

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка
до дипломного проекту (роботи)
магістра
(ступінь вищої освіти)

на тему КОМП'ЮТЕРНА СИСТЕМА ДЛЯ РОЗПІЗНАВАННЯ ФРАГМЕНТІВ
МАПИ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: студент 2 курсу, групи КНТ-622м
спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Спеціалізовані комп'ютерні системи

ІВАЩЕНКО В.Р.
(ПРИЗВИЩЕ та ініціали)
Керівник СКРУПСЬКИЙ С.Ю.
(ПРИЗВИЩЕ та ініціали)

Рецензент МАЛИЙ О.Ю.
(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) «Спеціалізовані комп'ютерні системи»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“12” листопада 2023 року

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

ІВАЩЕНКА Владислава Романовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Комп'ютерна система для розпізнавання фрагментів
мапи на основі штучного інтелекту

керівник проєкту (роботи) к. т. н., доцент, СКРУПСЬКИЙ Степан Юрійович
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) існуючі методи та алгоритми класифікації
об'єктів на супутниковому знімку, наявні супутникові знімки

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз технічного завдання, дослідження розпізнавання фрагментів мапи на
супутниковому знімку;

2) Розробка структури та алгоритму роботи системи;

3) Реалізація та дослідження системи;

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	приймав виконане завдання
1-3	СКРУПСЬКИЙ С.Ю., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз наявних методів та алгоритмів розпізнавання об'єктів на зображеннях, огляд сучасних нейронних мереж, огляд алгоритмів розпізнавання фрагментів мапи	10.10.2023 р.	
2	Розробка структури системи	15.10.2023 р.	
3	Розробка алгоритму роботи системи	20.10.2023 р.	
4	Реалізація системи розпізнавання фрагментів мапи	05.11.2023 р.	
5	Дослідження системи розпізнавання фрагментів мапи	20.11.2023 р.	
6	Оформлення отриманих результатів у ПЗ	25.11.2023 р.	
7	Оформлення графічного матеріалу	01.12.2023 р.	
8	Оформлення допоміжного матеріалу	10.12.2023 р.	

Студент  Владислав ІВАШЕНКО
(підпис) (ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)  Степан СКРУПСЬКИЙ
(підпис) (ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 88 с., 39 рис., 3 табл., 43 джерел.

КОМП'ЮТЕРНА СИСТЕМА, КАРТОГРАФІЯ, ГЕОІНФОРМАЦІЙНА СИСТЕМА, СЕМАНТИЧНА СЕГМЕНТАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, ФРАГМЕНТИ МАПИ

Об'єкт дослідження – комп'ютерна система розпізнавання фрагментів мапи.

Предмет дослідження – використання штучного інтелекту для розпізнавання фрагментів мапи.

Мета роботи – проектування та дослідження комп'ютерної системи для розпізнавання фрагментів мапи.

Наукова новизна полягає в застосуванні нейронних мереж для аналізу супутникових знімків, в розпізнаванні та класифікації об'єктів на мапі, що підвищує ефективність картографування супутникових знімків.

Матеріали, методи та технічні засоби: алгоритми розпізнавання зображень, проектування та дослідження комп'ютерної системи з розпізнавання фрагментів мапи. Комп'ютерна система дозволяє створювати та аналізувати певні області супутникових знімків з підтримкою штучного інтелекту.

Результати – розроблено комп'ютерну систему на основі сучасних нейронних мереж. Обрано відповідні бібліотеки для реалізації додатку та проведено дослідження використання впровадженого підходу.

Галузь використання – системи картографування.

ABSTRACT

Explanatory note to the master's work: 88 p., 39 figures, 3 tables, 43 sources.

COMPUTER SYSTEM, CARTOGRAPHY, GEOGRAPHIC INFORMATION SYSTEM, SEMANTIC SEGMENTATION, ARTIFICIAL INTELLIGENCE, MAP FRAGMENTS

The object of the study is a computer system for recognizing map fragments.

The subject of research is the use of artificial intelligence to recognize map fragments.

The purpose of the work is to design and research a computer system for recognizing map fragments.

Scientific novelty lies in the development of neural measures for the analysis of satellite images, in the recognition and classification of objects on the map, which increases the effectiveness of mapping satellite images.

Materials, methods and technical means: image recognition algorithms, design and research of a computer system for recognizing map fragments. The program allows you to create and analyse certain areas of satellite images with the support of artificial intelligence.

Results – a computer system based on modern neural networks. Appropriate libraries were selected for the implementation of the application, and research was conducted using the developed computer system.

Field of use - cartography systems.

ЗМІСТ

Скорочення та умовні позначки	8
Вступ.....	9
1 Постановка предметної області	10
1.1 Поняття геоінформаційні систем (ГІС).....	11
1.2 Задача розпізнавання образів	14
1.3 Дослідження методів ідентифікації об'єктів на зображеннях.....	15
1.4 Функції активації нейронних мереж	21
1.5 Архітектура згорткових нейронних мереж	24
1.6 Архітектури нейронних мереж семантичної сегментації	28
1.6.1 U-Net.....	28
1.6.2 LinkNet.....	29
1.6.3 DeepLabV3	31
1.7 Моделі нейронних мереж для розпізнавання фрагментів мапи	33
1.8 Реалізація розпізнавання фрагментів мапи	34
1.9 Висновки за розділом 1	40
2 Розробка системи	40
2.1 Огляд існуючих аналогів	40
2.2 Вибір інструментів розробки	42
2.2.1 NestJS для реалізації Backend	42
2.2.2 MongoDB для реалізації бази даних.....	44
2.2.3 React для реалізації Frontend	45
2.3 Визначення архітектури комп'ютерної системи.....	47

2.4 Вибір сервісів для проєктування додатку.....	50
2.5 Вибір бібліотеки для аналізування зображень.....	52
2.6 Реалізація компонентів системи	54
2.6.1 Реалізація ядра картографічного додатку.....	54
2.6.2 Реалізація сервісу зберігання файлів.....	59
2.6.3 Реалізація сервісу авторизації.....	60
2.6.4 Реалізація сервісу аналізу супутникових знімків	64
2.7 Висновки за розділом 2.....	68
3 Дослідження системи.....	69
3.1 Актуальність впровадженого підходу.....	69
3.2 Призначення й умови застосування програми.....	71
3.3 Аналіз результатів розпізнавання фрагментів мапи.....	76
3.4 Висновки за розділом 3.....	83
Висновки	84
Перелік джерел посилання	85

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ШІ	– Штучний інтелект;
ГІС	– Геоінформаційна система;
CNN	– Convolutional neural network;
RNN	– Recurrent neural network;
ELU	– Exponential Linear Unit;
GAN	– Generate adversarial networks;
IoU	– Intersection over Union
JSX	– JavaScript XML;
DOM	– Document Object Model.
API	– Application programming interface
JWT	– JSON Web Token

ВСТУП

Сучасний світ активно розвивається завдяки впровадженню інноваційних технологій в різні сфери життя суспільства. Одним із важливих напрямів технологічного прогресу є використання нейронних мереж та методів глибокого навчання для аналізу великих масивів даних і вирішення складних задач, які раніше вимагали значних зусиль та часу. Нейронні мережі здатні моделювати складні залежності в даних і забезпечувати точне розпізнавання об'єктів та патернів.

Особливо актуальним стає використання таких технологій в геоінформаційних системах, де існує потреба в аналізі супутникових знімків та визначенні характеристик об'єктів на мапі. Цей підхід дозволяє ефективно виділяти та класифікувати різні фрагменти мапи, такі як дахи будівель, дорожню сітку, лісову рослинність, забезпечуючи точне визначення їх розташування та характеристик.

У даній роботі буде розглянуто новизну застосування нейронних мереж для аналізу супутникових знімків, розпізнавання та класифікації об'єктів на мапі, а також буде проаналізовано ефективність розробленої системи.

1 ПОСТАНОВКА ПРЕДМЕТНОЇ ОБЛАСТІ

В сучасному світі актуальність даних на електронних картографічних мапах землі становить суттєву проблему. Це пов'язано з постійною зміною ландшафту, бо з'являються нові будівлі, будуються нові дороги, формуються нові райони тощо.

Традиційне цифрове картографування вимагає значного обсягу роботи та великої кількості спеціалістів, щоб створити карту навіть одного міста. Хоча процес цей може бути досить простим, але висока трудомісткість робить його дуже дорогим. Тому все більше вчених та інженерів звертаються до ідеї використання штучного інтелекту (ШІ) для картографування, заснованого на алгоритмах розпізнавання зображень та глибокого навчання.

По-перше, ШІ може підвищити точність електронних картографічних мап шляхом аналізу великих обсягів даних і виявлення закономірностей та аномалій, які можуть бути непоміченими людьми.

По-друге, алгоритми ШІ можуть швидко обробляти великі масиви даних, ефективно створюючи та оновлюючи мапи.

По-третє, ШІ може забезпечити персоналізацію карт, створюючи їх відповідно до індивідуальних потреб та переваг користувачів.

Проте існують також потенційні обмеження та етичні питання використання ШІ в картографії. Наприклад, алгоритми ШІ можуть бути упередженими, тим самим закріплюючи або посилюючи існуючу соціальну нерівність. Крім того, занадто велика залежність від ШІ може призвести до втрати контролю та відповідальності за можливі помилки чи неточності на мапах. Тому важливою технологією, яка може допомогти картографам прискорити свою роботу, є семантичний аналіз. При розробці мапи можна використовувати ШІ як інструмент для виявлення будівель, лісів, полів, доріг тощо. Для цього існує таке поняття як комп'ютерний зір.

Комп'ютерний зір – це така наукова і технічна сфера, яка займається розробкою систем, здатних розпізнавати та аналізувати зображення. Головне, що

робить комп'ютерний зір, він аналізує, що за річ зображена на картинці. Ця технологія корисна в медицині, наприклад, для того, щоб виявити хворобу по медичних знімках. Також вона використовується в промисловості, щоб контролювати якість товарів, і в робототехніці. Крім того, комп'ютерний зір може розуміти жести і емоції на обличчі людини. Також можна створити системи, які будуть краще спілкуватися з людьми, наприклад, керувати жестами або розуміти емоції. Сьогодні вже багато де використовується комп'ютерний зір, від роботизації виробництва до створення продуктів, які покращують життя людства. І ця галузь продовжує розвиватися, відкриваючи перед нами нові можливості [1].

Комп'ютерний зір використовує різні методи для аналізу зображень, наприклад, нейронні мережі, обробку сигналів, або статистичні алгоритми. Завдяки цьому комп'ютери можуть розуміти інформацію з реального світу і взаємодіяти з нею. Завдання комп'ютерного зору можна поділити на два типи.

Перший тип завдань може визначати різні об'єкти без додаткового навчання, тобто їх можна використовувати для розпізнавання будь-яких об'єктів. Це дає більшу гнучкість, але іноді може призводити до помилок при аналізі.

Другий тип завдань призначений для розпізнавання лише певного класу об'єктів, який визначений заздалегідь. Цей підхід менш гнучкий, але він точніший при розпізнаванні об'єктів [2].

1.1 Поняття геоінформаційні систем (ГІС)

Геоінформаційна система, або ГІС — це розумний архів, який допомагає збирати, зберігати, обробляти і показувати просторові дані, які мають свої координати на мапі. Перші такі системи з'явилися в 60-х роках минулого століття, коли комп'ютерні технології почали активно розвиватися [3]. На той час вже була велика кількість географічної інформації, наприклад, аерознімків та статистики, і виникла потреба якось їх упорядкувати і зробити доступними для багатьох людей.

Тому були розроблені саме ГІС для вирішення цієї проблеми, приклад якої показано на рисунку 1.1.

Веб-картографія ставить перед собою наступні ключові завдання [4]. Насамперед, це візуалізація просторових даних, яка спрощує їх обробку і аналіз. Користувачі можуть формувати запити за атрибутами, прокладати маршрути та використовувати інші функції, що базуються на геолокації об'єктів. Крім того розповсюдження просторових даних також є важливим завданням. Картографічні веб-додатки дозволяють завантажувати і обмінюватися даними, а також використовувати їх спільно з іншими користувачами. Наприклад, геопортали створюються з метою розповсюдження цих просторових даних. Також існує завдання формування масивів просторових даних, де користувачі можуть використовувати технології веб-ГІС для створення власних баз географічних даних, як це зробили, наприклад, користувачі проєкту OpenStreetMap, збираючи дані за допомогою космічних знімків і портативних GPS-приймачів. Сучасні картографічні веб-додатки надають користувачам доступ до різноманітних аналітичних функцій, таких як картометричні функції, пошук оптимального маршруту, статистичний аналіз і багато іншого.

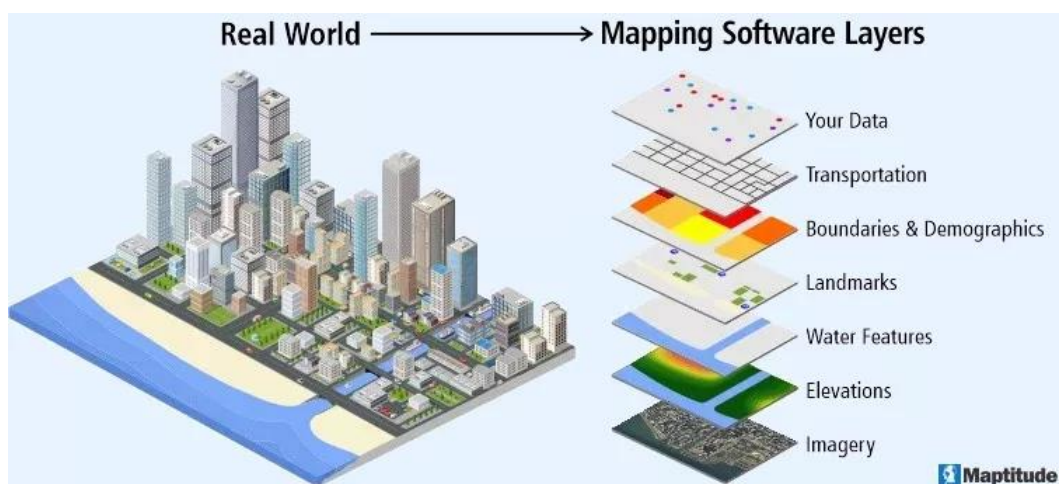


Рисунок 1.1 – Приклад ГІС

Картографічні веб-додатки або веб-ГІС забезпечують ряд переваг в порівнянні з їхніми настільними аналогами. Спершу та головне, вони усувають

проблему недоступності геоінформаційних ресурсів для користувачів, які знаходяться далеко від них. Із переваг веб-ГІС можна визначити [5]:

- глобальне покриття. Розробник може надати доступ до свого картографічного веб-додатку кому завгодно у світі. Користувач може працювати з програмою на своєму домашньому комп'ютері чи на смартфоні;

- кросплатформеність. Більшість веб-браузерів, таких як Google Chrome, Mozilla Firefox або Opera, підтримують картографічні веб-застосунки і їх можна використовувати на різних операційних системах, від Microsoft Windows до Android та IOS;

- низькі витрати на клієнта. Зазвичай веб-ГІС надається безкоштовно, що дозволяє організаціям скоротити витрати на придбання настільних ГІС для кожного користувача. Замість цього можна встановити одну веб-ГІС, якою можуть користуватися всі співробітники одночасно;

- легкість використання. Веб-додатки розробляються таким чином, щоб бути максимально зручними та інтуїтивно зрозумілими для широкої аудиторії;

- одноразове оновлення. Оновлення веб-ГІС автоматично поширюється на всіх користувачів, що значно полегшує керування. Це забезпечує легкість обслуговування та можливість надання інформації в режимі реального часу.

Для проєктування веб-ГІС використовують людей для аналізу супутникових знімків, що дозволяє збільшити точність та якість інтерпретації отриманих даних. Фахівці з обробки зображень та аналізу даних зможуть виділити важливу інформацію, виявити закономірності, які можуть бути непомітними для алгоритмів автоматичної обробки. Крім того, важливу роль у процесі аналізу грає досвід та інтуїція фахівців, які можуть інтерпретувати зображення на основі свого знання та досвіду, тим самим забезпечуючи більшу надійність та точність результатів.

