. Проте існує і значна кількість інших варіантів поліпшення зображень, наприклад, шумозаглушування або регулювання кольору і контрасту, які все одно не дозволяють відновити втрачену інформацію. Для супутникових знімків одним з показників якості є Ground Sample Distance (GSD), фізичний вимір, що представляється одним пікселем на зображенні. Для поліпшення таких зображень застосовують технологию, що називається «супердозволом». В англомовній літературі описано як подібні проблеми вирішуються методами SR (Super - Resolution). Ці методи дозволяють збільшити зображення, здолавши оптичну межу об'єктиву або роздільну здатність будь-якого іншого цифрового сенсора, за допомогою якого було знято зображення.

Технологія «супердозволу» (SR) виконує завдання поліпшення неякісних і стислих зображень високого розділення з дрібніших зображень (синтезується субпіксельна інформація невеликих зображень для обробки основного зображення) і має дуже важливе значення для застосування в областях аналізу супутникових і медичних зображень[1].

Синтез включає наступні методи:

- інтерполяція сусідніх пікселів в зображенні;
- интерполяция наближених кадрів в відео;

- частотна фільтрація для зменшення шуму.

Алгоритми SR дозволяють обчислити результуюче зображення двома способами: використовуючи фрагменти одного і того ж об'єкту [18] і використовуючи колекцію зразків різних зображень. Другий спосіб набагато більше масштабований за рахунок гнучкого налаштування параметрів структур з області штучного інтелекту, зокрема нейронних мереж.

У рамках цієї роботи планується вивчити поширені методи у напрямі використання штучних нейронних мереж. Для виконання поставленої мети пропонується досліджувати модель SRCNN, яка реалізується в наступному алгоритмі :

1) виконання інтерполяції для збільшення зображення з низьким дозволом до необхідного розміру;

2) формування нелінійної карти через тришарову згортальну мережу;

3) відновлення зображення з високим розділенням.

Як правило, для первинного збільшення зображення використовується алгоритм бікубічної інтерполяції.

Структура згортальної нейронної мережі для вирішення завдання SR (SRCNN)[19] допускає наскрізне перетворення (end - to - end mapping) елементів зображень низького і високого дозволу. Головною відмінністю цього методу від інших методів, заснованих на прикладах, являється те, що нейронна мережа використовується не для безпосереднього запам'ятовування колекцій фото, шукаючи збіги між зразком і поточним зображенням. Таким чином запропонована модель [19, 21] є більше інваріантною, ніж інші методи, засновані на прикладах.

Модель має бути простою, але доволі ефективною варіацією, якість роботи якої мала б бути кращою за інші аналогічні методи. Для перевірки роботи програми можуть бути використані такі метрики, як пікове співвідношення сигналу до шуму (PSNR), індекс структурної схожості (SSIM) і тому подібні. Результати планово мають перевершити методи бікубічної інтерполяції і подібний популярний метод заснований на прикладах - Sparse Coding [20].

Результати порівняння перетворених зображень за подібною методикою представлені на рисунку 2.1:

Рисунок 2.1. Визначення якості по методу PSNR

Цей метод є одним з найбільш швидких серед подібних, бо є алгоритмом прямого поширення і в ході його виконання не виникає проблем з оптимізацією процесу. Так само архітектура нейронної мережі для масштабування дозволяє одночасно обробляти усі три канали колірного простору (RGB і YCrCb), в якому було закодовано зображення, що підвищує її продуктивність.

Більшість попередніх методів SR працюють в монохромному режимі, тобто сам метод застосовується тільки до одного каналу, найчастіше для освітленості Y в просторі YCrCb, а інші канали масштабуються білінійною інтерполяцією. Так само цей метод грунтується не лише на ефективності згортальних шарів в роботі із зображеннями, але і використовує як активаційну функцію ReLU, оскільки ця функція підвищує швидкість обробки зображень (для різних завдань), при цьому зберігаючи прийнятну якість результату, та оберігаючи від згубних ефектів перенавчання при достатку прикладів в повчальній вибірці, як наприклад в завданнях у рамках конкурсу ImageNet, з якого і розпочався другий виток популярності цієї нейромережевої структури.

2.1. Алгоритм процесу обробки зображення моделлю SRCNN

Робота SRCNN розпочинається з єдиного процесу попередньої підготовки, початкового зображення, що полягає в масштабуванні, до бажаних розмірів за допомогою бікубичної інтерполяції. Позначимо Y (мал.) вхідне зображення, яке отримане в результаті інтерполяції, а зображення, з яким здійснюватиметься порівняння, як X. Завдання моделі повернути з Y зображення F (Y) і головним показником успішності цього процесу являється максимальна відповідність F (Y) оригінальному збільшеному зображенню X (ground - truth), яке використовується в процесі навчання, але абсолютно невідоме при цільовому використанні SRCNN. Для спрощення формулювання, Y далі асоціюватиметься з ординальным зображенням низького дозволу, незважаючи на той факт, що воно по розмірах рівно бажаному збільшеному зображенню, тобто X. Необхідно навчити систему наскрізному перетворенню (знайти функцію) F, яке концептуально складається з 3 основних етапів (рисунок 2.1.) :

1. Виділення ділянок зображення (patch extraction) :

ця операція витягає ділянки зображення низької роздільної здатності Y, що частково перекривають одна одну, які складаються з декількох пікселів. Необхідно перетворити кожен такий патч в набір карт ознак. Дані вектору містять в собі карти властивостей (feature maps), для яких їх кількість дорівнює розмірності цього вектору.

2. Нелінійне перетворення:

ця операція нелінійним чином співвідносить кожен високорозмірний вектор з іншим високорозмірним вектором. Кожен перетворений високорозмірний вектор (частина багатовимірного вектору) є тією ж ділянкою зображення, тільки з більш високою роздільною здатністю. Дані вектора так само містять в собі вже інші колекції карт властивостей.

3. Реконструкція зображення :

ця операція агрегує векторизовані образи ділянки зображення підвищеної роздільної здатності. Відтворюється зображення F(Y) коли для створення остаточного графічного представлення використовуються відображення високої

роздільності кожного патча, отримані на попередньому кроці. Передбачається, що результатом цього етапу буде зображення, максимально схоже на X.

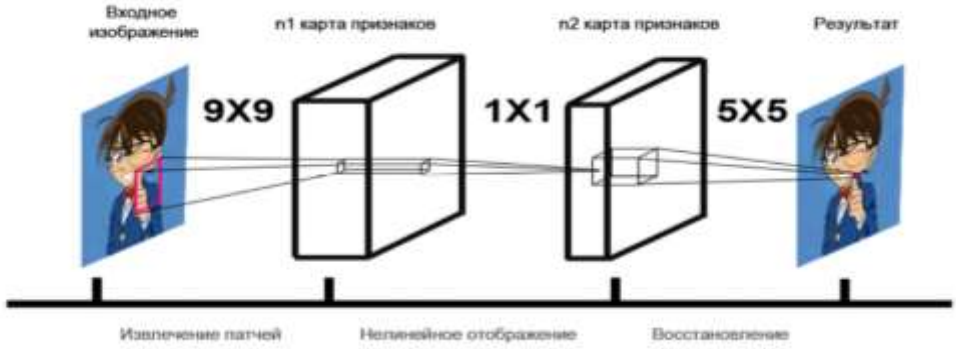

Рисунок 2.2.

