Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

Автоматизована система визначення архітектурної стилістики будівельних споруд

Виконав: студент 2 курсу, групи
2АКІТ-22м спеціальності 151 –
Автоматизація та комп'ютерно-
інтегровані технології

Дмитро ПЛЮЩЕВ
Ім'я ПРІЗВИЩЕ

Керівник:
к.т.н., доцент, доцент кафедри КСУ
степінь, звання, посада
Олена НИКИТЕНКО
Ім'я ПРІЗВИЩЕ
«01» 12 2023 р.

Опонент: к.т.н.,доцент кафедри АІІТ
степінь, звання, посада
Юрій ІВАНОВ
Ім'я ПРІЗВИЩЕ
«05» 12 2023 р.

Допущено до захисту
Т.в.о.Зав. кафедри КСУ
Марія ЮХИМЧУК
«07» 12 2023

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ

Т.в.о.Зав. кафедри КСУ

Марія ЮХИМЧУК

"09" жовтня 2023 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Плющеву Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи. Автоматизована система визначення архітектурної стилістики будівельних споруд
керівник роботи Никитенко Олена Дмитрівна
затверджені наказом ВНТУ від "18" вересня 2023 року №247
2. Термін подання студентом роботи "1 " грудня 2023 року
3. Вихідні дані до роботи: збір інформації для формування навчального датасету, оптимізація алгоритму навчання, керування базою даних, формування звітів.

4. Зміст текстової частини: огляд методів класифікації зображень, аналіз та розробка алгоритму, опис процесу навчання нейронної мережі, створення веб додатку, огляд існуючих підходів до розпізнавання об'єктів, тестування

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
вступ об'єкт, предмет та мета дослідження, задачі дослідження,, алгоритм навчання, діаграма класів, діаграма послідовностей, аналіз отриманих результатів, проведення технічного та економічного аудиту, висновки

Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
5	Бурєннікова Н. В., д.е.н., професор кафедри ЕПВМ.		
3 . 4	Ковтун В. В., д.т.н., професор, зав. кафедри КСУ		

Дата видачі завдання "09" жовтня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Огляд методів класифікації зображень	02.10.2023		
2	Аналіз алгоритму класифікації	22.10.2023		
	зображень та математична постановка			
	задачі			
3	Розробка програмного та технічного	05.11.2023		
	забезпечення		21.11.2023	
4	Тестування та дослідження ефектив-	21.11.2023		
	ності розроблених методів		24.11.2023	
5	Розрахунок економічної частини	25.11.2023		
6.	Підготовка графічних матеріалів	29.11.2023		

Студент (підпис) Дмитро ПЛЮЩЕВ (Ім'я ПРІЗВИЩЕ)

Керівник роботи (підпис) Олена НИКИТЕНКО (Ім'я ПРІЗВИЩЕ)

АНОТАЦІЯ

УДК 004.93

Плющев Д.О. Автоматизована система визначення архітектурної стилістики будівельних споруд. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2023.

На укр. мові. Бібліогр.: 55 назв; рис.: 27; табл. 11.

У ході виконання роботи було здійснено задачу класифікації зображення будівель за архітектурним стилем автоматизованим шляхом з залученням передових технологій штучного інтелекту. На меті якого стоїть знаходження міжкласових зв'язків між різними архітектурними стилями, за якими можна класифікувати споруду на відповідність до конкретного архітектурного стилю за допомогою підручних пристроїв, наприклад такі як смартфон з доступом до інтернету, тощо.

Також було розглянуто методи класифікації зображень, їх переваги та недоліки а також зроблено висновок що методи на основі глибокого навчання мають кращі результати за інші методи. Для вирішення задачі ідентифікації архітектурного стилю будівлі, було вирішено використовувати згортову нейронну мережу – нейронну мережу з попередньою обробкою зображень. Також була опрацьована стратегія, яка збільшує швидкість пошуку оптимального рішення за допомогою значення градієнта, і розглянута проблема перетренованості системи. Розглянуті основні бібліотеки для аналізу даних, як NumPy, SciPy, Sci-Kit Learn, BeautifulSoup та TensorFlow,та наведені переваги їх використання.

Розроблено веб-додаток за допомогою Angular framework і двох мікросервісів, один реалізований за допомогою Flask framework, а інший

ASP.NET Core. MsSQL Server використовується для збереження даних користувача та результатів класифікації зображень. Описано сценарій розгортання системи в Amazon Web Services.

Ключові слова: архітектурний стиль, класифікація зображень, тісні міжкласові зв'язки, архітектура згорткових нейронних мереж, штучний інтелект.

ABSTRACT

UDC 004.93

Pliushchev D.O. Automated system for determining the architectural style of building structures. Master's thesis on specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2023.

In Ukrainian speech Bibliography: 55 titles; Fig.: 27; table 11.

In the course of the work, the task of classifying the image of buildings by architectural style was carried out in an automated way with the involvement of advanced artificial intelligence technologies. The purpose of which is to find cross-class relationships between different architectural styles, according to which it is possible to classify a building according to a specific architectural style with the help of handy devices, such as a smartphone with Internet access, etc.

Image classification methods, their advantages and disadvantages were also considered, and it was concluded that methods based on deep learning have better results than other methods. To solve the problem of identifying the architectural style of the building, it was decided to use a convolutional neural network - a neural network with pre-processing of images. A strategy that increases the speed of finding the optimal solution using the gradient value was also developed, and the problem of overtraining of the system was considered. Major libraries for data analysis, such as NumPy, SciPy, Sci-Kit Learn, BeautifulSoup, and TensorFlow, are reviewed and the benefits of using them are given.

Developed a web application using Angular framework and two microservices, one implemented using Flask framework and the other ASP.NET Core. MsSQL Server is used to store user data and image classification results. The system deployment scenario in Amazon Web Services is described.

Keywords: architectural style, image classification, close interclass relations, architecture of convolutional neural networks, artificial intelligence.

Зміст

ВСТУП

В епоху цифрових технологій майже кожен має вільний доступ до пристроїв, які створюють цифрову інформацію — смартфонів, фотографій і відеокамер. Зі збільшенням їх кількості зростає їх обробка та класифікація, тобто автоматизація цих процесів. Класифікація інформації широко використовується в сферах медицини, безпеки та освіти, а також в екологічній та соціально-економічній сферах. Удосконалення методів і алгоритмів обробки інформації принесе величезну користь, і точність цих методів дуже важлива, якщо вони використовуються в таких сферах, як медицина.

Класифікація знімків, зроблених під час зйомки будівельних об’єктів, є важливою задачею цифрового обліку культурної та архітектурної спадщини. Велику кількість зображень необхідно обробити для запису, тому їх класифікація є виснажливою (що також означає схильність до помилок) і трудомістким завданням. Наявність автоматизованих методів для полегшення цих завдань класифікації покращить важливу частину процесу обробки цифрових документів. Крім того, правильна класифікація доступних зображень дозволяє краще керувати та ефективніше здійснювати пошук за конкретними термінами, тим самим допомагаючи досліджувати та інтерпретувати відповідні об’єкти спадщини.

Визначення архітектурних стилів також допомагає відвідувачам здобути нові знання, оскільки за допомогою системи можна отримати інформацію про описи архітектурних стилів та їх територіальні характеристики, завантажуючи фотографії будівель.

З іншого боку, дослідження задачі визначення архітектурних стилів будівель дозволяє знайти методи та розробити алгоритми, ефективні для завдань класифікації, де існують тісні міжкласові зв’язки, тобто коли присутня велика кількість основних ознак, а також спільні риси кількох стилів. На сьогоднішній день існує велика кількість алгоритмів, які вирішують задачі

класифікації за класами, які суттєво відрізняються один від одного, але тісно пов’язані – навпаки.

Розробка системи класифікації архітектурних стилів будівель на основі зображень є актуальною, оскільки подібної системи на ринку не знайдено.

У даній роботі розглядається використання нейронних мереж для вирішення задач класифікації зображень, а також розглядаються методи підвищення точності результатів і скорочення часу, необхідного для навчання алгоритму. Цей метод можна використовувати для вирішення будь-якої задачі класифікації зображень.

Метою дослідження є покращення алгоритмів класифікації зображень із тісними зв’язками між класами та скорочення часу навчання нейронної мережі, зберігаючи при цьому точність навчання.

Для досягнення мети необхідно виконати наступні завдання:

- Проаналізувати існуючі методи класифікації зображень;

- Модифікувати існуючі методи для підвищення точності та скорочення часу навчання;

- Створювати підібрані зображення будівель;

- програмне впровадження та розробка програмних алгоритмів;

- Порівняти ефективність модифікованого алгоритму з існуючими алгоритмами при вирішенні задачі визначення архітектурного стилю будівлі.

Об’єктом дослідження є процес навчання нейронних мереж для класифікації зображень з тісними міжкласовими зв’язками та для класифікації зображень будівель за архітектурним стилем.

Тема дослідження – оптимізаційні алгоритми та методи зміни властивостей нейронних мереж під час їх навчання, таких як ваги та швидкості навчання.

Наукова новизна отриманих результатів полягає в удосконаленні алгоритму коригування параметрів нейронної мережі під час навчання для вирішення задачі класифікації зображень з тісними зв’язками між класами.

Практичне значення отриманих результатів полягає у створенні системи ідентифікації архітектурного стилю будівель та методу навчання нейронних мереж, що забезпечує більшу точність за менший час навчання.

1. ОГЛЯД МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

Завдання класифікації зображень полягає у віднесенні об'єктів (ситуацій) до певної категорії в існуючому списку. У математичній статистиці завдання класифікації також називають завданнями дискримінантного аналізу. Існує багато способів вирішення певної задачі, кожен з яких має свої особливості та різну ефективність залежно від характеристик задачі та вихідних даних.

1.1 Аналіз існуючих рішень класифікації зображень будівель

Проаналізувавши наявні роботи, присвячені визначенню архітектурних стилів будівель, вдалося виділити наступні найбільш поширені підходи до розробки алгоритмів класифікації, які показали хороші результати:

– Локальні дискримінаційні плями або шаблони;

– Мультиноміальна латентна логістична регресія та SVM;

– Наївний байєсівський найближчий сусід (NBNN);

– мішок слів;

– Технологія глибокого навчання.

Локальні диференційовані плями або візерунки. Велика кількість існуючих алгоритмів, класифікованих за архітектурним стилем, зосереджуються на вилученні поточкових сегментів зображень - шаблонів - дискримінаційних локальних дескрипторів шаблонів для класифікації зображень [1][2][3][4][5]. Наприклад, у роботі «Visual Pattern Discovery for Architectural Image Classification and Product Image Classification» [2] реалізовано алгоритм класифікації на основі чотирьох архітектурних стилів. Дослідження категорій архітектурних стилів[4] продемонструвало ефективність методів пар слів і семантичних шаблонів, досягнувши майже ідеальних результатів на опублікованому наборі даних зображень будівель 5 архітектурних стилів.

Переваги: Цей алгоритм ефективний для класифікації зображень невеликої кількості архітектурних стилів.

Недоліки: абсолютно різні області зображення можуть мати близькі значення у функціональному просторі, що послаблює ефективність локальної корекції, для детальних архітектурних зображень такі алгоритми не можуть дати хороших результатів.

MLLR, латентна опорна векторна машина. «Мультиноміальна прихована логістична регресія» — це алгоритм, який додає приховані змінні до методу логістичної регресії. Таким чином, будуть розглянуті «менш важливі» ознаки, що оточують «вищі» ознаки, які можуть мати приховане значення для класифікатора.

Алгоритм, розроблений Z. Xu і представлений у «Класифікації архітектурних стилів з використанням мультиноміальної латентної логістичної регресії» [6], був розроблений з використанням зразків зображень будівель, класифікованих у 25 архітектурних стилях. Автор використовує ймовірність для аналізу співвідношення між архітектурними стилями, вводячи термін «ймовірність» в аналізі архітектурного стилю. Окрім точності класифікації, результати алгоритму показують зв’язки з усіма стилями та забезпечують аналіз архітектурного стилю окремих будівель. Алгоритм використовує модель деформації, яка описує частини зображення в багатомасштабному режимі на основі піраміди ознак HOG. Модель складається з трьох частин: кореневого фільтра, який показує приблизний контур об’єкта; набору детальних фільтрів для зображення з подвійною роздільною здатністю, ніж кореневий фільтр, що передає детальну інформацію про об’єкт; і витрат на деформацію, які показати відносну форму частини відносно кореня Відхилення від положення фільтра за замовчуванням.

MLLR і LSVM мають багато спільних рис - форму та напіввипуклу природу латентних змінних. Ці подібності дозволяють MLLR використовувати існуючі моделі латентних змінних, навчені LSVM, такі як

моделі на основі штамів. Однак між MLLR і LSVM є деякі значні відмінності, які роблять MLLR більш ефективним для завдань класифікації архітектурних стилів.

По-перше, SVM, як правило, неефективні при використанні незбалансованих даних навчання, Негативних прикладів більше, ніж позитивних, що неминуче при добірці образів, розділених за архітектурним стилем. Величезний клас «зображення-фон» у рамках виявлення об’єктів створює добре відомий дисбаланс між позитивними та негативними прикладами. У LSVM процес навчання повинен виправляти щонайбільше приховані значення позитивних прикладів, тоді як усі можливі приховані значення негативних прикладів мають бути збережені. По-друге, основний метод використання SVM для розв’язання багатокласових задач класифікації полягає в спрощенні багатокласової задачі у декілька бінарних задач. Оскільки кожна бінарна задача навчається незалежно, прийняття цієї стратегії є проблематичним, оскільки вона не може виявити кореляції між усіма класами. Тому вихідні значення рішень для кожної окремої задачі не можна порівнювати один з одним, що є відомою проблемою калібрування. MLLR навчає всі класи одночасно та вводить єдину цільову функцію, тому він не страждає від можливості різних упереджень через проблеми з кількома класами та незбалансовані навчальні дані. Нарешті, SVM не забезпечує хорошого імовірнісного аналізу "м'яких" меж.

Переваги: Результати класифікації враховують дрібні деталі зображення, алгоритм ефективний для великої кількості категорій.

Недоліки: погано працює при обробці будівель незвичайної форми (сучасні та новітні архітектурні стилі) або при обробці зображень з нестандартних ракурсів.

Наївний байєсівський найближчий сусід (NBNN). Реалізацію алгоритму NBNN для класифікації архітектурних стилів будівель було розглянуто в роботі «Автоматичне розпізнавання архітектурних стилів» [7]. Ідея цього алгоритму полягає в тому, щоб обчислити всі дескриптори навчальних

зображень і класифікувати їх за різними категоріями. Після цього для кожного дескриптора вхідного зображення найближчий сусід кожного класу обчислюється за допомогою наближення. Сума евклідових відстаней для кожного дескриптора представляє відстань від зображення до класу. Клас з найменшою дистанцією вибирається класом-переможцем. [7]

Переваги: Алгоритм простий у реалізації та швидко навчається, не потребує визначення додаткових параметрів, підтримує велику кількість класифікаційних класів, легко розширює кількість класів, уникає перенавчання.

Недоліки: потрібні дуже великі навчальні вибірки; експериментальні результати, наведені в статті, не демонструють високої точності; коли є кілька подібних класів - результати погані.

Тобто метод класифікації, запропонований у книзі «Архітектурно-стильова класифікація вікон фасадів будівель» [8], включає виділення фактури та форми зображення. Мета полягає в тому, щоб виділити характерні риси, такі як загострені арки, трикутники або округлі форми, мінімізуючи вплив елементів, які не мають відношення до аналізу. Під час навчання кожен знайдений елемент і його ознака кластеризується у візуальний словник – кодову книгу. Під час аналізу та тренувального процесу виявляються основні особливості, Після цього алгоритм визначає, яка книга кодів має найбільшу кількість збігів, і вибирає категорію, яка відповідає цій книзі, як переможця.

Переваги: Алгоритм дуже точно класифікує архітектурний стиль окремих структурних елементів будівлі (наприклад, куполів, веж, колон).

Недоліки: відображення від вікон, фактура будівельних матеріалів на стінах, додаткові об’єкти на зображенні суттєво впливають на результати аналізу, метод тестувався лише на окремій частині будівлі (вікнах), при порівнянні із зображеннями всієї будівлі, кількість ознак цих частин – невелика, тобто можна зробити висновок з роботи, що цей метод не дає хороших результатів для класифікації фасадів будівель у цілому.

Розглянемо технологію глибокого навчання. При розв’язанні проблем з класифікації архітектурних зображень використовувалися у таких роботах, як «Використання технології глибокого навчання для класифікації зображень архітектурної спадщини» [9] та «3D-анотація фасадів у складних сценах: практичне дослідження з використанням згорткових нейронних мереж і структур руху» [10]. Використовувалися у цьому випадку нейронна мережа та методи глибокого навчання. Існує велика кількість літератури, яка описує різні методи глибокого навчання для класифікації зображень, як загальні, так і специфічні, такі як медичні зображення, аерофотознімки, номерні знаки та транспортні засоби, класифікація мікроорганізмів, міське середовище тощо. Існує також деяка література про класифікацію зображень за архітектурним стилем з використанням таких методів, як пошук за зразком, фільтри Габора та опорні векторні машини, алгоритми комп’ютерного бачення, кластеризація та вивчення локальних особливостей або мультиноміальна прихована логістична регресія [6]; але на додаток до [9, 10] немає посилань із використанням глибокого навчання для класифікації зображень архітектурної спадщини.

У роботі [9] розглянуто класифікацію будівель за окремими архітектурними елементами (колони, дзвіниці, куполи тощо). Модель будується шляхом введення набору навчальних даних, категорії яких попередньо позначені для алгоритму, з якого можна навчатися. Модель використовується шляхом введення іншого набору вхідних даних, ніж той, який раніше використовувався для навчання, що дозволяє моделі прогнозувати категорії на основі даних, на яких вона навчалася. Для реалізації алгоритму класифікації були використані зразки зображень, що належать до десяти категорій. Жодні критерії не використовувалися для вибору цих категорій; це було просто створення початкової основи для подальшого розширення кількості категорій дослідження, щоб включити ті, які вважаються більш корисними для цього типу завдань.

Дослідження «3D Facade Labeling in Complex Scenes: A Case Study Using Convolutional Neural Networks and Structures from Motion» [10] фокусується на ідентифікації простих елементів фасаду, таких як двері, вікна, балкони та дахи. Складність цього завдання полягає в нескінченних варіаціях форми, складу матеріалу та непередбачуваних можливостях прикусу. Повний підхід цього методу складається з трьох етапів: контрольована модель CNN для семантичної сегментації; отримання геометрії сцени (3D-реконструкція) через конвеєр SfM, MVS, пост-обробку та 3D-маркери з використанням аналізу трасування променів. Створена авторами мережа складається з кодера з мережею VGG16 і декодера з архітектурою повністю згорткової мережі (FCN). Кодер із топологією згорткового шару VGG16 спочатку складається з 13 згорткових шарів, за якими йдуть відповідні шари об'єднання та повністю з'єднані (FC) шари. В алгоритмі FC кодер замінюється згортковим шаром 1 × 1, який приймає вихід останнього шару об'єднання (називається pool5) розміром 39 × 12 × 512 і генерує зображення з низькою роздільною здатністю розміром 39 × 12. розкол. Ця зміна робить мережу меншою та легшою для вивчення. Після вищевказаних операцій декодер FCN приймає матрицю 39 × 12 як вхідні дані для своїх 3 згорткових шарів і, нарешті, виконує операцію підвищення дискретизації для виконання прогнозування пікселів.

Переваги: Методи на основі глибокого навчання значно перевершують результати, досягнуті всіма іншими існуючими методами.

Недоліки: елементи не класифіковані спеціально за архітектурним стилем.

Тому, проаналізувавши існуючі рішення для класифікації архітектурних зображень, можна зробити висновок, що методи на основі глибокого навчання мають кращі результати, ніж інші існуючі методи. [11].

1.2 Дослідження класифікації за допомогою глибинного навчання

Роки досліджень у галузі маркування (класифікації) зображень привели до сьогоднішнього «золотого» стандарту для вирішення проблем сегментації та класифікації – використання глибокого навчання (DL). Глибоке навчання — це гілка машинного навчання, заснована на наборі алгоритмів, які моделюють високорівневі абстракції даних, використовуючи різні архітектури з багатьма нелінійними перетвореннями. Глибоке навчання може бути під наглядом або без нагляду.

За останні кілька років глибокі згорткові нейронні мережі та інші подібні варіанти (наприклад, залишкові мережі) стали однією з найпопулярніших архітектур для класифікації зображень. Поле комп'ютерного зору має велике значення для швидкого та масштабованого навчання, яке може забезпечити відмінні результати в розпізнаванні об'єктів, виявленні об'єктів, семантичній сегментації, розпізнаванні дій об'єктів, відстеженні об'єктів і багатьох інших завданнях. Ці моделі досягли великого успіху в сучасній графіці, розпізнаванні мови, виявленні об’єктів, облич тощо.

Глибоке навчання — це галузь машинного навчання, яка дозволяє обчислювальним моделям із кількома етапами обробки вивчати представлення на кількох рівнях абстракції. Функції верхнього рівня походять від функцій нижчого рівня для формування ієрархічного представлення, тому глибоке навчання також називають ієрархічним. Термін «глибина» відноситься до кількості шарів (стадій) обробки. Коротше кажучи, методи глибокого навчання можуть ідентифікувати структури у великих обсягах даних, використовуючи, наприклад, контрольоване навчання з концепцією зворотного поширення. [12]

У 1943 році з'явилася перша штучна нейронна мережа - ШНМ. За допомогою лише кількох підключень автори змогли продемонструвати, як комп’ютери можуть імітувати процес навчання людини. 1968 Hubel, D.H. Wiesel, T.N. Запропонуйте пояснення того, як ссавці візуально сприймають

світ за допомогою ієрархічної нейронної структури, подібної до мозку. Потім, у 1989 році, нейронна модель почала привертати увагу не лише завдяки своїм результатам, але й через свою подібність до біологічних візуальних систем, а також через її модулі обробки та сприйняття. LeCun, Y. та ін. Запропоновано складну нейронну модель для розпізнавання рукописних символів – Convolutional Neural Network (CNN) – саме через безперервні математичні операції згорток на зображеннях. Відтоді багато інженерів надихалися розробкою подібних алгоритмів розпізнавання образів у системі комп'ютерного зору.

1.3 Дослідження згорткових нейронних мереж (CNN)

Нейронна мережа — це алгоритм машинного навчання, побудований на організаційних і функціональних принципах біологічних нейронних мереж. Концепція виникла в 1943 році, коли Уоррен МакКаллох і Уолтер Піттс спробували змоделювати процеси, що відбуваються в мозку.

Нейронні мережі складаються з окремих одиниць, які називаються нейронами. Нейрони розташовані в багатьох групах-шарах. Нейрони кожного шару з'єднані з нейронами наступного шару. Дані проходять уздовж цих шарів від вхідного рівня до вихідного. Принципова схема нейронної мережі представлена на рисунку 1.1 .

Кожен окремий вузол виконує прості математичні обчислення. Потім він передає дані на всі вузли, до яких підключений.

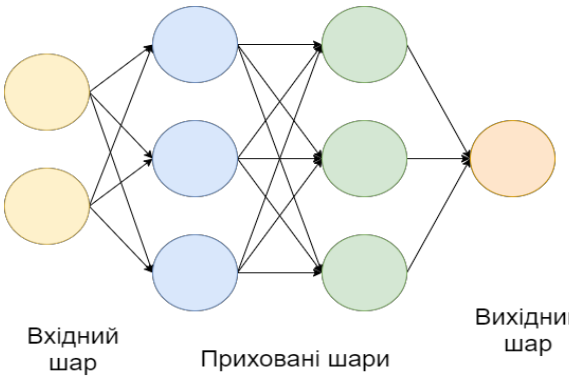


Рисунок 1.1 – Схема простої нейронної мережі

У міру зростання обчислювальної потужності та збільшення досвіду роботи з нейронними мережами з’явилося глибоке навчання, а структури нейронних мереж стали складнішими та здатними вирішувати більш широкий спектр завдань, які раніше неможливо було ефективно вирішити за допомогою нейронних мереж. Яскравим прикладом є класифікація зображень.