Однак, для ефективного аналізу супутникових знімків важливо також використовувати інноваційні технології та інструменти обробки даних, такі як машинне навчання та штучний інтелект, які дозволяють автоматизувати процес обробки великих обсягів даних, виділяти важливу інформацію та виявляти закономірності, які можуть бути непомітними для людського ока. Таким чином,

комбінація людського аналізу та автоматизованих методів обробки даних може забезпечити найкращий результат при проєктуванні ГІС на основі супутникових знімків.

1.2 Задача розпізнавання образів

Головна мета розпізнавання образів – це навчити машину виявляти ключові характеристики об'єктів та призначати їм певні мітки чи категорії. Для цього потрібно використовувати величезну кількість даних і розробляти спеціалізовані моделі. Задача розпізнавання образів може ділитися на два види [6]:

- класифікація зображень, яка передбачає процес навчання, під час якого системі представляють об'єкти, для яких вже відомо, до якої категорії вони належать. Вивчаючи особливості кожної категорії та їх представників, систему вчать розрізняти їх між собою. Після завершення навчального процесу розпочинається етап розпізнавання, під час якого системі подають нові об'єкти (один або декілька), і вона визначає, до якої категорії вони належать;

- кластеризація зображень ставить мету визначити схожість між об'єктами і розділити їх на відповідні категорії. На відміну від методів навчання, де вже відомо, до якої категорії належить кожен об'єкт, у випадку кластеризації ці категорії потрібно виявити самостійно під час процесу розпізнавання.

Метод класифікації зображень широко використовує техніки глибокого навчання та нейронні мережі. Зокрема, конволюційні нейронні мережі (CNN), які показали свою ефективність у вирішенні цих завдань, адже вони здатні вчитися внутрішньому представленню об'єктів.

Надалі, з розвитком комп'ютерного зору та технологій глибокого навчання, системи класифікації зображень будуть ставати все точнішими і потужнішими. Тому це дасть можливість автоматизувати багато процесів, для яких раніше знадобилася б участь людини, наприклад, проєктування веб-ГІС.

1.3 Дослідження методів ідентифікації об'єктів на зображеннях

Метод Віюлі-Джонса був одним із перших методів які мав високу продуктивність при аналізі зображень. Цей метод використовує особливості, які ґрунтуються на ознаках Хаара, що є хвилями прямокутної форми з однаковими високим і низьким інтервалами. У двовимірному вигляді, прямокутна функція представляє собою пару сусідніх прямокутників, один з яких світлий, а інший – темний, які зображено на рисунку 1.2. На практиці, комбінації прямокутників, які використовуються для візуального впізнавання об'єктів, фактично є справжніми ознаками Хаара. Вони містять в собі прямокутні структури, які оптимально підходять для вирішення завдань розпізнавання в зображеннях [7].

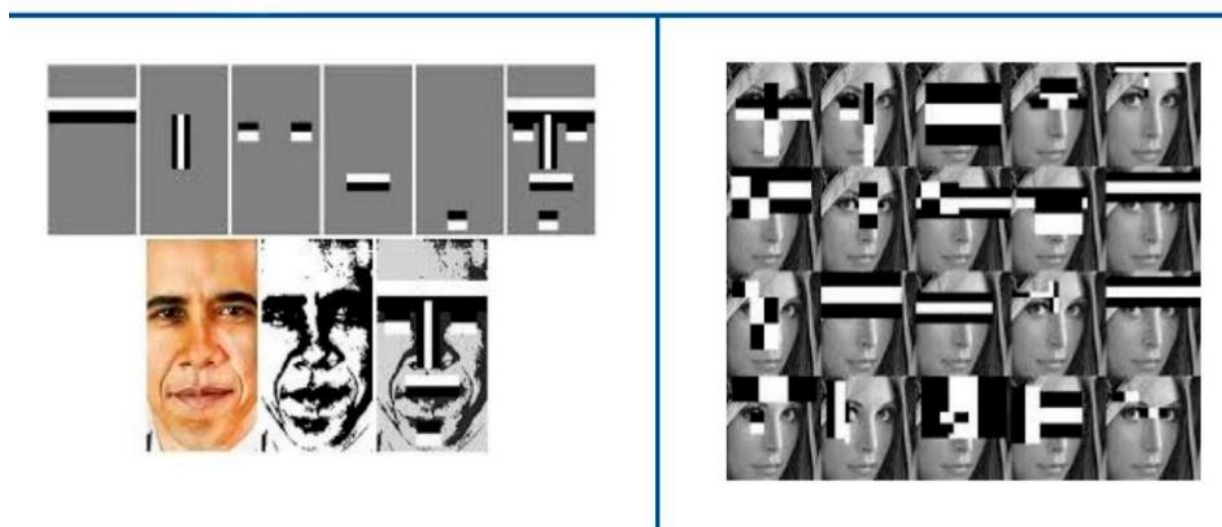
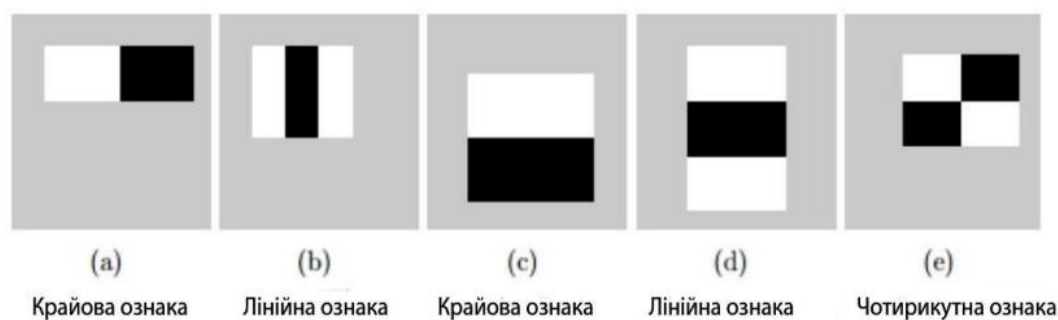


Рисунок 1.2 – Ознаки Хаара

Для вибору конкретних ознак Хаара та встановлення порогових рівнів, Віола і Джонс застосовує метод машинного навчання під назвою AdaBoost. Цей метод поєднує масив слабких класифікаторів з метою створення одного потужного класифікатора. Слабкий означає, що такий класифікатор допомагає вгадувати правильні відповіді не набагато краще, ніж випадкове вгадування. Тобто такий класифікатор сам по собі не є дуже ефективним, але коли є багато таких класифікаторів і кожен з них надає трохи більше правильних відповідей, можна отримати потужний і точний класифікатор. AdaBoost обирає набір слабких класифікаторів для об'єднання і присвоює кожному з них вагу. Приклад зображено на рисунку 1.3. Ця збалансована комбінація створює саме міцний класифікатор [7].

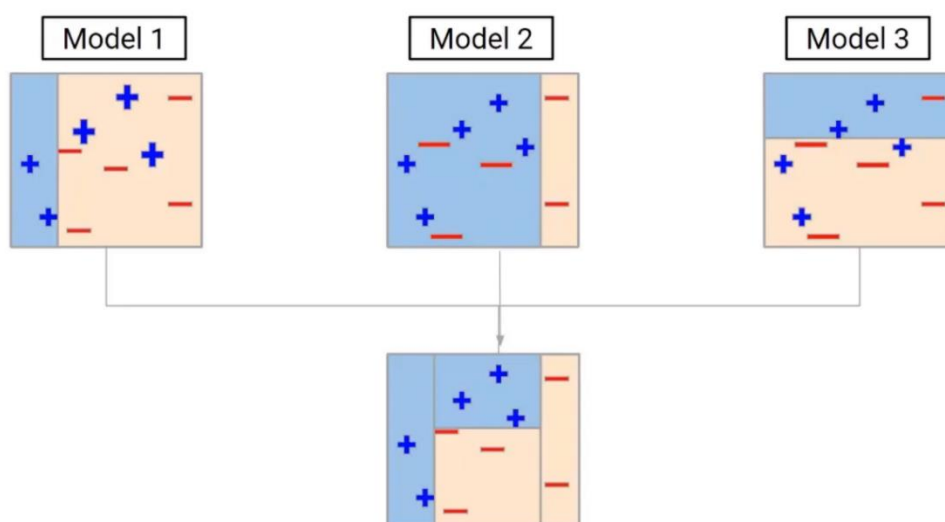


Рисунок 1.3 – Алгоритм AdaBoost

У методі Віоли-Джонса AdaBoost використовується як для відбору ознак, так і для навчання класифікатора. Оскільки, кількість ознак у формі прямокутників є великою, обчислення всіх цих ознак виявляється надто витратним завданням. Однак не всі ознаки є обов'язковими для створення ефективного класифікатора, і можна відібрати невелику підмножину ознак, яка допоможе сформувати ефективний класифікатор. Отже, основним завданням є відшукування цих ключових ознак [7].

На відмінну від методу Віоли-Джонса бажано використовувати нейроні

мережі, які являють важливу складову штучного інтелекту, що бере своє натхнення з функціонування людського мозку. Вони представляють собою комп'ютерні системи, призначені для моделювання і обробки інформації, схожої на те, як це роблять нейрони в біологічних системах.

Основна ідея нейронних мереж полягає у тому, щоб наслідувати структуру та функціонування нейронів у мозку. Нейронні мережі складаються з великої кількості обчислювальних вузлів, які називають "штучними нейронами". Вони взаємодіють між собою у складну мережу. Кожен нейрон приймає вхідні сигнали, обробляє їх та генерує вихід, який може бути переданий іншим нейронам чи використаний для прийняття рішень [8].

Нейронні мережі можуть мати різні архітектури, включаючи:

- нейронні мережі прямого зв'язку, де інформація переміщується в одному напрямку, від вхідного до вихідного шару. Це найпоширеніший тип нейронних мереж і використовується для завдань, таких як розпізнавання зображень та мови;

- багатошарові нейронні мережі, включають заглиблені нейронні мережі (CNN) і рекурентні нейронні мережі (RNN). Вони мають багато шарів нейронів і здатні вирішувати складні завдання, де важливі нелінійні залежності та контекст. Заглиблені нейронні мережі (CNN) спеціалізуються на обробці зображень і широко використовуються для завдань комп'ютерного зору, наприклад для класифікації зображень та розпізнавання обличч, а рекурентні нейронні мережі (RNN) спеціалізуються на обробці та аналізі послідовностей даних.

Розробка нейронних мереж включає в себе процес навчання, де мережа вивчає оптимальні ваги зв'язків між нейронами на основі великої кількості тренувальних даних. Тому, завдяки цьому процесу, навчанні мережі стають здатними вирішувати конкретні завдання, від розпізнавання образів до передбачення результатів [8].

Нейронні мережі знаходять застосування у різних сферах, включаючи обробку природної мови, комп'ютерний зір, автономні автомобілі, медицину, фінанси та ігрову індустрію. Вони мають значний вплив на розвиток штучного інтелекту та надають інструменти для створення інтелектуальних систем, які

можуть адаптуватися до нових завдань і навіть самостійно навчатися.

Однією з ключових завдань, яка стоїть перед штучними нейронними мережами є розпізнавання образів. Ця задача полягає в тому, щоб навчити комп'ютерну систему розпізнавати та ідентифікувати об'єкти, подібно до того, як це робить людський мозок. Нейроні мережі використовуються для вирішення цієї задачі завдяки їх здатності вивчати та адаптуватися до нових даних, що робить їх потужними інструментами у багатьох сферах, включаючи комп'ютерний зір, обробку природної мови тощо.

Основна ідея, яка лежить в основі нейронних мереж, полягає в імітації роботи нейронів у людському мозку. Тобто, існує багато нейронів які з'єднані між собою у мережу та можуть передавати сигнали один одному. Тому така передача сигналів у формі вагованих зв'язків між нейронами дозволяє нейронній мережі вивчати складні залежності між вхідними даними та вихідними. Можна відзначити, що процес навчання нейронних мереж включає в себе підбір оптимальних ваг зв'язків так, щоб мережа могла точно розпізнавати об'єкти на подано на вхід.

Центральним елементом нейронної мережі є нейрон, який виконує обробку вхідних даних та генерує вихідні дані. Нейрони в мережі організовані у шари, з яких можна відмітити такі як: вхідний шар, який приймає необроблені дані; приховані шари, які виконують проміжні обчислення; вихідний шар, який формує вихідне рішення [9].

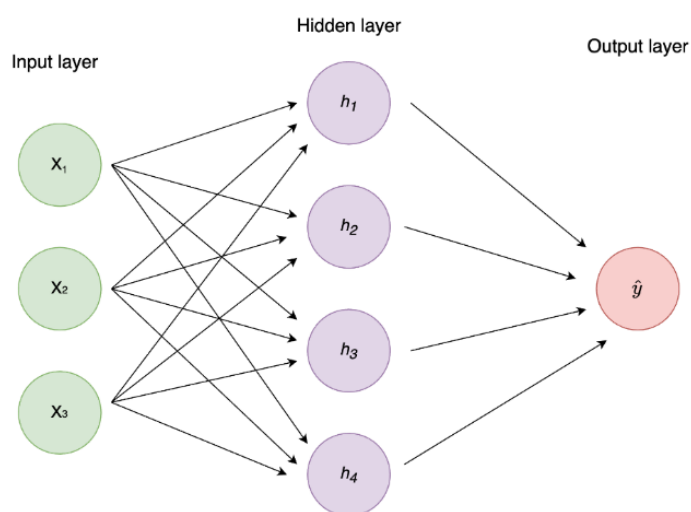


Рисунок 1.4 – Схема простої нейронної мережі

На рисунку 1.4 зображена проста схема нейронної мережі, де:

- зеленим кольором позначені вхідні нейрони, які передають отримані дані;
- фіолетовим приховані нейрони, які виконують основні обчислення;
- червоним позначений вихідний нейрон.

Кожен нейрон в шарі з'єднаний з нейронами попереднього та наступного шарів, і кожному такому зв'язку присвоюють ваги (Weight), що визначають важливість вхідних даних для нейрона. Під час навчання ці ваги коригуються для підвищення продуктивності мережі [10]. Крім того, використовується таке поняття як зміщення (Bias), що також є значущим аспектом, яке враховує усі можливі або спостережувані фактори. Тому, нейроні мережі намагаються враховувати ці непередбачені або невидимі фактори, щоб покращити точність. Кожен нейрон, який не розташований на вхідному рівні, має своє власне зміщення, яке подібно до ваги, має свою важливість. Зображення на рисунку 1.5 надає ілюстрацію цього процесу.

Приділяючи увагу шарам прихованих та вихідних нейронів, видно що до кожного нейрону веде малий червоний або синій вектор. Помітно також, що ці малі вектори не мають вихідного нейрона.