Усі ці 3 етапи реалізовані на базі згортальних шарів нейронної мережі.

1. Перший шар.

Використовується для отримання патчів. Етап витягання традиційно для завдань SR представлений як процес щільного витягання ділянок зображення певної розмірності з подальшим застосуванням до них методу головних компонент, дискретного косинусного перетворення або перетворення Хаару для зменшення розмірності вхідних параметрів з мінімальними втратами. Цей процес аналогічний конволюції (згортці) зображення набором фільтрів. Таким чином роботу першого шару нейронної мережі можна представити у вигляді формули:

$$F_1(Y) = \max(0, W_1 * Y + B_1),$$

де W_1 та B_1 відповідають фільтру і зміщенню (bias) відповідно, а знак "*" означає операцию конволюції *(згортки).

W_1 являє собою набір фільтрів в кількості n_1 з розмірами $c * f_1 * f_1$. Фільтри W_1 можна представити формулою

$$W_1 = c \times f1 \times f1,$$

де c дорівнює кількості каналів зображення Y, а f_1 просторовий розмір фільтра. Фільтри W_1 здійснюють n_1 операцій згортки зображення, й кожна згортка має ядро згортання розміром $c * f_1 * f_1$. Вихідні дані слоя містять n_1 карт ознак. Ваги B_1 представляють собою n_1-мірний вектор, кожен елемент якого сопоставлений з

елементом фільтра W_1 . В якості активаційної функції першого слоя застосовують функцию ReLU (Rectified Linear Unit)

f(x)=$\max(0, x)$

де x – вхідний сигнал..

Результат є набором карт властивостей у кількості n_1.

2. Другий шар.

Етап нелінійного перетворення, що слідує за витяганням, приймає на вхід вектор розмірності n_1 і перетворює їх у вектори розмірності n_2. Ця операція рівнозначна конволюції з фільтрами розмірами 1*1 у кількості n_2. До речі, ця інтерпретація дійсна тільки для фільтрів з одиничною розмірністю. Таким чином операцію можна описати формулою:

$$F_2(Y) = \max(0, W_2 * F_1(Y) + B_2)$$

де параметри W_2 і B_2 аналогічні відповідному фільтру і зміщенню (bias) для F_1, а знак "*" означає чергову операцію конволюції. W_2 знову є набором фільтрів у кількості n_2 з розмірністю $n_1 * f_2 * f_2$.

Тепер фільтри здійснюють n_2 операцій, а ваги представлені n_2 - мірним вектором. До конволюції (згортки) так само застосовується функція ReLU.

Кожен новий вектор розмірності n_2 концептуально є ділянкою високої роздільної здатності, пов'язаний з аналогічною ділянкою низького дозволу. Можливо збільшити кількість прихованих згортальних шарів для збільшення рівня нелінійності перетворення, але ця модифікація ускладнить систему і приведе до збільшеного часу її навчання, тому у рамках цієї роботи нелінійне перетворення складатиметься з однієї конволюції.

3. Третій шар.

Шар здійснює фінальну операцію відновлення зображення. На етапі реконструкції в традиційних методах SR значення областей, що перекриваються, усереднюються, і у результаті складаються в результуюче зображення високого розділення. Процес усереднення так само може бути інтерпретований, як конволюція встановленим заздалегідь фільтром для карти властивостей, де кожна

позиція є "згладженою" областю високого розділення. Зважаючи на це припущення, останній шар можна описати за допомогою формули:

$$F(Y) = W_3 * F_2(Y) + B_3,$$

де W_3 - означає набор з лінійних фільтрів розмірності $n_2 * f_3 * f_3$ в кількості c, а B_2 є вектором розмірності c.

Вихідними даними цього шару є зображення високого розділення. Для активації цього шару використовується лінійна функція:

$$f(x) = x,$$

де x – це вхідний сигнал.

Як функція втрат використовується середньоквадратична помилка MSE (Mean Squared Error) :

$$L(\theta) = 1/n \sum_{i=1}^{n} | F(Y_i) - X_i |^2$$

де Θ – це конфігурація нейронної мережі $W_1W_2W_3B_1B_2B_3$, n – число семплів навчальної вибірки {2}.

Поліпшення якості відео передбачає кілька додаткових операцій по розкладанню на кадри і зворотну зборку відеодоріжки. В цьому випадку кількість даних для обробки буде істотно збільшена в порівнянні з простою обробкою серії зображень. Це означає, що для поліпшення одного відеофайлу в розумні терміни необхідно буде використати графічні процесори для прискорення процесу.

Структура нейронної мережі зображена на рисунку 2.2:

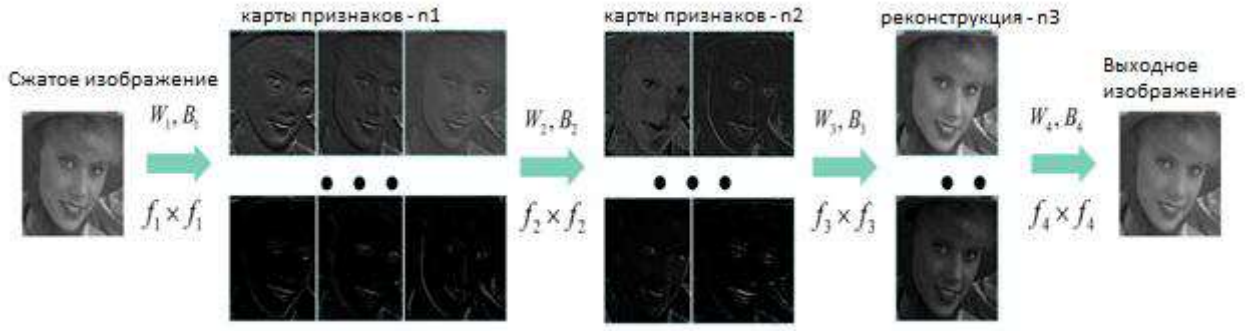

Рисунок 2.3. Структура SRCNN

Для навчання моделей можуть використовуватися різні набори зображень. Проте кожне перенавчання може як поліпшити, так і погіршити вихідну якість для конкретної моделі, тому слід обгрунтовано вибирати набори даних під кожне завдання.

2.2. Можливість модифікації для моделі SRCNN

Ця модель заснована на ідеях, розвинених в топології NiN [15]. На етапі нелінійного перетворення використовується згортальний шар з фільтром одиничної розмірності, що дозволяє ефективно зв'язати вектори різної розмірності, тим самим здійснивши прив'язку зображень низької і високої якості. Для першого і останнього етапу автори рекомендують використати фільтри розмірністю 9 і 5 відповідно, що так само характерно для прогресивних топологій, на зразок VGG, NiN і Inception.