Згорточна нейронна мережа (CNN) — це особлива архітектура нейронної мережі, запропонована Яном Лекуном у 1988 році. CNN використовує деякі особливості та механізми зорової кори ока, що робить його найефективнішим серед інших нейромережевих архітектур для обробки зображень. Наприклад, Facebook використовує CNN для алгоритмів автоматичного позначення зображень, Amazon використовує його для створення рекомендацій щодо продуктів, а Google використовує його для пошуку фотографій користувачів.

Основним завданням класифікації зображень є прийняття вхідного зображення та визначення його категорії. Це навичка, якою люди володіють з народження, що дозволяє їм легко розпізнати, що на зображенні зображений кіт. Але комп’ютери бачать зображення зовсім по-іншому. На рисунку 1.2 показано різницю між тим, як око сприймає зображення, і тим, як його бачить комп’ютер: замість зображення комп’ютер бачить масив пікселів.

Наприклад, якщо розмір зображення 300 x 300 пікселів, розмір масиву буде 300x300x3, де 300 — ширина, наступні 300 — висота, 3 - Значення каналу RGB. Кожному номеру комп’ютера присвоюється значення від 0 до 255. Це значення описує піксельну інтенсивність кожної точки.

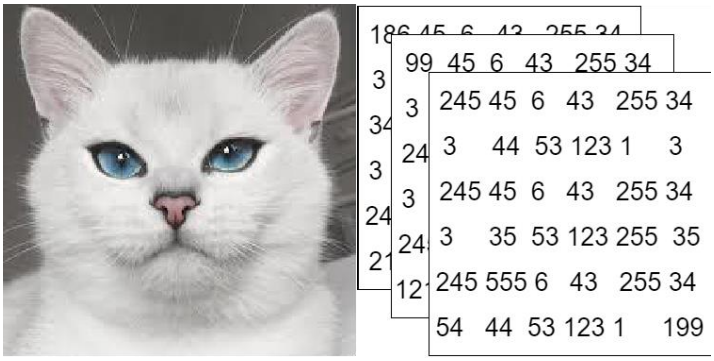


Рисунок 1.2 – Представлення рисунку у пам’яті комп’ютера

Щоб вирішити цю проблему, комп’ютери шукають функції базового рівня. У людському розумінні цими ознаками є, наприклад, очі або вуха. Для комп’ютера ці особливості є межами або викривленнями. Потім за допомогою груп згорткових шарів комп’ютер будує більш абстрактні концепції. Зображення проходить через серію згорткових шарів, нелінійних шарів, шарів об’єднання та повністю зв’язаних шарів перед тим, як створити вихідні дані.

Згортковий шар. Спочатку завжди є згортковий шар. Вхідне зображення (матриця зі значеннями пікселів). Уявімо, що зчитування вхідної матриці починається з лівого верхнього кута зображення. Далі програмне забезпечення вибирає меншу матрицю під назвою фільтри (або нейрони чи ядра). Потім фільтр згортається, тобто переміщується вздовж вхідного зображення. Завдання фільтра полягає в тому, щоб помножити його значення на початкове значення пікселя. Усі ці множення додаються. Результатом є число. Оскільки фільтр починає обробляти зображення у верхньому лівому куті, він переміщується на 1 одиницю праворуч і робить щось подібне. Після фільтрації на всіх позиціях виходить нова матриця, яка менша за вхідну, як показано на малюнку 1.3. Як правило, мережа складається з кількох згорткових шарів, змішаних з нелінійними шарами та шарами об’єднання. Коли зображення проходить через згортковий шар, вихід першого шару стає входом другого шару. Ця операція повторюється для кожного згорткового шару мережі.

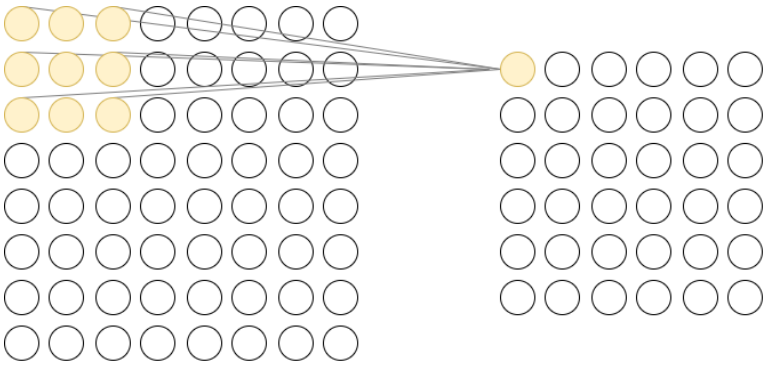


Рисунок 1.3 – Процес згортки

Ця операція схожа на те, як людське око визначає межі та прості кольори зображення. Але для того, щоб розпізнати атрибути вищого рівня, такі як очі чи вуха, необхідні всі рівні мережі.

Нелінійний шар. Після кожної операції згортки додається нелінійний шар. Він має функцію активації, яка забезпечує нелінійні характеристики. Без цієї властивості мережа недостатньо «розумна» і не може моделювати відповідь (мітка класу).

Шар об'єднання. За нелінійним шаром слідує шар об’єднання. Він обробляє ширину та висоту зображення та виконує над ними операції зменшення роздільної здатності. В результаті розмір зображення зменшується. Це означає, що якщо певні особливості (наприклад, межі) були визначені в попередніх операціях згортки, детальне зображення більше не потрібне для подальшої обробки та стискається в менш детальне зображення.

Ці шари дуже прості - їх завдання полягає в тому, щоб розділити зображення на області, виконати певні операції над кожною областю, а їх виходом є елементи результуючого зображення. Декілька варіантів операцій можна виконувати з регіонами, найпопулярнішими є середнє, глобальне та максимальне об’єднання. Розглянемо кожен більш детально.

Average Pooling — це операція об’єднання, яка обчислює середнє значення площ пікселів у кожній частині зображення.

Max Pooling — це операція об’єднання, під час якої вибирається максимальне значення площі пікселів у кожній частині зображення.

Вони використовуються для підкреслення наявності певних особливостей місцевості. На практиці максимальне об’єднання дає кращі результати, ніж середнє об’єднання. Причина полягає в тому, що об’єкти часто кодують просторову присутність певних шаблонів або концепцій на різних ділянках графіка об’єктів (у кількох вимірах матриці зображення), і розгляд максимальної присутності різних об’єктів є більш інформативним, ніж їх середня присутність.

Глобальне об’єднання — це операція об’єднання, під час якої обчислення виконується не для окремої області, а для всього зображення. Цю операцію можна використовувати, коли є потреба «активно» підкреслити наявність певних особливостей зображення, наприклад вертикальної лінії, що проходить через усе зображення. Іноді замість повністю підключеного рівня використовується глобальний рівень об’єднання, а результати передаються безпосередньо перед виведенням моделі.

Повністю зв'язаний шар. Після проходження згорткового шару, нелінійного шару та уніфікованого шару зображення потрапляє на повністю зв’язаний шар. Цей рівень приймає вхідні дані від згорткової мережі. Повністю зв’язані шари зводять зображення до N-вимірного вектора, де N — кількість категорій, з яких модель вибирає потрібну категорію. Операція класифікації виконується на повністю зв’язаному шарі. [13].

1.4 Дослідження архітектур згорткових нейронних мереж

В останні роки з’явилося багато різних архітектур згорткових нейронних мереж. Збільшення обчислювальної потужності дозволило розробити глибокі моделі до точки, коли їх стало неможливо повністю візуалізувати, і більшість людей розглядають їх як «чорні ящики». Однак між цими архітектурами є відмінності, і їх необхідно розуміти, щоб вирішити дану проблему. Розглянемо найвідомішу архітектуру CNN.

Мережа VGG. Цей набір архітектур вважається найпростішим серед інших, але для деяких завдань його точність перевищує найскладніші архітектури. Існує 6 архітектур VGGNet, найпопулярнішими є VGG-16 і VGG-19. Ідея VGG полягає в тому, щоб збільшити розмір фільтра в кожному наступному згортковому шарі, тобто якщо шар 1 має 16 фільтрів, то шар 2 повинен мати 16 або більше фільтрів. Ще одна особливість полягає в тому, що всі фільтри у VGG 3x3. Ідея полягає в тому, що два фільтри 3x3 мають майже

таке ж покриття, як і фільтр 5x5, але «дешевші» з точки зору загальної кількості операцій множення.

Недоліком цієї архітектури є те, що вона не підходить для глибоких мереж, тому що чим глибша мережа, тим більша ймовірність того, що градієнт зникне. Створення глибокого VGG вимагає більше підготовки та параметрів. Проте мережі VGG ***добре підходять*** для перенесення навчання та невеликих завдань класифікації. [14]

Залишкова мережа. Залишкові мережі були першими дійсно глибокими мережами - 152 шари показали хороші результати. До цієї архітектури навчання глибоких мереж не давало задовільних результатів. Однією з особливостей ResNet, яка використовується для навчання такої глибокої (152-рівневої) мережі, є наявність залишкових з'єднань. У VGG кожен шар пов'язаний зі своїм попереднім шаром і отримує від нього вхідні дані.

При поширенні з одного шару на інший найбільш корисні властивості передаються, але менш важливі властивості втрачаються. Таким чином, останній рівень не отримає інформацію, отриману першим шаром. ResNet вирішує цю проблему, не лише з'єднуючи попередній шар із поточним, але й з'єднуючи шари перед попереднім, як показано на малюнку 1.4. Тому один шар може не тільки бачити спостереження попереднього шару.

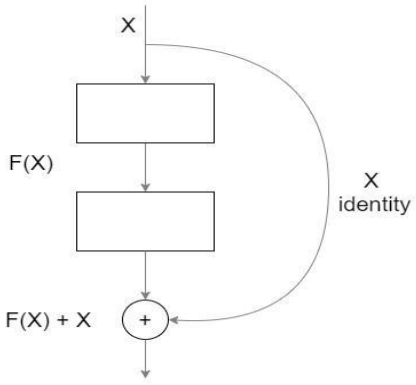


Рисунок 1.4 – Підключення декількох шарів в архітектурі ResNet

Такі глибокі залишкові мережі можна навчити, використовуючи шар нормалізації після кожного шару згортки. Рівень нормалізації покращує значення ваги і, отже, досягає вищої точності під час навчання, що прискорює навчання та мінімізує проблему зникнення градієнтів.

Існує багато варіацій ResNet, але основна ідея полягає у використанні виходу згорткового шару Conv2D як вхідного параметра x. Що потрібно зробити, так це застосувати групу згорткових шарів Conv2D до вхідного параметра x, а потім додати вихід до x і надіслати його на вхід наступного шару.

Перевага ResNet полягає в тому, що ми можемо навчати глибші мережі за допомогою цієї архітектури. [15]

Густа мережа. У ResNet вихідні дані групи згорткових шарів додаються до вхідного рівня, тоді як у DenseNet для кожного поточного шару всі інші шари перед ним підключаються до поточного шару. Завдяки такому з'єднанню шарів можна зменшити кількість фільтрів, тим самим мінімізуючи проблему зникнення градієнта, оскільки всі шари безпосередньо пов'язані з вихідними даними, а градієнти можна обчислити безпосередньо з вихідних даних кожного шару.

У порівнянні з ResNet, DenseNet має більше проміжних з'єднань. Крім того, це дозволяє використовувати менше фільтрів у щільних шарах, що є великою перевагою для невеликих моделей [16].

Ідея Inception Net полягає в створенні ширшої мережі. Це можна зробити шляхом розпаралелювання кількох шарів різними фільтрами, а потім об'єднання всіх цих паралельних шарів для переходу до наступного шару, як показано на малюнку 1.5.

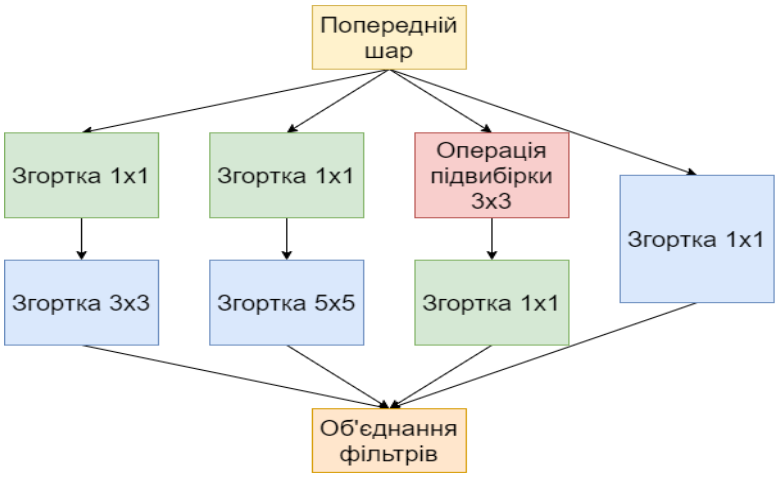


Рисунок 1.5 – Модуль архітектури Inception для зменшення розмірності

Існує багато варіацій Inception Net, основні відмінності яких:

– Використовування фільтру 3x3 замість 5x5 для підвищення ефективності обчислень (наприклад, VGGNet);

– Використовування згорткового шару 1x1 Conv2D із меншою кількістю фільтрів перед виконанням будь-якого згорткового шару Conv2D із більшим розміром фільтра, оскільки фільтри 1x1 зменшують глибину вхідних даних і підвищують ефективність обчислень;

– Використовування фільтру 1x3 замість фільтра 3x3, потім використання фільтру 3x1. Це також значно підвищить ефективність обчислень.

Inception Net містить майже всі переваги розглянутої вище архітектури та багато іншого, оскільки ця мережа є не тільки глибшою, але й ширшою порівняно з іншими архітектурами, а кількість параметрів навчання стає меншою зі збільшенням кількості рівнів. [17]

Xception Net є вдосконаленням InceptionNet з точки зору обчислювальної ефективності. Різниця полягає в тому, що Inception Net виконує звичайні операції згортання, тоді як Xception Net виконує поглиблені операції згортання.

Згортка з розділенням по глибині відрізняється від звичайної згортки тим, що в звичайному шарі Conv2D для зображення розміром 32x32x3 ми

можемо використовувати будь-яку кількість фільтрів на рівні згортки, і кожен фільтр використовуватиметься у всіх трьох роботах на каналі, а результат буде сума всіх відповідних значень. Але в згортці з роздільною глибиною існує лише одне ядро на канал для виконання згортки. Таким чином, виконуючи згортку з розділеними глибинами, ми можемо зменшити обчислювальну складність, оскільки кожне ядро є лише двовимірним і виконує згортку лише на одному каналі.

Xception Net є гарною альтернативою для малопотужних пристроїв, оскільки згорткові шари є менш обчислювально інтенсивними, але точність дуже подібна до традиційних згорткових шарів [18].

1.5 Алгоритми зміни атрибутів нейронної мережі

Під час навчання нейронної мережі використовується ітераційний процес для коригування вагових і порогових функцій — зміна значень вагових коефіцієнтів, зміщень і швидкості навчання для мінімізації функції втрат.

Існуючі методи можна використовувати для знаходження мінімального значення функції, і їх переваги та недоліки обговорюються нижче.

Градієнтний спуск — це алгоритм оптимізації, який використовується в навчанні нейронної мережі для знаходження локального мінімуму диференційованої функції. Він заснований на ітераційному налаштуванні параметрів опуклої функції для знаходження її локального мінімуму. Цей алгоритм популярний через свою обчислювальну простоту та легкість реалізації, але він також має очевидні недоліки: дуже ймовірно, що він зупиниться на локальному мінімумі; ваги мережі оновлюються після обчислення градієнта всього набору вхідних даних, тому він важко знайти для великих обсягів даних. Мінімум може зайняти багато часу; обчислення потребує багато пам’яті для обчислення градієнта.

Алгоритм стохастичного градієнтного спуску усуває деякі недоліки алгоритму градієнтного спуску. У цьому алгоритмі похідна кожної точки

обчислюється окремо, на відміну від градієнтного спуску, який вимагає обчислення всіх точок одночасно. Однак це створює іншу проблему - час обчислення 1 епохи навчання довший, ніж градієнтний спуск.

Існує ще одна модифікація - стохастичний градієнтний спуск з імпульсом, - який має всі переваги алгоритму стохастичного градієнтного спуску і сходиться швидше, ніж алгоритм градієнтного спуску. Однак на кожній ітерації для кожного оновлення потрібно обчислювати змінну. Алгоритм стохастичного градієнтного спуску можна використовувати лише в неглибоких нейронних мережах.

Розглянуті вище алгоритми не впливають на такі параметри, як швидкість навчання нейронної мережі, тому ми розглянемо AdaGrad - алгоритм, який використовує адаптивні рівні навчання для кожної ваги. AdaGrad регулює загальну швидкість вивчення кожного параметра на кожному кроці часу на основі попередніх значень градієнта. Головна перевага полягає в тому, що немає необхідності вручну встановлювати швидкість навчання, оскільки вона змінюється адаптивно з ітераціями. Основним недоліком є накопичення квадратів значень градієнта, сума яких зростає під час тренування, внаслідок чого швидкість тренування постійно знижується.

Проблему уповільнення навчання вирішує алгоритм AdaDelta - замість того, щоб зберігати суму попередніх квадратів градієнтів, вона рекурсивно обчислюється як середнє значення всіх минулих квадратів градієнтів. Існує алгоритм, який точно такий же, як AdaDelta - це RMSprop, і вони були розроблені приблизно в той же час, щоб усунути недоліки AdaGrad.

Найпопулярнішим алгоритмом є Адам — поєднання RMSprop і стохастичного градієнтного спуску з імпульсом. Він поєднує в собі переваги обох алгоритмів з найнижчою вартістю навчання. Однак є недолік - якщо градієнт певної ваги великий (тобто значення функції втрат сильно змінюється), Тоді алгоритм не такий ефективний, як стохастичний градієнтний спуск. [19].

1.6 Постановка задачі

Проблема класифікації зображень є дуже актуальною в наш час. Обсяг даних продовжує зростати, а технологія цифрової фотографії дозволила оснастити камерою кожен мобільний телефон. Для великої кількості цифрових зображень виникає проблема автоматизованого аналізу.

Розробка спрямована на прогнозування архітектурного стилю будівлі на основі зображень різного формату та якості.

Метою розробки є спрощення процесу визначення архітектурного стилю будівлі та зменшення похибок у визначенні стилю за рахунок аналізу отриманих зображень.

Завдяки розробленим алгоритмам та створеній системі визначення архітектурного стилю будівель, облік архітектурної спадщини можна автоматизувати, що скоротить час, необхідний для запису, та підвищить достовірність результатів порівняно з ручним визначенням архітектурного стилю. Вивчаючи еволюцію будівельних конструкцій, це має сприяти збереженню, підтримці та відновленню будівель.

Крім того, завдання класифікації за архітектурним стилем має певні характеристики, які дозволять вивчати класифікацію образів за наявності тісних міжкласових зв’язків.

Тому необхідно створити алгоритмічне забезпечення, яке зможе ефективно класифікувати зображення будівель. Алгоритм повинен враховувати наявність тісних міжкласових зв'язків і мати мінімальну похибку.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- аналізувати методи класифікації зображень;

- Модифікувати існуючі методи для підвищення точності нейронних мереж і скорочення часу навчання;

- Створення серії зображень будівель, розділених на різні архітектурні стилі;

- Програмно реалізовувати алгоритми та розробляти системи аналізу архітектурного стилю;

- Порівняти ефективність модифікованого алгоритму з існуючими алгоритмами при вирішенні задачі визначення архітектурного стилю за зображеннями будівель.

Загалом, класифікація зображень — це базове завдання, яке намагається зрозуміти й проаналізувати зображення в цілому. Мета — проаналізувати зображення та призначити йому певну мітку.

Проаналізовано та порівняно існуючі алгоритми вирішення задачі ідентифікації архітектурного стилю будівлі та описано їх характеристики. Однак досі не виявлено системи, яка б могла визначити архітектурний стиль. Останніми роками з’явилося багато різних архітектур згорткових нейронних мереж. Ці архітектури є найпопулярнішими в даний час, але не було знайдено жодного дослідження, яке б спеціально класифікувало архітектурні стилі.

Завдання класифікації архітектурних стилів будівель відрізняється від стандартних завдань класифікації тим, що між різними стилями існують міжкласові зв’язки, тобто велика кількість ознак і характеристик є загальними для стилів.

Вивчаючи літературу, що відповідає класифікації зображень будівель за архітектурним стилем, Знайшов багато дуже хороших рішень проблеми. Наприклад, алгоритм, розроблений Z. Xu, представлений у «Класифікації архітектурного стилю з використанням мультиноміальної латентної логістичної регресії» [6], враховує навіть дрібні деталі зображення, які впливають на результати класифікації. Алгоритм здатний обробляти велику

кількість категорій, оскільки він був розроблений з використанням високоякісної добірки зображень будівель у 25 архітектурних стилях.

Проаналізувавши рішення для класифікації архітектурних зображень «Architectural Heritage Image Classification Using Deep Learning Techniques» [9] і «3D Facade Labeling of Complex Scenes: A Case Study Using Convolutional Neural Networks and Structures from Motion» [10], ми можемо зробити висновок, що глибокі методи навчання значно перевершують результати, досягнуті всіма іншими методами.

Однак існуючі методи, описані в літературі, або взагалі не розглядають завдання класифікації за архітектурним стилем і, отже, не враховують специфіку цього завдання, або алгоритми непридатні для детальних архітектурних зображень. Розглянуті існуючі методи зміни властивостей нейронних мереж мають той недолік, що вони є вирішальними для вирішення задач класифікації зображень, оскільки проблема вимагає ефективних обчислень і часу навчання, оскільки зображення є «складними» даними.

Отже, зазначені недоліки є суттєвими та унеможливлюють ефективне застосування існуючих методів вирішення проблеми класифікації будівельних образів за архітектурними стилями. Однак ідеї, описані в дослідницькій літературі, є корисними та можуть бути застосовані для розробки алгоритмів для вирішення заданої проблеми.

2. РОЗРОБКА АЛГОРИТМУ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

2.1 Опис загальних понять для розв’язання задачі

Для вирішення задачі ідентифікації архітектурного стилю будівлі, тобто класифікації зображень, будемо використовувати згортову нейронну мережу – нейронну мережу з попередньою обробкою зображень.

Згорточна нейронна мережа — це архітектура нейронної мережі, яка працює подібно до механізму зорової кори ока, що робить цю архітектуру найбільш ефективною для обробки зображень. Згорткові нейронні мережі імітують роботу людського мозку — ця архітектура нейронної мережі працює так само, як мозок сприймає вхідні дані, обробляє їх і створює певні результати. Прийом вхідних даних, обробка отриманої інформації та генерація вихідних даних представлені у вигляді шарів нейронної мережі (вхідний рівень, прихований шар і вихідний рівень). Таким чином, зображення пропускається через згорткові шари, нелінійні шари, рівномірні шари та повнозв’язані шари, на виході яких ми отримуємо вектор із призначеного для навчання класу.

Зображення в пам'яті комп'ютера. Зображення, представлене в пам'яті комп’ютер як матриця n×n×3, де 3 відповідає трьом кольоровим каналам - червоний, синій, зелений, матриця, що містить значення від 0 до 255 - Кількість певного кольору в певному пікселі (RGB). Цей метод дуже ефективний для зберігання інформації про зображення (256 значень точно вміщуються в один байт) і достатній для потреб людського ока, яке може розрізняти обмежену кількість відтінків одного кольору.

Обробка зображення. Алгоритм CNN складається з процесів згортки та об’єднання, представлених у вигляді шарів нейронної мережі. Після проходження матриці зображення через шари згортки та об’єднання розмір зображення достатньо зменшено та готове для подальшого аналізу. Ця матриця переходить у «повністю зв’язаний шар», де матриця зображення

проектується на вектор вихідного шару. Така структура мережі дає змогу аналізувати зображення різного розміру та якості. [20]

Повний процес навчання нейронної мережі складається з двох етапів: прямого поширення та зворотного поширення.