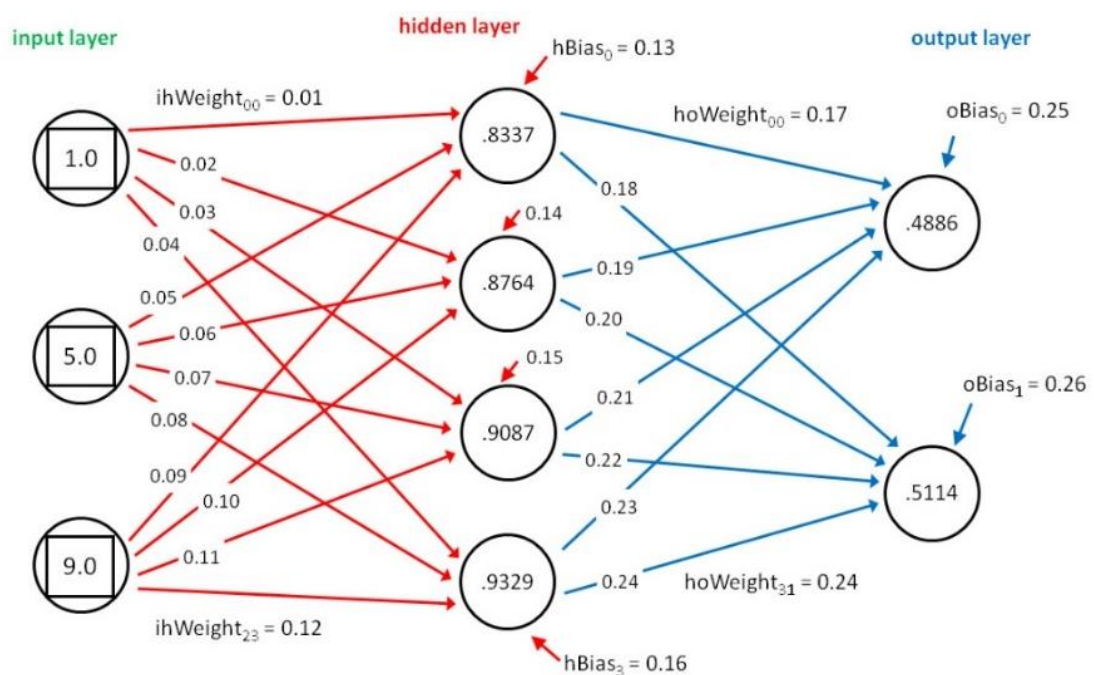


Рисунок 1.5 – Ваги нейронної мережі

Розпізнавання образів за допомогою нейронних мереж є однією з ключових галузей штучного інтелекту, яка продовжує розвиватися та надавати нові можливості для автоматизації процесів та покращення роботи комп'ютерних систем. Нейронні мережі стають все більш потужними та ефективними, завдяки чому вони можуть виконувати складні завдання розпізнавання образів з великою точністю. В сучасних застосуваннях, розпізнавання образів за допомогою нейронних мереж має великий практичний потенціал. Від розпізнавання обличчь, автомобілів та предметів на фотографіях до визначення рукописних символів в оптичному текстовому розпізнаванні. Ця технологія знаходить своє застосування у багатьох сферах, включаючи медицину, автономні транспортні засоби та робототехніку. Приклад розпізнавання образів показано на рисунку 1.6.

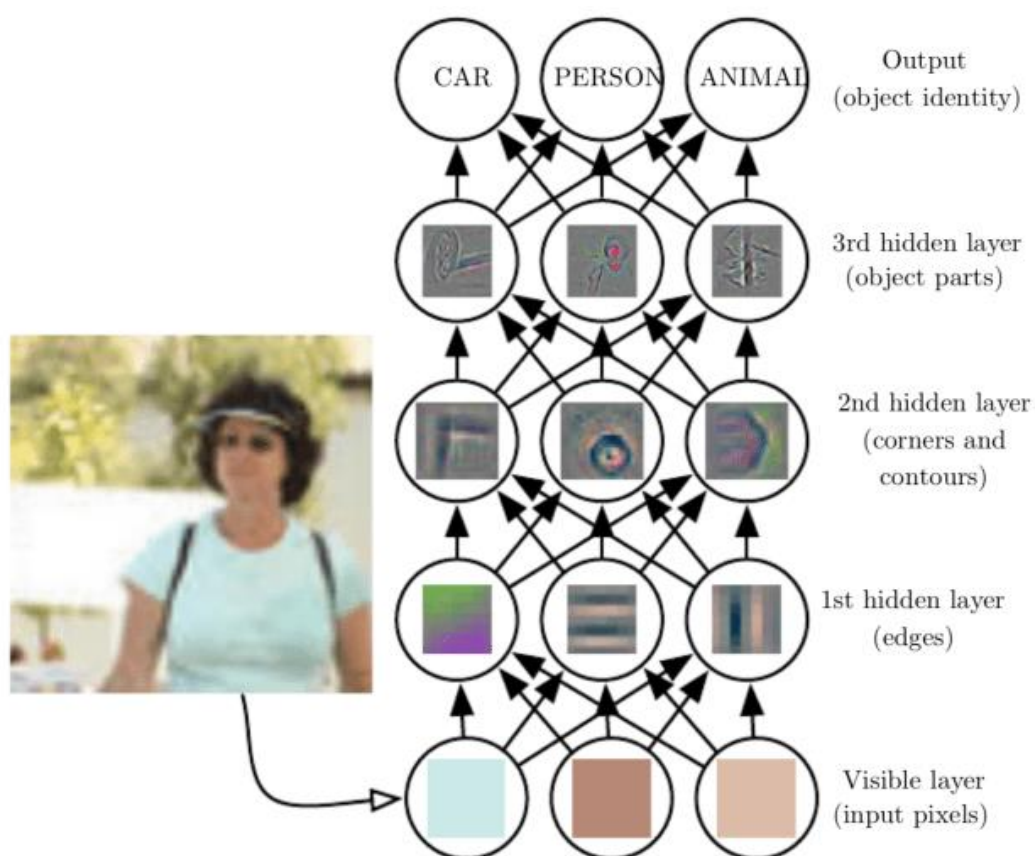


Рисунок 1.6 – Розпізнавання образів за допомогою нейронної мережі

1.4 Функції активації нейронних мереж

Функції активації в нейронних мережах це математичні функції, які використовуються для регулювання активності нейронів у мережі та визначення їх вихідних значень на основі вхідних даних. Основна роль цих функцій полягає в тому, щоб визначити, чи і наскільки нейрон активується відповідно до своїх вхідних сигналів. Математична модель нейрона зображена на рисунку 1.7. Це допомагає нейронним мережам моделювати складні нелінійні залежності в даних та вирішувати різноманітні завдання, такі як класифікація, регресія, аналіз зображень та інші [10].

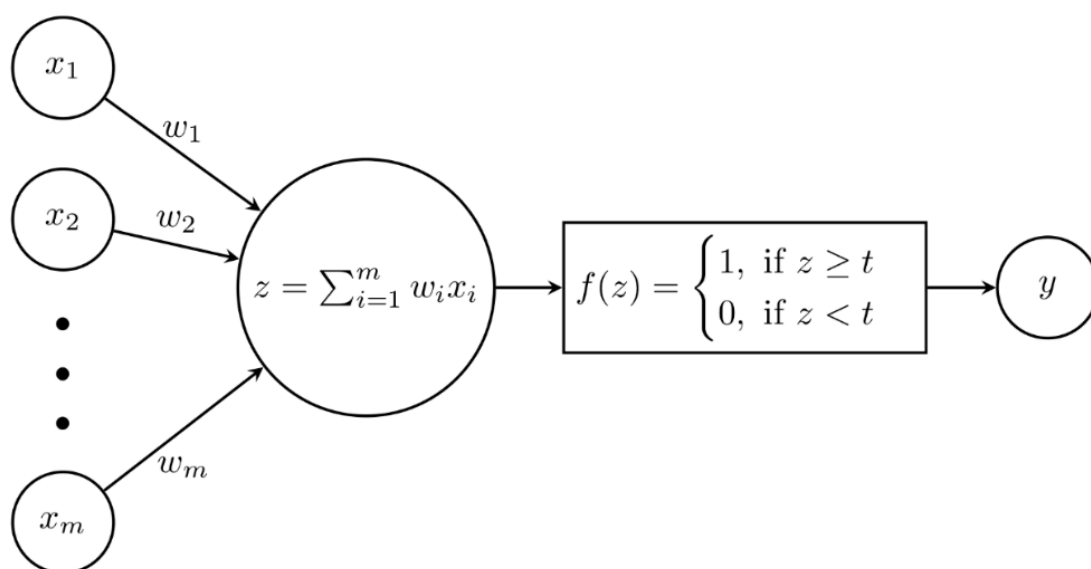


Рисунок 1.7 – Математична модель нейрона

У нейронних мережах існує ряд різних функцій активації, і кожна з них має свої особливості та застосування. Наприклад, найпоширеніші функції активації зображені на рисунку 1.8 [11]:

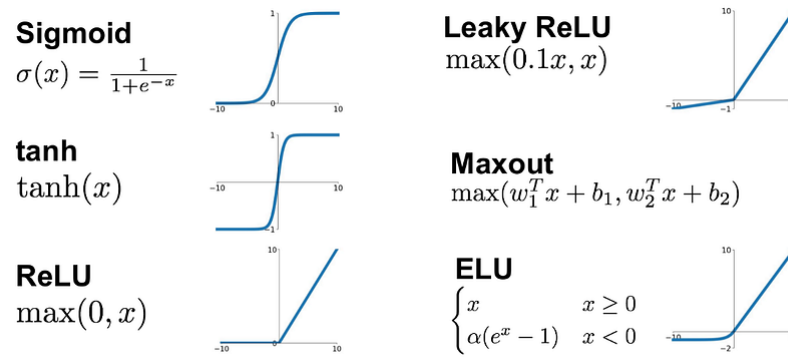


Рисунок 1.8 – Приклад функцій активації

Сигмоїдальна функція (Sigmoid) з формули 1.1 зазвичай використовується у шарах з прихованими нейронами. Вона перетворює входні значення в діапазон від 0 до 1, що може інтерпретуватися як ймовірність у формулі.

$$f(x) = \frac{1}{1+e^{-x}} \quad (1.1)$$

Гіперболічна тангенс (Tanh). Також використовується в прихованих шарах. Ця функція перетворює входні значення в діапазон від -1 до 1 у формулі 1.2.

$$f(x) = \tanh(x) \quad (1.2)$$

ReLU (Rectified Linear Unit). Дуже популярна функція активації, яка використовується в прихованих шарах та наведена у формулі 1.3. Вона не має верхньої обмеженості, що дозволяє нейронам навчатися швидше. Тобто, вона дозволяє обрати максимум між 0 та значенням.

$$f(x) = \max(0, x) \quad (1.3)$$

Leaky ReLU може бути розглянута як поліпшення ReLU, функція активації, яка наведена у формулі 1.4. Вона має всі переваги ReLU, але має важливий додатковий бонус, вона ніколи не призводить до проблеми "мертвого" нейрона,

коли активація завжди дорівнює нулю.

$$f(x) = \max(0.1x, x) \quad (1.4)$$

Експоненційні лінійні одиниці (ELU) – це ще один спосіб активації в нейронних мережах, що є вдосконаленою версією ReLU, особливо для випадків, коли значення x менше за нуль. Ця функція активації наведена у формулі 1.5. ELU зберігає всі переваги ReLU і має наступні особливості:

- уникнення проблеми "мертвих" нейронів, яка іноді виникає в ReLU;
- середнє значення активації ближче до нуля, порівняно з Leaky ReLU, що може бути корисним в певних випадках;
- ELU вимагає більше обчислень через включення експоненційної функції, тому це слід враховувати при виборі функції активації для нейронної мережі.

$$\begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases} \quad (1.5)$$

Функція активації Maxout є однією із специфічних активаційних функцій, яка використовується в нейронних мережах. Вона відзначається своєю здатністю вивчати складні нелінійні залежності в даних. Основна ідея Maxout полягає в тому, що вміст кожного нейрона формується як максимум з набору лінійних комбінацій вхідних ознак. Це відрізняє Maxout від інших активаційних функцій, таких як ReLU або Sigmoid, які використовують фіксовану нелінійність.

Maxout надає моделі більше гнучкості в навчанні, оскільки може вивчати різноманітні форми нелінійних залежностей. Однак це може призводити до більшого числа параметрів у нейронних мережах, що може вимагати більше даних для ефективного навчання і уникнення перенавчання. Ця функція активації наведена у формулі 1.6.

$$f(x) = \max(\omega_1^T x + b_1, \omega_2^T x + b_2, \dots, \omega_k^T x + b_k) \quad (1.6)$$

1.5 Архітектура згорткових нейронних мереж

Згорткова нейронна мережа, яка також відома як CNN, представляє собою особливий клас нейронних мереж, розроблений для обробки даних, які мають топологію у вигляді сітки, зокрема зображень. Цифрові зображення представляють собою двійкове подання візуальних даних. Вони складаються з ряду пікселів, розташованих у вигляді сітки, і містять значення пікселів, які вказують на їхню яскравість і кольоровий відтінок. Приклад наведено на рисунку 1.8.

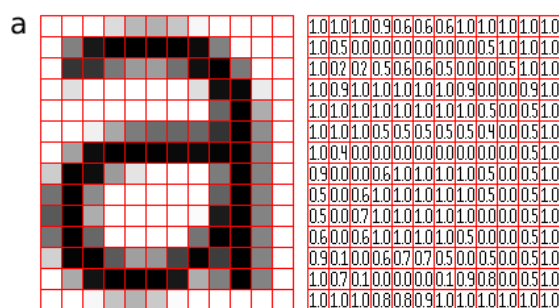


Рисунок 1.8 – Представлення зображення у вигляді даних

Людський мозок має неймовірну здатність обробляти величезну кількість інформації, що надходить, коли ми дивимося на зображення. Кожен нейрон у нашому мозку функціонує в межах свого рецептивного поля і пов'язаний з іншими нейронами так, що покриває всю область нашого зору. Аналогічно, у світі нейронних мереж, таких як згорткові нейронні мережі (CNN), кожен нейрон обробляє дані тільки в межах свого відомого поля.

Шари нейронів у CNN розташовані таким чином, що вони спершу виявляють прості візерунки, такі як лінії і криві, а потім поступово агрегують їх, щоб розпізнати більш складні рисунки, такі як обличчя та об'єкти. Ця ієрархічна структура дозволяє нейронним мережам "розуміти" вміст зображень на різних рівнях абстракції. Використовуючи CNN, можна надати комп'ютерам здатність аналізувати візуальну інформацію, що відкриває безліч можливостей.

CNN зазвичай має три рівні: згортковий рівень, рівень об'єднання та повнозв'язковий рівень [12]. Така архітектура наведена на рисунку 1.9.

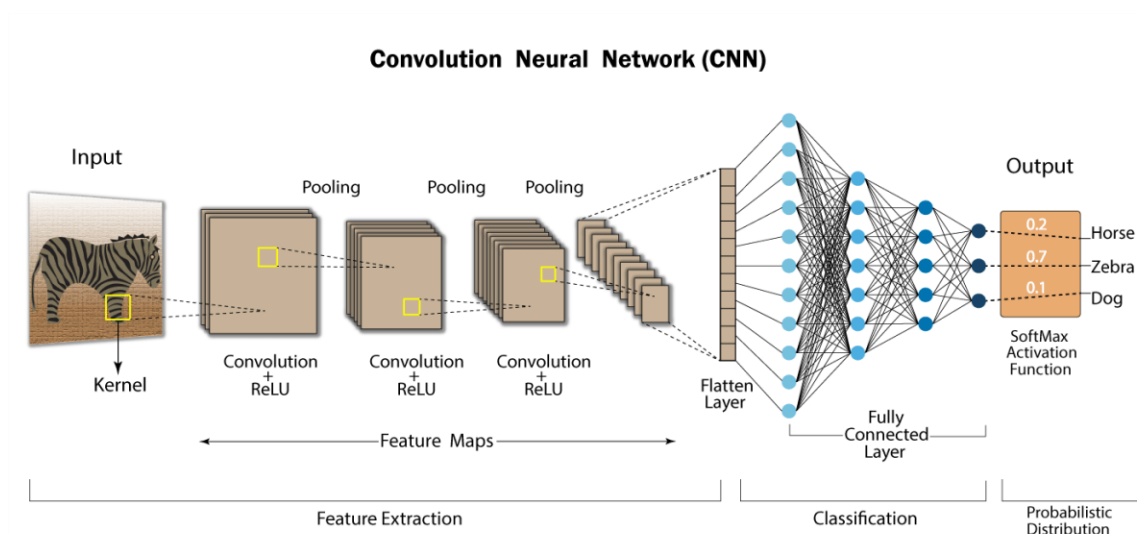


Рисунок 1.9 – Архітектура CNN

Шар згортки є одним із ключових компонентів у побудові згорткових нейронних мереж (CNN). Він бере на себе значну частину обчислювального навантаження в мережі. Приклад операції згортки показано на рисунку 1.10.

На цьому рівні відбувається скалярний добуток двох матриць: одна матриця – це навчальні параметри, відомі як ядро, а інша матриця – це обмежена частина рецептивного поля. Ядро просторово менше зображення, але має більшу глибину. Це означає, що, наприклад, для зображень із трьох RGB каналів, висота та ширина ядра малі просторово, але його глибина розповсюджується на всі три канали.