Під час аналізу сучасних топологій НМ і співвідношення їх із структурою SRCNN, можна припустити, що нейромережу можна поліпшити, використавши більшу кількість шарів. При цьому з історичних причин, топологія VGG відмінно справляється з цим завданням. Так само, для спрощення моделі, можна спроектувати нейромережу із згортальними шарами однакової розмірності фільтру 3×3, проігнорувавши при цьому шар одиничної розмірності, та замінивши його активаційною функцією ReLU після кожного згортального шару.

2.3. UML діаграми для системи.

Універсальна мова моделювання (Unified Modelling Language або UML) — це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення. UML не є методом розробки. Іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає наочно переглядати компонування системи, покращує сприйняття задачі і полегшує співпрацю з членами команди розробників. Тож для полегшення подальшої розробки бажано створити кілька діаграм.

Функціональні вимоги до додатка передбачають:

- Обробку зображень для збільшення з в 2 рази.
- Можливість в реальному часі відстежувати прогрес обробки зображення (виведення інформації про поточний етап).
- Можливість отримання зображення як в режимі реального часу через ajax, так і по прямому посиланню через унікальний тікет (ідентифікатор) запиту.

На рисунку 2.4 представлена діаграма варіантів використання програми:

Рисунок 2.4. Діаграма варіантів використання веб-додатку

На рисунку 2.5. наведена діаграма класів модуля масштабування

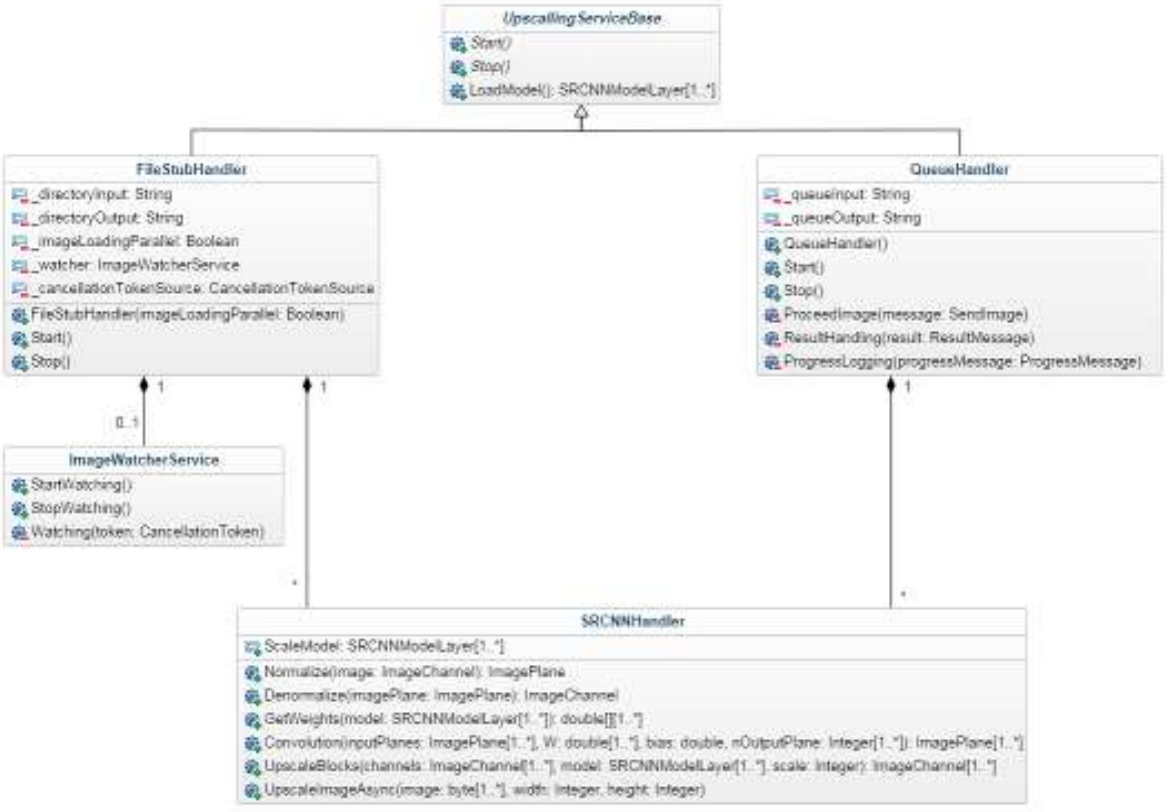

Рисунок 2.5. Діаграма класів модуля масштабування

З інтерфейсною частиною проекта повязана діаграма послідовності завантаження зображень. (рисунок 2.6.)

Рисунок 2.6. Діаграма послідовностей завантаження зображень для масштабування

2.4. Вибір інструментальних засобів

Початкові коди програмного продукту мають бути реалізовані на мові C# для серверної частини веб-додатка та мовою JavaScript для клієнтської частини.

C# – це багатоцільова мова програмування що згодиться для всього: мобільних додатків (Xamarin та MAUI), для сайтів є Blazor. Також C# це і серверні програми, десктопні програми, сайти та багато іншого. Також C# – це розробка Microsoft та основна мова для платформи .NET, отже ви отримуєте доступ до інструментів Microsoft: Visual Studio, що є потужним середовищем для розробки, або ж її аналог Visual Studio Code, яку використовують багато розробників JavaScript, PHP, Java, Go, Python та інших мов. Окрім середовища розробки Microsoft дає можливість використовувати GitHub Copilot ─ штучний інтелект, який допомагає у написанні коду, що важливо і для початківців, і для просунутих користувачів, адже можна скоротити час на рутину чи уникнути типових помилок. C# — строго типізована мова, компілятор перевіряє код помилки ще до його виконання. Визначаючи потенційні помилки заздалегідь, C# позбавляє від складнощів налагодження, забезпечує надійність і стійкість програм. C# постійно розвивається, отримуючи нові функції. Такі часті оновлення — це гарантія актуальності і конкурентоспроможності C# в світі програмування. Величезною перевагою використання C# є вбудований збирач сміття, що автоматично звільняє пам'ять від змінних, які більше не використовуватимуться. А отже це значить, що більше не потрібно буде турбуватися про ручне управління пам'яттю й забути про витік пам'яті та збої, котрі пов'язані з цим.

JavaScript — це високорівнева скриптова мова програмування для розробки динамічних вебсайтів і вебдодатків. Створена компанією Netscape в 1995 році, вона є найпопулярнішою мовою програмування в світі та в Україні, посідаючи друге місце в рейтингу вибору для вивчення. Найчастіше цією мовою створюють інтерактивні елементи на сайтах, за допомогою яких сайти «оживають», починають реагувати на дії користувачів. Однак дана мова популярна не тільки в створенні сайтів. Її використовують для розробки прикладних програм, наприклад, браузерів, додатків для SMART-телевізорів, фітнес-трекерів, розумних годинників, приставок, в макросах для офісних програм.