Пряме поширення. Зображення подаються на вхідний рівень мережі як числові матриці, кожна матриця є інтенсивністю пікселя в зображенні. Нейрони мережі в прихованому шарі застосовують до цих значень певні математичні операції (докладний опис операцій наведено нижче). Для виконання математичних операцій певні значення параметрів необхідно випадково встановити на початку навчання. Результати виконання передаються на вихідний рівень, який генерує остаточні результати класифікації.

Зворотне поширення. Після отримання результату його необхідно порівняти з правильним значенням, щоб визначити, наскільки він близький до фактичного значення, тобто розраховується помилка, або втрата. На основі значення помилки параметри мережі оновлюються. Процес прямого поширення повторюється з використанням нових значень параметрів і генерування нових результатів, поки помилка не буде мінімізована або не буде досягнуто задану кількість ітерацій.

Гіперпараметри нейронних мереж. Важливим етапом формування нейронної мережі є вибір параметрів обраної архітектури: кількості шарів, кількості нейронів на шар, функції активації, значення швидкості навчання та кількості епох навчання. Функція активації є одним із ключових елементів нейронних мереж: без неї нейронна мережа стає комбінацією лінійних функцій. Нелінійні елементи забезпечують гнучкість створення складних функцій під час навчання. Функція активації істотно впливає на швидкість навчання, що є одним з основних критеріїв її вибору[21].

2.2 Опис процесу виділення кордонів та меж

Операція згортання подібна до процесу, за допомогою якого люди виділяють краї та межі об’єктів на зображенні. Завдання згорткових шарів — зменшити зображення для аналізу без втрати важливої інформації.

2.2.1 Згортка зображення

Для згортки зображення виберіть матрицю, яка є частиною загальної матриці - фільтра або ядра. З цієї матриці згортка виходить шляхом множення значення фільтра на початкове значення пікселя. Значення згортки Z обчислюється за формулою (2.1), де X – вхідне зображення, f – фільтр, m і n – індекси рядків і стовпців результуючої матриці:

$$Z_{m,n} = (X * f)_{m,n} = \sum_j \sum_k f_{j,k} X_{m-j,n-k}. \quad (2.1)$$

Математично згортку часто позначають зірочкою *: Z = X * f.

Цей фільтр застосовується до різних частин зображення, на кожному елементі виконується множення, його значення складаються, і в результаті на виході виходить число. Подібна операція виконується шляхом повторення цієї операції для кожного фільтра, отриманого переміщенням його вбік або вниз на s одиниць (один крок). Після фільтрації по всіх позиціях виходить нова, менша матриця. Оскільки зображення зменшується щоразу, коли виконується згортка, її можна виконувати лише обмежену кількість разів, перш ніж зображення повністю зникне. Щонайбільше, коли фільтр рухається по зображенню, пікселі, розташовані по краях зображення, мають набагато менший вплив, ніж пікселі в центрі зображення. Щоб виправити цю проблему, додаємо відступ p, значення якого розраховується згідно з рівнянням (2.2), де f — розмір фільтра:

$$p = \frac{f-1}{2}. \quad (2.2)$$

Враховуючи зсув p і розмір кроку s вхідної матриці розмірності f, розмірність вихідної матриці p можна обчислити за формулою (2.3):

$$n_{out} = \left\lfloor \frac{n_{in}+2p-\mathrm{f}}{s} + 1 \right\rfloor. \quad (2.3)$$

Існує кілька обмежень під час роботи з кольоровими зображеннями: фільтр і зображення, до якого застосовано фільтр, повинні мати однакову кількість каналів; щоб використовувати кілька фільтрів на одному зображенні, кожен фільтр потрібно згортати окремо, а результати додавати до кожного інше і злито в одне. Розмірність вихідної 3D матриці розраховується за формулою (2.4):

$$[n, n, n_c] * [\mathrm{f}, \mathrm{f}, \mathrm{f}_c] = \left[\left\lfloor \frac{n+2p-\mathrm{f}}{s} + 1 \right\rfloor, \left\lfloor \frac{n+2p-\mathrm{f}}{s} + 1 \right\rfloor, n_{\mathrm{f}} \right], \quad (2.4)$$

де n — розмір зображення, f — розмір фільтра, nc — кількість каналів у зображенні, p — зсув, s — розмір кроку, nf — кількість фільтрів. [двадцять один].

2.2.2 Шари згортки у нейронній мережі

Згорткові шари виконують такі операції над зображеннями:

- Для поточного шару згортки обчисліть значення згортки вхідного зображення, використовуючи результат згортки попереднього шару та додайте зсув.

-Застосувати нелінійну функцію активації до проміжних значень.

Така поведінка забезпечує нелінійність і покращує результати навчання.

2.2.3 Зворотнє поширення у шарах згортки

Мета зворотного розповсюдження в згортковому шарі така ж, як і в повністю пов’язаному шарі – обчислити похідні, які будуть використовуватися для оновлення значень параметрів у згортковому шарі. Процес зворотного поширення подібний до описаного в 2.4.

2.3 Процес зменшення зображення

Щоб забезпечити ефективну обробку алгоритмів великих зображень, ми додамо шари, які ще більше зменшать розмір зображення. Оскільки основні характеристики зображення визначаються після операції згортання, зображення з великими деталями більше не потрібні для подальшої обробки.

Об'єднання шарів дозволить змінити розмір піксельної матриці - зменшивши розміри стовпців і рядків.

Ця операція дозволить навчати нейронні мережі на великих зразках зображень, не впливаючи на швидкість навчання.

2.3.1 Шари об’єднання

Щоб зменшити зображення, ми використаємо метод максимального об’єднання – розділимо матрицю на частини Xk,o, у кожній частині виберіть максимальне значення та розмістіть його у відповідній позиції вихідної матриці Pi,j. Як і у випадку згорткових шарів, використовується фільтр розміру f2. При комбінуванні багатоканальних зображень кожен канал комбінується окремо. [22]

$$P_{i,j} = \max X_{k,o},$$
$$k = \overline{\imath \ldots \imath + \mathrm{f}_2}, \quad \mathrm{o} = \overline{\jmath \ldots \jmath + \mathrm{f}_2}.$$

2.3.2 Зворотнє поширення у шарах об'єднання

У об’єднаному шарі немає параметрів, які потрібно оновлювати, тому просто розподіліть градієнт належним чином. У прямому проході максимальне значення кожної області вибирається для максимального об’єднання та передається на наступний рівень. Таким чином, під час зворотного поширення градієнти не повинні впливати на елементи матриці, які не були включені до прямого поширення. На практиці це досягається

шляхом створення маски, яка запам'ятовує положення значень, використаних на першому кроці, які пізніше можна використовувати для передачі градієнтів.

2.4 Опис процесу навчання нейронної мережі

Навчання нейронної мережі передбачає використання ітераційного процесу для налаштування параметрів нейронної мережі – вагових коефіцієнтів і швидкості навчання – для мінімізації функції втрат. Щоб знайти мінімум функції, ви можете використовувати різні методи, описані в пункті 1.5, але всі вони мають деякі недоліки, які є вирішальними для вирішення задач класифікації зображень.

На кожній ітерації t потрібно розрахувати градієнт функції втрат, який покаже напрямок, у якому функція втрат grad(L) зростає найшвидше, а ваги мережі W перераховуються з урахуванням швидкості навчання a .

$$g_t = grad(L), \tag{2.5}$$

$$W_t = W_{t-1} - \alpha g_t, \tag{2.6}$$

де a — швидкість навчання, a grad — градієнт функції.

Вибір швидкості навчання a має вирішальне значення - якщо вона встановлена занадто низько, нейронна мережа навчатиметься повільно, а якщо вона встановлена занадто високо, мінімальне значення може не бути досягнуто. Щоб визначити швидкість навчання, необхідно провести масштабні експерименти з даними, які використовуються для навчання мережі, на що можуть піти роки, тому рекомендується використовувати адаптивну швидкість навчання для кожної ітерації.

Змінюючи вагові коефіцієнти відповідно до рівняння (2.6), ми знайдемо локальний мінімум функції втрат, але ми не маємо способу контролювати зміну вагових коефіцієнтів – беручи більше значення «кроку», щоб пришвидшити навчання, коли градієнт робить не сильно змінювати, Або навпаки, менше значення, щоб не «пропустити» мінімальне значення. Для

цього при розрахунку градієнта обчислюємо такі значення ковзного середнього градієнта і його квадрата:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad (2.7),$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \quad (2.8).$$

де beta1 і beta□2 – параметри, які контролюють вплив минулих значень на зміни ваги.

Змінюючи ваги мережі, ми контролюємо «розмір кроку», помноживши зміну на швидкість навчання a:

$$W_t = W_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \varepsilon}. \quad (2.9)$$

Отже, для малих змін градієнта mit приблизно дорівнює sqrt(vt), тому зміна ваг повністю контролюється швидкістю навчання. Оскільки градієнт різко змінюється, mit значно менше, ніж sqrt(vt), що означає, що швидкість навчання зменшується.

Під час навчання нейронної мережі важливо підтримувати невелике значення W, що дозволить уникнути перенавчання та створити більш загальну модель. Існуючі методи вирішують цю проблему шляхом додавання члена мкВт до формули розрахунку градієнта (2.5), але це впливає на значення mit і vit - вони відстежують не тільки зміни градієнта, але і регулюючу вагу мережі. Щоб вирішити цю проблему, ми додаємо коригування ваги, як показано в (2.10):

$$W_t = W_{t-1} - \alpha \frac{\beta_1 m_{t-1} + (1-\beta_1)(g_t + \mu W_{t-1})}{\sqrt{v_{t-1}} + \varepsilon}. \quad (2.10)$$

У цьому випадку зменшення ваги також нормалізується значенням sqrt(vt).Коли значення W велике або значення градієнта велике (воно різко змінюється), значення vt також матиме велике значення ., тому вага істотно не зміниться. Те саме стосується й малих величин. [23]

Нейронні мережі зазвичай використовують не тільки ваги в кожному нейроні, але й такі параметри, як зміщення, швидкість навчання, градієнт і

ковзне середнє його квадрата. Зміна параметрів подібна до описаного вище процесу. Загальна схема навчання нейронної мережі показана на рисунку 2.1.

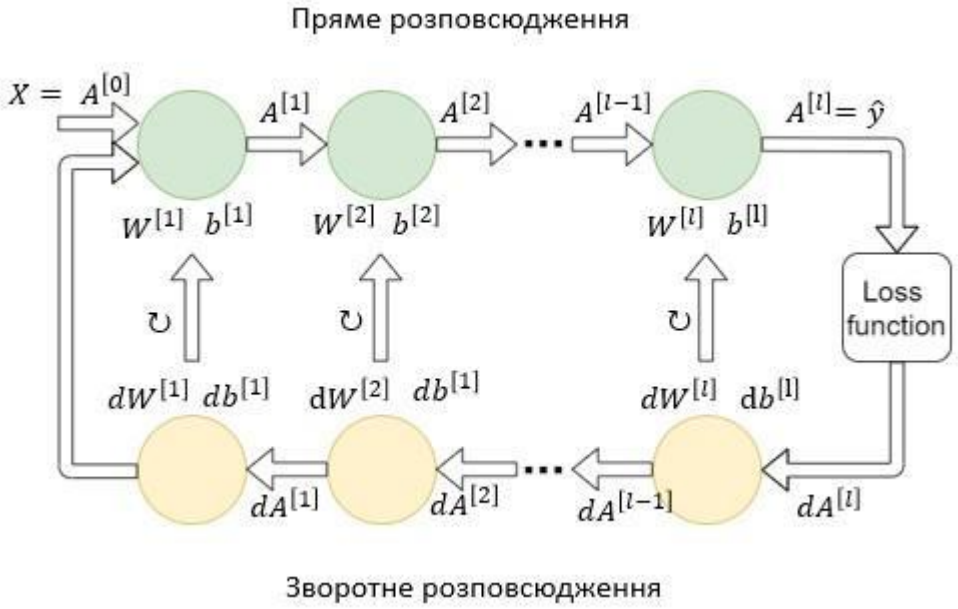


Рисунок 2.1 – Схема прямого і зворотнього поширення

Функція втрат. Функція втрат, функція помилки та вартість навчання показують оцінку, яка підсумовує різницю між фактичним і прогнозованим розподілами ймовірностей для всіх класів проблеми. Це значення показує ефективність моделі класифікації та варіюється від 0 до 1. Цю частку слід мінімізувати, тобто ідеальне значення перехресної ентропії має дорівнювати 0.

2.5 Математична постановка задачі

Сформулюємо цільову функцію для заданої задачі. Для випадкових вибірок зображень, розподілених за класом c, нейронну мережу потрібно навчити мінімізувати функцію втрат.

Оскільки:

си – класифікаційна категорія, тобто архітектурний стиль;

Xj - зображення будівлі;

N – кількість зображень;

pj – ймовірність того, що зображення Xj належить класу cXj;

Цільова функція вирішення задачі:

Необхідно мінімізувати значення функції втрат, яка відповідає за сумарну похибку мережі під час навчання. Функція втрати показує, які частини зображення можуть бути неправильно класифіковані за допомогою мережі:

$$L = -\frac{1}{N}\sum_{j=1}^{N} \ln p_j \to min. \quad (2.11)$$

З обмеженнями:

$i, j, N \in N;$

$1 \leq N < \infty;$

$2 \leq i < \infty;$

$0 \leq L < \infty;$

2.6 Схема алгоритму розв’язання задачі

2.6.1 Алгоритм навчання

Вивчення нейронної мережі передбачає коригування вагових коефіцієнтів і порогових функцій за допомогою ітераційного процесу. Розглянемо загальний процес навчання нейронної мережі:

Крок 1: Ініціалізація всіх вагових коефіцієнтів W(l,u,v) з’єднань нейронної мережі випадковими значеннями та зсувом b, де W(l,u,v) – це вагові коефіцієнти з’єднання від рівня нейрона u (l – 1) до нейрона v у шарі l.

Крок 2. Виконайте наступне:

Крок 2.1: Встановіть дані в навчальній вибірці як вхідне значення мережі, а вихідне значення – як вихідний рівень.

Крок 2.2: обчисліть вхідне значення кожного нейрона в наступному прихованому шарі та застосуйте функцію активації. Повторіть для всіх наступних шарів:

$$R^{(l)} = Y^{(l-1)}W^{(l-1)},$$
$$Y^{(l)} = \sigma\left(X^{(l)}\right).$$

Крок 3. Запустіть у зворотному напрямку:

Крок 3.1: обчисліть матрицю градієнта від передостаннього шару до останнього вихідного шару (l = n).

Крок 3.2: обчисліть значення градієнта попередніх шарів і оновіть ваги відповідно до рівняння 2.10. Цей процес повторюється, доки не оновляться всі ваги нейронної мережі.

Крок 4: Якщо задану кількість ітерацій (епоха) завершено або функція втрат досягла мінімального значення, процес навчання завершено. В іншому випадку перейдіть до кроку 2.

2.6.2 Алгоритм класифікації

Алгоритми класифікації використовують уже навчені нейронні мережі із заданими вагами та іншими значеннями параметрів. Тому під час класифікації необхідно лише «пропустити» зображення через усі шари нейронної мережі та інтерпретувати отримані вектори:

Крок 1: ми беремо вхідні дані та встановлюємо значення вхідного рівня мережі (рівень 0) у значення даних, а значення результату – у вихідний рівень.

Крок 2: обчисліть вхідне значення кожного нейрона в наступному R(l,j) і застосуйте функцію активації, щоб отримати вихідне значення наступного прихованого шару. Повторіть для всіх наступних шарів:

$$R^{(l)} = Y^{(l-1)}W^{(l-1)},$$
$$Y^{(l)} = \sigma\left(R^{(l)}\right).$$

Крок 3: зіставте вихідний вектор з назвою класу та поверніть результат класифікації. [24].

2.6.3 Обґрунтування правильності роботи алгоритму

Оскільки під час навчання нейронної мережі дуже ймовірно, що мінімальне значення функції втрат не буде досягнуто, але зупиниться на локальному мінімумі, або мінімум буде пропущено, алгоритм має бути дуже обережним щодо зміни вагових коефіцієнтів мережі. Якщо розмір кроку занадто малий, час навчання буде дуже довгим, а якщо розмір кроку занадто великий, точність навчання може вплинути. Для вирішення цієї проблеми використовується адаптивна швидкість навчання - коли значення градієнта мало, вага змінюється швидко і прискорюється до мінімальної точки, тоді як коли градієнт великий, це означає, що функція втрат починає зменшуватися, і це Варто робити менші кроки, щоб не упустити мінімум. Ця стратегія може збільшити швидкість пошуку оптимального рішення, не впливаючи на точність навчання.

Ще одна проблема навчання нейронної мережі – це перетренованість. Щоб уникнути цієї проблеми, змінюйте вагові значення нейронної мережі на низькому рівні. Тому щоразу, коли ви змінюєте вагу, виконується масштабування - значення коригується так, щоб значення ваги не збільшувалося. Існуючий алгоритм використовує цей метод, але до формули розрахунку градієнта додає коригування ваги, що впливає на процес пошуку мінімального значення функції втрат і може призвести до вибору неправильного напрямку зміни ваги. Для вирішення цієї проблеми до формули для розрахунку нових ваг мережі після розрахунку градієнта були додані правила, які дозволяють зберігати низькі значення ваг, не впливаючи на процес знаходження мінімуму.

У розділі «Розробка алгоритмів класифікації зображень» наведено математичну постановку задачі, цільову функцію задачі та схему алгоритмів навчання та класифікації зображень. Розглянуто загальні концепції та методи, які використовуються при розробці алгоритмів класифікації, а також розглядаються процес навчання нейронних мереж і гіперпараметри, що

використовуються при їх створенні. У цьому розділі детально описані методи і процеси виділення кордонів і кордонів об'єктів на зображеннях, а також вибрані методи зменшення зображення. Одночасно перевіряється правильність алгоритму навчання нейронної мережі.

2.7 Аналіз мови програмування Python і специфіка її використання для нейронних мереж

Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Його розширені вбудовані структури даних у поєднанні з динамічною типізацією та динамічним зв’язуванням роблять його дуже привабливим для швидкої розробки додатків і для використання як мови сценаріїв або мови зв’язування для з’єднання існуючих компонентів. Простий у вивченні синтаксис Python підкреслює читабельність, таким чином зменшуючи витрати на обслуговування програми. Python підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python і велика стандартна бібліотека вільно доступні у вихідному коді або двійковій формі для всіх основних платформ і можуть вільно поширюватися [23]. Оскільки етапу компіляції немає, цикл редагування-тестування-налагодження виконується дуже швидко. Налагодження програм на Python просте: помилки або неправильний ввід ніколи не призведуть до помилки сегментації. Натомість, коли інтерпретатор виявляє помилку, він створює виняток. Якщо програма не перехопить виключення, інтерпретатор надрукує трасування стека. Налагоджувач рівня вихідного коду дозволяє перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати контрольні точки, крок за рядком коду тощо. Сам налагоджувач написаний мовою Python, що є свідченням можливостей самоаналізу Python. З іншого боку, часто це найшвидший спосіб налагодити програму.

Часто вважають, що програми Python працюють повільніше, ніж програми Java, але час розробки набагато менший. Програми на Python зазвичай у 3-5 разів коротші за еквівалентні програми на Java. Цю різницю можна пояснити вбудованими в Python типами даних високого рівня та його динамічною типізацією. Наприклад, програмісти Python не витрачають час на оголошення типів параметрів або змінних, а потужні поліморфні типи списків і словників Python (розширена підтримка синтаксису яких вбудовано в мову) можна використовувати майже в кожній програмі Python. Через тип середовища виконання Python має бути довшим, ніж Java. Наприклад, коли обчислюється вираз a+b, об'єкти a і b потрібно спочатку перевірити, щоб дізнатися їхні типи, які невідомі під час компіляції. Потім він викликає відповідну операцію додавання, яка може бути перевантаженим методом, визначеним користувачем. Java, з іншого боку, може виконувати ефективне додавання цілих чи плаваючих ком, але вимагає оголошення змінних для a і b і не дозволяє перевантажувати оператор + для екземплярів визначених користувачем класів.

З цих причин Python краще підходить як «склеюча» мова, тоді як Java краще підходить як мова реалізації низького рівня. Насправді, обидва разом створюють чудову комбінацію. Компоненти можна розробляти на Java та комбінувати для створення програм на Python [24].

Згідно з опитуванням, NumPy, Pandas і Matplotlib є основними бібліотеками, які використовують експерти з аналізу даних.

Pandas — це одна з бібліотек інтелектуального аналізу даних, яка надає розширені структури даних та інструменти для простого маніпулювання даними. Забезпечення простого, але ефективного підходу до аналізу даних вимагає здатності індексувати, витягувати, розділяти, об'єднувати, реорганізовувати та виконувати різноманітні інші аналізи багатовимірних та одновимірних даних [25].

Переваги використання Pandas:

• Pandas робить представлення даних легким і простим, покращуючи аналіз і розуміння даних. Для проектів наукових даних таке просте представлення даних допомагає отримати кращу інформацію.

• Pandas дуже ефективний, оскільки дозволяє виконувати будь-яке завдання, написавши лише кілька рядків коду.

• Pandas надає користувачам різноманітні команди для швидкого аналізу даних.

NumPy — це бібліотека Python для числових і наукових обчислень. NumPy надає велику кількість функцій, які ентузіасти та програмісти Python можуть використовувати для роботи з високопродуктивними масивами та матрицями. Масиви NumPy забезпечують векторизацію математичних операцій, що покращує продуктивність порівняно з конструкціями циклу Python [26].

Переваги використання NumPy

• NumPy забезпечує ефективне, масштабоване зберігання даних і краще керування даними для математичних обчислень.

• Масиви Numpy містять багато функцій, методів і змінних, які спрощують обчислення матриці.

SciPy — це набір математичних функцій і алгоритмів, побудованих на розширенні Python NumPy. SciPy надає різноманітні високорівневі команди та класи для обробки та візуалізації даних. SciPy корисний для обробки даних і створення прототипів систем. Він містить такі модулі, як статистика, оптимізація, інтеграція, лінійна алгебра, перетворення Фур’є, обробка сигналів і зображень, вирішувач ODE [27].

Переваги використання SciPy

• Використовуйте розширені команди та класи для візуалізації та маніпулювання даними.

• Для паралельного програмування існують класи, веб-процедури та процедури бази даних.

Sci-Kit Learn має керовані та неконтрольовані алгоритми машинного навчання, придатні для виробничих програм. Sci-Kit Learn зосереджується на якості коду, документації, простоті використання та продуктивності, оскільки бібліотека надає алгоритми навчання [28].

• бібліотеки scikit-learn є корисними інструментами для прогнозування поведінки клієнтів, розробки нейровізуалізації тощо.

• Він простий у використанні та абсолютно безкоштовний.

TensorFlow — це безкоштовна наскрізна платформа машинного навчання з відкритим вихідним кодом, яка включає різноманітні інструменти, бібліотеки та ресурси. Вперше його опублікувала команда Google Brain 9 листопада 2015 року. TensorFlow спрощує розробку та навчання моделей машинного навчання за допомогою API високого рівня, таких як Keras. Він також забезпечує різні рівні абстракції, що дозволяє вибрати найкращий підхід для вашої моделі. TensorFlow також дозволяє розгортати моделі машинного навчання в різних середовищах, включаючи хмару, браузери та персональні пристрої. TensorFlow надає набір робочих процесів із інтуїтивно зрозумілими API високого рівня для початківців і експертів для створення моделей машинного навчання кількома мовами. Розробники можуть розгортати моделі на багатьох платформах, таких як сервери, хмара, мобільні та периферійні пристрої, браузери та багато інших платформ JavaScript. Це полегшує розробникам перехід від побудови моделі та навчання до розгортання. [29].

OpenCV з ліцензією BSD — це безкоштовна бібліотека машинного навчання та комп’ютерного зору. Він пропонує загальну архітектуру програм

комп’ютерного бачення для оптимізації реалізації комп’ютерного бачення в комерційних продуктах [30].