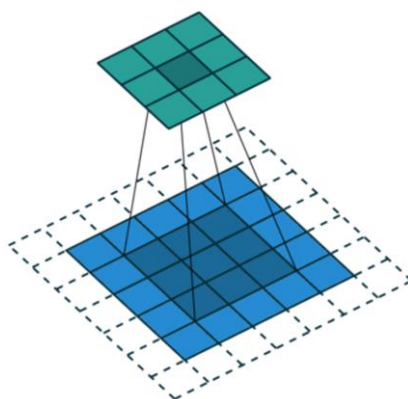


Рисунок 1.10 – Операція згортки

Під час проходження зображення через шар згортки, ядро обходить його вздовж висоти та ширини, створюючи уявлення про рецептивну область. Цей процес породжує двовимірну карту активацій, де кожне ядро відповідає певному положенню на зображенні.

Шар пулу відіграє важливу роль у зменшенні просторових розмірів уявлення в мережі. Він замінює вихідні дані мережі у певних областях, збираючи статистику навколишніх вихідних даних. Це сприяє зменшенню кількості обчислень та параметрів, необхідних для мережі. Кожен фрагмент зображення обробляється окремо під час операції об'єднання. Приклад операції пулінгу показано на рисунку 1.11.

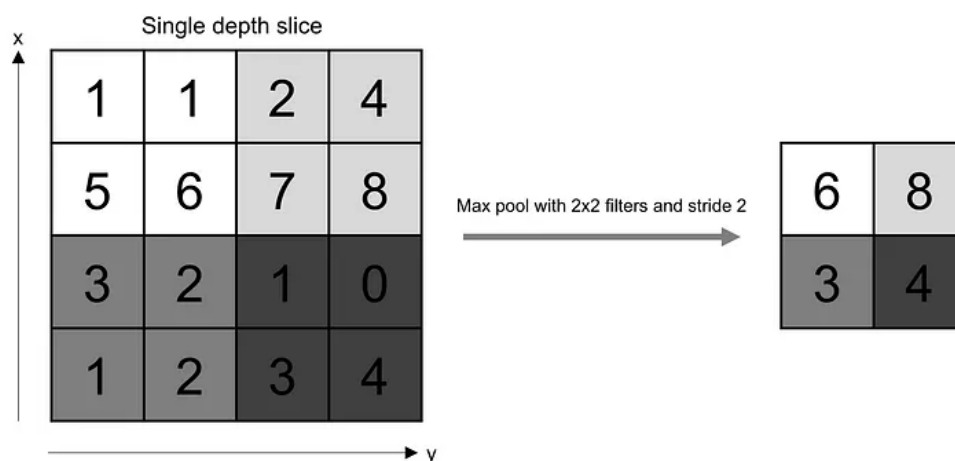


Рисунок 1.11 – Операція пулінгу

Повністю зв'язковий шар мережі є спрощеним і ефективним. Нейрони у цьому шарі мають зв'язок із кожним нейроном у попередньому та наступному шарі, що відрізняє його від інших типів шарів у нейронних мережах.

Обчислення в цьому шарі можна здійснити шляхом множення матриць та подальшого застосування відповідних функцій активації. Цей процес допомагає узгодити уявлення мережі між вхідним та вихідним шарами, що є ключовим для правильної роботи нейронної мережі.

Шари нелінійності грають важливу роль у нейронних мережах, оскільки зображення і дані зазвичай мають складну нелінійну структуру. Оскільки згортка,

як правило, є лінійною операцією, шари нелінійності зазвичай додаються безпосередньо після згортки, щоб впровадити нелінійність до функцій активацій [13].

Зазвичай, коли йде обробка з зображеннями меншого розміру та меншою кількістю каналів кольорів, звичайна нейронна мережа може дати задовільні результати. Проте, якщо розмір зображення та його складність збільшуються, це призводить до збільшення вимог до обчислювальної потужності та ресурсів, що може потребувати великої та дорогої обчислювальної техніки.

Крім того, з часом виникає ще одна проблема – це перенавчання, коли нейронна мережа намагається вивчити занадто багато деталей з навчальних даних. Це може призвести до навчання мережі шуму даних, що негативно вплине на її продуктивність при роботі з тестовими наборами даних. У кінцевому підсумку, звичайна нейронна мережа не може ефективно виділити особливості або закономірності в наборі даних, а отже і сам об'єкт.

На відмінність від цього, згорткова нейронна мережа (CNN) використовує підхід спільного використання параметрів. На кожному рівні CNN кожен вузол з'єднується з іншим, а ваги залишаються фіксованими, коли фільтри шарів переміщуються по зображенню. Ця властивість відома як спільне використання параметрів. Це робить систему CNN менш обчислювально-інтенсивною порівняно зі звичайною нейронною мережею.

Згорткові нейронні мережі вже застосовуються у різних областях комп'ютерного зору та розпізнавання зображень [13]. Наприклад CNN може використовуватися у охороні здоров'я для виявлення аномалій на медичних зображеннях, включаючи ракові клітини. У автомобільній промисловості досліджуються для застосування у безпілотних автомобілях. У соціальних медіа використовують CNN для ідентифікації людей на фотографіях та відзначення друзів. Тому, таку мережу можна використовувати для аналізу фрагментів мапи зі супутникових знімків.

1.6 Архітектури нейронних мереж семантичної сегментації

1.6.1 U-Net

U-Net це архітектура глибокого навчання, яка широко використовується для біомедичних зображень. Основна мета цієї архітектури полягає у вирішенні проблеми нестачі анотованих даних у галузі медицини. Основним досягненням є її здатність ефективно використовувати обмежений обсяг даних, забезпечуючи при цьому швидкість та точність [14].

Архітектура унікальна завдяки своїм шляхам стиснення і розширення. Шлях стиснення містить кодуючі шари, які збирають контекстну інформацію та зменшують просторовий розмір введення. З іншого боку, розширений шлях містить декодуючі шари, які відновлюють закодовані дані і використовують інформацію зі шляху стиснення через пропущені з'єднання для створення карти сегментації.

Шлях стиснення U-Net визначає відповідні функції вхідного зображення. Кодуючі шари виконують згорткові операції, які зменшують просторову роздільну здатність картин об'єктів, одночасно збільшуючи їх глибину. Цей шлях подібний до шарів прямого зв'язку в інших згорткових нейронних мережах. Розширений шлях займається декодуванням закодованих даних та виявленням об'єктів, зберігаючи просторовий розмір вхідних даних. Декодуючі шари в розширеному шляху підвищують дискретизацію об'єктів і виконують згорткові операції. Пропущені з'єднання зі шляху стиснення допомагають зберегти просторову інформацію, втрачену у шляху стиснення, що сприяє більш точному визначенню розташування об'єктів шарами декодера [14].

На рисунку 1.12 продемонстровано, як мережа U-Net змінює вхідне зображення у відтінках сірого розміром $572 \times 572 \times 1$ у двійкову сегментовану вихідну карту розміром $388 \times 388 \times 2$. Можна відзначити, що розмір виходу менший за розмір входу, оскільки не використовуються відступи. Проте застосування відступів дозволяє зберегти розмір введення. Під час проходження шляху стиснення вхідного зображення поступово зменшується за висотою та шириною,

але збільшується кількість каналів, що дозволяє мережі взяти високорівневі функції. У вузькому місці застосовується остання операція згортки для створення карти об'єктів розміром $30 \times 30 \times 1024$. Потім розширений шлях бере карту об'єктів з вузького місця і перетворює її назад у зображення такого ж розміру, що й вихідні дані. Це досягається за допомогою шарів, що підвищують дискретизацію, які збільшують просторову роздільну здатність карти об'єктів, зменшуючи кількість каналів. Пропущені з'єднання зі шляху стиснення використовуються для того, щоб допомогти шарам декодера знаходити і уточнювати елементи зображення. Нарешті, кожен піксель вихідного зображення є міткою, що відповідає певному об'єкту або класу вхідного зображення. У цьому випадку вихідна карта представляє собою двійкову карту сегментації, де кожен піксель відображає область переднього або заднього плану [15].

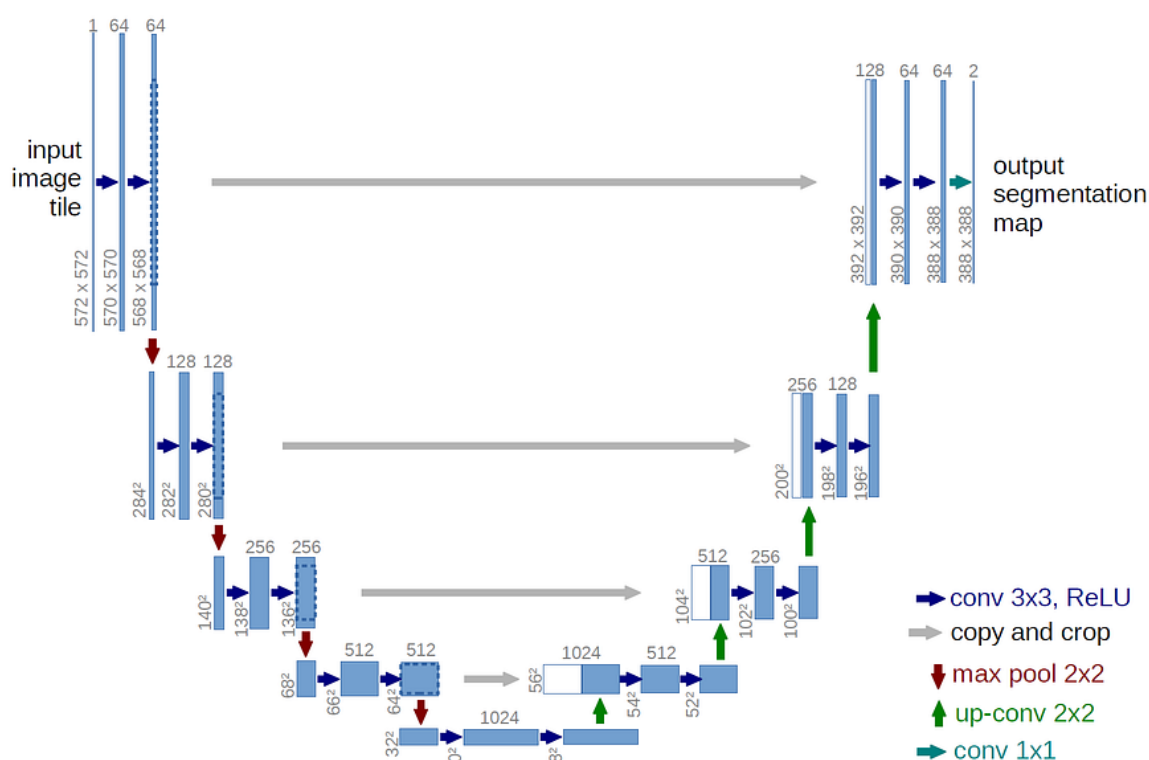


Рисунок 1.12 – Архітектура мережі U-Net

1.6.2 LinkNet

LinkNet це легка архітектура глибокої нейронної мережі, розроблена для виконання семантичної сегментації в реальному часі, яку можна використовувати

для таких завдань, як безпілотні транспортні засоби, доповнена реальність тощо. Вона здатна забезпечувати продуктивність у реальному часі як на графічних процесорах, так і на вбудованих пристроях, таких як NVIDIA TX1. Ця мережа була розроблена членами e-Lab в університеті Пердью. LinkNet може обробляти вхідне зображення з роздільною здатністю 1280x720 на TX1 і Titan X зі швидкістю 2 та 19 кадрів в секунду відповідно [16].

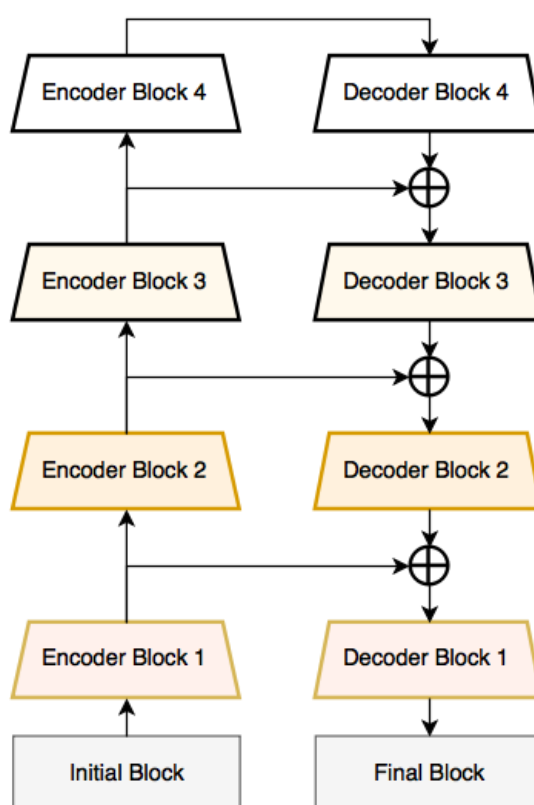


Рисунок 1.13 – Архітектура мережі LinkNet

На рисунку 1.13 запропонована архітектура намагається ефективно обмінюватися інформацією, отриманою кодувальником після кожного блоку зменшення дискретизації. Такий спосіб виявляється кращим, ніж використання індексів об'єднання в декодері, або просто використання повністю згорткових мереж. Така техніка пересилання функцій не тільки може дати гарні значення точності, але також дає змогу мати декілька параметрів у декодері. Тому LinkNet часто використовується в програмах, де сегментація в реальному часі має вирішальне значення. Приклад такої обробки зображено на рисунку 1.14.

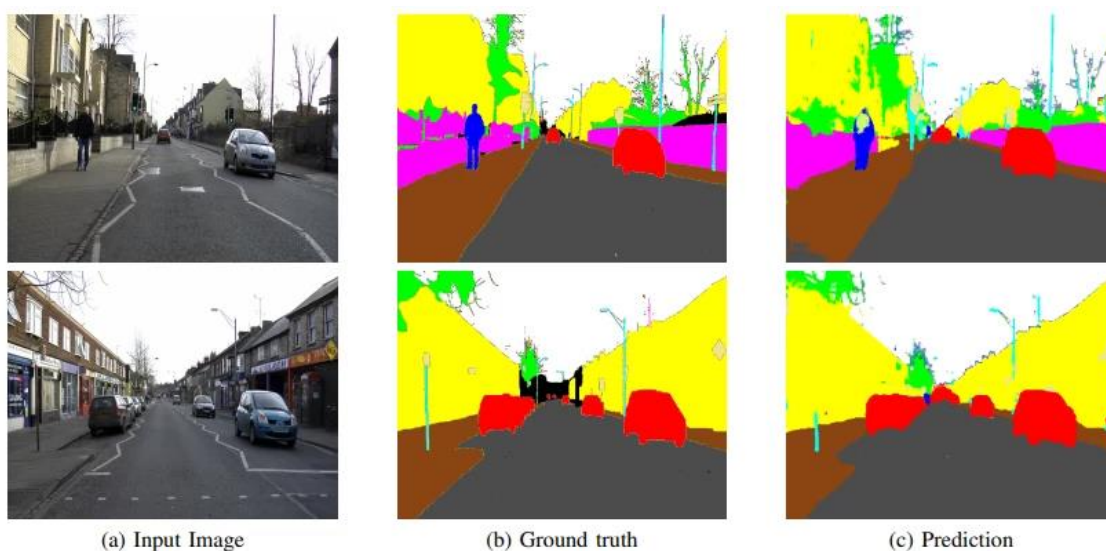


Рисунок 1.14 – Приклад роботи архітектури LinkNet

1.6.3 DeepLabV3

Повністю згорткові нейронні мережі часто використовуються для семантичної сегментації. Однією з проблем, пов'язаних із використанням таких мереж на зображеннях для завдань сегментації, є те що карти вхідних функцій стають меншими під час проходження через шари згортки, що спричиняє втрату інформації на зображенні та призводить до вихідних даних із низькою роздільною здатністю, а межі об'єктів стають нечіткі. Модель DeepLab вирішує цю проблему за допомогою модулів Atrous Convolutions і Spatial Pyramid Pooling (ASPP). Також, варто відмітити що ця архітектура розвивалася протягом кількох поколінь [17]:

- DeepLabV1 використовує Atrous Convolution і для керування роздільною здатністю, з якою обчислюються характеристики зображення;
- DeepLabV2 використовує Atrous Spatial Pyramid Pooling (ASPP) для розгляду об'єктів у різних масштабах і сегментації зі значно покращеною точністю;
- DeepLabV3 окрім використання Atrous Convolution, також використовує вдосконалений модуль ASPP, включаючи пакетну нормалізацію та функції рівня зображення.