Висновки до другого розділу

У другому розділі було розглянуто алгоритми збільшення зображень без втрати їх якості. В ході роботи проведено порівняльний аналіз сучасних підходів до масштабування зображень і обґрунтовано вибір застосування глибокої згорткової нейронної мережі. Для реалізації задачі пропонується модель SRCNN.. Алгоритм процесу обробки зображення даною моделлю передбачає:

1) виконання інтерполяції для збільшення зображення з низьким дозволом до необхідного розміру;

2) формування нелінійної карти через тришарову згортальну мережу;

3) відновлення зображення з високою роздільною здатністю.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Розробка програмного модуля зі збільшення зображень передбачає програмну реалізацію мережі SRCNN в оригінальному і дещо модифікованому вигляді зі зміненими налаштуваннями. Також для можливості демонстрації результатів роботи даного модуля розробляється інтерфейс користувача.

3.1. Використані технології

Для навчання мережі було обрано зображення з колекції бази даних ImageNet. Це — проект зі створення і супроводу масивної бази анотованих зображень, призначена для опрацювання і тестування методів розпізнавання образів і машинного зору. На даний час вона містить більше чотирнадцяти мільйонів URL із зображеннями, які пройшли ручну анотацію для ImageNet, в анотаціях перераховувалися об'єкти, що потрапили на зображення, і прямокутники з їх координатами. База даних з анотацією та URL зображень від третіх осіб доступна безпосередньо через ImageNet, але при цьому самі зображення не належать проекту.

Модель нейронної мережі складається з 7 шарів. Якщо застосовувати обробку зображень в браузері, без серверної компоненти, то може не вистачати обчислювальних ресурсів і пам’яті конкретного комп’ютера. Тож, через обмеження що можуть накладатись апаратною складовою, краще здійснювати реалізацію з переносом основного логічного навантаження в бік сервера.

Загалом запропонована реалізація програмного додатку для роботи з нейронними мережами являє собою клієнт-серверну систему. Клієнтська частина створювалась засобами мови JavaScript (ECMAScript 6). Це відносно нова версія популярної мови програмування JavaScript. Ecmascript відповідає за синтаксис мови, типи, прототипи і спадкоємство, стандартну бібліотеку вбудованих об'єктів і функцій, а саме: JSON, Math, методи масивів, методи об'єктів і багато іншого. Крім того покриває ті аспекти мови, які представлені не лише у веб-браузерах, але і в платформах типу node.js. ECMAScript 6 має досить багато відмінностей від попередніх версій мови. Ці нововведення є результатом довгої і копіткої роботи фахівців в цій галузі. Саме вони створили багато нових і корисних можливостей

ECMA. Можна стверджувати, що це найзначиміше оновлення JS, яке було з його появи.

Серверна частина веб-додатку реалізована на платформі.net Framework мовою C#. Було взято за приклад бібліотеку з ресурсу гітхаб з відкритим вихідним кодом «Реалізація на C++ згорткової нейронної мережі». Для зручності демонстрації роботи додатку створено користувацький інтерфейс, який забезпечує виавантаження вихідних зображень з обраного каталога та розміщення нових отриманих зображень в інший каталог призначення. Частини програмного коду з описом створених класів наведено в додатку А. Зокрема маємо класи:

public class SRCNN_handler

public class Image_blocks

public class Image_channel

public class Image_channels

public class Image_plane

public class Model_config

public class Queue_handler

public abstract class Upscalling_service_base

public static class Reflection_help

public class Block_upscall

public class Progress_message

public class Result_message

public class File_handler

public class Image_watch

Докладніше ці класи наведено в додатку до дипломної роботи.

3.2. Складові архітектури веб-додатку

Програмний додаток логічно поділений на дві складові – інтерфейсну і функціональну. Користувачу надається простий веб-інтерфейс, написаний з застосуванням технологій JavaScript, CSS3 и HTML5.

Розташування елементів користувацького інтерфейса реалізовано мовою розмітки HTML, а стилі розроблялись за допомогою синтаксиса CSS-препроцесора Less. Як виглядає інтерфейс представлено на Рисунку 3.1:

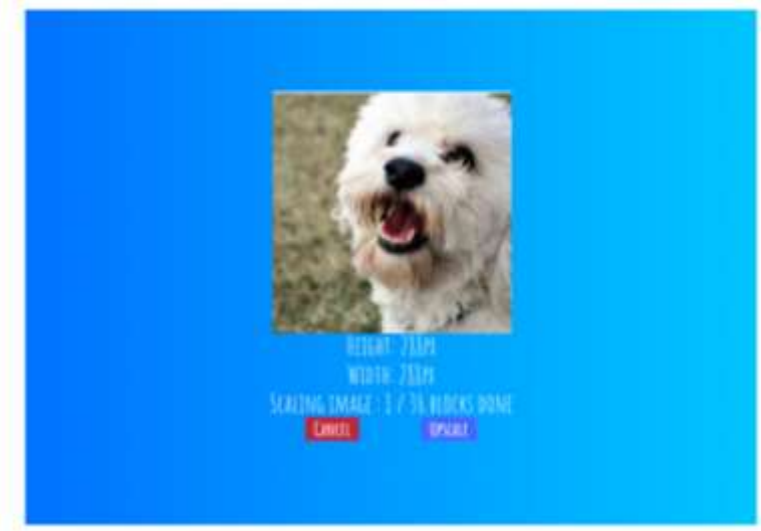

Рисунок 3.1. Інтерфейсна частина додатку

3.2.1. Клієнтська частина додатка реалізує процеси:

- вибору/завантаження файлу користувачем;
- запуск процесу масштабування зображення;
- виведення результатів попередньої операції.

Реалізація клієнтської логіки здійснювалась мовою JavaScript , точніше ECMAScript 6, яка має зручний API для роботи з колекціями і масивами, а також технологію управління асинхронними операціями Promises. Ця версія мови підтримується переважною більшістю сучасних і актуальних браузерів останніх версій.

Для реалізації структурного патерна Model-View-ViewModel використана бібліотека VueJS. Це фреймворк, що працює на JavaScript, створений для розробки користувацьких інтерфейсів, і надає можливості декларативно програмувати користувацькі інтерфейси будь-якої складності на основі його компонентів. Своїм успіхом Vue зобов'язаний компонентній моделі. Робота з компонентами Vue, як і в React чи Angular, означає розбиття інтерфейсу на маленькі блоки. Це сприяє застосуванню модульного підходу до розробки, повторного використання таких блоків та підтримки одноманітності зовнішнього

вигляду додатків. Усі компоненти Vue, крім того, є і екземплярами Vue, а це означає, що вони приймають однакові об'єкти з параметрами (за винятком деяких особливих параметрів, потрібних для кореневого об'єкту) і надають розробникові однакові можливості по обробці подій їх життєвого циклу. Технічно, Vue.js визначена як ViewModel шар шаблону MVVM. Вона сполучає модель і представлення в двостороннє зв'язування даних. Поточні DOM-зміни і форматований вивід абстрагуються в Директивах і Фільтрах.