SQLAlchemy — це набір інструментів бази даних Python, який допомагає вам ефективно отримувати доступ до сховищ даних. Він містить найпоширенішу високопродуктивну модель доступу до бази даних. SQLAlchemy ORM і SQLAlchemy Core є двома основними компонентами SQLAlchemy. Ядро SQLAlchemy додає рівень абстракції, застосовуючи API бази даних Python і функціональність. Він також надає користувачам інструкції та схеми SQL. SQLAlchemy ORM є автономним об’єктно-реляційним картографом. SQLAlchemy дозволяє розробникам контролювати свою базу даних, одночасно автоматизуючи надлишкові операції [31].

BeautifulSoup — це бібліотека збору та аналізу даних Python. Він збирає необроблені дані HTML і XML. Це дає змогу дослідникам даних розробити веб-сканер, який може сканувати веб-сайти. BeautifulSoup може отримувати дані та створювати їх у потрібному форматі. Зібрані дані HTML містять велику кількість зашифрованих веб-даних, які користувачі не можуть інтерпретувати. Його остання версія, BS4 (BeautifulSoup 4), організовує хаотичні веб-дані в прості для розуміння XML-структури для аналізу даних. BeautifulSoup автоматично визначає кодування та плавно інтерпретує документи HTML, включаючи спеціальні символи [32].

Ggplot — це бібліотека візуалізації даних Python, яка базується на мові програмування R ggplot2 і має рейтинг близько 3000 зірок на Github. Ggplot може використовувати API високого рівня для створення візуалізацій даних, таких як гістограми, кругові діаграми, гістограми, діаграми розсіювання, графіки помилок тощо. Це також дозволяє комбінувати різні компоненти або шари візуалізації даних у складені візуалізації.

2.8 Огляд середовища розробки

Вибрано середовище Jupyter як середовище розробки програмного забезпечення. Проект Jupyter — це некомерційний проект із відкритим вихідним кодом, який народився на основі проекту IPython у 2014 році. Розвиваючись, він підтримує інтерактивну науку про дані та наукові обчислення всіма мовами програмування. Jupyter завжди буде на 100% відкритим програмним забезпеченням і безкоштовним для всіх і на ліберальних умовах модифікованої ліцензії BSD [33]. Jupyter розробляється публічно на GitHub завдяки консенсусу спільноти Jupyter. Усі онлайнові та особисті взаємодії та комунікації, безпосередньо пов’язані з проектом, регулюються Кодексом поведінки Jupyter. У цьому Кодексі поведінки встановлюються вимоги щодо того, щоб різноманітна спільнота користувачів і співавторів брала участь у проекті шанобливо та безпечно.

Основні переваги використання середовища Jupyter:

• Редагування коду в браузері з автоматичним підсвічуванням синтаксису, відступами та завершенням табуляції/самоаналізом.

• Можливість виконання коду з браузера та додавання результатів обчислень до коду, який їх згенерував.

• Показувати результати обчислень за допомогою мультимедійних представлень, таких як HTML, LaTeX, PNG, SVG тощо. Наприклад, індикатори якості публікації, відтворені бібліотекою matplotlib, можуть бути вбудовані.

• Редагуйте форматований текст у браузері за допомогою мови розмітки Markdown, яка надає коментарі до коду, а не лише звичайний текст.

• Можливість легко включати математичні символи в комірки цін за допомогою LaTeX і відтворювати їх у MathJax.

Документи Jupyter Notebook містять вхідні та вихідні дані інтерактивних сеансів, а також додатковий текст, який супроводжує код, але не призначений для виконання. Таким чином, файли блокнота можуть служити повним

обчислювальним записом сеансу, вкраплюючи виконуваний код пояснювальним текстом, багатими представленнями математики та об’єктами результатів. Ці документи є внутрішніми файлами JSON і зберігаються з розширенням .ipynb [34].

Крім того, будь-який документ блокнота .ipynb, доступний за загальнодоступною URL-адресою, можна поділитися через Jupyter Notebook Viewer <nbviewer>. Ця служба завантажує документи блокнота з URL-адреси та відображає їх як статичні веб-сторінки. Таким чином, результатами можна поділитися з колегами або у вигляді публічної публікації в блозі, не вимагаючи від інших користувачів самостійно встановлювати блокноти Jupyter. Таким чином, звичайний робочий процес у Блокноті дуже схожий на стандартний сеанс IPython, за винятком того, що замість використання команди %run magic для повторного запуску одного сценарію ви можете редагувати клітинку кілька разів, доки не отримаєте бажаний результат. [35].

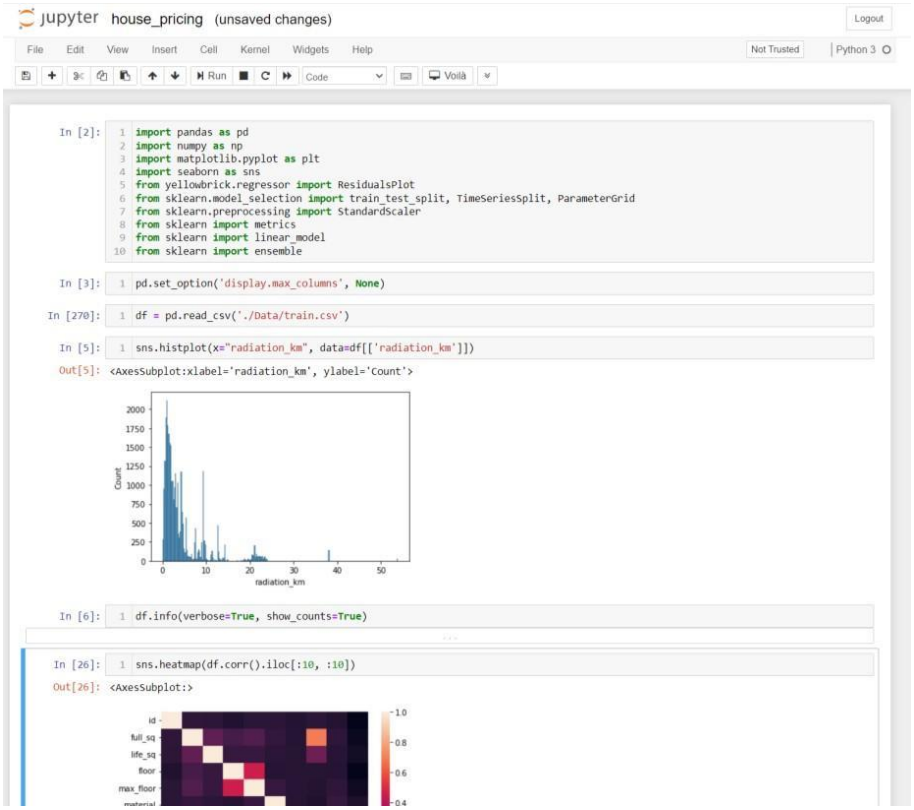


Рисунок 2.2 – Інтерфейс Jupyter ноутбука

Як правило, ви розв’язуєте задачу обчислення частинами, організовуєте пов’язані ідеї в клітинки та рухаєтеся вперед, коли попередні частини працюють належним чином. Для інтерактивного дослідження це набагато зручніше, ніж розбивати обчислення на сценарії, які потрібно виконувати разом (що раніше було необхідно), особливо якщо деякі з них виконуються довго.

JupyterLab — це користувальницький інтерфейс наступного покоління для проекту Jupyter, який забезпечує всі знайомі будівельні блоки класичного блокнота Jupyter — блокнот, термінал, текстовий редактор, файловий браузер і розширений вихід — у гнучкому користувацькому інтерфейсі.

JupyterLab можна розширити за допомогою пакетів npm, які використовують публічний API. Попередньо скомпільовані розширення можна поширювати через PyPI, conda та інші менеджери пакетів. Вихідні розширення можна встановити безпосередньо з npm, але для цього потрібні додаткові кроки збірки.

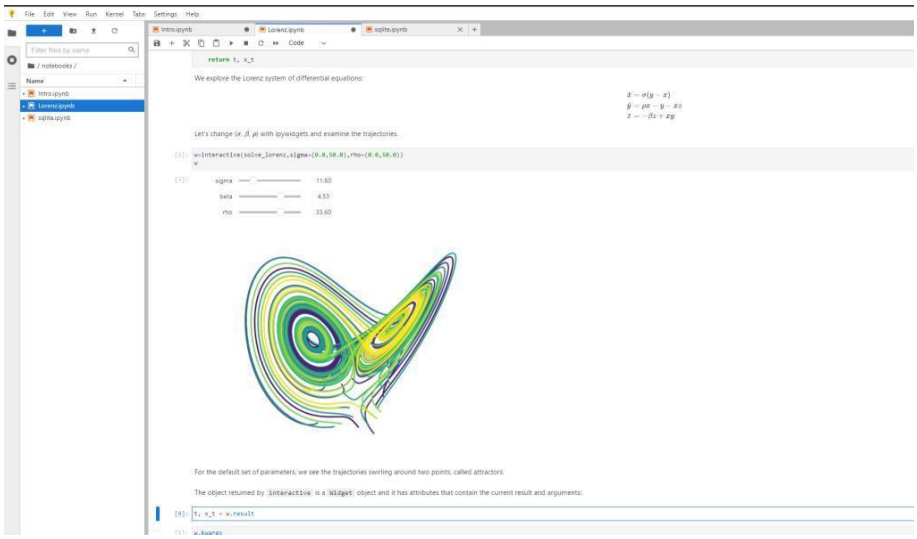


Рисунок 2.3 – Інтерфейс Jupyter lab

Крім того, JupyterLab дозволяє використовувати текстовий редактор, термінал, засіб перегляду файлів даних та інші настроювані компоненти разом із блокнотами в робочій області з вкладками.

• Використовуйте перетягування, щоб переставляти та копіювати комірки між блокнотами.

• Блоки коду виконуються інтерактивно безпосередньо з текстових файлів (.py, .R, .md, .tex тощо).

• Консоль коду можна зв’язати з ядром блокнота для інтерактивного вивчення коду, не захаращуючи блокнот спеціальними редагуваннями.

• Можливість редагувати такі популярні формати файлів, як Markdown, JSON, CSV, Vega, VegaLite тощо з попереднім переглядом у реальному часі.

JupyterLab побудовано на системі розширення, яка дозволяє налаштовувати та покращувати ваше середовище. Насправді всі вбудовані функції самого JupyterLab, тобто блокнот, термінал, файловий браузер, система меню тощо, забезпечуються базовим набором розширень.

Оскільки важлива частина навчання нейронних мереж вимагає частих виправлень, вибору параметрів та інших подібних речей, доцільно розробляти програмну частину в цьому середовищі.

2.9 Аналіз даних для навчання та бібліотек взаємодії з ними

Набір даних про архітектуру будівельних споруд із веб-сайту конкурсу kaggle.com було обрано як навчальні дані з таких причин:

1) Вибраний набір даних містить уже підготовлені дані, тому не потрібно витрачати час на їх очищення, і ви можете зосередитися на меті роботи.

2) Набір даних містить приблизно 193 000 навчальних зображень, яких достатньо для навчання нейронної мережі.

3) Зображення надаються з різними погодними умовами, наближеннями, якістю та іншими відмінностями, що дозволяє тренувати мережу в більшості можливих ситуацій

4) Платформа може порівнювати результати розпізнавання з іншими учасниками, і хоча багато фіналістів використовують ансамблі нейронних мереж або вдаються до хитрощів, ви можете отримати приблизне уявлення про

те, наскільки точно створена нейронна мережа справляється зі своїм завданням

Kaggle — це платформа онлайн-спільноти для спеціалістів із обробки даних та ентузіастів машинного навчання. Kaggle дозволяє користувачам співпрацювати з іншими користувачами, знаходити та публікувати набори даних, використовувати ноутбуки з інтегрованими графічними процесорами та конкурувати з іншими дослідниками даних для вирішення завдань науки про дані. Мета цієї онлайн-платформи (заснованої в 2010 році Ентоні Голдблюмом і Джеремі Говардом і придбаної Google у 2017 році) полягає в тому, щоб допомогти професіоналам і студентам досягти своїх цілей у сфері наукових даних за допомогою потужних інструментів і ресурсів, які вона надає. Станом на сьогодні (2021) Kaggle має понад 8 мільйонів зареєстрованих користувачів [36].

2.10 Аналіз існуючих підходів до розпізнавання об’єктів

Розпізнавання об’єктів — це загальний термін, що описує набір пов’язаних завдань комп’ютерного зору, пов’язаних з ідентифікацією об’єктів на цифрових фотографіях [37].

Класифікація зображень передбачає передбачення класу окремих об’єктів на зображенні. Позиціонування об’єкта означає розміщення одного чи кількох об’єктів на зображенні та малювання прямокутника навколо їхнього розміру. Виявлення об’єктів поєднує ці два завдання та визначає місцезнаходження та класифікує один або кілька об’єктів на зображенні.

Тому ми можемо виділити ці три завдання комп’ютерного зору:

1 Класифікація зображень: завдання призначення міток або категорій цілим зображенням. Для кожного зображення може бути лише один клас. Моделі класифікації зображень приймають зображення як вхідні дані та повертають передбачення щодо класу, до якого належить зображення. У

класифікаторах на основі CNN базовий рівень — це згортковий рівень нейронної мережі, що складається від кількох шарів до 100 (наприклад, ResNet 101), залежно від програми, обсягу даних і доступних обчислювальних ресурсів. Кількість шарів сама по собі є великою областю дослідження. Після рівня CNN слідує рівень об’єднання, за яким слідують один або два повністю з’єднані рівні. Останній шар є вихідним шаром, який дає ймовірність присутності об'єкта на зображенні. Наприклад, припустимо, що алгоритм ідентифікує 100 різних об’єктів на даному зображенні. Останній шар генерує масив довжиною 100 зі значеннями в діапазоні від 0 до 1, що представляє ймовірність присутності об'єкта на зображенні [38].

Вхід: зображення з одним об’єктом, наприклад фотографія.

Вихід: мітка класу (наприклад, одне або кілька цілих чисел, зіставлених з міткою класу).

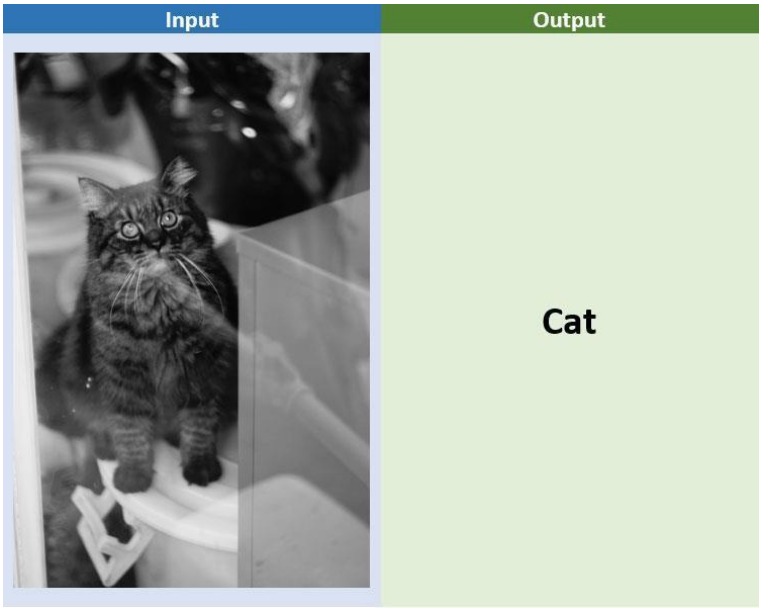


Рисунок 2.4 – Приклад класифікації зображень

2 Локалізація об’єкта: визначте наявність об’єктів на зображенні та вкажіть їх розташування за допомогою обмежувальних рамок. Локалізація зображення є побічним продуктом традиційних алгоритмів зору CNN. Ці алгоритми передбачають класи з дискретними числами. Під час локалізації

об’єкта алгоритм передбачає набір із 4 чисел, а саме координати x, координати y, висоти та ширини, щоб намалювати обмежувальну рамку навколо цільового об’єкта [39].

Вхід: зображення, що містить один або кілька об’єктів, наприклад фотографію.

Вихід: одна або кілька обмежувальних рамок (наприклад, визначених точками, шириною та висотою).

Рисунок 2.5 – Приклад локалізації об’єктів

Виявлення об’єктів: техніка комп’ютерного зору, яка використовується для визначення місцезнаходження об’єктів на зображеннях або відео. Алгоритми виявлення об’єктів часто використовують машинне або глибоке навчання для отримання значущих результатів. Це робиться за допомогою обмежувальних рамок і типу або категорії об’єкта на зображенні, щоб визначити присутність об’єкта. По суті, це складне завдання, яке поєднує концепції локалізації зображення та класифікації. За наявності зображення алгоритми виявлення об’єктів повертають обмежувальні прямокутники навколо всіх цікавих об’єктів і призначають їм клас [40].

Вхід: зображення, що містить один або кілька об’єктів, наприклад фотографію.

Вихід: одна або кілька обмежувальних рамок (наприклад, визначених точками, шириною та висотою) і мітка класу для кожної обмежувальної рамки.

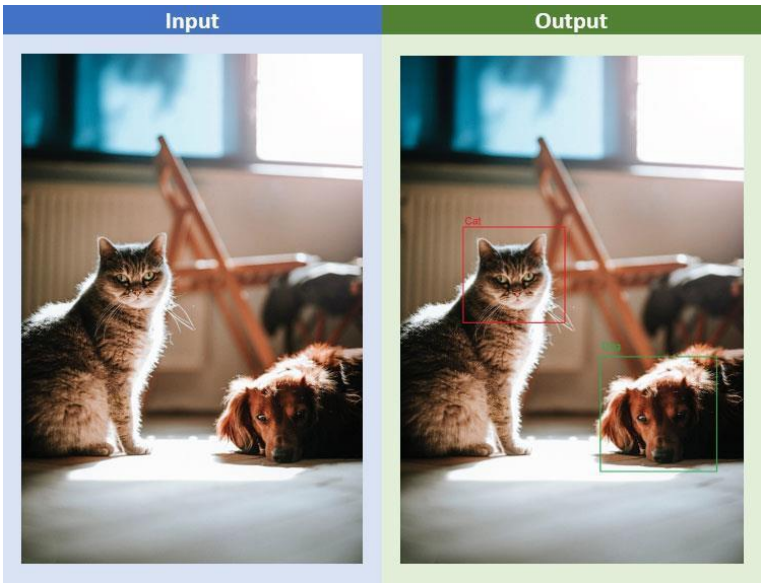


Рисунок 2.6 – Приклад виявлення об’єктів

Ефективність моделі класифікації зображень була оцінена за допомогою середньої помилки класифікації прогнозованих міток класу. Оцініть ефективність моделі при локалізації окремого об’єкта, використовуючи відстань між очікуваними та прогнозованими обмежувальними рамками очікуваного класу. Ефективність моделі розпізнавання об’єктів оцінюється точністю та запам’ятовуванням кожної найкращої відповідності обмежувальної рамки відомого об’єкта на зображенні.

Метою будь-якого алгоритму машинного навчання є прогнозування значень, які максимально наближені до справжньої ситуації. З цією метою будь-який контрольований алгоритм машинного навчання використовує функцію втрат, і алгоритм навчається шляхом мінімізації втрат через оптимізацію ваги або параметрів.

Оскільки локалізація об’єкта є проблемою регресії, можна використовувати будь-яку регресійну функцію втрат, придатну для N-вимірних масивів. Наприклад, втрати відстані L1, втрати відстані L2, втрати Губера тощо [41].

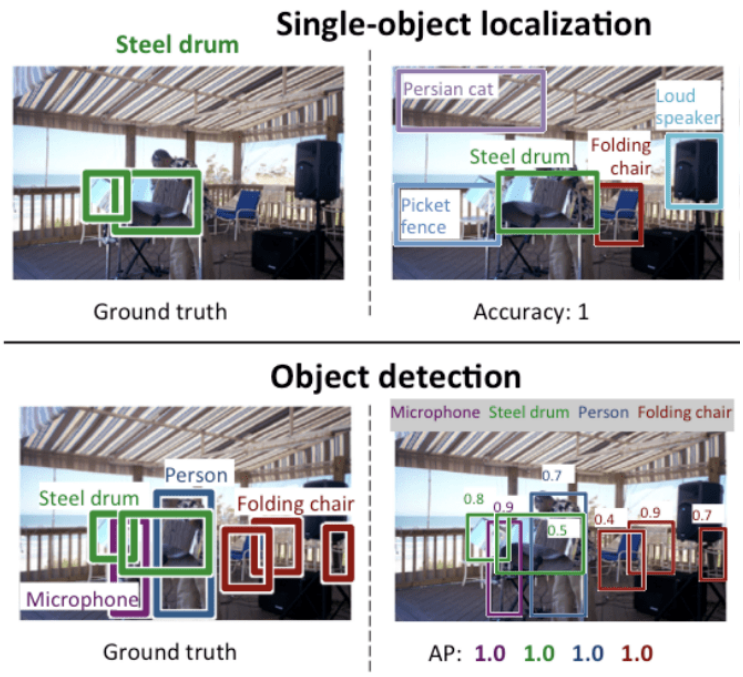


Рисунок 2.7 – Порівняння між локалізацією окремого об’єкта та виявленням об’єкта

Мережа R-CNN є одним із існуючих методів розпізнавання, і це перше масштабне та успішне застосування згорткових нейронних мереж у задачах локалізації, сегментації та виявлення об’єктів [42].

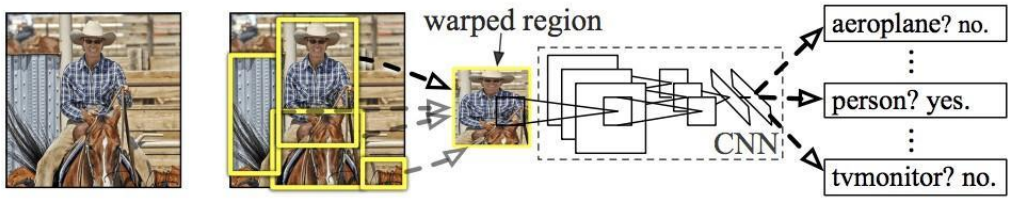


Рисунок 2.8 – Архітектура R-CNN моделі

Структуру мережі R-CNN можна розділити на 3 модулі:

• Модуль 1: Регіональні пропозиції – створюйте та видаляйте незалежні від категорії пропозиції регіону, наприклад кадри кандидатів.

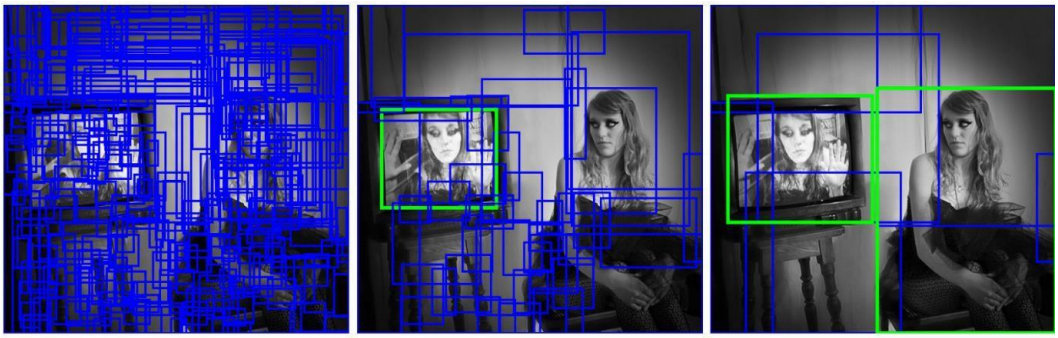


Рисунок 2.9 – Регіональна пропозиція

• Модуль 2: Функціональний екстрактор – витягує ознаки з кожної кандидатської області, наприклад, за допомогою глибокої згорткової нейронної мережі. Під час процесу навчання CNN «вивчає» оптимальні значення матриці фільтра, що дозволяє витягувати значущі характеристики (текстури, краї, форми) з вхідної карти функцій. Зі збільшенням кількості фільтрів, застосованих до вхідних даних (глибина вихідної карти ознак), кількість ознак, які CNN може витягти, також збільшується. Однак компроміс полягає в тому, що фільтри займають більшу частину ресурсів, які споживає CNN, тому час навчання збільшується, коли додається більше фільтрів. Крім того, кожен фільтр, доданий до мережі, забезпечує меншу додаткову цінність, ніж попередній фільтр, тому інженери прагнуть побудувати мережі, які використовують мінімальну кількість фільтрів для виділення функцій, необхідних для точної класифікації зображень.