Модель DeepLabV3 має наступну архітектуру. Спочатку функції витягуються з магістральної мережі (VGG, DenseNet, ResNet). Потім для того щоб контролювати розмір карти функцій, в останніх кількох блоках використовується додаткова

згортка. На додаток до витягнутих функцій з магістралі додається мережа ASPP для класифікації кожного пікселя відповідно до його класів. Вихідні дані з мережі ASPP проходять через згортку 1x1 щоб отримати фактичний розмір картинки, яке буде остаточною сегментованою маскою для зображення [18]. Архітектуру цієї мережі зображено на рисунку 1.15.

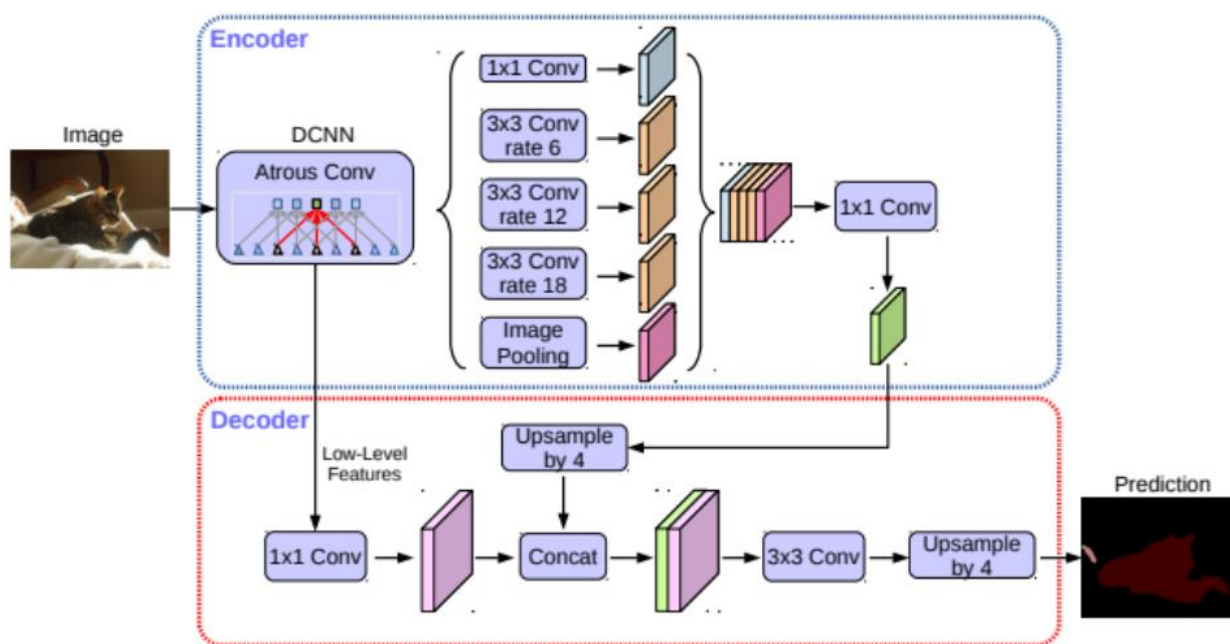


Рисунок 1.15 – Архітектура мережі DeeplabV3

Ці вдосконалення допомагають отримувати досконалі мапи функцій для довгострокових контекстів. Це експоненціально збільшує сприйнятливий поле без зменшення/втрати просторового виміру та покращує продуктивність завдань сегментації. Deeplab успішно працює в різних програмах, включаючи аналіз сцени, сегментацію об'єктів і сегментацію зображень загалом.

Таким чином, незважаючи на те, що U-Net, LinkNet та Deeplab спрямовані на вирішення завдань сегментації зображень, вони відрізняються своєю архітектурою та областями фокусування. U-Net часто використовується в біомедичній візуалізації, LinkNet призначений для сегментації в реальному часі, а Deeplab чудово справляється зі збором багатомасштабної контекстної інформації для завдань семантичної сегментації.

1.7 Моделі нейронних мереж для розпізнавання фрагментів мапи

Для аналізу та розпізнавання різних фрагментів мапи за допомогою штучних нейронних мереж, потрібно використовувати різні моделі, кожна з яких повинна спеціалізуватися на визначенні конкретних об'єктів або фрагментів. Наприклад, можна запровадити такі моделі як:

- модель для розпізнавання дахів будівель. Така модель призначається для аналізу супутникових знімків і може ефективно виділяти області, які відповідають дахам будівель. Вона повинна використовувати глибокі нейронні мережі для виявлення та вирізання цих областей з зображення. Такі області включають основні категорії: житлові будівлі з багатьма квартирами, приватні будинки та промислові споруди. Висота будівлі може оцінюватися на основі вимірювання довжини її тіні та видимої частини стін. Визначення типу будівництва відбувається через аналіз класифікації об'єктів на супутникових знімках високої роздільної здатності. Така модель повинна бути ключовою для аналізу мапи та визначення розташування будівель. Під час навчання, ця модель повинна вивчати різні архітектурні особливості дахів, такі як їх форма, кут нахилу, матеріал тощо. Вона також повинна враховувати контекст оточення будівель, щоб визначити їхнє точне розташування на мапі. Результатом роботи моделі є створення маски, яка виділяє області, що відповідають дахам будівель на супутниковому знімку;

- модель для розпізнавання лісової рослинності. Така модель повинна спеціалізуватися на визначенні лісових областей на мапі. Вона може використовувати зготовані нейронні мережі для виділення та класифікації дерев та лісової рослинності. Виділення категорій лісової рослинності з високороздільних зображень за їхнім типом, густотою та висотою. Така модель може бути використана як інструмент для прийняття рішень при плануванні робіт з видалення лісових урочищ;

- модель для розпізнавання дорожньої сітки. Така модель може призначатися для визначення доріг та дорожніх об'єктів на мапі. Вона повинна автоматично

визначати сітку доріг на супутникових знімках високої роздільної здатності. Використання такої моделі дозволить ефективно відслідковувати і оновлювати інформацію про будівництво нових доріг у різних частинах світу.

Наприклад, застосування першої моделі для виявлення дахів будівель може бути різноманітним [19]:

- у картографії. Розпізнані дані можуть бути використані для покращення цифрових мап і навігаційних систем, що допоможе користувачам швидше і точніше знаходити своє місцезнаходження та планувати маршрути;

- геодезії. Модель може бути використана для автоматичного визначення розміщення будівель на земельній ділянці для цілей забудови та земельного кадастру;

- моніторинг забудови. Можливо, навіть виявлення нових будівель або змін в існуючих структурах на основі регулярного аналізу супутникових знімків;

- рятувальних операцій. Під час надзвичайних ситуацій, таких як природні катастрофи, модель може використовуватися для визначення розміщення будівель і служити як інструмент для координації рятувальних операцій. Отже, модель для розпізнавання дахів будівель є важливим інструментом для картографії та геоінформаційних систем, яка може допомогти зробити картографію більш точною і ефективною, а також знайти своє застосування в багатьох інших областях.

Отже, ці моделі можуть бути навчанні на великих наборах даних, які містять зображення мапи, щоб навчити їх розпізнавати відповідні об'єкти та фрагменти.

1.8 Реалізація розпізнавання фрагментів мапи

Для ефективної реалізації розпізнавання фрагментів мапи потрібно також обробляти отримані результати після проходження навченої моделі нейронної мережі. Приклад такої обробки наведено на рисунках 1.16 та 1.17. Якщо опрацювати багато супутникових знімків, то можна отримати багато даних, які

перевірити та обробити вручну є нереальним завданням. Навіть при задовільних результатах для використання в статистичних моделях просторового аналізу територій, такі контури можуть не відповідати вимогам для готової картографічної продукції. З візуальної точки зору користувача вони не виглядають естетично. Тому якщо оброблені дані обробити з використанням алгоритмів полігонізації, тоді візуальний результат значно поліпшиться. Таким чином, до розпізнавання фрагментів мапи штучним інтелектом можна також додати ще декілька етапів. Наприклад, можна додати такі етапи обробки фінального результату як:

- виключення накладень полігонів, який буде допомагає вирішити проблему накладення контурів, розширюючи полігони та зменшувати їх площу;
- заповнення невеликих дірок для уникнення помилок у формі будівель, які мають внутрішні двори, тобто цей алгоритм може заповнювати невеликі порожнечі всередині полігонів;
- фільтрація за площею або класом для відсіювання занадто малих об'єктів, які, ймовірно, не є житловими або не входять в клас, який відповідає завданню.

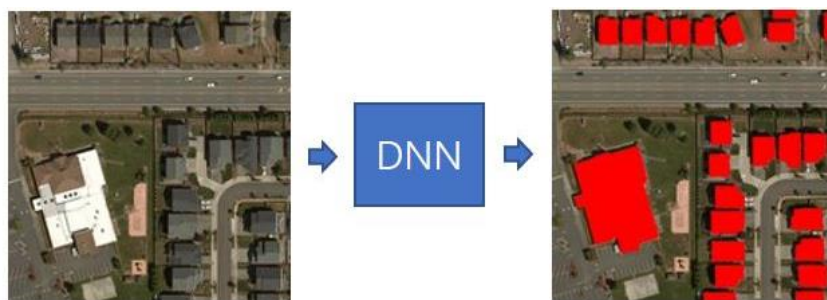


Рисунок 1.16 – Приклад розпізнавання мапи

Перший алгоритм який можна включити в обробку результатів нейронної мережі це полігонізація. Якщо подивитися на будь-яку мапу, то можна визначити, що більшість будівель можна апроксимувати прямокутною формою, а більшість кутів саме прямими. Тому за допомогою цих чинників можна створити алгоритм полігонізації, який буде створювати форми будівель. Приклад застосування такого алгоритму зображено на рисунку 1.17. Проте, такий процес може бути

трудомістким при додавання в його в нейронну мережу, і в кінцевому результаті він може займати багато часу. Тому, як варіант, можна використовувати інше рішення, яке називається уточненням маски за допомогою мережі GAN [20].

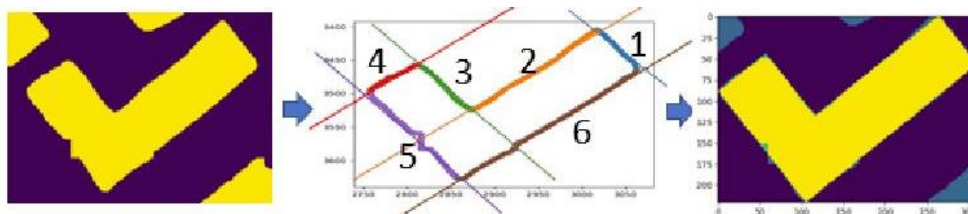


Рисунок 1.17 – Приклад застосування алгоритму полігонізації

Мережа GAN навчається на парах об'єктів, які мають правильну полігонізацію. Фактично, коли процес доходить до уточнення полігонів за допомогою мережі GAN, то нова мережа застосовується до вибраних об'єктів, які не були полігонізовані. Це поліпшує спрощення, але не впливає на метрики. Приклад такої мережі зображено на рисунку 1.18, де перше зображення це маска сегментації, друге уточнена маска сегментації, а третє спрощена уточнена маска сегментації.



Рисунок 1.18 – Приклад застосування маски сегментації

Крім правильної форми будівель, потрібно також включити метод вирівнювання будівель, розташованих вздовж доріг. Такий алгоритм дозволяє вирівнювати будівлі вздовж вулиць та відносно одна одної з метою досягнення максимальної точності відображення обстановки на місцевості. Такий алгоритм визначає середнє відхилення поворотів будівель всередині кластера щодо лінії дороги та інших будівель у кластері. Якщо середнє відхилення перевищує

встановлений поріг, алгоритм переорієнтовує всі відповідні будівлі під прямим кутом до найближчої дороги та, за необхідності, усуває накладення будівель. Незважаючи на те, що цей алгоритм не є ідеальним і може призводити до помилок у повороті, його використання спрямоване на пошук балансу між автоматизацією та точністю. Приклад застосування такого алгоритму показано на рисунку 1.19.



Рисунок 1.19 – Приклад застосування алгоритму вирівнювання будівель

Крім того, можна додати алгоритм оцінки параметрів будівлі. Зараз існують алгоритми які дозволяють побачити будівлю у тривимірному вигляді [21]. Існує безліч варіантів, які вже давно доступні для супутникових даних, наприклад, радіолокаційні зображення можуть показувати висоту об'єкта, а стереопари (пари зображень, зроблених під різними кутами у відомих умовах) можуть показати достатньо, щоб побачити будівлю в тривимірному вигляді, що як і у 3D-фільмі зветься фотограмметрію. Приклад якої наведено на рисунку 1.20.



Рисунок 1.20 – Приклад фотограмметрії

Щоб зрозуміти висоту будівлі можна використовувати простий математичний алгоритм, де з огляду на розташування контуру будівлі, потрібно намітити тінь, яку будівля відкидає на землю, і виконати прості математичні обчислення [21]. Наприклад, для рисунку 1.21 потрібно застосовувати формулу 1.7.

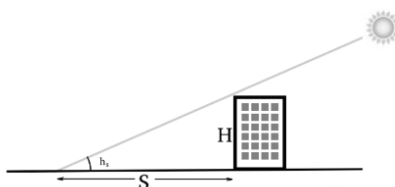


Рисунок 1.21 – Приклад вимірювання висоти будівлі

$$H = S \cdot \tan(h) \quad (1.7)$$

де S – довжина тіні; h – кут падіння сонця, який відомий на основі часу захоплення зображення.

Крім того, деякі області взагалі можуть не оброблятися, тобто, не всі супутникові знімки зроблені безпосередньо зверху (у надирі), а багато з них зроблені під ухилом (поза надиром). Тому, можна оновити геометрію сцени, яка буде виглядати як на рисунку 1.22.

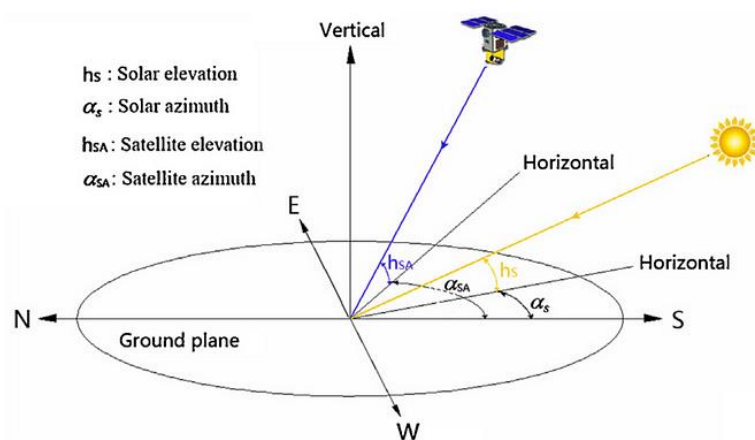


Рисунок 1.22 – Приклад складного варіанту сцени вимірювання висоти будівлі

У більшості випадків, кути мають висоту та азимут сонця, а також висоту та азимут супутника, які можна отримати у постачальників зображень. І найголовніше те, що якщо отримати дах, стіну та тінь будівлі, можна реконструювати всі тривимірні параметри, включаючи висоту та площу основи. Це важливо, оскільки на зображенні сліду не видно, і його пряма сегментація є складним завданням. Тому нейронним мережам не потрібно вгадувати невидимі частини, а лише описувати видимий дах, тінь та стіну [22]. Приклад такого застосування зображено на рисунку 1.23.