Структурний патерн Model - View - ViewModel є аналогом патерну Model - View - Controller і дозволяє розділити логіку, модель даних і представлення на 3 незалежних компоненти, а VueJS інкапсулює механізм двосторонньої прив'язки даних (Model) і представлень (View) через прошарок (ViewModel), з подальшим взаємним оновленням стану.

В Vue.js View-Model'и створюються через клас Vue. Тут View-Model'і є об’єктами, що спрощують відображення даних моделі (сожна співвідносити любий об’єект як модель до HTML элемента в якості представлення).

При створенні представлення. спочатку створимо порожню сторінку :

```
<div id="my_view">
</div>
```

Для створення об’єкта моделі в JavaScript його можна розташувати в тезі <script>, але краще винести його в окремий файл.

```
var myModel = {
  name: "Image_1",
  height: 299px
};
```

Коли є наявні представлення і модель, створюється View-Model (Vue інстанс), що поєднає їх разом:

```
var myViewModel = new Vue({
  el: '#my_view',
  data: myModel
});
```

Тут властивість el вказує на представлення (мозна застосувати любий CSS селектор), а властивість data показує на об’єкт моделі.

Щоб відображати дані моделі всередині представлення, застосовуються спеціальні токени заключнені в фігурні скобки. У нас , щоб відобразити властивість height, треба додати токен {{ height }} всередині представлення. В наступному фрагменті кода, використано обідві властивості моделі:

```
<div id="my_view">
 File {{ name }}  height is {{ height }}.
  <!-- Evaluated to "File Image_1 height is 299px" -->
</div>
```

Це забезепечує те що всі зміни в моделі відразу буде показано в представленні.

Токени у вигляді фігурних дужок - це один із способів зв'язування даних. Вони дозволяють відображати моделі, але не змінювати їх. Щоб представлення змінювало дані, то треба реалізувати двосторонній зв'язок, використовуючи директиву v - model. Щоб наше представлення містило input атрибут v-model якого посилається на властивість моделі name:

```
<div id="my_view">
  <label for="name">Enter name:</label>
  <input type="text" v-model="name" id="name" name="name" />
  <p> File {{ name }}  height is {{ height }}.</p>
</div>
```

Забезпечує моментальну зміну значення в моделі при його введенні.

Також в токенах застосовуються фільтри (функції). Перед фільтром завжди ставлять символ pipe (вертикальна риска |). Наприклад, щоб назва була в верхньюму регістрі, то токен буде наступним:

```
{{ name | uppercase }}
```

Фільтри lowercase і capitalize застосовуються подібним чином.

Для виведення вмісту масивів користуємось директивою v-for, додавши її наприклад до елемента <li> списка. Це виглядатиме приблизно так:

```
var myModel = {
```

```
  name: "Image_1",
  height: 299px,
related: [
   { name: " Image_11", height: 600px },
   { name: " Image_12", height: 300px },
   { name: " Image_13", height: 900px }
   ]
};
```

Для виведення (отримання) влестивості height у кожного об’єкта в масиві пишемо:

```
<div id="my_view">
  <ul>
    <li v-for=" related in related "> {{ related.name }} </li>
  </ul>
</div>
```

Коли треба змінити порядок виведення елементів зі списку, застосовується фільтр orderBy всередині директиви v-for. Коли нам треба відсортувати наші елементи-файли за висотою то воно виглядає так:

```
<div id="my_view">
  <ul>
    <li v-for=" related in related | orderBy ' height '"> {{ related. height }}</li>
  </ul>
</div>
```

Щоб задавати умову виведення елементів застосовується фільтр filterBy, за його допомогою здійснюється пошук. Щоб вивести лише елементи розміром(висотою) 600 напишемо.

```
<div id="my_view">
  <ul>
    <li v-for=" related in related | filterBy  '600' in  ' height'"> {{ related. height }} </li>
  </ul>
```

```
</div>
```

В Vue.js, при обробці подій, треба додавати обробники подій у властивість methods нашої View-Model. Всередині обробників Vue.js, бажано застосовувати ключове слово this, для доступа до даних. Для обробки кліка маємо:

```
var myViewModel = new Vue({
  el: '#my_view',
  data: myModel,
  // A click handler inside methods
  methods: {
    myClickHandler: function(e) {
      alert("Go_Run " + this.name);
    }
  }
});
```

Для зв'язування обробника подій, що був визначений в View-Model, з елементами інтерфейса в представленні, користуємось директивою v-on. Елемент <button> з представлення, за допомогою директиви v-on викликає обробник myClickHandler:

```
<div id="my_view">
  Name: <input type="text" v-model="name">
  <button v-on:click="myClickHandler"> Go_Run </button>
</div>
```

Також Vue.js дозволяє створювати власні HTML елементи для представлень. Це дозволяє адаптувати код для потреб конкретного розробника. Користувацький HTML-елемент, створюється через метод component класа Vue. За допомогою властивості template визначається вміст конкретного елемента користувача..

3.2.2. Функціональна частина.

Друга частина клієнт-серверного застосування теж логічно та фізично розділена на 2 частини. Це невеличке застосування, що приймає запити від клієнта і виводить головну веб-сторінку системи, та відокремлений модуль

безпосередньої обробки зображень, який реалізує роботу нейронної мережі SRCNN. Для повноцінного функціонування цей модуль винесено в окрему бібліотеку, оскільки окрім веб-інтерфейсу (GUI) ще має працювати інтерфейс командного рядка (CLI), тож прив'язка до одного з них всередині одного модуля була небажаною.

Серверна частина веб-додатка реалізована через фреймворк ASP.net Core, який може забезпечити роботу веб-додатка як у рамках стандартного для Windows веб-сервера IIS, так і у рамках власного процесу (розгортає окремий веб-сервер для додатка). Також для можливості розгортання додатка на операційній системі Linux усі проекти були написані для зборки на новій версії платформи.net Core.

У складних системах, системах з обчислювальними завданнями, а також таких що вимагають гарантованої доставки важливо організувати розподілену архітектуру і налагодити комунікацію між компонентами. Через те що процес масштабування зображення є досить довгим, то при великому завантаженні сервера і великих розмірах зображень, взаємодію ASP.net застосування і модуля SRCNN краще здійснювати через брокера черг повідомлень з відкритим початковим кодом RabbitMQ. Цей брокер є найбільш популярним безкоштовним рішенням, що реалізовує протокол для черг повідомлень Advanced Message Queuing Protocol (AMQP)[22], а ще має реалізацію для платформи.net Core, що було основною мотивацією вибору RabbitMQ для нашої роботи.