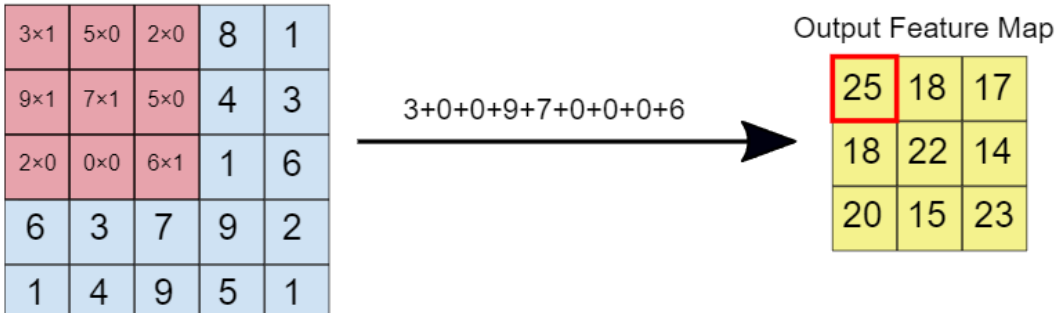


Рисунок 2.10 – Принцип згортки зображення

• Модуль 3: Класифікатор – класифікує об’єкт за однією з відомих категорій, наприклад, використовуючи модель лінійного класифікатора SVM.

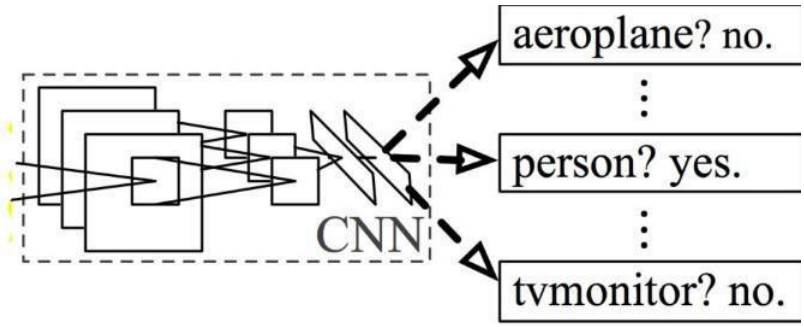


Рисунок 2.11 – Передбачення останнього модуля

Наскрізна структура моделі CNN показана на малюнку нижче.

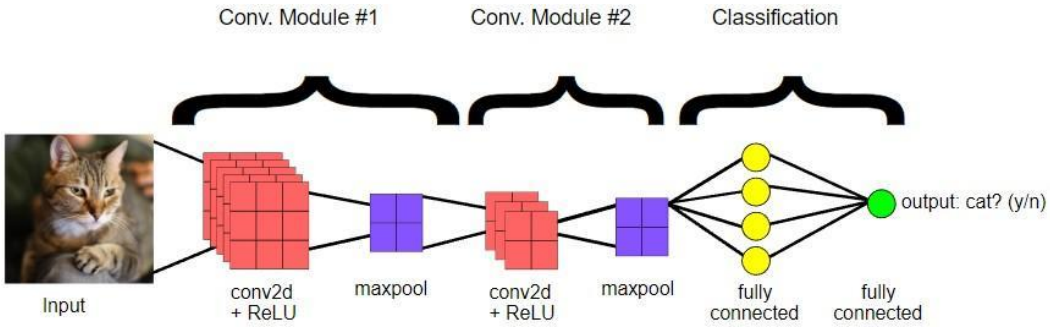


Рисунок 2.12 – Наскрізна структура CNN

До недоліків цієї моделі (зазначених в її описі) можна віднести:

• Навчання – це багатоетапний процес. Він передбачає підготовку та роботу трьох окремих моделей.

• Навчання є дорогим у просторі та часі. Підготовка глибоких CNN до такої великої кількості регіональних пропозицій на кожному зображенні дуже повільна.

• Виявлення об'єктів відбувається повільно. Прогнозування для такої великої кількості регіональних пропозицій дуже повільне за допомогою глибоких CNN.

Іншим методом ідентифікації можна вважати сімейство моделей YOLO. Модель YOLO була спочатку описана Джозефом Редмоном та іншими в статті 2015 року під назвою «You Only Look Once: Unified, Real-Time Object Inspection». Зауважте, що Росс Хіршик, розробник R-CNN, також був автором і учасником цієї роботи та працював у Facebook AI Research у той час.

Метод включає в себе єдину нейронну мережу, навчену за допомогою відеоматеріалу, яка приймає фотографії як вхідні дані та безпосередньо прогнозує обмежувальну рамку та мітку класу для кожної обмежувальної рамки. Технологія має нижчу точність передбачення, наприклад більше помилок локалізації, хоча вона працює зі швидкістю 45 кадрів на секунду, а версії, оптимізовані за швидкістю моделі, працюють зі швидкістю до 155 кадрів на секунду.

Зокрема, це дає можливість використовувати модель на вуличних камерах, дронах та інших подібних засобах.

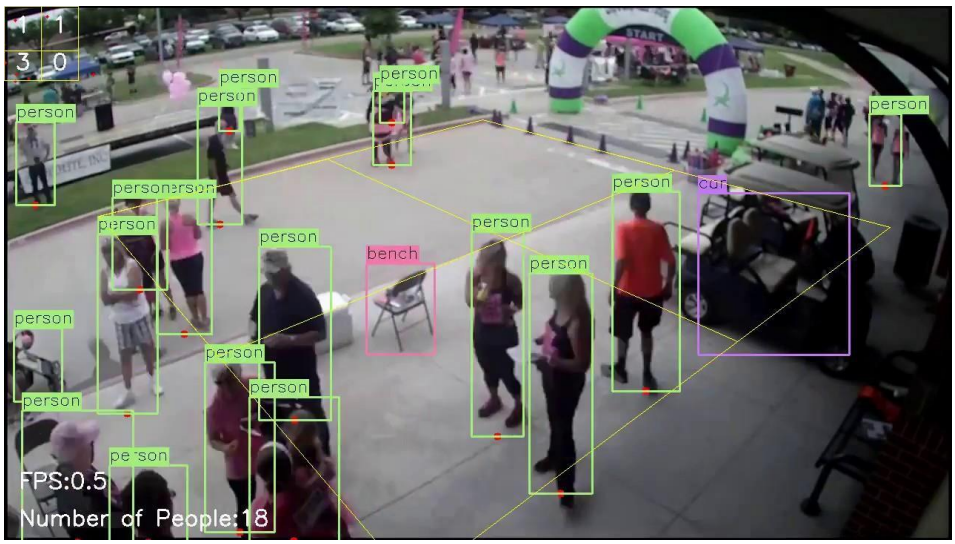


Рисунок 2.13 – робота YOLO моделі

Модель працює спочатку розбиваючи вхідне зображення на сітку клітинок, де кожна клітинка відповідає за прогнозування обмежувальної рамки, якщо центр обмежувальної рамки потрапляє в клітинку. Кожна клітинка сітки передбачає обмежувальну рамку, що включає координати x, y,

ширину, висоту та достовірність. Прогноз класу також базується на кожній клітинці.

Наприклад, зображення можна розділити на сітку 7×7, і кожна клітинка в сітці може передбачати 2 обмежувальні прямокутники, що призводить до 94 запропонованих обмежувальних рамок. Карта ймовірностей класів і обмежувальні прямокутники з достовірністю потім об'єднуються в остаточний набір обмежувальних рамок і міток класів.

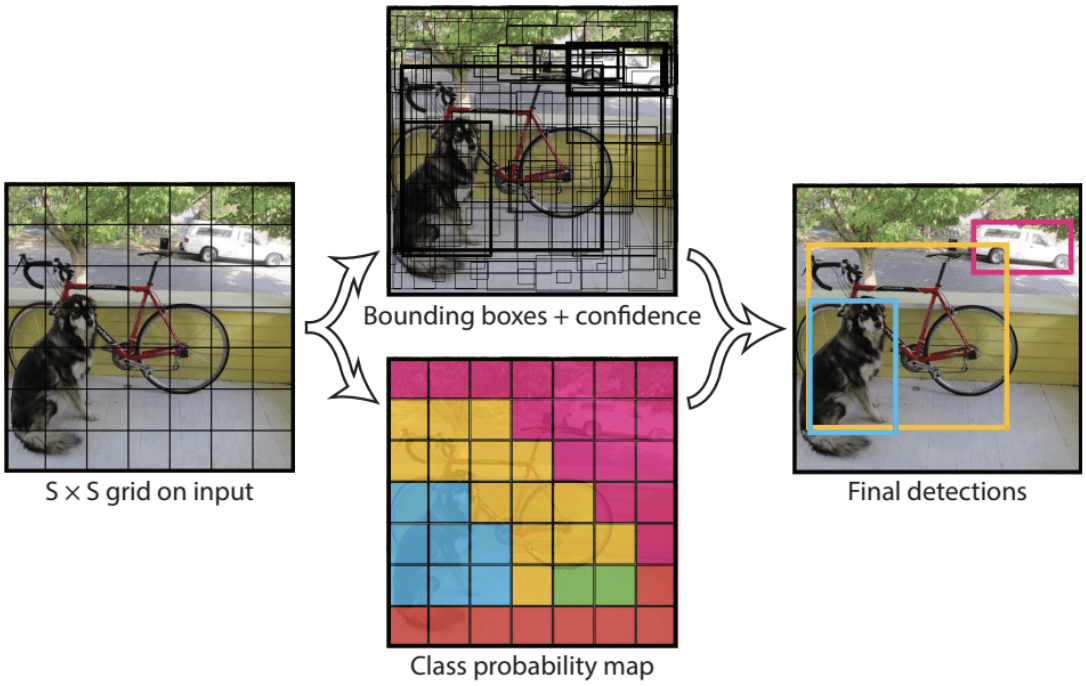


Рисунок 2.14 – Структура YOLO моделі

У даному розділі були розглянуті обрані технічні засоби з реалізації нейронної мережі. Описано обрану мову її переваги та деякі необхідні бібліотеки. Була приділена увага середовищу розробки, та даним що будуть використовуватись для навчання, оглянуто деякі існуючи підходи до розпізнавання.

3. ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вимоги до програмного продукту

3.1.1 Вимоги до функціональних характеристик

Система повинна надавати користувачам такі функціональні можливості:

а) створювати облікові записи;

б) увійдіть у свій обліковий запис;

в) Визначте архітектурний стиль будівлі:

1) Завантажте зображення;

2) Класифікуйте його ;

3) езультати аудиту;

4) Додайте опис зображення;

d) Перегляд попередньо збережених результатів класифікації;

е) Перегляд документів і посібників користувача.

3.1.2 Вимоги до надійності

Особиста інформація користувачів (паролі) повинна зберігатися в зашифрованому вигляді. Введені користувачем дані повинні бути перевірені на формальну правильність.

3.1.3 Вимоги до складу і параметрів технічних засобів

Для правильного запуску мережевих програм мінімальні вимоги до апаратної частини ПК:

а) принаймні 1 Гб оперативної пам'яті;

б) Процесор з тактовою частотою не менше 1 ГГц в) Підключення до Інтернету.

3.2 Засоби розробки

Алгоритм реалізовано за допомогою платформи Python та мови програмування, яка є найпопулярнішою мовою для реалізації алгоритмів машинного навчання. Ця мова досить проста, але має велику кількість модулів і бібліотек, які реалізують різні математичні операції, що дозволяє швидко реалізовувати алгоритми. [43]

При розробці нейронних мереж і алгоритмів обробки зображень використовувалися такі бібліотеки, які використовуються в науці про дані та аналізі даних: NumPy, SciPy, Scikit-Learn, Matplotlib. Ці бібліотеки широко використовуються для створення алгоритмів машинного навчання і є дуже ефективними. Крім того, Python надає готові реалізації популярних методів, таких як класифікація, рекомендація, регресія та кластеризація[44].

Для створення та навчання нейронних мереж використовували Python версії 3.5.2 і менеджер пакетів pip. Використовуються такі бібліотеки Python:

- h5py — це інтерфейс Pythonic для двійкового формату даних HDF5. Дозволяє зберігати великі обсяги числових даних і легко маніпулювати ними за допомогою NumPy;

- keras – бібліотека, що містить API для використання згорткових нейронних мереж, які можуть обробляти зображення на CPU та GPU. Ця бібліотека використовує TensorFlow;

- matplotlib – бібліотека для побудови графіків функцій, гістограм, діаграм, діаграм розсіювання тощо;

- numpy – бібліотека, що надає можливість використовувати готові функції, призначені для роботи з багатовимірними масивами;

- opencv-python – неофіційний, але дуже поширений пакет OpenCV для мови програмування Python[46]. Дозволяє маніпулювати зображеннями та виконувати операції редагування пікселів, геометричні перетворення тощо;

-pillow – бібліотека для обробки растрових зображень;

- scipy - бібліотека, що містить функції модуля для оптимізації, обробки зображень, розв'язання диференціальних рівнянь тощо. Ця бібліотека була розроблена розробниками Scilab і Matlab;

-Tensorflow — це бібліотека з відкритим кодом для створення моделей машинного навчання, розроблена Google[45].

Розробив веб-додаток за допомогою Angular framework і двох мікросервісів, один реалізований за допомогою Flask framework, а інший ASP.NET Core. MsSQL Server використовується для збереження даних користувача та результатів класифікації зображень.

3.3 Архітектура програмного забезпечення

3.3.1 Діаграма класів

Під час розробки інформаційної системи була створена структура класів для доменної області системи, як показано на структурній діаграмі класів у Додатку А. Область домену складається з користувачів, результатів аналізу зображень і зображень. і дозвіл на перегляд збережених результатів іншим користувачам. Результат аналізу завжди належить існуючому користувачеві, який його створив, і завжди містить посилання на зображення.

3.3.2 Діаграма послідовностей

У Додатку Г наведена діаграма структурної послідовності інформаційної системи, яка класифікує зображення будівель за структурою будівлі.

Схема структурної послідовності для автентифікації користувача, наведена в Додатку Г, показує схему послідовності автентифікації для зареєстрованих користувачів. Користувачеві необхідно ввести логін і пароль, натиснути кнопку «Вхід», після перевірки введених даних користувач буде перенаправлений на головну сторінку. Після успішної перевірки даних автентифікації служба генерує маркер автентифікації, який зберігається в

браузері. Усі запити до компонента аналізу та збереження результатів надсилатимуться з отриманим токеном для перевірки користувача.

Структурна схема послідовності реєстрації користувача наведена в Додатку Г.

Користувачеві необхідно заповнити реєстраційну форму з ім'ям, адресою електронної пошти та паролем, натиснути на кнопку «Зареєструватися», після перевірки введених даних користувач буде перенаправлений на головну сторінку. Після успішного створення нового користувача служба негайно генерує маркер автентифікації, який зберігається в браузері. Це необхідно, щоб уникнути процесу входу після реєстрації.

Структурна схема послідовності аналізу зображення наведена в Додатку Г. Користувач повинен завантажити зображення і натиснути кнопку «Аналізувати». Сторінка аналізу зображень доступна як для авторизованих, так і для анонімних користувачів, але лише авторизовані користувачі можуть зберігати результати аналізу та переглядати раніше збережені результати.

3.3.3 Діаграма компонентів

Структурна схема компонентів інформаційної системи класифікації зображень будівель за архітектурним стилем наведена в додатку Г. Система складається з 4 компонентів:

– Компонент аналізу зображення з використанням навчених нейронних мереж. Цей компонент відкритий для всіх користувачів і не вимагає попередньої реєстрації для використання;

– Компонент збереження результатів: дозволяє зберігати результати аналізу та переглядати раніше збережені результати. Цей компонент доступний лише зареєстрованим користувачам;

– Компонент автентифікації: дозволяє створювати нових користувачів, отримувати токени автентифікації користувачів і перевіряти токени;

– Компоненти інтерфейсу користувача: містить форми для автентифікації, реєстрації, аналізу зображень, збереження переглядів результатів і документації.

3.4 Інструкція користувача

Щоб почати використовувати програмне забезпечення, користувачі повинні перейти на сторінку веб-додатку. Інструкції щодо розгортання програмного забезпечення у вашому локальному середовищі наведено в інструкціях до сховища. [47][48][49]

Розглянемо більш детально вигляд домашньої сторінки користувача, головного меню сайту та вміст сторінки, доступний для незареєстрованих користувачів (рис. 3.1):

- Назва програми;

– посилання на головну сторінку;

– Посилання на сторінки документації;

– Посилання на сторінку, що визначає стиль архітектурного зображення;

– Посилання на сторінку входу.

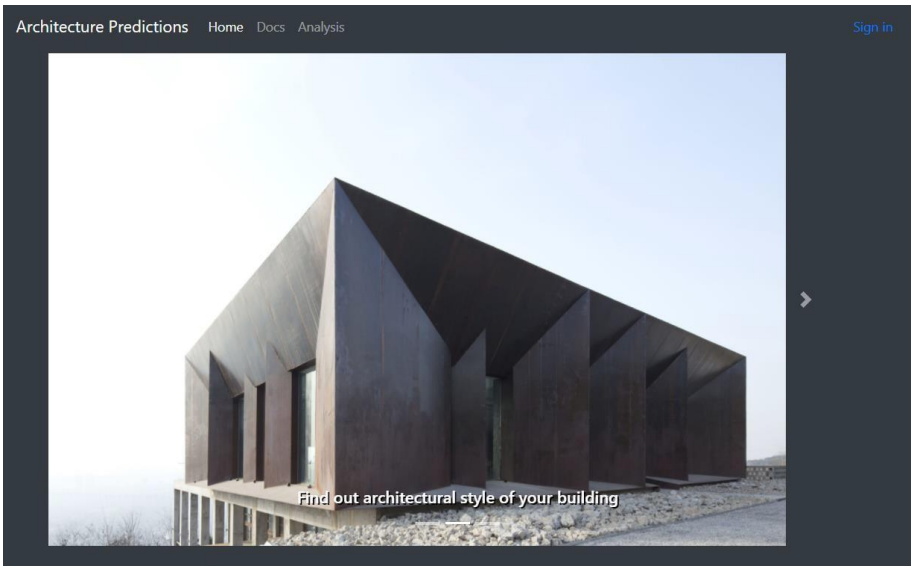


Рисунок 3.1 – Домашня сторінка

Розглянемо форму входу (рис. 3.2). Тут користувач повинен заповнити дані для входу та натиснути кнопку. Якщо користувач не має облікового запису в системі, він може створити його через реєстраційну форму, натиснувши на кнопку Реєстрація (рис. 3.3).

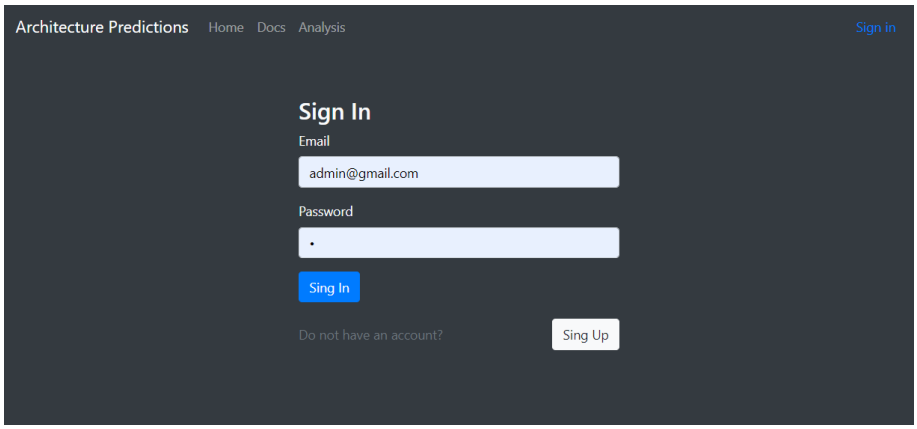


Рисунок 3.2 – Форма входу в систему

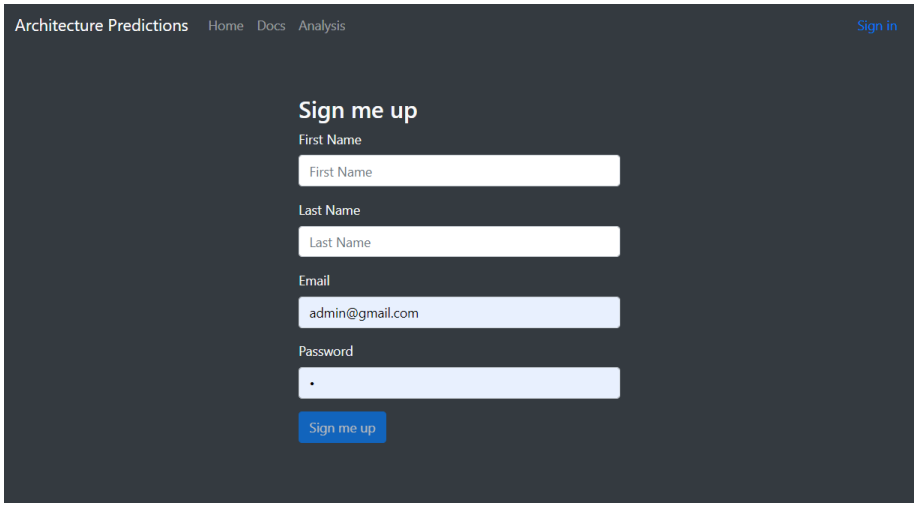


Рисунок 3.3 – Форма реєстрації

Розглянемо сторінку документації (рис. 3.4), де користувач переходить на сторінку з описом архітектурного стилю та результатів.

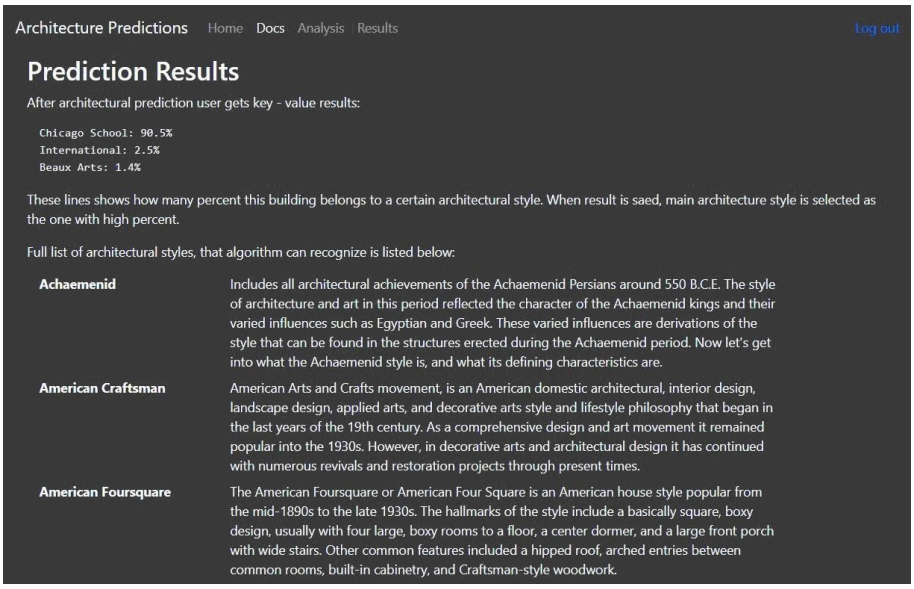


Рисунок 3.4 – Форма документації

Заходимо на сторінку визначення архітектурного стилю та завантажуємо зображення для аналізу (рис. 3.5). На цій сторінці є компонент завантаження зображень, який відображає опис завантаженого файлу або назву завантаженого файлу. Поруч розташовано кнопки для вибору файлів і надсилання зображень на аналіз, які не активні, доки зображення не вибрано. Після натискання кнопки «Аналіз зображення» поруч із зображенням відобразяться результати аналізу. Ця форма та кнопка «Зберегти результати» доступні лише для авторизованих користувачів.