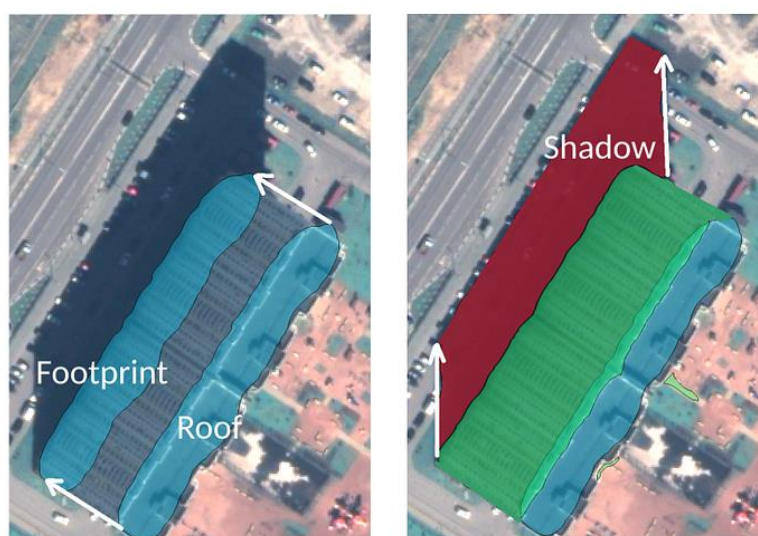


Рисунок 1.23 – Приклад модифікованого вимірювання висоти будівлі

Тому якщо потрібно виміряти тінь та розміри стіни, тоді прямий вимір зведе до багатьох проблем, оскільки деякі тіні з'єднуються один з одним, а стіна часто частково затінена. Тому, можна використовувати метод моделювання будівлі у 3D. Спочатку обчислюється положення даху, а потім визначається, чи будівля має задану висоту, та наскільки добре їй підходить тінь та стіна. Далі йде підганяння, яке означає, що йде обчислення IoU між результатами сегментації та результатами моделювання та береться висота, яка дає кращий результат (усереднений для стіни та тіні). Тому середня абсолютна похибка для цього методу становить трохи більше 3 метрів, що означає похибку ± 1 поверх у звичайному багатоквартирному будинку.

1.9 Висновки за розділом 1

За результатами аналізу та опису архітектури систем, які використовують нейронні мережі для аналізу та розпізнавання фрагментів мап, можна виділити кілька ключових висновків. Перш за все, сучасні технології нейронних мереж глибокого навчання здатні ефективно вирішувати складні задачі аналізу зображень, що відкриває широкі можливості для їх застосування в різних областях, включаючи аналіз супутникових знімків для визначення характеристик об'єктів на мапі.

Функції активації нейронних мереж відіграють важливу роль у процесі навчання та роботи нейронних мереж, допомагаючи моделювати складні нелінійні залежності в даних. Згорткові нейронні мережі (CNN) виявилися ефективними для обробки зображень, і їх можна застосовувати для розпізнавання певних об'єктів на зображенні, наприклад, дахів будівель.

2 РОЗРОБКА СИСТЕМИ

2.1 Огляд існуючих аналогів

На сьогоднішній день існує велика кількість картографічних сервісів, що надають користувачам доступ до різноманітних географічних інформаційних ресурсів. Ця різноманітність картографічних сервісів відображає розвиток сучасних технологій та зростання популярності геоінформаційних систем (ГІС) та картографічних додатків, але мало де використовуються саме нейронні мережі для картографування супутникових знімків.

- OpenStreetMap є відкритою картографічною платформою, яка базується на зібраній та підтримуваній спільнотою інформації. Вона надає безкоштовний доступ до глобальних геоданих та інструментів для редагування мап. Основною перевагою OSM є її відкритий тип мавпування, що сприяє постійному оновленню та

вдосконаленню мап. Проте в даному сервісі відсутня підтримка нейронних мереж;

- Google Maps є комерційним сервісом, який надає великий спектр можливостей для користувачів та розробників. Він включає в себе навігацію, геолокацію, інтеграцію з іншими Google-сервісами та сторонніми додатками. Основною перевагою Google Maps є його доступність для багатьох платформ та зручність використання, але тут теж відсутня підтримка будь-яких нейронних мереж для прискорення картографування;

- MS Buildings є спеціалізованим продуктом від Microsoft, спрямований на аналіз та візуалізацію геоданих, пов'язаних з будівлями та геопросторовими структурами. В наявності вже було виявлено понад 999 мільйонів будівель за зображеннями Bing Maps по 2021 рік, включаючи зображення Махар та Airbus. Ці дані доступні для скачування та використання безкоштовно в рамках ліцензії ODbL. На відміну від інших аналогів, цей сервіс використовує нейронні мережі для аналізу супутникових знімків [23].

Як бачимо, на сучасному ринку геоінформаційних сервісів можна відзначити значну нестачу картографічних платформ, які активно використовують нейронні мережі для аналізу супутникових знімків. Хоча технології глибинного навчання та штучного інтелекту отримують все більшого значення у сфері обробки геоданих, але існує обмежений вибір інструментів та платформ, які надають такий функціонал. Сучасні нейронні мережі вже здатні до точного аналізу та обробки супутникових знімків, дозволяючи виявляти об'єкти, визначати зміни в ландшафті, прогнозувати ризики та багато іншого, тому використання нейронних мереж для аналізу супутникових знімків можуть поліпшати можливості користувачів та підприємств у вирішенні деяких різноманітних геоспецифічних завдань.

Такі платформи можуть забезпечити значний внесок у розвиток галузі, дозволяючи ефективно використовувати великі обсяги геоданих для розв'язання проблем середовища, транспорту, міського планування та інших важливих сфер. Зростаючий попит на картографічні сервіси, що використовують нейронні мережі, свідчить про необхідність розвитку цього напрямку індустрії.

2.2 Вибір інструментів розробки

Після того, як було проведено детальний аналіз предметної області, було прийнято рішення щодо вибору відповідних фреймворків для реалізації проєкту. У процесі аналізу були розглянуті різні аспекти, такі як вимоги до функціональності, масштабованість, швидкодія, підтримка сучасних мов програмування та інші фактори.

На основі отриманих даних та оцінок [24] були обрані фреймворки, які оптимальні для вирішення поставлених завдань та досягнення поставлених цілей проєкту. Кожен з обраних фреймворків має свої унікальні переваги та можливості, які відповідають конкретним вимогам проєкту та сприяють його успішному виконанню.

2.2.1 NestJS для реалізації Backend

NestJS це фреймворк для створення серверних програм на базі Node.js, який робить процес розробки масштабованих застосунків більш простим та швидким [25]. Він використовує як JavaScript, так і TypeScript, і об'єднує три основні концепції програмування. Перша версія NestJS була представлена у 2017 році і вона дозволяє розробникам створювати програми, які перевершують можливості попереднього рішення - Express.js, для роботи з Node.js.

NestJS використовує модульну систему, яка дозволяє створювати налаштовані модулі без надмірних зусиль. Вона включає в себе такі складові, як контролери, провайдери, і модулі, і кожна з них відповідає за певний аспект розробки.

Контролери обробляють вхідні запити і надають відповідь клієнту.

Провайдери можуть бути усіма типами, такими як сервіси, репозиторії, фабрики, допоміжні функції та інші, і вони можуть бути використані в контролерах та інших провайдерах.

Модулі організують структуру додатка і кожна програма NestJS має кореневий модуль, який служить для впорядкування всіх інших модулів [25].

Крім цього, NestJS надає інші компоненти, такі як Pipes, Guards, і Interceptors, які допомагають в розробці та контролі за додатком.

Загалом можна відзначити такі переваги NestJS як [25]:

- надає дуже гнучку модульність, яка дозволяє структурувати додаток у модулі, що полегшує організацію та підтримку коду. Кожен модуль має свої компоненти, контролери та сервіси, що допомагає зберігати код чистим і розширюваним;

- використання патерну dependency injection, який сприяє створенню зв'язків між компонентами додатку, що полегшує управління залежностями і покращує перевірку коду;

- надання великою ефективності, яка пропонує ряд оптимізацій та інструментів для покращення продуктивності, такі як кешування, лімітування запитів і багато іншого;

- використання мови програмування TypeScript, яка дозволяє писати код з типами, що полегшує виявлення помилок на ранніх етапах розробки та підвищує читабельність коду.

Незважаючи на багато переваг, NestJS також має і деякі недоліки, які можуть бути важливими для деяких проєктів [25]:

- зайвий рівень абстракції, при використанні Dependency Injection та інших паттернів, що може призвести до зайвої абстракції в коді та може ускладнити розуміння та підтримку проєкту;

- може бути більше важчим за інші Node.js фреймворки через використання TypeScript та ряд оптимізацій для структурування коду, що може вплинути на продуктивність в окремих випадках;

- завдяки високій рівні абстракції і модульності, код може бути більшим, ніж у менш абстрактних фреймворків, що може вимагати більше зусиль для розробки та його підтримки.

Комп'ютерна система, що розробляється матиме багато ендпоінтів та складну структуру для взаємодії, тому NestJS виконує усі умови сучасного Backend серверу.

2.2.2 MongoDB для реалізації бази даних

MongoDB наразі є однією з найпопулярніших систем управління базами даних типу NoSQL, особливо в сфері документоорієнтованих систем. Ця база даних потрапляє до десятки найбільш використовуваних систем цього типу в світі [26]. Основна відмінність MongoDB від традиційних реляційних баз даних полягає в тому, що вона зберігає дані у вигляді документів, а не рядків. Документи можуть мати більш складну структуру, ніж рядки, і вони можуть бути розглянуті як сховища ключ-значення [27].

У MongoDB, як і в реляційних базах даних, є поняття унікального ідентифікатора для кожного запису, який називається "_id". Відмінність поля "_id" від первинного ключа в реляційних базах полягає в тому, що MongoDB генерує значення "_id" автоматично, якщо ви не вказуєте його явно.

У реляційних базах даних кожне поле має певне значення, і може бути встановлено значення null для тих полів, які не мають значення. У MongoDB, якщо значення не зазначено для певного ключа, цей ключ просто може бути опущений у документі. Крім того в реляційних базах також використовуються таблиці для зберігання однотипних, жорстко структурованих об'єктів, тоді як у MongoDB дані зберігаються в колекціях, які можуть містити різні об'єкти з різною структурою та набором властивостей [27].

Загалом можна відзначити такі переваги як:

- використання сховища даних на основі документів, де кожен документ може мати різні поля та структуру. Це дає можливість легко адаптувати схему даних для різноманітних типів об'єктів в додатку без необхідності міграції даних;
- підтримка горизонтального масштабування, яке дозволяє додавати нові сервери для обробки додаткових навантажень без значного уникнення, що робить MongoDB гарним вибором для великих та зростаючих додатків;
- підтримка різноманітних запитів до бази даних, включаючи складні запити, агрегацію даних та можливість виконувати запити на основі JavaScript;
- підтримка високо доступних конфігурацій з реплікацією та розподіленим кластеруванням для забезпечення надійності даних.

Загалом, MongoDB є потужною NoSQL базою даних, яка може бути корисною для багатьох типів додатків, особливо тих, де потрібна велика гнучкість у сховищі та масштабованості. Однак перед вибором MongoDB для реалізації проєкту важливо також розглянути і обмеження цієї бази даних [27]:

- MongoDB добре підходить для додатків які мають мінливу структуру даних, але вона може бути менш ефективною для додатків з жорсткою та структурованою схемою даних;

- може бути вимогливою до ресурсів, особливо при великому обсязі даних та високих навантаженнях, що може призвести до необхідності великої кількості серверів та обладнання для підтримки цієї бази даних;

- історично MongoDB не підтримувала транзакції в реляційному стилі, хоча більш пізні версії фреймворку додали деяку підтримку транзакцій. Таким чином, якщо система вимагає сильної гарантії цілісності даних, то це може стати великою проблемою;

- запис даних в MongoDB може бути повільним у порівнянні з іншими системами баз даних, особливо при великому обсязі записів.

Комп'ютерна система, що розробляється матиме різноманітну структуру об'єктів, тому незважаючи на недоліки такої бази даних, MongoDB ідеально підходить для даного завдання.

2.2.3 React для реалізації Frontend

Для розробки графічної частини проєкту була обрана бібліотека React. Ця бібліотека використовується для створення складних односторінкових додатків (SPA) на базі JavaScript. Слід також зауважити, що React на сьогоднішній день можна назвати більше фреймворком, ніж просто бібліотекою, оскільки вона надає структуру для розробки додатків [28].

React був створений компанією Facebook з метою поліпшення процесу розробки та візуалізації інтерфейсів, особливо для великих проєктів, таких як Facebook. Для вирішення проблеми з погіршенням продуктивності при зміні багатьох елементів на сторінці був розроблений віртуальний DOM (Document Object Model), який дозволяє ефективно управляти елементами сторінки. Він

представляє собою легку копію звичайного DOM. Якщо потрібна інформація про стан елемента, React звертається до віртуального DOM. І якщо потрібно змінити елементи сторінки, React знаходить різницю між новим і старим станом та оновлює реальний DOM з мінімальними операціями. Це дозволяє досягти високої ефективності при взаємодії з елементами.

Однією з особливостей React є використання JSX (JavaScript XML), який поєднує в собі код JavaScript і HTML. Крім цього, React базується на концепції компонентів. Кожен компонент є окремою частиною інтерфейсу, яку можна використовувати повторно або переносити з проєкту в проєкт як шаблон. Просто описавши, як різні частини інтерфейсу повинні виглядати в різних станах додатку, React автоматично оновлює лише необхідні компоненти [28].

Для реалізації багатьох компонентів інтерфейсу було обрано бібліотеку Material UI (MUI) [29]. Material UI – це бібліотека відкритого коду для React, яка реалізує дизайн Material Design від Google. Вона включає в себе повний набір готових компонентів, які можна використовувати у виробництві безпосередньо. Інтерфейс Material UI має привабливий дизайн і надає можливість налаштування параметрів для реалізації власного дизайну на базі цих компонентів.

З переваг можна відзначити [30]:

- використання віртуального DOM (Virtual DOM) для ефективного оновлення інтерфейсу, що дозволяє уникнути зайвих операцій в DOM і підвищити продуктивність веб-додатків;

- використання компонентної архітектури, де інтерфейс розділяється на невеликі, незалежні компоненти. Це полегшує розробку, тестування та підтримку коду;

- використання JavaScript або JSX для опису інтерфейсу, що спрощує розробку, оскільки розробники можуть дуже гнучко підтримувати додаток;

- активна спільнота розробників, що означає багато ресурсів, бібліотек та інструментів, які допомагають в розробці.

Незважаючи на багато переваг, React також має і деякі недоліки та обмеження [30]:

- великі проекти можуть потребувати додаткової оптимізації, щоб забезпечити високу продуктивність;
- використання великої кількості компонентів та стану може призвести до збільшення обсягу завантажуваного JavaScript, що може вплинути на швидкість завантаження сторінок;
- у великих додатках керування станом може стати складним завданням, і вимагати використання додаткових бібліотек, таких як MobX.

Велика частина цих недоліків може бути вирішена з правильним плануванням, гарно підбраною архітектурою та використанням додаткових бібліотек та інструментів для допомоги у розробці. Тому для комп'ютерної системи що розробляється React підходить ідеально.

2.3 Визначення архітектури комп'ютерної системи

Проектування архітектури програмного забезпечення – це один з найважливіших етапів при розробці програмного продукту, він представляє компонування компонентів системи, які найкраще відповідають бажаним якісним характеристикам системи. Тобто, розробник слідкує про те щоб система була швидкою, надійною та доступною. Менеджер проекту слідкує щоб система була поставлена вчасно та в рамках бюджету. Команда підтримки слідкує щоб система була простою для розуміння та налагодження. Неможливо розробити систему відповідаючи всім, не жертвуючи якістю системи. Отже, при розробці архітектури програмного забезпечення потрібно вирішити, які атрибути якості мають найбільше значення для завдання:

- продуктивність, скільки потрібно чекати, перш ніж завдання буде виконане;
- можливість модифікації, якщо розробники хочуть додати нову функцію до системи, наскільки легко це зробити;
- безпека, наскільки має бути надійною система;

- масштабованість, якщо швидко збільшується базу користувачів, чи зможе система легко масштабуватися для задоволення нового трафіку.