RabbitMQ дозволяє обмінюватися повідомленнями між різними програмами, він підтримує наступні мови: Python, Java, Ruby, PHP, C#, JavaScript, Go, Elixir, Objective - C, Swift, Spring AMQP. RabbitMQ можна використати в разніх випадках: для фонової обробки даних та для інтеграції всередині додатків і між ними, тобто в якості брокера повідомлень між мікросервісами.

Фонова обробка даних є оптимальним сценарієм використання RabbitMQ. Тут можна просто помістити повідомлення в чергу без його негайної обробки. Якщо треба завантажити звіт з додатка, то додаток обробляє інформацію і генерує PDF -файл впродовж 15-20 хвилин, а потім відправляє його по електронній пошті.

Черга повідомлень дозволяє виконати усі завдання асихронно. Черги RabbitMQ служать шинами подій і дозволяють веб-серверам швидше реагувати на запити замість того, щоб виконувати трудомісткі завдання на місці.

Також RabbitMQ часто використовується для мікросервісної архітектури, де він виступає засобом зв'язку між додатками і допомагає уникати вузьких місць при передачі повідомлень. Черги повідомлень дозволяють балансувати навантаження на обчислювальний модуль SRCNN, бо обробляють саме стільки зображень, скільки дозволяє продуктивність сервера. Так само цей підхід дозволяє оперативно відстежувати навантаження на модуль через доволі зручний інтерфейс, представлений на Рисунку 3.2:

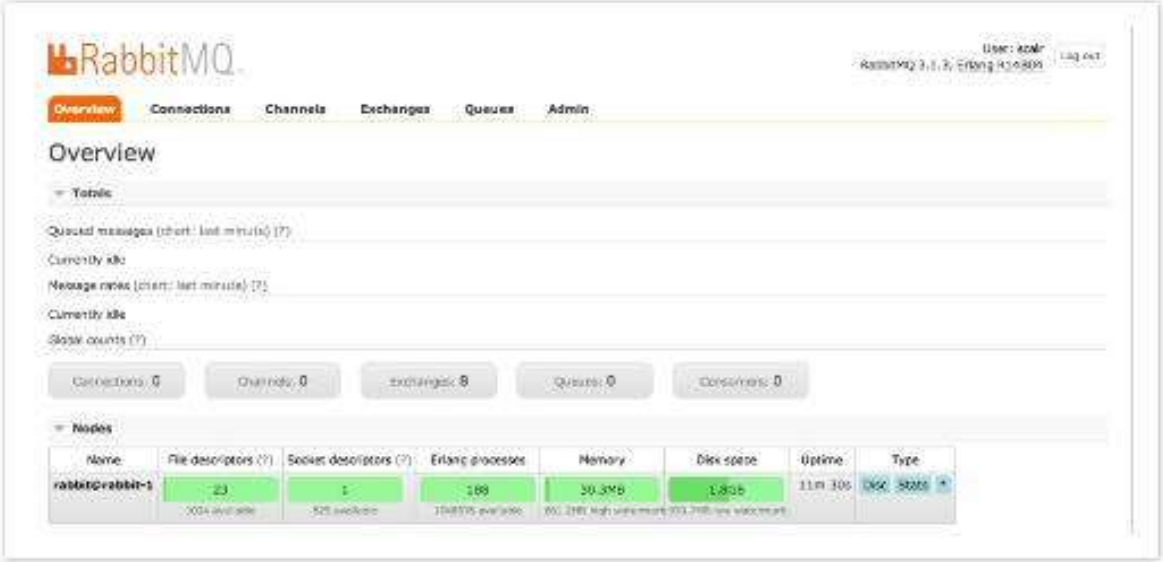

Рисунок 3.2. Моніторинг черг в RabbitMQ

В разі потреби є можливість підключати додаткові обробники (модулі мережі SRCNN) для паралельної роботи з чергою зображень, що поступає. Це дозволить збільшити пропускну спроможність усього застосування. Також черги повідомлень дозволяють найбільш безпечним способом реалізовувати асинхронну обробку завдань, тому що після того, як ASP.net застосування відправило зображення до черги, воно не простоює в очікуванні до закінчення обробки, а повертає ідентифікатор завдання (тікет), за допомогою якого клієнтське застосування може опитувати брокер про поточний стан обробки і про отриманий результат у вигляді масштабованого зображення.

Для реалізації асинхронної операції отримання збільшеного зображення у брокері повідомлень RabbitMQ створено три черги: ImageInput, ImageProgress та ImageOutput. на основі класу QueueHandler (в UpscallingServiceBase).(Додаток А)

При отриманні зображення від клієнта, Asp.net застосування формує об'єкт повідомлення для черги, розміщуючи саме зображення у бінарному виді в тіло повідомлення (Body) і привласнюючи цьому завданню унікальний ідентифікатор Guid, шо описано в методі void PublishMessage (IModel outputChannel, Guid ticket, byte [] messageBody) (див додаток А) який прикріпляється до усіх повідомлень, пов'язаних з цим завданням, наприклад, до повідомлень про зміну статусу процесу масштабування зображення або до повідомлення про отримання кінцевого результату.

Після ініціалізації повідомлення, воно поміщається в чергу ImageInput, а клієнтській частині додатка повертається тікет, за допомогою якого вона періодично опитує Asp.net застосування про стан завдання. Модуль SRCNN отримує повідомлення з черги ImageInput, витягає з нього тікет і зображення, та запускає процес обробки зображення.

При зміні етапу або статусу завдання, модуль SRCNN так само асихронно відправляє повідомлення з даними про поточний статус в чергу ImageProgress, прикріплюючи до нього збережений на самому початку тікет завдання. Ці повідомлення отримує додаток Asp.net, коли до нього приходить запит на отримання останнього актуального стану завдання з вказаним тікетом. По закінченні операції, Asp.net застосування отримує результат з черги ImageOutput.
Самі процеси передачі файла і отримання зображення повністю відповідають діаграмі послідовностей що наведена в розділі 2 дипломної роботи.

Модуль масштабування може працювати як з чергою повідомлень, так і за допомогою тек, які імітують черги. Це також дуже зручно. Сам алгоритм масштабування передбачає розпаралелювання процесів масштабування зображення, шляхом розділення зображення на блоки однакових розмірів, які оброблятимуться алгоритмом в різних потоках. Для цих цілей використовується

бібліотека Task Parallel Library (TPL), що входить до числа стандартних бібліотек платформи.net.

В основі бібліотеки TPL лежить концепція задач, кожна з яких описує окрему тривалу операцію. У бібліотеці класів .NET завдання представлене спеціальним класом - класом Task, який знаходиться в просторі імен System.Threading.Tasks. Цей клас описує окреме завдання, яке запускається асихронно в одному з потоків з пулу потоків. Хоча його також можна запускати синхронно в поточному потоці.

Для визначення і запуску завдання можна використати різні способи.

Перший спосіб створення об'єкту Task і виклик у нього методу Start :

Task task = new Task(() => Console.WriteLine("Started Task"!));

task.Start();

В якості параметра об'єкт Task приймає делегат Action, тобто ми можемо передати будь-яку дію, яка відповідає цьому делегатові, або посилання на який-небудь метод. У прикладі при виконанні завдання на консоль виводитиметься рядок " *Started Task".* А метод Start() власне запускає завдання.