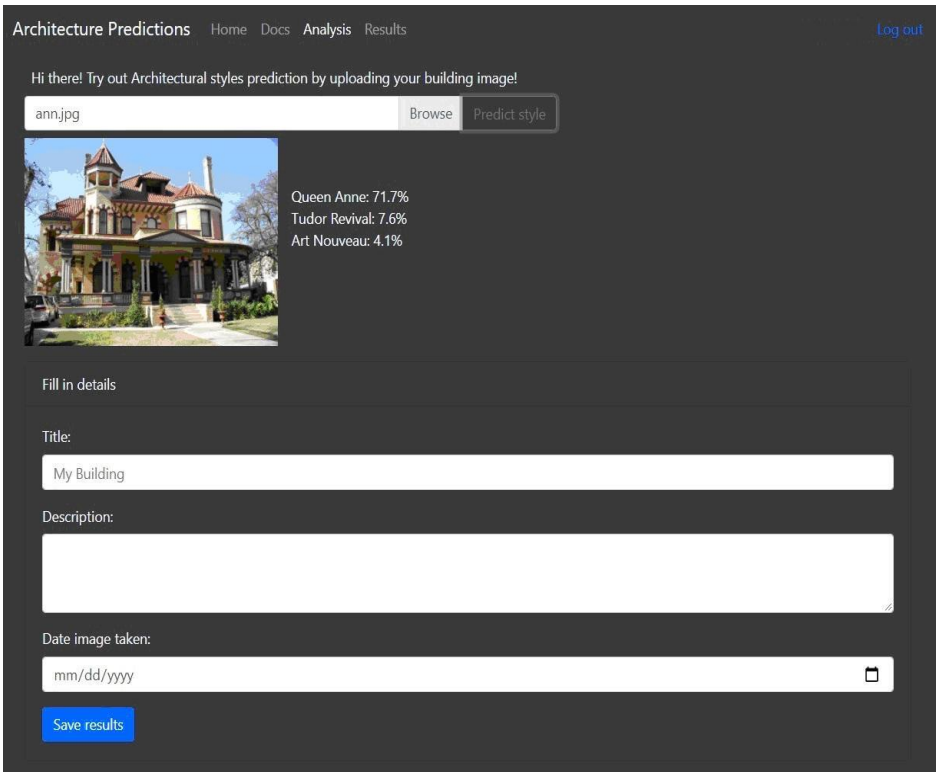


Рисунок 3.5 – Форма документації

Після аналізу та збереження результатів користувач може переглянути їх у формі Збережені результати (рис. 3.6). Результати тут відображаються у вигляді таблиці, яка підтримує сортування та видалення результатів за деякими полями.

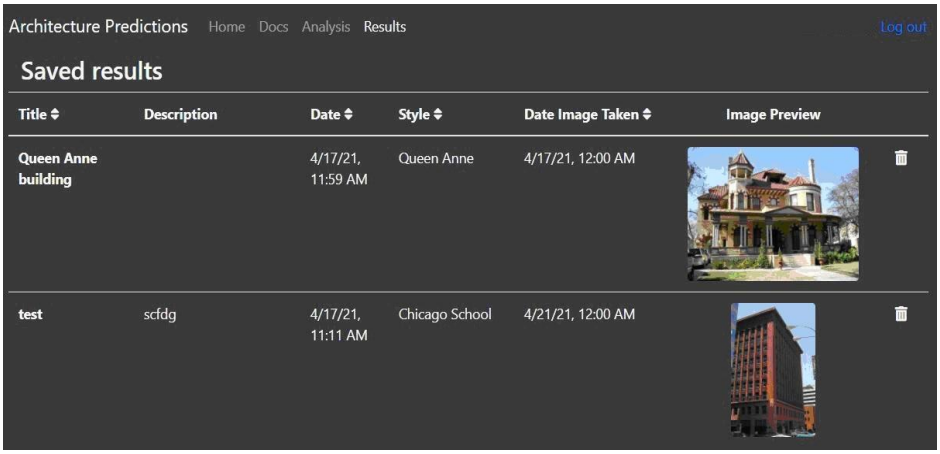


Рисунок 3.6 – Форма збережених результатів

3.5 Опис технічного забезпечення

Структурний план розміщення інформаційної системи класифікації архітектурних стилів зображення будівлі показано в Додатку Г. Схема структурного плану розміщення.

Оскільки система складається з 2 мікросервісів і програми інтерфейсу користувача, обмін повідомленнями між ними відбувається за протоколом HTTPS.

Цей сценарій готує вас до розгортання системи в середовищі Amazon Web Services за допомогою рідної технології постачальника. Amazon Web Services пропонує широкий спектр глобальних хмарних продуктів, включаючи обчислення, сховище, бази даних, аналітику, мережі, мобільні пристрої, інструменти розробника, керування, IoT, безпеку та корпоративні програми. Ці послуги допоможуть проектам розвиватися швидше, скоротять IT-витрати та масштабуються. AWS довіряють найбільші підприємства та найпопулярніші стартапи для забезпечення різноманітних робочих навантажень, зокрема: веб- та мобільних додатків, розробки ігор, обробки та зберігання даних, зберігання, архівування тощо. [50]

Система складається з 4 компонентів: компонент аналізу зображення за допомогою навченої нейронної мережі; компонент зберігання результатів; компонент автентифікації; і компонент інтерфейсу користувача.

Як система доменних імен Amazon Route 53 використовується для перенаправлення кінцевих користувачів до Інтернет-додатків шляхом перетворення доменних імен (наприклад, www.architecture-recognition.com) у числовий формат IP-адреси (наприклад, 192.0.2.1). Служба Amazon Route 53 спрямовує запити користувачів до інфраструктури AWS, наприклад екземплярів Amazon EC2, Elastic Load Balancing або сегментів Amazon S3. У нашому випадку запит клієнта буде перенаправлено до Amazon CloudFront – служби глобальної мережі доставки контенту (CDN), яка забезпечує швидку

та безпечну доставку даних, відео, програм і API клієнтам із низькою затримкою та високою швидкістю.

Залежно від запиту CloudFront перенаправляє його в сховище статичних файлів Amazon S3 або Elastic Load Balancing, де зберігаються файли інтерфейсу користувача.

Еластична балансування навантаження автоматично розподіляє вхідний трафік програми між кількома цілями (контейнери AWS Fargate). Кожен контейнер є одним із компонентів: компонент аналізу зображення за допомогою навченої нейронної мережі; компонент, який зберігає результати; і компонент автентифікації. AWS Fargate — це безсерверний обчислювальний механізм для контейнерів. Fargate позбавляє від необхідності налаштовувати та керувати серверами, дозволяє визначати та оплачувати ресурси для кожної програми, а також покращує безпеку шляхом розробки ізольованих програм. Fargate виділяє потрібний обсяг обчислень, усуваючи необхідність вибору екземплярів і масштабування потужності кластера. Fargate запускає кожне завдання або підсистему у власному ядрі, надаючи завданням і підсистемам власне ізольоване обчислювальне середовище. Це дає змогу вашій програмі досягти ізоляції робочого навантаження та вдосконалення дизайну в кожному конкретному випадку.

Щоб зберігати дані про користувачів і зберігати результати, ми будемо використовувати Amazon Aurora Serverless, яка є службою реляційної бази даних, Він підтримує конфігурацію автоматичного масштабування. Він забезпечує автоматичне підключення, відключення та розширення ресурсів залежно від потреб програми. Для зберігання зображень будівель ми будемо використовувати Amazon S3 – дешеве та надійне сховище файлів.

База даних і сегмент Amazon S3, які використовуються для зберігання зображень будівель, мають бути розгорнуті в приватній підмережі, щоб запобігти неавторизованим клієнтам отримати доступ до даних користувача.

У цьому розділі розглядаються технічні аспекти інформаційної системи класифікації зображень будівель за архітектурним стилем.

Описує технології, обрані для створення та розгортання інформаційних систем – Python Flask Microframework, .NET Core Web API, СУБД Ms SQL та фреймворк для створення SPA – Angular. Обґрунтовано вибір технології та наведено переваги її використання. Він також описує вимоги до обслуговування системи, функціональні вимоги та вимоги до надійності.

За допомогою діаграми послідовності показані основні способи використання користувачем системи - авторизація користувача, аналіз зображення і його зображення.

Архітектура системи описана у вигляді діаграм класів, діаграм компонентів і діаграм розгортання з детальним описом які розміщені в додатку Г.

4. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Бенчмарк

Щоб перевірити ефективність вдосконаленого алгоритму, ми беремо на прикладі завдання визначення архітектурних стилів будівлі, щоб порівняти час і результати навчання нейронної мережі. Для порівняння рекомендується використовувати вдосконалені алгоритми Adagrad і RMSprop на архітектурах нейронної мережі ResNet і DenseNet[47]. Для навчання була сформована навчальна вибірка з 1000 зображень і тестова вибірка з 500 зображень, які містили зображення будівель, класифікованих на 6 архітектурних стилів.

Навчання проводилося в ідентичних умовах - апаратне та програмне забезпечення не змінювалися, інші операції не виконувалися паралельно.

Основними критеріями порівняння є час навчання та значення функції втрат під час навчання. На малюнку 4.1 показано зміни у функції втрати навчання. Покращений алгоритм працює краще, ніж алгоритми Adagrad і RMSprop на архітектурах ResNet і DenseNet. Після навчання функція втрат моделі, навченої за вдосконаленим алгоритмом, у 1,5 рази менша, ніж інші моделі.

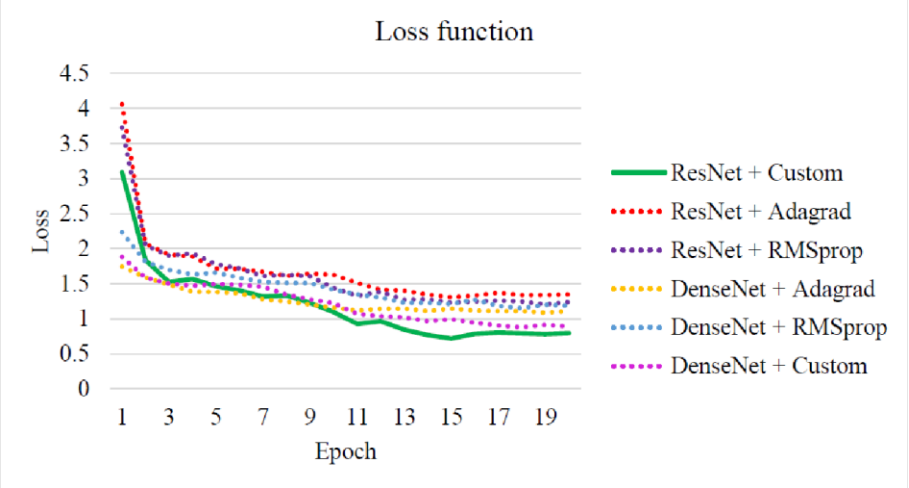


Рисунок 4.1 – Графік залежності функції втрат від епохи

На малюнку 4.2 показано час (у секундах), необхідний для навчання моделі протягом 20 епох. За допомогою покращеного алгоритму навчання моделі завершується в 1,13 рази швидше, ніж за допомогою інших алгоритмів. [53].

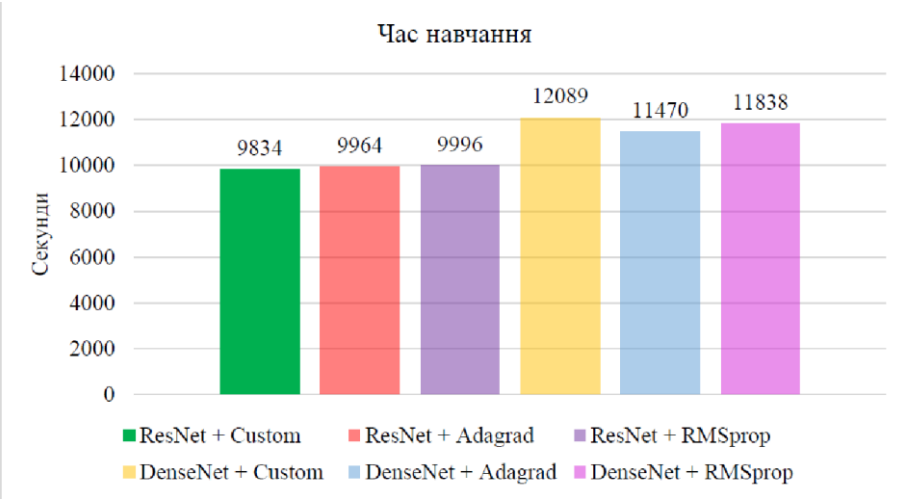


Рисунок 4.2 – Час затрачений на навчання мережі

4.2 Порівняння результатів аналізу зображень

Ми використовуємо 12 контрольних зображень будівель 6 архітектурних стилів, щоб порівняти точність класифікації навчених алгоритмів[31].

Результати наводяться як ймовірність того, що зображення має певний архітектурний стиль, який показує, наскільки навчена мережа впевнено класифікує його.

Правильний результат 1, оскільки зображення на 100% відповідає вказаному архітектурному стилю. Таблиця результатів враховує значення ймовірності архітектурного стилю, до якого належить зображення, і не враховує інші архітектурні стилі.

Імовірність правильної класифікації розраховується як середнє значення всіх проведених експериментів і показує, наскільки навчена мережа впевнено класифікує зображення.

Як видно, ймовірність дуже низька, що пов'язано з вибором кількості епох для навчання. Для досягнення більш високої точності потрібна більша кількість навчальних ітерацій.

Таблиця 4.1. Порівняння результатів аналізу

Зображення	**Правильний результат**	**ResNet + Adagr ad**	**ResNet + Custo m**	**ResNet + RMSprop**	**DenseNet + Adagra d**	**DenseNet + Custom**	**DenseNet + RMSprop**
Baroque 1	1	0.55	0.68	0.63	0.53	0.64	0.61
Baroque 2	1	0.44	0.57	0.52	0.42	0.53	0.5
Chicago School	1	0.49	0.62	0.57	0.47	0.58	0.34
Chicago School	1	0.51	0.61	0.56	0.49	0.57	0.54
Deconstructivism 5	1	0.49	0.61	0.56	0.47	0.57	0.54
Deconstructivism 6	1	0.49	0.62	0.57	0.47	0.58	0.45
Gothic 7	1	0.46	0.59	0.34	0.46	0.42	0.52
Gothic 8	1	0.5	0.63	0.58	0.48	0.59	0.58
Novelty 9	1	0.54	0.67	0.62	0.52	0.63	0.6
Novelty 10	1	0.52	0.65	0.6	0.5	0.58	0.55
Romanesque 11	1	0.36	0.49	0.44	0.34	0.45	0.42
Romanesque 12	1	0.39	0.52	0.47	0.37	0.48	0.45
Ймовірність правильної класифікації		**0.48**	**0.61**	**0.54**	**0.46**	**0.55**	**0.51**

У цьому розділі розглядається порівняння розробленого алгоритму навчання нейронної мережі з існуючими алгоритмами на прикладі завдання класифікації зображень будівель за архітектурним стилем.

Запропонований метод навчання нейронної мережі для класифікації архітектурних стилів будівель на основі зображень будівель досягає вищої точності з меншим часом навчання. У середньому запропонований алгоритм

має в 1,5 рази менші значення функції втрат наприкінці навчання і в 1,13 рази швидше за інші алгоритми.

Імовірність правильної класифікації також вища, ніж в інших мережах, що означає вищу точність класифікації за той самий час навчання.

5. ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Автоматизована система визначення архітектурної стилістики будівельних споруд» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями **[Козловський, Лесько, Кавецький]**.

Таблиця 5.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	3	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	2	2	2
4. Ринкові переваги (технічні властивості)	2	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	2	3	2

10. Практична здійсненність (необхідність нових матеріалів)	2	2	2
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	32	34	34
Середньоарифметична сума балів $СБ_с$	33,3		

За результатами розрахунків, наведених в таблиці 5.1, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в **[Козловський, Лесько, Кавецький]**.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Автоматизована система визначення архітектурної стилістики будівельних споруд» становить 33,3 бала, що, відповідно до **[Козловський, Лесько, Кавецький]**, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок узагальненого коефіцієнта якості розробки

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою **[Кавецький практикум 2016]**:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \qquad (5.1)$$

де, k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *і-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *і-го* технічного показника якості нової розробки.

Результати порівняння зведемо до таблиці 5.2.

Таблиця 5.2 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований продукт	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість розпізнаваних архітектурних стилів	од	5	25	5	0,3
Точність розпізнавання	%	0,9	0,97	1,08	0,2
Потреба в попередній обробці зображення	бал	4	8	2	0,15
Швидкість аналізу	бал	5	9	1,8	0,25
Універсальність методу	бал	6	9,5	1,58	0,1

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 5 \cdot 0,3 + 1,08 \cdot 0,2 + 2 \cdot 0,15 + 1,8 \cdot 0,25 + 1,58 \cdot 0,1 = 2,62.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,62 рази.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Автоматизована система визначення архітектурної стилістики будівельних споруд», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою **[Козловський, Лесько, Кавецький]**:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.2)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 18500{,}00 \cdot 56 / 22 = 47090{,}91 \ грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.3 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	18500,00	840,91	56	47090,91
Інженер-програміст 1-ї категорії	18000,00	818,18	54	44181,82
Консультант (провідний фахівець галузі архітектури)	18000,00	818,18	5	4090,91
Технік	7000,00	318,18	44	14000,00
Всього				109363,64

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Автоматизована система визначення архітектурної стилістики будівельних споруд» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.3)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.4)$$

де M_M – розмір мінімальної місячної заробітної плати, приймемо *Мм*=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення (табл. Б.2, додаток Б) **[Козловський, Лесько, Кавецький]**;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок;

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}35 / (22 \cdot 8) = 56{,}53$ *грн.*
$З_{р1} = 56{,}53 \cdot 4{,}00 = 226{,}13$ *грн.*

Таблиця 5.4 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка обчислювального обладнання для проведення досліджень	4,00	2	1,10	56,53	226,13
Підготовка робочого місця розробника програмного забезпечення	5,00	3	1,35	69,38	346,90
Встановлення програмного забезпечення розробки програмних засобів оцінки параметрів	4,00	4	1,50	77,09	308,35

Підготовка дослідних баз даних архітектурних зображень	14,00	3	1,35	69,38	971,31
Ведення кодів базових модулів аналізу параметрів	4,50	5	1,70	87,37	393,15
Компіляція системних блоків	3,00	5	1,70	87,37	262,10
Налагодження системи	4,00	6	2,00	102,78	411,14
Всього					2919,07

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (109363,64 + 2919,07) · 11 / 100% = 12351,10 грн.

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (109363,64 + 2919,07 + 12351,10) · 22 / 100% = 27419,44 грн.

5.3.3 Сировина та матеріали

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (5.7)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, ($K_j = 1,1 \ldots 1,15$);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$M_1 = 1 \cdot 259,00 \cdot 1,06 - 0 \cdot 0 = 274,54$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
USB флеш накопичувач Transcend 64Gb JetFlash 700 (TS64GJF700)	259,00	1	0	0	274,54
Прибор настільний 13 предметів 6300-01 Buromax чорний	240,00	2	0	0	508,80
Папір офісний Офіс Центр А5 80г/м2 500 аркушів клас С	112,00	3	0	0	356,16
Папір А4 500 аркушів клас-С Crystal Print&Copy UPM	188,00	3	0	0	597,84

Набір канцелярський офісний FAX	207,00	2	0	0	438,84
Картридж для принтера	2032,00	1	0	0	2153,92
Диск оптичний CD-RW	27,00	5	0	0	143,10
Всього					4473,20

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Автоматизована система визначення архітектурної стилістики будівельних споруд», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (5.8)$$

де $Н_j$ – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*).

$К_в = 10 \cdot 22{,}00 \cdot 1{,}05 = 231{,}00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Кабель Ethernet cat.6a(1 Gb\s)	10	22,00	231,00
Конектори RG-45	5	12,00	63,00
Процессор для сервера (Intel XEON Ten Core E5-2470 V2 2.40 GHz)	1	2118,00	2223,90
Зовнішня пам’ять для сервера (2 шт Жорский диск Seagate IronWolf 10 TB)	2	10782,00	22642,20
Всього			25160,10

5.3.5 Спецустаткування для наукових (експериментальних) робіт

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \quad (5.9)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ = *6699,00 · 1 · 1,04 = 6966,96 грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Смартфон Xiaomi Redmi 12 8/256Gb Midnight Black	1	6699,00	6966,96
Сервер для зберігання БД (Сервер DELL R420 (4x3.5) LFF	1	28899,00	30054,96
Всього			37021,92

5.3.6 Програмне забезпечення для наукових (експериментальних) робіт

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \quad (5.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *8610,00 · 1 · 1,01 = 8696,10 грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Підписка AWS (Amazon Route 53, Amazon Simple Storage Service (S3)	1	8610,00	8696,10
ОС Windows	1	4299,00	4341,99
Прикладний пакет Microsoft Office	1	5240,00	5292,40
Підписка AWS(Amazon Aurora Serverless)	1	7800,00	7878,00
Програмний засіб розробки IDE – PyCharm	1	4500,00	4545,00
Програмний продукт Microsoft SQL Server 2022	1	3780,00	3817,80
Всього			34571,29

5.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (25125{,}00 \cdot 3) / (2 \cdot 12) = 3140{,}63$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер	25125,00	2	3	3140,63
Обчислювально-графічна система програмної розробки	38599,00	2	3	4824,88
Робоче місце розробника програмного забезпечення	9580,00	5	3	479,00
Пристрій виводу інформації HP-5500	8765,00	4	3	547,81
Оргтехніка	8925,00	4	3	557,81
Приміщення лабораторії	255500,00	25	3	2555,00
Мультимедійний проектор Epson 1800A	18400,00	5	3	920,00
Всього				13025,13

5.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (5.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; приймемо Ц$_e$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,20 \cdot 400,0 \cdot 7,50 \cdot 0,95 / 0,97 = 600,00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер	0,20	400,0	600,00
Обчислювально-графічна система програмної розробки	0,40	400,0	1200,00
Робоче місце розробника програмного забезпечення	0,07	400,0	210,00
Пристрій виводу інформації HP-5500	0,22	4,0	6,60
Оргтехніка	0,45	2,0	6,75
Мультимедійний проектор Epson 1800A	0,22	10,0	16,50
Сервер для зберігання БД (Сервер DELL R420 (4x3.5) LFF	0,10	400,0	300,00
Всього			2339,85

5.3.9 Службові відрядження

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.13)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (109363,64 + 2919,07) · 20 / 100% = 22456,54 грн.

5.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.14)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (109363,64 + 2919,07) · 30 / 100% = 33684,81 грн.

5.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (109363,64 + 2919,07) · 50 / 100% = 56141,35 грн.

5.3.12 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (109363,64 + 2919,07) · 100 / 100% = 112282,70 грн.