Залежно від того, яке програмне забезпечення проєктується або покращується, певні атрибути можуть мати більше значення. Архітектура допускає такі атрибути якості, наприклад, однорангова архітектура матиме високу доступність через децентралізований характер системи, а архітектура мікросервісів матиме високу модифікованість через розподіл обов'язків [31].

Архітектура фокусується на компонованні, а не на створенні компонентів. Замість того, щоб зосередитися на тому, як написаний код, архітектура змушує думати про те, як компоненти в системі взаємодіють один з одним. Існує безліч архітектур на вибір [31], але не всі з них підходять під певну задачу та іноді вимагають багаторічного досвіду для правильної реалізації. Приклад двох сучасних архітектур зображено на рисунку 2.1.

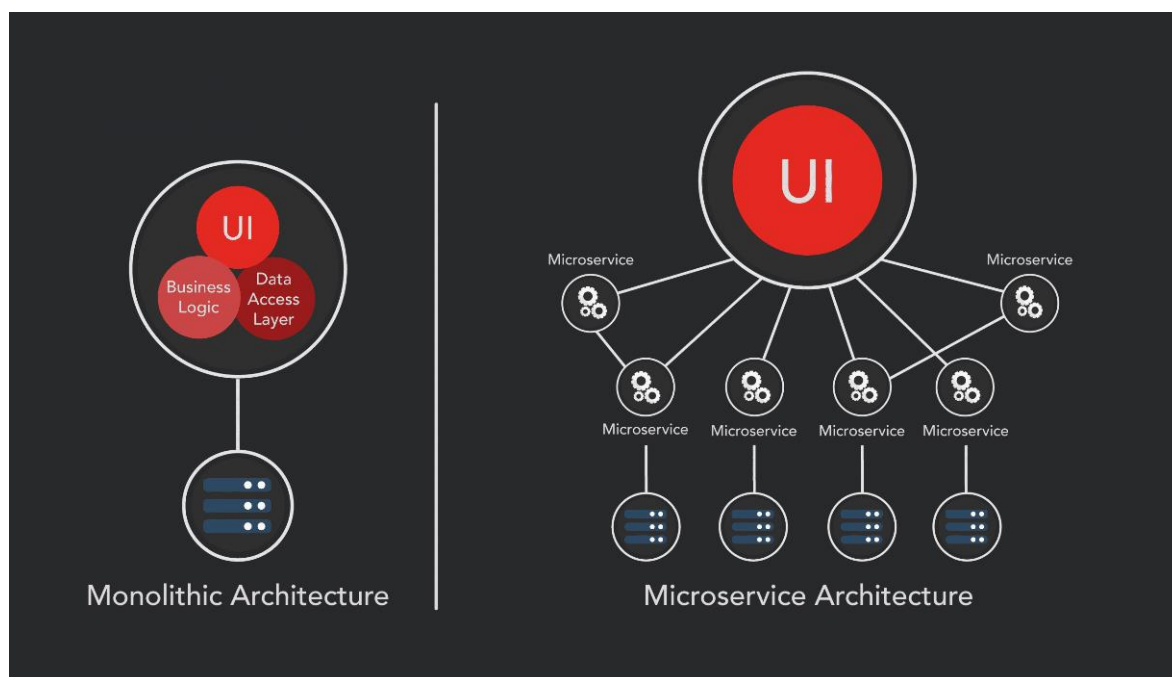


Рисунок 2.1 – Архітектури комп'ютерних систем

Монолітна архітектура вважається звичайним методом створення комп'ютерних системи, вона представляє із себе неподільний блок, що складається з різних рівнів. Ці рівні включають базу даних, серверний додаток, що опрацьовує

запити клієнта та взаємодіє із базою даних та виконує бізнес-логіку, та інтерфейс користувача на стороні клієнта, що зазвичай складається з HTML сторінок, що запускаються в браузері. Усі ці компоненти об'єднуються, утворюючи єдиний логічний блок. Монолітна система зазвичай базується на одній великій кодовій базі та не має модульності. При потребі в оновленнях розробники мають збирати та розгортати весь стек одночасно [32].

Архітектура мікросервісів полягає в розбитті програмного застосунку на невеликі незалежні модулі. Ці частини виконують свої завдання як окремі служби, кожна з яких виконує конкретні функції і має власну бізнес-логіку та базу даних. Архітектура мікросервісів це спосіб розробки програми, де функціональність розділена на невеликі служби. Уся функціональність розділена на незалежні модулі, які взаємодіють один з одним через конкретні API. Кожна служба має свій власний обсяг застосування і може бути розгорнута, оновлена та масштабована незалежно. Централізоване управління цими службами практично відсутнє. На відміну від монолітної архітектури, функціональність мікросервісів формалізується через бізнес-орієнтовані API. Ці служби адаптовані для використання в різних контекстах і можуть бути використані повторно в різних бізнес-процесах або через різні канали. Завдяки застосуванню принципу слабкого зв'язку, залежності між службами та їхніми споживачами зводяться до мінімуму. Це дає власникам служб можливість розгортати будь-які оновлення реалізації служб без впливу на користувачів [33].

Для реалізації програмного забезпечення було обрано саме мікросервісну архітектуру, бо для реалізації картографічного додатку потрібно буде використовувати різні зовнішні сервіси. Наприклад, для відображення мапи зі супутниковими знімками можна використовувати один сервіс, для обробки файлів інший, логіка роботи додатку буде оброблюватися на самописному сервері тощо.

2.4 Вибір сервісів для проєктування додатку

У процесі розробки сучасного веб додатку потрібно використовувати низку передових технологій та сервісів, кожен з яких відіграє ключову роль у забезпеченні функціональності та зручності для користувачів. Тому для реалізації комп'ютерної системи було обрано такі сервіси як: MongoDB Atlas для створення віддаленої бази даних, Mapbox GL для використання існуючого шару векторної та супутникової мапи, Dropbox як сервіс для зберігання файлів та Google OAuth як сервіс для можливості авторизації через існуючий акаунт. Усі вони пропонують різноманітні функції та інструменти, тому варто роздивитися кожен із них більш детально.

MongoDB Atlas для створення віддаленої бази даних. Як вже було зазначено, MongoDB це NoSQL база даних, що працює з документоорієнтованим підходом. Ця технологія дозволяє зберігати великі об'єми структурованих та неструктурованих даних у форматі, який є гнучким і легко масштабованим. Тому використання MongoDB Atlas як віддаленої бази даних дозволяє розгортати, керувати і масштабувати бази даних в хмарі з використанням провідних хмарних провайдерів, таких як AWS (Amazon Web Services), Google Cloud Platform і Microsoft Azure. З переваг можна виділити [34]:

- забезпечення легкого горизонтального масштабування, що дозволяє збільшувати та зменшувати ресурси в залежності від потреби, без перерви в роботі сервісу;
- автоматична обробка реплікації даних між різними зонами доступності, що забезпечує високий рівень доступності та стійкість до збоїв;
- автоматизоване резервне копіювання та можливість відновлення даних, що гарантує, що додаток може швидко відновитися після аварійних ситуацій;
- можливість обирати регіони розміщення своїх даних, щоб оптимізувати швидкість відгуку для своїх користувачів.

Через це використання MongoDB Atlas в розробці додатків є вибором

багатьох стартапів та великих підприємств, оскільки воно значно спрощує управління базами даних і дозволяє розробникам зосередитися на створенні інноваційних рішень, тому цей сервіс було використано у розробці картографічного додатку.

Mapbox GL для створення існуючого шару векторної мапи. Цей сервіс є потужною бібліотекою для відображення інтерактивних мап, що дозволяє інтегрувати кастомізовані векторні мапи у веб та мобільні додатки. Використовуючи Mapbox GL, можна створити детальні та адаптивні мапи, які не тільки покращують досвід користувачів, але й забезпечують високу продуктивність навіть на мобільних пристроях з обмеженими ресурсами [35].

Dropbox як сервіс для зберігання файлів та зображень. Один із варіантів збереження файлів є використання власного серверу, але щоб заощадити можна використовувати саме Dropbox, бо він є одним з лідерів на ринку хмарних сервісів для зберігання файлів, і надає швидкий доступ до файлів [36].

Google OAuth як сервіс для можливості авторизації через вже існуючий акаунт. Авторизація через цей сервіс забезпечує користувачам безпечний та швидкий варіант входу до додатку. Цей метод використовує існуючі облікові записи Google, щоб уникнути необхідності створювати новий логін та пароль, тим самим зменшуючи кількість паролів, які користувачу потрібно запам'ятовувати та підвищуючи загальний рівень безпеки [37].

Комбінація цих сервісів дає змогу створити додаток, який є одночасно функціональним, зручним у використанні та безпечним. MongoDB гарантує гнучке керування даними, Mapbox GL надає інструменти для інтеграції кастомізованих мап, Dropbox забезпечує надійне зберігання файлів, а Google OAuth пропонує простий та безпечний спосіб авторизації. Тому разом всі ці компоненти формують технологічний функціонал, який відповідає сучасним вимогам ринку та високим стандартам користувацького досвіду.

2.5 Вибір бібліотеки для аналізування зображень

Перед початком створення будь-якого проєкту, пов'язаного з розпізнаванням зображень, потрібно правильно обрати бібліотеку, бо це вибір визначає шлях успішної реалізації завдання. На ринку існує безліч бібліотек для обробки та аналізу зображень, тому можна роздивитись лише кілька з них. Однією з найпопулярніших бібліотек у цьому напрямку є OpenCV, яка володіє широким спектром функціоналу для обробки та аналізу зображень. Вона дозволяє виконувати такі завдання, як розпізнавання обличч, виявлення об'єктів, фільтрація та підсумовування зображень.

OpenCV (Open Source Computer Vision Library) – це бібліотека з відкритим вихідним кодом, яка надає інструменти для обробки зображень та комп'ютерного зору. Вона реалізована на C++ та має інтерфейси для мов програмування, таких як Python, Java та інші [38]. Вона була розроблена для найвищої ефективності та продуктивності в задачах машинного зору, які вимагають значних ресурсів. Таким чином, компанія надає особливу увагу реалізації програм машинного зору в режимі реального часу. Вона може взаємодіяти з багатоядерними процесорами для реалізації багатопоточності. Головною метою OpenCV є забезпечення простої та зручної інфраструктури для комп'ютерного зору, яка допомагає швидко розробляти складні програми для машинного зору. Можна виділити що OpenCV широко використовується в перевірці якості заводської продукції, медичній візуалізації, системах безпеки, інтерфейсах людина-машина, калібруванні камер та робототехніці. Вона має розширені можливості обробки зображень, такі як обробка відеопотоків, злиття зображень з декількох камер, калібрування камери та виконання різних завдань попередньої обробки зображень. З огляду на роль машинного навчання в галузі комп'ютерного зору, OpenCV включає повну бібліотеку машинного навчання загального призначення, орієнтовану на статистичне розпізнавання образів і кластеризацію [38].

Крім того також існує ще й і SimpleCV, що є ще однією з варіантів бібліотек,

побудованою на основі OpenCV, яка відома своєю легкістю використання та високорівневим інтерфейсом. Вона може бути особливо корисною для швидкого прототипування та розробки на базовому рівні. SimpleCV спрощує та прискорює реалізацію завдань комп'ютерного зору, надаючи оболонку базового складного коду у бібліотеці OpenCV. На додаток до цього, вона також надає такі функції, як оптичне розпізнавання символів за допомогою простих у використанні функцій [39].

Також можна виділити таку бібліотеку як Dlib, яка є ще одним потужним інструментом для завдань аналізу зображень, зокрема розпізнавання обличч та виявлення об'єктів. Вона використовує високоефективні алгоритми, які можуть забезпечити точні та швидкі результати. Вона розроблена на мові програмування C++, Dlib відзначається великим спектром функцій, що роблять її популярним інструментом в галузі розпізнавання обличч, виявлення об'єктів та інших завдань комп'ютерного зору. Однією з ключових особливостей цієї бібліотеки є її вражаючий арсенал алгоритмів розпізнавання обличч, які можуть працювати в реальному часі та показують високу точність. Бібліотека також включає ефективні інструменти для виявлення об'єктів на зображеннях та роботи з контурними моделями. Dlib становить цінний ресурс для розвитку застосувань у сферах від медичного зображення та візуальної аналітики до розпізнавання жестів та реалізації алгоритмів трекінгу об'єктів у відеопотоках [40].

Але на відміну від усіх цих бібліотек, для розпізнавання фрагментів мапи потрібно використовувати дещо інші алгоритми, тому також можна розглядати такий варіант як Mapflow AI. Цей сервіс спрямований на автоматизацію картографії за допомогою навчених алгоритмів розпізнавання зображень супутникових знімків. Система використовує неймережі, які були навчені на фотографіях, а потім перенавчені на великих картографічних наборах даних. Глибоке навчання використовується для обробки великої кількості фрагментів супутникових знімків. Платформа Mapflow має доступ до глобальних даних супутникових знімків, механізм управління ланцюгами обробки та моделями штучного інтелекту для семантичного аналізу та класифікації зображень. Ця платформа може

використовується у проєктах автоматичного картографування будівель, моніторингу охоронних зон та інших застосуваннях [41].

Тому Mapflow може бути відмінним варіантом, оскільки ця платформа вже має готові моделі штучного інтелекту для семантичного аналізу та класифікації зображень, що значно спрощує впровадження інтелектуальних аналітичних можливостей у проєкт.

2.6 Реалізація компонентів системи

2.6.1 Реалізація ядра картографічного додатку

Головне що є в картографічному додатку це взаємодія з мапою, тому для цього використовується сервіс MapboxGL, який впроваджує WebGL для відображення інтерактивних та анімованих карт. Він забезпечує високу продуктивність та підтримку великого обсягу геоданих.

Перед тим як розпочати впровадження розпізнавання фрагментів мапи з використанням Mapbox GL, важливо ретельно розібратися з тим, як працюють шари у цьому картографічному інструменті. Mapbox GL відзначається гнучкістю та високою ефективністю завдяки концепції шарів, яка дозволяє ефективно організовувати та відображати географічні дані. Шари у Mapbox GL це ключовий елемент для відображення інформації на мапі. Кожне джерело представляє собою окремий набір географічних об'єктів, таких як точки, лінії або полігони, які можна стилізувати та налаштовувати в шарах [42]. Приклад зображено на рисунку 2.2.

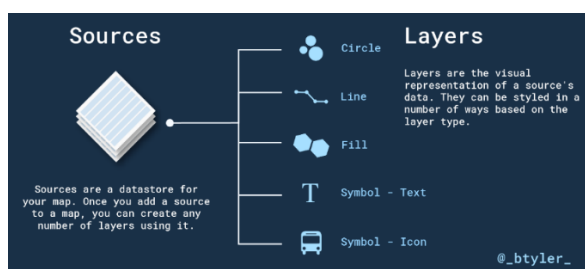


Рисунок 2.2 – Приклад структури шарів

Найбільш поширеним варіантом додавання просторових даних на мапу є додавання джерела векторних шарів, які розміщуються в самій бібліотеці Mapbox. Загалом дані, що передається, представляються в основному через JSON або XML. Таким чином, щоб безперешкодно взаємодіяти з існуючою логікою та стандартами, було введено такий формат як GeoJSON, що є відкритим форматом обміну геопросторовими даними [43]. Він використовується для представлення простих географічних об'єктів та відображення їх непросторових атрибутів. Цей формат базується на нотації об'єктів JSON, та призначений для кодування різноманітних структур географічних даних. Приклад показано на лістингу 2.1.

Лістинг 2.1 – Приклад структури GeoJSON

```
{ "type": "FeatureCollection",
  "features": [
    { "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [102.0, 0.5]
      },
      "properties": {
        "prop0": "value0"
      }
    },
    { "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]
        ]
      },
      "properties": {
        "prop0": "value0",
        "prop1": 0.0
      }
    }
  ]
}
```

Для створення відображення шару мапи використовується React-компонента, за допомогою бібліотеки Mapbox GL. Компонента має ім'я MainMap та використовується для рендерингу інтерактивної карти на веб-сторінці. Основні елементи коду включають в себе використання хука useRef для створення

посилання на DOM-елемента мапи, використання хука `useEffect` для виконання дій при монтуванні та розмонтуванні компоненти, і встановлення токена доступу Mapbox з оточення виконання. Компонент також має можливість передавати функцію `onLoaded`, яка викликається при завантаженні мапи.