Другий спосіб полягає у використанні статичного методу Task.Factory.StartNew(). Цей метод також в якості параметра приймає делегат Action, який вказує, яка дія виконуватиметься. При цьому цей метод відразу ж запускає завдання:

Task task = Task.Factory.StartNew(() => Console.WriteLine("Started Task"!));

В якості результату метод повертає запущене завдання.

Третій спосіб визначення і запуску завдань представляє використання статичного методу Task.Run() :

Task task = Task.Run(() => Console.WriteLine("Started Task"!));

Метод Task.Run() також параметром може приймати делегата Action - виконувану дію а повертає об'єкт Task.

Все це дозволяє задіяти відразу декілька ядер процесора. Таким чином на багатоядерних обчислювальних машинах процес виконуватиметься швидше.

Архітектура для оптимізованого распараллеленого застосування представлена на рисунку 3.3.

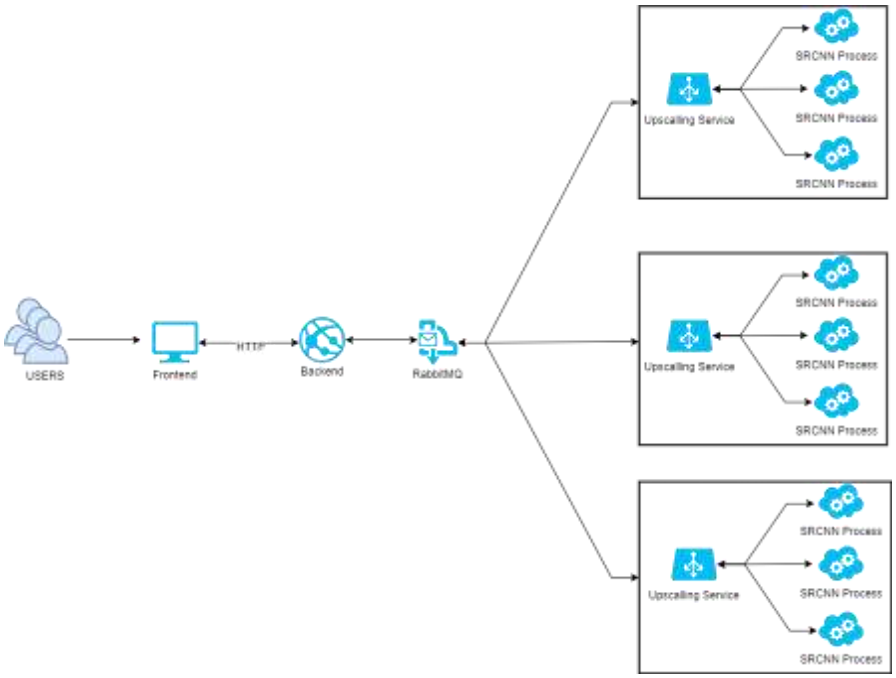

Рисунок 3.3 Архітектура розпаралелювання

Однією з основних зручностей застосування черг повідомлень і розпаралелювання задач є перетворення додатка на горизонтально масштабовану систему. Для підвищення пропускної спроможності можна спробувати додати додаткові обчислювальні машини, що неможливо реалізувати у разі типового рішення на базі клієнтського Javascript –додатку. Проте у цього є і свої недоліки. Збільшення кількості вузлів системи, неминуче підвищить складність архітектури і природно кількісно зростуть часові витрати, необхідні для взаємодії окремих модулів між собою.

3.3. Результати роботи

Для тестування роботи застосунку використовувалися 5 зображень. Результати порівнювалися із зображеннями, отриманими з тих же вихідних але за допомогою алгоритму бікубічної інтерполяції.

Для оцінки якості зображень є два підходи: кількісна оцінка за допомогою використання математичних методів (середньоквадратична помилка, Lp -норма [1-4], засоби, що враховують особливості сприйняття зображення зоровою

системою людини і т.п.) і суб'єктивна оцінка на основі експертних оцінок. З іншого боку, суб'єктивні і кількісні оцінки якості зображень можуть бути абсолютними або порівняльними. Абсолютна міра якості використовується для оцінки одного зображення, тобто зображенню привласнюється відповідна категорія в рейтинговій шкалі. Порівняльні заходи використовуються для ранжирування набору зображень в якісній шкалі від «найкраще» до «найгірше» або взаємного порівняння двох зображень, наприклад, початкового і відфільтрованого (або отриманого в різні дні, різними камерами і т.д)

Для оцінки якості зображення пропонується метрика PSNR (Peak Signal to Noise Ratio), де для оцінки якості передачі відео чи статичних зображень використовується пікове відношення сигналу до шуму. Дана метрика є інженерним терміном, що означає співвідношення між максимумом можливого значення сигналу і потужністю шуму, що спотворює значення сигналу. Оскільки багато сигналів мають широкий динамічний діапазон, PSNR зазвичай вимірюється в логарифмічній шкалі в децибелах. PSNR найчастіше використовується для виміру рівня спотворень при стискуванні зображень. Простіше усього його визначити через середньоквадратичне відхилення (MSE). Для двох монохромних зображень I і K розміру m×n, одне з яких вважається «зашумленим» наближенням іншого, MSE обчислюється так:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |I(i,j) - K(i,j)|^2 ,$$

де I і K є двома зображенням розміром m×n, причому одне є зашумленою варіацією іншого. Таким чином сам вимір PSNR визначається так:

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

де MAX_I — це максимальне значення, що приймається пікселем зображення. Коли пікселі мають розрядність 8 біт. Якщо $MAX_I = 255$, це, коли значення сигналу представлені лінійно з B бітами на значення, максимально можливе значення MAX_I буде 2^B -1. Для кольорових зображень з трьома компонентами RGB на піксель застосовується таке ж визначення PSNR, але MSE вважається по

усіх трьох компонентах (і ділиться на потрійний розмір зображення). Ця метрика, по суті, аналогічна середньоквадратичному відхиленню, проте користуватися нею дещо зручніше за рахунок логарифмічного масштабу шкали. Їй властиві ті ж недоліки, що і середньоквадратичному відхиленню. Слід зазначити, що «хороший» PSNR не завжди гарантує хорошу якість зображення, через те що зорова система людини має нелінійну поведінку.

Результати тестування двох методів і їх порівняння представлені в Таблиці 3.1:

Таблиця 3.1. Зображення при порівнянні

Картинка	PSNR для бікубічної інтерполяції	PSNR для SRCNN
	 33.56 дБ	 37.45 дБ
	 30.59 дБ	 32.53 дБ
	 23.45 дБ	 32.50 дБ
	 31.53 дБ	 35.14 дБ
	 26.85 дБ	 35.18 дБ

Навіть візуально помітна різниця між бікубічною інтерполяцією і SRCNN. SRCNN дозволяє отримати більш деталізовані і чіткіші зображення, особливо це стосується невеликих деталей, як то хутро тварин на всіх зображеннях і на візерунках аксесуарів на першому і передостанньому. Незважаючи на доволі тривалий час виконання процесу, результат вірно передає малопомітні деталі першого зображення, що робить даний підхід виграшним в плані якості. Так само

результат модифікованої моделі SRCNN є повністю задовільним, що доводить позитивний ефект від збільшення кількості згортальних шарів.