Витрати на проведення науково-дослідної роботи на тему «Автоматизована система визначення архітектурної стилістики будівельних споруд» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.17)$$

$B_{заг}$= *109363,64 +2919,07 +12351,10 +27419,44 +4473,20 +25160,10 +37021,92 + 34571,29 + 13025,13 +2339,85 +22456,54 +33684,81 +56141,35 +112282,70 = 493210,13 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 493210,13 / 0,95 = 519168,56 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

Результати дослідження проведені за темою «Автоматизована система визначення архітектурної стилістики будівельних споруд» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	3000	4000	4000	2000

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 21000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 550,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 243,20 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [Козловський, Лесько, Кавецький]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.19)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(243,20·21000,00+793,20·3000)·0,83·0,4·(1-0,18/100%)=2038206,43 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(243,20·21000,00+793,20·7000)·0,83·0,4·(1-0,18/100%)=2901969,50 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(243,20·21000,00+793,20·11000)·0,83·0,4·(1-0,18/100%)=3765732,58 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(243,20·21000,00+793,20·13000)·0,83·0,4·(1-0,18/100%)=4197614,11 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.20)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,12;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП = 2038206,43/(1+0,12)^1+2901969,50/(1+0,12)^2+3765732,58/(1+0,12)^3+$
$+4197614,11/(1+0,12)^4=1819827,17+2313432,32+2680374,07+2667659,65=9481$
293,21 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{5.21}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 519168,56 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 519168,56 = 1038337,12$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.22}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 9481293,21 грн;

PV – теперішня вартість початкових інвестицій, 1038337,12 грн.

$E_{абс} = ПП - PV = 9481293,21 - 1038337,12 = 8442956,09$ грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.23)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 8442956,09 грн;

PV – теперішня вартість початкових інвестицій, 1038337,12 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+8442956,09/1038337,12)$^{1/4}$= 0,74.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.24)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,12;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,3.

$\tau_{мін}$ = 0,12+0,3 = 0,42 < 0,74 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Автоматизована система визначення архітектурної стилістики будівельних споруд» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.25}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,74 = 1,35 р.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Автоматизована система визначення архітектурної стилістики будівельних споруд» становить 33,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,62 рази.

Також термін окупності становить 1,35 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Автоматизована система визначення архітектурної стилістики будівельних споруд».

ВИСНОВКИ

Тому в цій роботі розглядається використання нейронних мереж для вирішення проблеми класифікації зображень, яка відрізняється від інших проблем класифікації тим, що існують міжкласові зв’язки між різними стилями. і шляхи підвищення точності результатів і скорочення часу, необхідного для навчання алгоритмів. Актуальність розробки даного алгоритму полягає в тому, що він дозволить класифікувати зображення з тісними міжкласовими зв’язками не тільки для класифікації архітектурних стилів будівель, але й для зображень з інших областей. Наукова новизна полягає в удосконаленому алгоритмі налаштування параметрів нейронної мережі під час навчання для вирішення задачі класифікації зображень із тісними міжкласовими зв’язками. Практичне значення отриманих результатів полягає у створенні системи ідентифікації архітектурного стилю будівель та методу навчання нейронних мереж, що забезпечує більшу точність за менший час навчання.

У ході роботи проаналізовано та порівняно існуючі алгоритми вирішення задачі ідентифікації архітектурного стилю будівлі, описано їх характеристики та встановлено, що на даний момент системи з подібними функціями не існує. Дослідження показало, що методи, засновані на глибокому навчанні, значно перевищили результати, досягнуті всіма іншими методами.

У розділі Розробка алгоритмів класифікації зображень описано математичну постановку задачі, цільову функцію, алгоритм навчання мережі та схему класифікації зображень з її допомогою. Розглядаються загальні поняття та методи, які використовуються під час розробки алгоритму класифікації, навчання нейронної мережі та налаштування параметрів. У цьому розділі описано методи та процедури виділення кордонів і кордонів об’єктів на зображеннях, а також методи зменшення зображень. Одночасно перевіряється правильність алгоритму навчання нейронної мережі.

В описі розділу програмно-технічного забезпечення розглянуто технічні аспекти інформаційної системи класифікації зображень будівель за архітектурним стилем. Описує обрані технології для створення та розгортання інформації, обґрунтовує їх вибір та наводить переваги їх використання – Python Flask Microframework, .NET Core Web API, СУБД Ms SQL та фреймворки для створення SPA – Angular. Також описані вимоги до технічної підтримки системи, функціональні вимоги та вимоги до надійності. За допомогою діаграми послідовності показані основні способи використання користувачем системи - авторизація користувача, аналіз зображення і його зображення.

У частині аналізу отриманих результатів вдосконалений алгоритм навчання нейронної мережі порівнюється з існуючим алгоритмом на прикладі завдання класифікації побудови зображень різних архітектурних стилів. Запропонований метод навчання нейронної мережі для класифікації архітектурних стилів будівель на основі зображень будівель досягає вищої точності з меншим часом навчання. У середньому запропонований алгоритм має в 1,5 рази менші значення функції втрат наприкінці навчання і в 1,13 рази швидше за інші алгоритми. Імовірність правильної класифікації також вища, ніж в інших мережах, що означає вищу точність класифікації під час навчання.

У розділі економічного обгрунтування згідно проведених досліджень рівень комерційного потенціалу розробки свідчить про комерційну важливість проведення даних досліджень При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,62 рази.

Також термін окупності становить 1,35 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

ПЕРЕЛІК ПОСИЛАНЬ

1. Berg, A.C. Parsing images of architectural scenes / Berg, A.C, Grabler, F, Malik, J // IEEE 11 th International Conference on Computer Vision, / Berg, A.C, Grabler, F, Malik, J., 2007. – p. 2–8.

2. Wei-Ta Chu. ICMR '12: Proceedings of the 2nd ACM International Conference on Multimedia Retrieval. / Wei-Ta Chu, Ming-Hung Tsai // 27. – 2012. – URL: https://doi.org/10.1145/2324796.2324831.

3. What makes paris look like paris? / Doersch C, Singh S, Gupta A, Sivic J. // ACM Transactions on Graphics (TOG). – 2012. – №31.

4. Goel A. Are buildings only instances?: exploration in architectural style categories/ Goel A, Juneja M, Jawahar C, 2012. – (Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing). – p. 1–12.

5. Object retrieval with large vocabularies and fast spatial matching / Philbin J, Chum O, Isard M, Sivic J // Computer Vision and Pattern Recognition / Philbin J, Chum O, Isard M, Sivic J., 2007. – p. 1–8.

6. Architectural Style Classification using Multinomial Latent Logistic Regression /

Z. Xu, D. Tao, Y. Zhang, J. Wu. // European Conference on Computer Vision (ECCV2014),. – 2014.

7. AUTOMATIC ARCHITECTURAL STYLE RECOGNITION / M. Mathias, A. Martinovic, J. Weissenberg, S. Haegler. // PSI/VISICS, Department of Electrical Engineering. – 2011. – p. 1–6.

8. Shalunts G. Architectural Style Classification of Building Facade Windows / G. Shalunts, Y. Haxhimusa, R. Sablatnig. // Vienna University of Technology, Institute of Computer Aided Automation Computer Vision Lab, Institute of Computer Graphics and Algorithms, Pattern Recongition and Image Processing Lab. – 2011.

9. Classification of Architectural Heritage Images Using Deep Learning Techniques/ J.Llamas, P. Lerones, R. Medina, E. Zalama. // Applied Sciences. – 2017. – №992.– p. 1–25.

10. 3D Façade Labeling over Complex Scenarios: A Case Study Using Convolutional Neural Network and Structure-From-Motion / Rodolfo Georjute Lotte, Norbert Haala, Mateusz Karpina, Luiz Eduardo Oliveira e Cruz de Aragão. // Remote Sensing. – 2018. – №1435. – p. 1–28.

11. Новіченко Н. В. Порівняння алгоритмів класифікації зображень / Неля Валеріївна Новіченко. // Матеріали V всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського» – 2020. – С. 119–123.

12. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. Nature 2015, 521, 436–444.URL: https://pubmed.ncbi.nlm.nih.gov/26017442/.

13. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems; MIT Press: Cambridge, MA, USA, 2012; p. 1097–1105.

14. Simonyan K. Very Deep Convolutional Networks for Large-Scale Image Recognition / K. Simonyan, A. Zisserman. // Computer Vision and Pattern Recognition. – 2015. – №6.

15. Dive into Deep Learning / Zhang A., Lipton Z. C., Mu Li, Smola A. J.. – 2020. –№15. – p. 263–296.

16. Densely Connected Convolutional Networks / Gao Huang, Zhuang Liu, Kilian Q. Weinberger, Laurens van der Maaten. // arXiv. – 2016. – p. 1–12.

17. Going deeper with convolutions / Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet. // arXiv. – 2014. – p. 1–12.

18. Chollet F. Xception: Deep Learning With Depthwise Separable Convolutions / Francois Chollet. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – №1. – p. 1251–1258.

19. Khandelwal R. Overview of different Optimizers for neural networks [Електронний ресурс] / Renu Khandelwal // DataDrivenInvestor. – 2019. – URL: https://medium.datadriveninvestor.com/overview-of-different-optimizers-for-neural-networks-e0ed119440c3.

20. Sorokina K. Image Classification with Convolutional Neural Networks / Ksenia Sorokina // A Medium Corporation. – 2018. – URL: https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8.

21. Kinnikar A., Husain M., Meena S.M. Face Recognition Using Gabor Filter AndConvolutional Neural Network // Proceedings of the International Conferenceon Informatics and Analytics (Pondicherry, India, August 25–26, 2016), 2016.

22. Rossana M. S. Cruz. Artificial Neural Networks and Efficient Optimization Techniques for Applications in Engineering / Rossana M. S. Cruz, Helton Maia Peixoto, Rafael Magalhães // Artificial Neural Networks - Methodological Advances and Biomedical Applications / Rossana M. S. Cruz, Helton Maia Peixoto, Rafael Magalhães. – London: IntechOpen, 2011. – p. 1–26.

23. Гавриленко О. В. Розпізнавання архітектурних стилів будівель за допомогою методів машиннного навчання / О. В. Гавриленко. // Матеріали IV всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ- 2020) – 2021. – С. 120–125.

24. Python 3.7.10 documentation – 2021. [Електронний ресурс] - URL: https://docs.python.org/3.7/.

25. Shelly Singh, 15 Python Libraries for Data Science and Machine Learning. - 2023 [Електронний ресурс] - URL: https://www.projectpro.io/article/top-5-libraries-for-data-science-in-python/196#mcetoc_1fu5v5t8dl

26. NumPy Introduction. [Електронний ресурс] – URL: https://www.w3schools.com/python/numpy/numpy_intro.asp

27. Scipy 1.9.3. [Електронний ресурс] - URL: https://pypi.org/project/scipy/

28. Scikit Learn Tutorial. [Електронний ресурс] - URL:

https://www.tutorialspoint.com/scikit_learn/index.htm

29.TensorFlow. [Електронний ресурс] - URL: https://opensource.google/projects/tensorflow

30. Learning OpenCV. [Електронний ресурс] – URL: https://www.oreilly.com/library/view/learning-opencv/9780596516130/

31. The Python SQL Toolkit and Object Relational Mapper. [Електронний ресурс] - URL: https://www.sqlalchemy.org/

32. Beautiful Soup: Build a Web Scraper With Python. [Електронний ресурс] - URL: https://realpython.com/beautiful-soup-web-scraper-python/

33. Річард Столмен, Проблема ліцензій BSD. – 2022. [Електронний ресурс] - URL: https://www.gnu.org/licenses/bsd.uk.html

34. Jupyter Notebook [Електронний ресурс] - URL: https://www.onworks.net/uk/software/windows/app-jupyter-notebook

35. Jupyter Team, Що таке файл IPYNB – 2015. [Електронний ресурс] - URL: https://jupyter-notebook.readthedocs.io/en/stable/notebook.html#notebook-documents

36. Marian Dumitru, Architectural styles dataset – 2020. [Електронний ресурс] - https://www.kaggle.com/datasets/dumitrux/architectural-styles-dataset

37. A Gentle Introduction to Object Recognition With Deep Learning. [Електронний ресурс] – https://machinelearningmastery.com/object-recognition-with-deep-learning/

38. Image Classification. [Електронний ресурс] – https://huggingface.co/tasks/image-classification

39. Chen Wang. Convolutional Neural Network for Image Classification [Електронний ресурс] / Chen Wang, Yang Xi. – 2015. URL: http://www.cs.jhu.edu/~cwang107/files/cnn.pdf.

40. Шквиря О.В.. Система розпізнавання облич - м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського» – 2019. – 65с.

41. . Rifai, Y. N. Dauphin, P.Vincent, Y. Bengio, X. Muller. The manifold tangent classifier. Advances in neural information processing systems, 24, 2011.

42. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition / Girshick J et. al., CVPR: IEEE Conference, 2014. 541 p.;

43. Python documentation – 2021. [Електронний ресурс] - URL: https://docs.python.org/3.7/.

44. Törmä M., Kohonen. Self-organizing feature mapin pattern recognition, Institute of Photogrammetry and Remote Sensing Helsinki University of Technology, 2005. 14 p.;

45. Андрій Хмельницький. Перші кроки в NLP: розглядаємо Python-бібліотеку TensorFlow та нейронні мережі в реальному завданні – [Електронний ресурс] dou.ua – 2020.- URL: https://dou.ua/lenta/articles/first-steps-in-nlp-tensorflow/

46. OpenCV-Python Tutorials [Електронний ресурс] - URL: https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html

47. Плющев Д. Architecture Prediction Web API [Електронний ресурс] / Дмитро Плющев. URL: https://github.com/happybeer63/arcitecture-prediction-api/blob/master/README.md.

48. Плющев Д. Architecture Recognition Client [Електронний ресурс] / Дмитро Плющев. URL: https://github.com/happybeer63/arcitecture-recognition-client/blob/master/README.md.

49. Плющев Д. Architecture Recognition Web API [Електронний ресурс] /Дмитро Плющев. URL: https://github.com/happybeer63/arcitecture-recognition-api/blob/master/README.md.

50. Wittig A. Amazon Web Services in Action / A. Wittig, M. Wittig. – Shelter Island, NY, USA: Manning Publications Co, 2019. – p. 498 .

51. Dive into Deep Learning / Zhang A., Lipton Z. C., Mu Li, Smola A. J.. – 2020. – №15. – p. 263–296.

52. Densely Connected Convolutional Networks / Gao Huang, Zhuang Liu, Kilian Q. Weinberger, Laurens van der Maaten. // arXiv. – 2016. – p. 1–12.

53. Гавриленко О. В. Дослідження оптимізаторів нейронної мережі для класифікації зображень будівель за архітектурним стилем / О. В. Гавриленко, // Системні технології. – 2021. – №5.

54.Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

55. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

Додатки

ДОДАТОК А

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: **«Автоматизована система визначення архітектурної стилістики будівельних споруд»**

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 88,5% Схожість 11,5%

Аналіз звіту подібності (відмітити потрібне)

☑ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Дмитро ПЛЮЩЕВ
(підпис) (прізвище, ініціали)

Керівник роботи _____ Олена НИКИТЕНКО
(підпис) (прізвище, ініціали)

Додаток Б

(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО

Т.в.о. зав. кафедри КСУ ВНТУ,

д.т.н., доцент

_____Марія ЮХИМЧУК

"06" 10 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Автоматизована система визначення архітектурної стилістики будівельних споруд

08-33.МКР.12.00.000 ТЗ

Студент групи 2АКІТ-22М
Плющев Д.О.

Дмитро Плющев
Ім'я ПРІЗВИЩЕ

Підпис

Керівник Никитенко О.Д.

Олена НИКИТЕНКО
Ім'я ПРІЗВИЩЕ

Підпис

Вінниця 2023

1. Назва та галузь застосування

1.1. Назва – Автоматизована система визначення архітектурної стилістики будівельних споруд.

1.2. Галузь застосування – архітектурна індустрія та будівельна сфера

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від №247 від 18-09-2023р.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є удосконалення алгоритмів класифікації зображень з тісними міжкласовими взаємозв’язками, зменшення часу навчання нейронної мережі при збереженні точності навчання.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Новіченко Н. В. Порівняння алгоритмів класифікації зоображень / Неля Валеріївна Новіченко. // Матеріали V всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського» – 2020. – С. 119–123.

2. Rossana M. S. Cruz. Artificial Neural Networks and Efficient Optimization Techniques for Applications in Engineering / Rossana M. S. Cruz, Helton Maia Peixoto, Rafael Magalhães // Artificial Neural Networks - Methodological Advances and Biomedical Applications / Rossana M. S. Cruz, Helton Maia Peixoto, Rafael Magalhães. – London: IntechOpen, 2011. – p. 1–26

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- збір інформації для формування навчального датасету ;
- оптимізація алгоритму навчання;
- керування базою даних;
- формування звітів.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10;
- Хмарне середовище Amazon Web Services
- PyCharm
- Python Flask Microframework
- Microsoft SQL Server

5.2.2. Умови експлуатації системи:

- Робота на пристроях з підключенням до мережі інтернет;
- можливість цілодобового функціонування системи;
- дані оновлюються і є актуальними.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі автоматизації процесами в об’єкті управління відповідно до теми кваліфікаційної роботи. Постановка задач дослідження «03»_09__ 2023 р.
2. Удосконалення технології прийняття рішень при автоматизації об’єкту управління « 02 » 10 2023 р.
3. Визначення технічних характеристик системи « 14 » 10 2023 р.
4. Розробка програмного забезпечення системи « 05 » 11 2023 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи « 05 » 11 2023 р.
2. Розробка моделі бази даних системи « 10 » 11 2023 р.
3. Тестування програмного забезпечення « 21 » 11 2023 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до « 28 » 11 2023 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «11» 12 2023р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до « 20 » 12 2023 р.

Додаток В

Лістинг основного модуля (main)

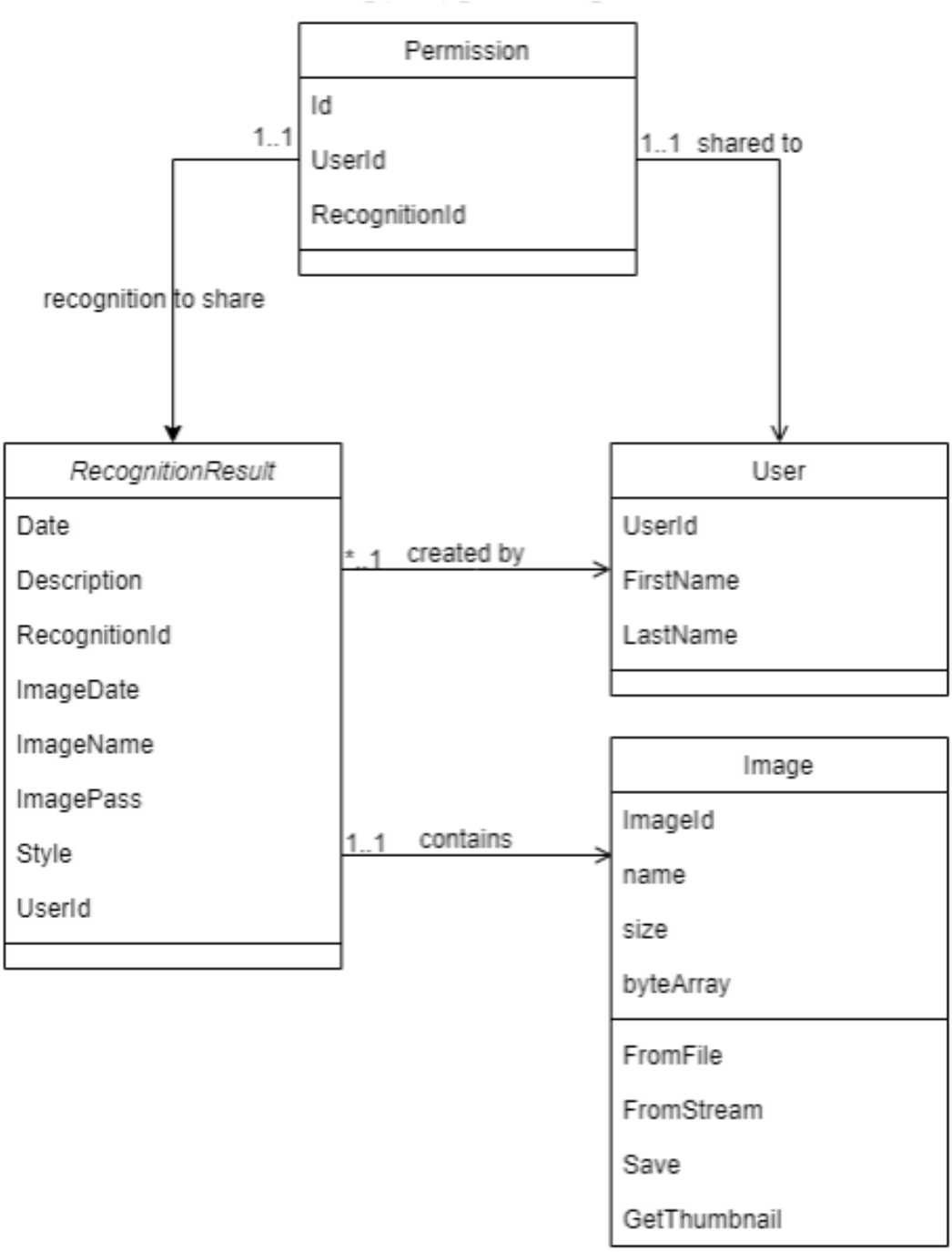
```
# Importing all necessary libraries
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K

img_width, img_height = 224, 224
train_data_dir = 'v_data/train'
validation_data_dir = 'v_data/test'
nb_train_samples =400
nb_validation_samples = 100
epochs = 10
batch_size = 16
if K.image_data_format() == 'channels_first':
        input_shape = (3, img_width, img_height)
else:
        input_shape = (img_width, img_height, 3)
model = Sequential()
model.add(Conv2D(32, (2, 2), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(64, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy',
                        optimizer='rmsprop',
                        metrics=['accuracy'])
train_datagen = ImageDataGenerator(
        rescale=1. / 255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='binary')
```

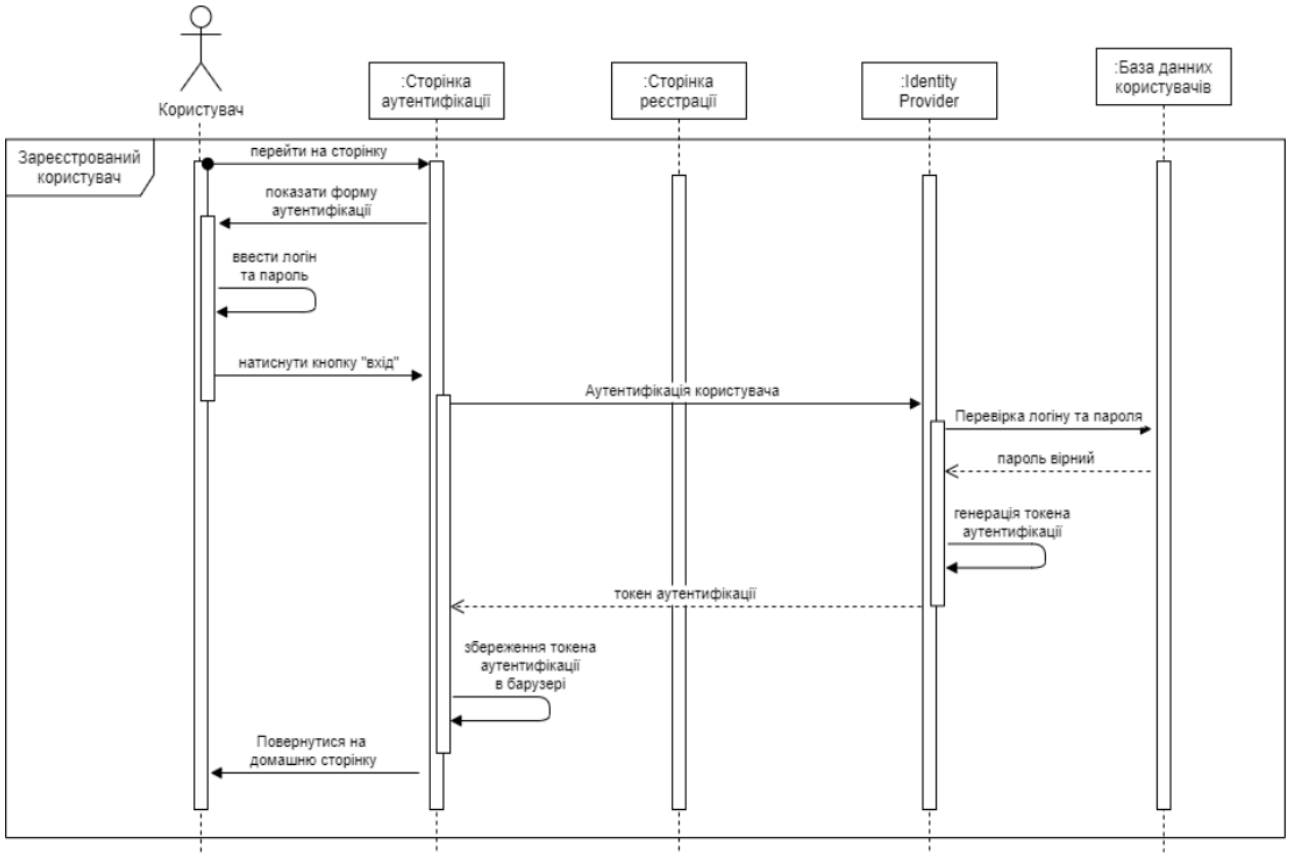
```
model.fit_generator(
        train_generator,
        steps_per_epoch=nb_train_samples // batch_size,
        epochs=epochs,
        validation_data=validation_generator,
        validation_steps=nb_validation_samples // batch_size)
model.save_weights('model_saved.h5')
from keras.models import load_model
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16
import numpy as np

from keras.models import load_model

model = load_model('model_saved.h5')

image = load_img('v_data/test/planes/5.jpg', target_size=(224, 224))
img = np.array(image)
img = img / 255.0
img = img.reshape(1,224,224,3)
label = model.predict(img)
print("Predicted Class (0 - Cars , 1- Planes): ", label[0][0])
```