Висновки до третього розділу

У даному розділі була описано програмну реалізацію згорткової нейронної мережі, спеціалізованої на обробці структурованих даних. Згорткові нейронні мережі широко використовуються в завданнях обробки зображень і в інших сферах, що робить їх однією з найпопулярніших архітектур нейронних мереж.

Для поліпшення відновлення стислих зображень були розглянуті кращі моделі згорткових нейронних мереж, включаючи мережу ARCNN, яка є розширенням моделі SRCNN і розв'язує завдання супер-роздільної здатності за допомогою додавання покращувального шару для виділення ознак зображення.

На основі цього дослідження була розроблена інша модель згорткових нейронних мереж, яка показала кращі результати на експериментальних даних порівняно з ARCNN. Хоча час обробки трохи збільшився, але це не критично в даному контексті, оскільки ці алгоритми не призначені для реального часу. Основний увага була зосереджена на універсальності обробки та забезпеченні якості. Експерименти також вказали на те, що якість відновлення в значній мірі залежить від вибору моделі згорткової нейронної мережі, а не від розміру навчального набору даних.

Загальні висновки

У дипломній роботі була створена модифікована тришарова згорткова нейронна мережа, яка використовує топологію VGG для масштабування зображень. Ця модифікація відрізняється від оригінальної моделі SRCNN та покращує якість збільшених зображень, навіть при додаванні додаткових шарів.

Запропонована нейронна мережа оброблює блоки зображень паралельно та рівномірно розподіляє навантаження між обчислювальними ядрами процесора. Використання черг повідомлень дозволяє розподілити обчислення масштабування зображень між різними комп'ютерами, що покращує швидкість обробки. Проте ця нейронна мережа поки що не готова до використання в режимі реального часу, і для цього може бути доцільніше використовувати більш просту топологію Network in Network з меншою кількістю шарів.

Можливі напрямки подальшого вдосконалення включають оптимізацію через додаткові приховані шари та оптимізацію представлення кольорових каналів зображень. Крім того, швидкість обчислень може бути підвищена за допомогою графічних адаптерів та програмного забезпечення CUDA від Nvidia.

Для перевірки ефективності запропонованої моделі було створено два варіанти оптимізації. Перший варіант вимагав значних часових витрат, і тому були внесені зміни в алгоритм та архітектуру для покращення продуктивності. Під час розробки також використовувалися навички розробки багатопоточних розподілених веб-застосунків

.

СПИСОК ЛІТЕРАТУРИ

1.Learning a Deep Convolutional Network for Image Super-Resolution [електронний_ресурс]. URL: http://personal.ie.cuhk.edu.hk /~ccloy /files/ eccv_2014_deepresolution.pdf

2. Image Super-Resolution Using Deep Convolutional Networks [електронний_ресурс]. URL: https://arxiv.org/pdf/1501.00092.pdf

3. Колесник А. В. - Розподілена программна система для розпізнавання зображень. [електронний ресурс]. – Режим доступу:

http://masters.donntu.org/2016/fknt/kolesnik/diss/index.htm

4. Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. - 2012. - С.1097-1105.

5. Krizhevsky A. ImageNet classification with deep convolutional neural .networks / A. Krizhevsky, I. Sutskever, G. E. Hinton. // Advances in Neural Information Processing Systems (NIPS). – 2012. – No25. – С. 1097–1105.

6. Фільтрація зображень різними способами. Швидке перетворення Фур'е [електронний ресурс] – Режим доступу: https://sites.google.com/site/alexeysidnev/docs/filter_fft

7. Згорткова нейронна мережа [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа.

8. Rudenko O., Bezsonov O.,Romanyk O., Lebediev O. Analysis of convergence of adaptive algorithms for the identification of non-stationary objects. Eastern-European Journal of Enterprise Technologies, 4 (97), 2019. pp. 6-14.

9. Bezsonov O., Rudenko O., Udovenko S., Dudinova O. Processing of noisy digital images with use of evolving autoencoders. Eastern-European Journal of Enterprise Technologies, 6/9 (90), 2017 . pp. 63-69.

10. Dong C., Loy C.C., Tang X. Accelerating the super-resolution convolutional neural network // European Conference on Computer Vision. Springer. 2016. P. 391–407.

11. Dong C., Loy C.C., He K. Image super-resolution using deep convolutional networks // IEEE transactions on pattern analysis and machine intelligence. 2016. №38(2). P. 295-307.

12. Generative Adversarial Networks – Hot Topic in Machine Learning. URL : http://www.kdnuggets.com/2017/01/generative-adversarialnetworks-hottopic- machine-learning.html (дата звернення 23.10.2023).

13. ImageNet – Image database for machine learning. URL : http://www.image-net.org/ (дата звернення 23.10.2023).

14. Implementation of Super Resolution CNN in Tensorflow. URL : https://github.com/tjvandal/srcnn-tensorflow (дата звернення 23.10.2023).

15. OpenCV library. URL : https://opencv.org/ (дата звернення 23.10.2023).

16. Peyman Milanfar.: Enhance! RAISR Sharp Images with Machine Learning. URL : https://research.googleblog.com/2016/11/enhance-raisr-sharpimages-withmachine.html (дата звернення 23.10.2023).

17. Super Resolution using Generative Adversarial Network (SRGAN) URL : https://github.com/tadax/srgan (дата звернення 23.10.2023).

18. The Most Popular Language For Machine Learning and Data Science Is … URL : http://www.kdnuggets.com/2019/01/most-popularlanguagemachine-learning-data-science.html (дата звернення 23.10.2023).

19. Khan S., Rahmani H., Ali Shakh S. A., Bennamoun M. A Guide to Convolutional Neural Networks for Computer Vision. Morgan &Claypool Publishers, 2018. 207 p.

20.Brownlee J. Loss and Loss Functions for Training Deep Learning Neural Networks, 2019. URL : https://machinelearningmastery.com/loss-andloss-functions-for-training-deep-learning-neural-networks/

21.Brownee J. How to Choose Loss Functions When Training Deep Learning Neural Networks. 2019. URL : https://machinelearningmastery.com/how-to-choose-loss-functions-whentraining-deep-learning-neural-networks/

22. Romano Y., Isidoro J., Milanfar P. RAISR: rapid and accurate image super resolution [Text] // IEEE Transactions on Computational Imaging. – 2017. – Т. 3. – №. 1. – С. 110-125.

ДОДАТОК А Окремі фрагменти програмного коду

ДОДАТОК Б ПРЕЗЕНТАЦІЯ