Додаток Г

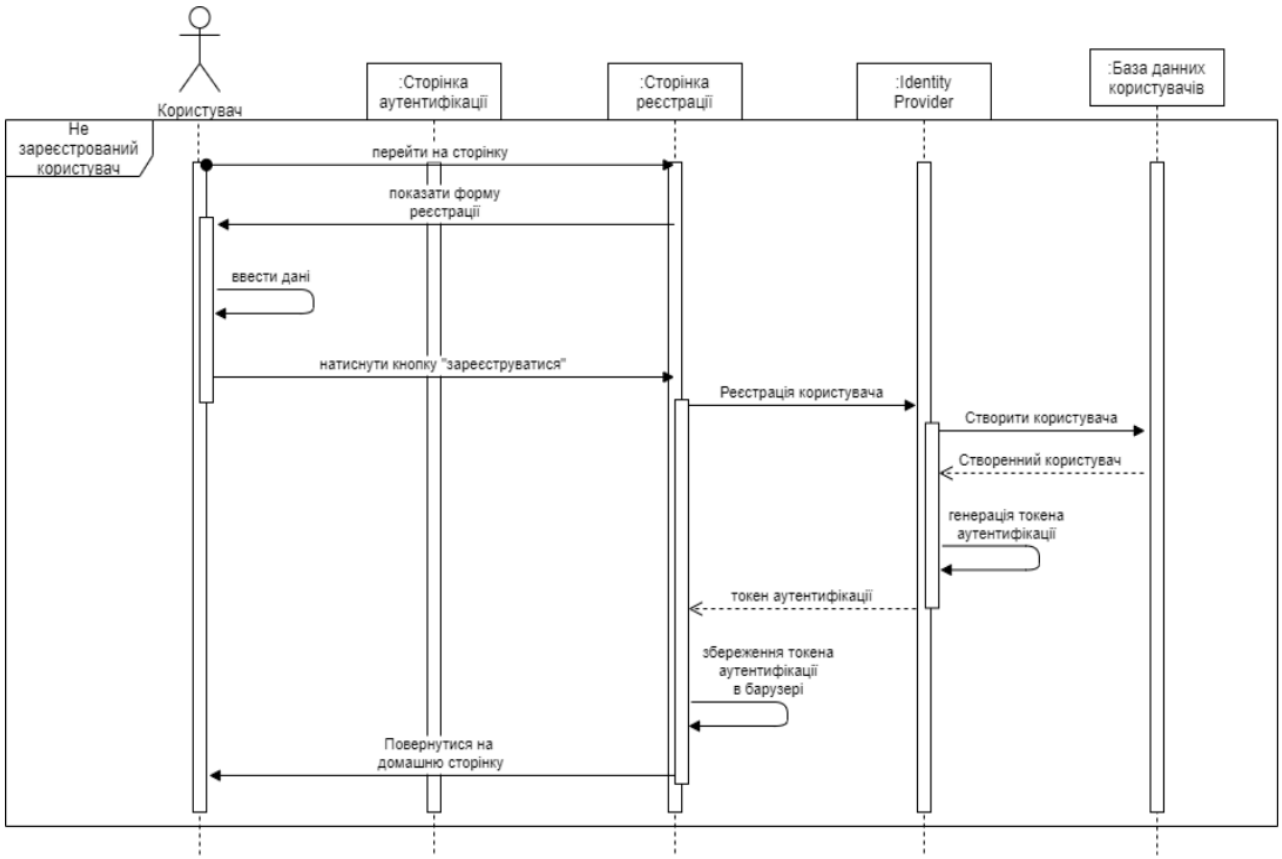
Структурна схема діаграми класів



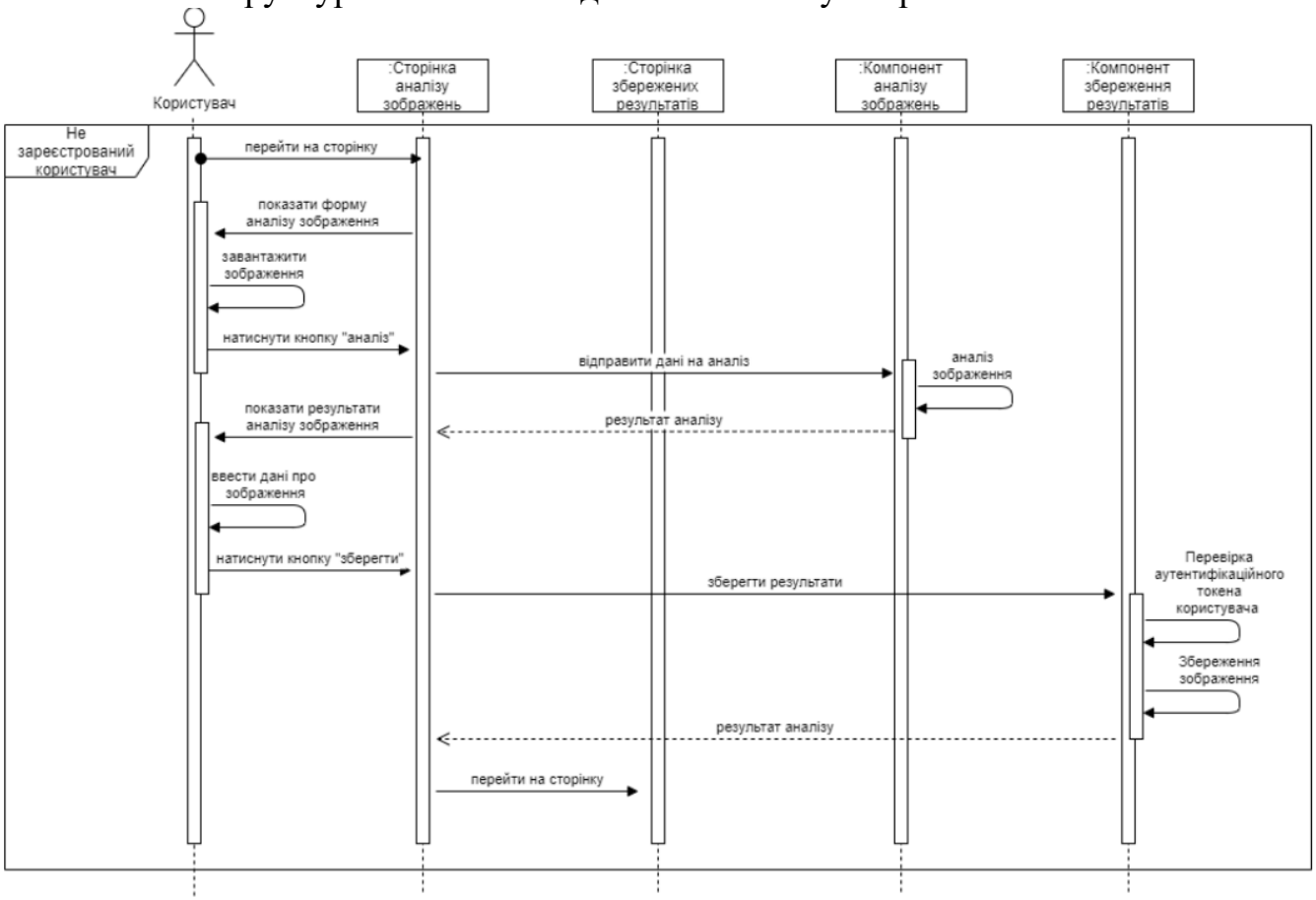
Структурна схема послідовності аутентифікації користувача



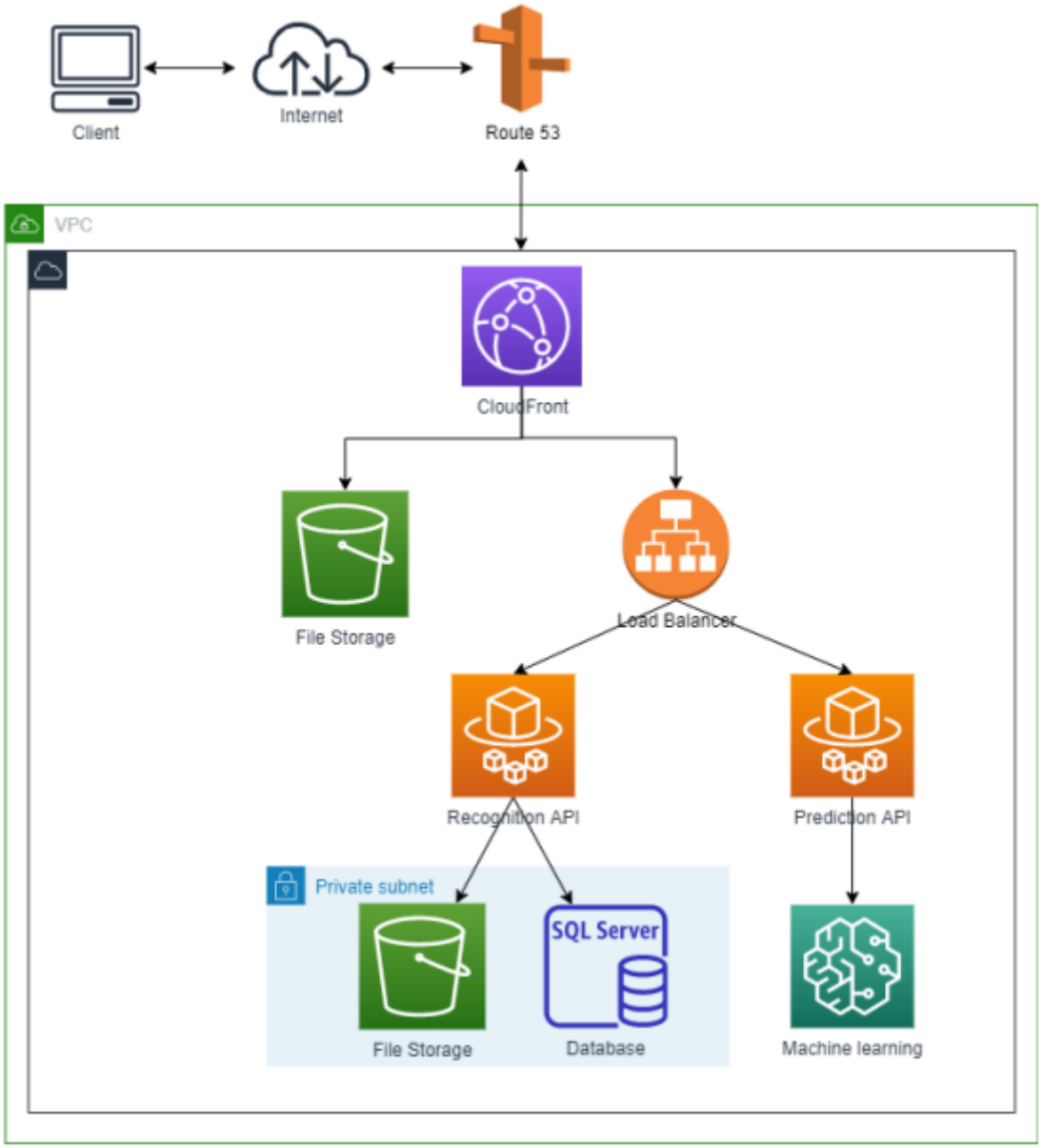
Структурна схема послідовності реєстрації користувача



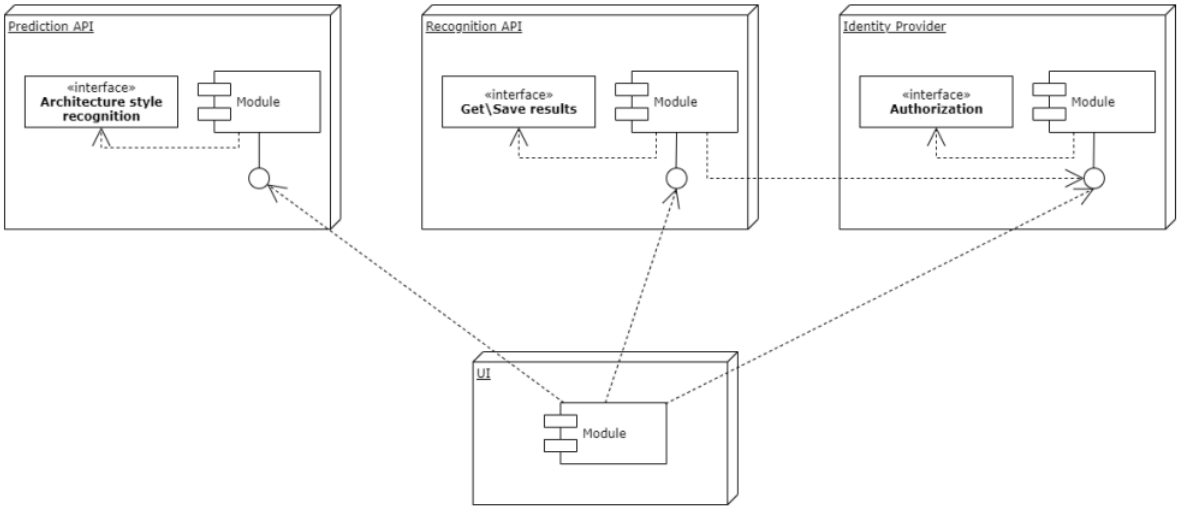
Структурна схема послідовності аналізу зображення



Структурна схема розгортання



Структурна схема компонентів



Додаток Д

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Автоматизована система визначення архітектурної стилістики будівельних споруд

1. Об'єкт, предмет та мета дослідження;
2. Задачі дослідження;
3. Алгоритм навчання
4. Діаграма класів
5. Діаграма послідовностей
6. Веб інтерфейс
7. Схема розгортання сервісу використовуючи хмарні технології
8. Порівняння результатів аналізу
8. Проведення економічного аудиту,
9. Висновки

Студент групи 2АКІТ-22м

Дмитро ПЛЮЩЕВ

Підпис *Ім'я ПРІЗВИЩЕ*

Керівник к.т.н.,доцент кафедри КСУ

Олена НИКИТЕНКО

Підпис *Ім'я ПРІЗВИЩЕ*

Автоматизована система визначення архітектурної стилістики будівельних споруд

ВИКОНАВ СТ ГР 2АКІТ-22М ПЛЮЩЕВ ДМИТРО
КЕРІВНИК К.Н.Т., ДОЦЕНТ КАФ КСУ НИКИТЕНКО ОЛЕНА

1.Об'єкт, предмет та мета дослідження

Метою дослідження є покращення алгоритмів класифікації зображень із тісними зв'язками між класами та скорочення часу навчання нейронної мережі, зберігаючи при цьому точність навчання.

Об'єктом дослідження є процес навчання нейронних мереж для класифікації зображень з тісними міжкласовими зв'язками та для класифікації зображень будівель за архітектурним стилем.

Тема дослідження – оптимізаційні алгоритми та методи зміни властивостей нейронних мереж під час їх навчання, таких як ваги та швидкості навчання.

Наукова новизна отриманих результатів полягає в удосконаленні алгоритму коригування параметрів нейронної мережі під час навчання для вирішення задачі класифікації зображень з тісними зв'язками між класами

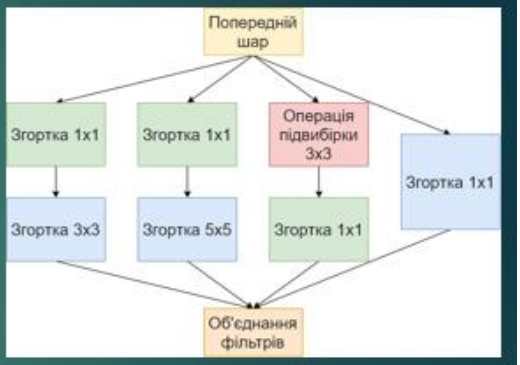
2.Задачі дослідження

Для досягнення мети необхідно виконати наступні завдання:

- Проаналізувати існуючі методи класифікації зображень;
- Модифікувати існуючі методи для підвищення точності та скорочення часу навчання;
- Створювати підібрані зображення будівель;
- програмне впровадження та розробка програмних алгоритмів;
- Порівняти ефективність модифікованого алгоритму з існуючими алгоритмами при вирішенні задачі визначення архітектурного стилю будівлі.

3.Алгоритм навчання

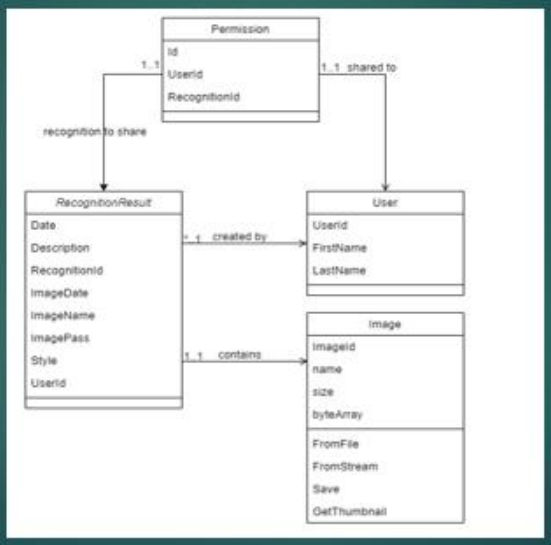
У ResNet вихідні дані групи згорткових шарів додаються до вхідного рівня, тоді як у DenseNet для кожного поточного шару всі інші шари перед ним підключаються до поточного шару. Завдяки цьому можна зменшити кількість фільтрів, чим мінімізувати проблему зникнення градієнта, оскільки всі шари безпосередньо пов'язані з вихідними даними, а градієнти можна обчислити безпосередньо з вихідних даних кожного шару.



Inception Net

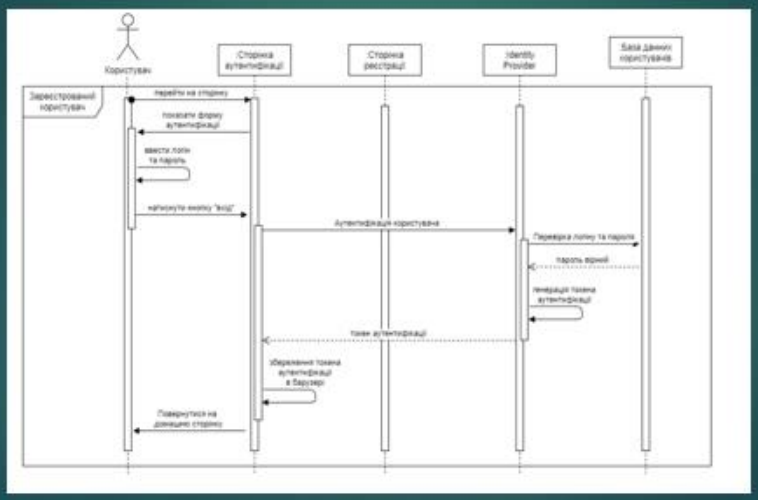
Ідея Inception Net полягає в створенні ширшої мережі ніж ResNet, DenseNet. Це можна зробити шляхом розпаралелювання кількох шарів різними фільтрами, а потім об'єднання всіх цих паралельних шарів для переходу до наступного шару

4. Діаграма класів



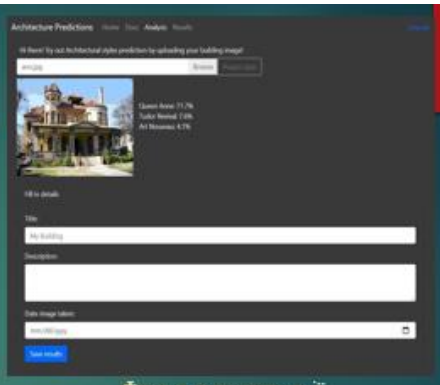
5.1 Діаграма послідовностей

Структурна схема послідовності аутентифікації користувача



6. Веб інтерфейс

Форма входу в систему



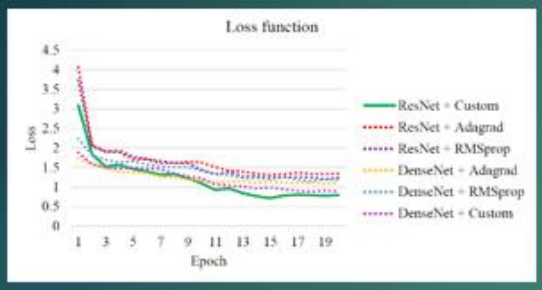
Форма документації

Форма реєстрації

Форма збережених результатів

7.Схема розгортання сервісу використовуючи хмарні технології

8. Проведення технічного аудиту



Показано час (у секундах), необхідний для навчання моделі протягом 20 епох. За допомогою покращеного алгоритму навчання моделі завершується в 1,13 рази швидше, ніж за допомогою інших алгоритмів

Показано зміни у функції втрати навчання. Покращений алгоритм працює краще, ніж алгоритми Adagrad і RMSprop на архітектурах ResNet і DenseNet. Після навчання функція втрат моделі, навченої за вдосконаленим алгоритмом, у 1,5 рази менша, ніж інші моделі.

10. Економічна частина

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Автоматизована система визначення архітектурної стилістики будівельних споруд» становить 33,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,62 рази.

Також термін окупності становить 1,35 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки

- У ході роботи проаналізовано та порівняно існуючі алгоритми вирішення задачі ідентифікації архітектурного стилю будівлі, описано їх характеристики та встановлено, що на даний момент системи з подібними функціями не існує. Дослідження показало, що методи, засновані на глибокому навчанні, значно перевищили результати, досягнуті всіма іншими методами.
- У розділі Розробка алгоритмів класифікації зображень описано математичну постановку задачі, цільову функцію, алгоритм навчання мережі та схему класифікації зображень з її допомогою. У частині аналізу отриманих результатів вдосконалений алгоритм навчання нейронної мережі порівнюється з існуючим алгоритмом на прикладі завдання класифікації побудови зображень різних архітектурних стилів. Запропонований метод навчання нейронної мережі для класифікації архітектурних стилів будівель на основі зображень будівель досягає вищої точності з меншим часом навчання. У середньому запропонований алгоритм має в 1,5 рази менші значення функції втрат наприкінці навчання і в 1,13 рази швидше за інші алгоритми. Імовірність правильної класифікації також вища, ніж в інших мережах, що означає вищу точність класифікації під час навчання.
- У розділі економічного обґрунтування згідно проведених досліджень рівень комерційного потенціалу розробки свідчить про комерційну важливість проведення даних досліджень При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,62 рази.
- Також термін окупності становить 1,35 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.