Лістинг 2.2 – Головна компонента мапи

```
export const MainMap: React.FC<MainMapProps> = ({ onLoaded }) => {
  const mapNode = React.useRef(null);

  useEffect(() => {
    const node = mapNode.current;
    if (typeof window === 'undefined' || node === null) return;

    mapboxgl.accessToken = process.env.REACT_APP_MAP_TOKEN ?? '';
    const mapboxMap = new mapboxgl.Map({
      container: node,
      style: mapStore.mapType as string,
      hash: 'map'
    });

    mapboxMap.once('load', () => onLoaded(mapboxMap));

    return () => {
      mapboxMap.remove();
    };
  }, []);

  return <div ref={mapNode} className='map' />;
};
```

Після того як мапа була створена потрібно завантажити колекцію GeoJSON, для цього в хуку `useEffect` виконується код з лістингу 2.3. Тут спочатку додаються джерела різних шарів, а після чого вже додаються і самі шари стилів, які встановлюються під конкретне джерело.

Лістинг 2.3 – Код ініціалізації джерел шарів

```
const init = async () => {
  const mapData = await loadData();
  if (!mapData) return;
  for (const source of mapData.sources) {
    mainMap.addSource(source.id, {
```

```

        type: 'geojson',
        data: source.featureCollection,
        promoteId: 'id'
    });
}

for (const layer of mapData.layers) {
    if (mapData.sources.find(s => s.id === layer.source)) {
        mainMap.addLayer(layer as mapboxgl.AnyLayer);
    }

    if (layer.id.includes('label')) {
        continue;
    }
}
};

```

Дані завантажуються функцією `loadData`, яка використовує асинхронний підхід для завантаження даних на мапу. Основні кроки включають в себе отримання географічних координат області видимості мапи, потім встановлюється подія `'sourcedata'` для картографічного об'єкту `mainMap`. Після отримання даних встановлюється значення `performingUpdate` в глобальному стані `mapStore` на `false`, що означає що дані були завантажені.

На кінець, викликається асинхронна функція `updateMapData` об'єкту `mapStore`, яка отримує різні параметри, такі як географічні координати області, масштаб, координати верхнього лівого кута, висота та ширина канвасу мапи. Вона зробить запит на сервер, отримає дані та оновить їх у глобальному стані.

Всі ці дії призначені для оновлення даних на мапі та викликаються асинхронно для забезпечення коректності та продуктивності завантаження даних на мапу.

Лістинг 2.4 – Оновлення даних на мапі

```

const loadData = async () => {
    const bounds = mainMap.getBounds();
    const coords = [bounds.getWest(), bounds.getNorth()];

    mainMap.once('sourcedata', () => {
        mapStore.performingUpdate = false;
    });
};

```

```

return await mapStore.updateMapData(
  bounds.toArray(),
  Number.parseInt(mainMap.getZoom().toFixed(0)),
  coords,
  mainMap.getCanvas().height,
  mainMap.getCanvas().width
);

```

Функція `getMapData` у NestJS використовує асинхронні запити до бази даних мапи через клас сервісу та отримує дані картографії для вказаної області. Після чого йде обробка результатів відповідно до виду `GeoJSON`.

Лістинг 2.5 – Оброблювач запиту даних мапи

```

@Get()
async getMapData(@Query() areaQuery: AreaQuery): Promise<MapData> {
  const mapFeatures = await this.mapService.getAllMapFeatures(areaQuery);
  const types = await this.mapService.getFeatureTypesLayers();
  const layers: LayerSource[] = [];
  const sources: Source[] = [];

  for (const type of types) {
    for (const layer of type.layers) {
      LayerUtil.formatZoom(layer);
      layers.push({ ...layer, id: type.id + layer.id, source: type.id });
    }

    const featureCollection: FeatureCollection<GeometryProp, MapFeatureProps> =
    {
      type: 'FeatureCollection',
      features: [],
    };

    featureCollection.features.push(...mapFeatures.filter((feature) =>
    feature.properties.type?.id == type.id));
    if (featureCollection.features.length > 0) {
      sources.push({ id: type.id, featureCollection });
    }
  }

  return {
    version: 1,
    sources, layers, };
}

```

2.6.2 Реалізація сервісу зберігання файлів

У додатку для керування файлами був створений окремий сервіс, який включає завантаження, збереження та обробку файлів. Механізм роботи включає синхронізацію з хмаровою службою, такою як Dropbox. На лістингу 2.7 можна виділити асинхронну функцію для завантаження зображень в хмарне сховище. Основна ідея полягає в оптимізації зображення перед завантаженням та визначенні параметрів завантаження відповідно до типу файлу.

Спочатку відбувається оптимізація та зміна розміру зображення для заощадження дискового простору, де для цього використовується бібліотека `sharp` яка оптимізує та змінює розміри зображення, які зберігається у форматі JPEG з певною якістю, яка залежить від типу файлу (оригінал, мініатюра чи малий розмір). Розмір зображення також може бути змінений шляхом зміни ширини, і при цьому важливо уникнути збільшення розміру. Для завантаження у хмару Dropbox використовується об'єкт глобальний `dbx` для виклику методу `filesUpload`, який приймає параметри, такі як шлях, вміст файлу (зображення) тощо. Шлях формується на основі параметрів, таких як підпапка, ім'я файлу та типу файлу.

Лістинг 2.6 – Код завантаження файлів

```
private async uploadFile(data: Buffer, fileName: string, fileType: FileType,
width: number, subFolder: string): Promise<number> {
  console.log(width);

  const image = await sharp(data)
    .jpeg({ mozjpeg: true, quality: fileType == FileType.ORIGINAL ? 90 : 50 })
    .resize({ width: width, withoutEnlargement: true })
    .toBuffer();

  const res = await this.dbx.filesUpload({
    path: `/${subFolder ? `${subFolder}/` : ''}${fileName}${fileType ==
FileType.ORIGINAL ? '' : `_${fileType}`}${EXT}`,
    contents: image.buffer,
  });

  return res.status;
}
```

Оскільки ім'я завантаженого файлу (зображення) знаходиться у базі даних, то можна отримати зображення із хмари Dropbox через ендпоінт `getFeatureMedia`. На

лістингу 2.7 подано код, який представляє контролер у фреймворку, який відповідає за обробку HTTP-запитів типу GET для отримання медіафайлів з серверу. Функція має маршрут `/feature/media/:file`, що вказує на те, що в запиті очікується параметр `file` (назва файлу). Викликається метод `downloadFiles` сервісу `filesService`, якому передаються параметри, такі як ім'я файлу (`file`), підпапка та тип файлу. Далі через глобальний об'єкт `dbx` завантажується необхідний файл. Після чого встановлюється статус відповіді 200 та налаштовуються заголовки відповіді, такі як `Content-Disposition` та `Content-Type`. Ці налаштування вказують браузеру, що відповідь є медіафайлом для завантаження.

Лістинг 2.7 – Код отримання файлів

```
@Get('feature/media/:file')
async getFeatureMedia(@Param('file') file: string, @Res() res: Response,
@Query() optionsQuery: FileOptionsQuery) {
  try {
    const media = (await this.filesService.downloadFiles([file], { subfolder:
MEDIA_FOLDER, fileType: optionsQuery.type }))[0];

    res.writeHead(200, {
      'Content-Disposition': `attachment; filename="${media.filename}"`,
      'Content-Type': media.mimeType,
    });

    res.end(media.buffer);
  } catch (e) {
    console.log(e);

    throw new NotFoundException();
  }
}
```

2.6.3 Реалізація сервісу авторизації

У додатку для забезпечення реєстрації та авторизації використовується JSON Web Token (JWT) для забезпечення безпечного та ефективного механізму аутентифікації. Цей стандарт передбачає використання JSON-об'єкта для передачі інформації між сторонами у вигляді компактного та самостійного способу. Одним з типів інформації, який може бути переданий через JWT це дані для аутентифікації

користувача під час реєстрації у додатку. Приклад такого процесу зображено на рисунку 2.3.

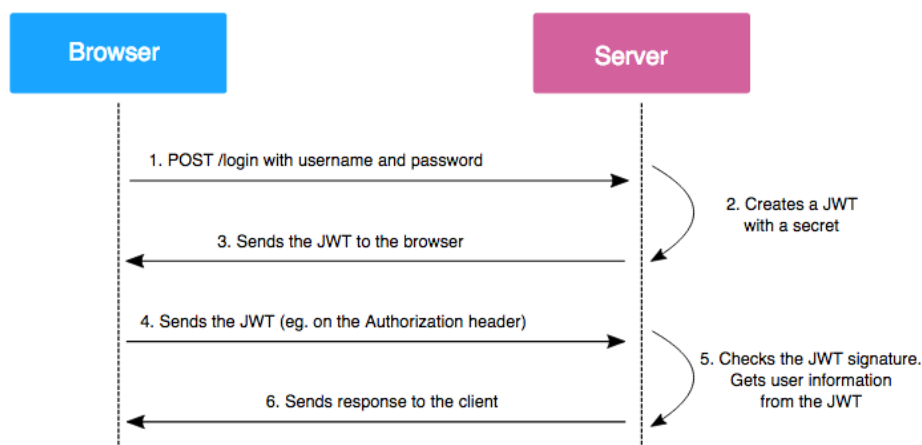


Рисунок 2.3 – Приклад застосування JWT токена

Процес реєстрації у додатку виглядає наступним чином. Після того, як користувач вводить свої реєстраційні дані, наприклад такі як ім'я користувача та пароль, сервер обробляє ці дані та перевіряє їх на вірність. Після успішної перевірки сервер створює JWT, який містить інформацію про користувача та його ролі в системі. Приклад подано на лістингу 2.8.

Сам JWT складається з трьох частин: заголовка, тіла та підпису. У заголовку вказується тип токена та алгоритм шифрування. Тіло містить інформацію про користувача. Підпис гарантує цілісність даних та перевіряє їх автентичність.

Отримавши JWT, клієнт зберігає його та використовує при подальших запитах до серверу, який вносить у спеціальний заголовок запиту. Сервер перевіряє підпис та дешифрує тіло, отримуючи інформацію про користувача. Це дозволяє забезпечити безпеку та автентифікацію передачі пароля кожного разу при запиті.

Використання JWT для реєстрації у додатку дозволяє забезпечити ефективний та безпечний спосіб обміну даними між клієнтом та сервером. Цей стандарт забезпечує не лише автентифікацію, а й збереження конфіденційності та цілісності інформації.

Лістинг 2.8 – Авторизації користувача

```

@UseInterceptors (AvatarInterceptor)
@Post('register')
async register(@Body() createUserDto: CreateUserDto) {
  if (await this.usersService.findUserByUsername(createUserDto.username)) {
    throw new HttpException('User with this login is already exists',
HttpStatus.CONFLICT);
  }

  if (await this.usersService.findUserByEmail(createUserDto.email)) {
    throw new HttpException('User with this email is already exists',
HttpStatus.CONFLICT);
  }

  return await this.authService.createToken(await
this.authService.register(createUserDto));
}

@Post('login')
async login(@Body() loginUserDto: LoginUserDto) {
  const user = await this.authService.login(loginUserDto);

  if (!user) {
    throw new UnauthorizedException();
  }

  return await this.authService.createToken(user);
}

```

Крім того можна забезпечити авторизацію через сторонній сервіс, такий як Google, для того щоб була можливість швидко авторизуватися та почати користуватися сайтом.

Лістинг 2.9 – Компонента для авторизації через Google

```

<GoogleLogin
  className='signIn__googleBtn'
  clientId={process.env.REACT_APP_GOOGLE_CLIENT_ID as string}
  buttonText={t('GOOGLE_SIGN_IN')}
  onSuccess={handleGoogleLogin}
  onFailure={handleGoogleFailure}
  cookiePolicy='single_host_origin'
/>

```

Спочатку, на фронті потрібно додати компоненту з лістингу 2.9, яка буде показувати вікно для вибору відповідного акаунту Google. Коли користувач обрав потрібного користувача, буде викликано функцію `handleGoogleLogin` при успішній дії, або `handleGoogleFailure` при виникненні помилки.

У функції буде зроблено запит на сервер з відповідним токеном. На лістингу 2.10 подано код, який представляє собою обробник запиту для цього маршруту з методом POST. У тілі запиту буде об'єкт з одним полем `"tokenId"`. Спочатку створюється новий екземпляр `OAuth2Client` за допомогою ідентифікатора клієнта, отриманого з оточення. Виконується перевірка ідентифікаційного токenu (`tokenId`), отриманого з тіла запиту, за допомогою створеного `OAuth2Client`. З отриманого об'єкта `ticket` витягуються такі дані як ім'я, електронна пошта та аватар користувача. Далі викликається метод сервісу користувачів для пошуку наявного користувача з такою електронною поштою, де після чого перевіряється наявність користувача в базі даних. Якщо користувач існує, викликається метод оновлення користувача з типом, який вказує що даний акаунт був авторизований як Google-користувач. Якщо користувача немає, тоді викликається метод реєстрації нового Google-користувача. Після чого повертається токен автентифікації

Лістинг 2.10 – Авторизації через Google

```
@Post('google')
async googleLogin(@Body() data: { tokenId: string }) {
  const client = new OAuth2Client(process.env.CLIENT_ID);
  const ticket = await client.verifyIdToken({
    idToken: data.tokenId,
    audience: process.env.CLIENT_ID,
  });

  const { name, email, picture } = ticket.getPayload();

  const user = await this.userService.findUserByEmail(email);
  const googleUserDto: GoogleUserDto = {
    username: name,
    email,
    googleAvatar: picture,
  };
}
```

```

    if (user) {
        return await this.authService.createToken(await
this.authService.updateAsGoogleUser(user, googleUserDto));
    } else {
        return await this.authService.createToken(await
this.authService.registerAsGoogleUser(googleUserDto));
    }
}

```

2.6.4 Реалізація сервісу аналізу супутникових знімків

Для аналізу супутникових знімків спочатку потрібно розробити інтерфейс який буде впроваджувати можливість вибору області на мапі для подальшого аналізу. Тому була розроблена компонента яка дозволяє обирати область на шарі мапи. Для цього в залежності від типу обраного режиму картографування з'являється відповідний інструмент. Код наведено на лістингу 2.11.

Лістинг 2.11 – Код вибору режиму картографування

```

switch (editorStore.drawMode) {
    case GeometryConstant.POLYGON:
        mode = 'draw_polygon';
        break;
    case GeometryConstant.MULTI_LINE_STRING:
        mode = 'draw_line_string';
        isLine = true;
        break;
    case GeometryConstant.MULTI_POLYGON:
        isPolygon = true;
        mode = 'draw_polygon';
        break;
}

const mapboxDraw = new MapboxDraw({
    displayControlsDefault: false,
    controls: {
        line_string: isLine,
        polygon: isPolygon,
        trash: true
    },
    defaultMode: mode
});

mainMap?.addControl(mapboxDraw, 'bottom-right');

setDraw(mapboxDraw);

```

Коли користувач завершив виділення області, виконується функція `onCompleteDrawing`, в якій спочатку беруться намальовані об'єкти (features) та в залежності від обраного режиму формується масив координат для подальшої обробки.

Лістинг 2.12 – Функція завершення картографування

```
const onCompleteDrawing = () => {
  const features = draw?.getAll().features;
  if (!features) {
    return;
  }

  if (editorStore.drawMode == GeometryConstant.POLYGON) {
    const coordinates = (features[0].geometry as Polygon).coordinates;
    editorStore.setCreatedGeometry({ type: GeometryConstant.POLYGON, coordinates
  });
  } else if (editorStore.drawMode == GeometryConstant.MULTI_POLYGON) {
    const coordinates: Position[][][] = [];

    for (const feature of features) {
      coordinates.push((feature.geometry as Polygon).coordinates);
    }

    editorStore.setCreatedGeometry({ type: GeometryConstant.MULTI_POLYGON,
coordinates });
  } else if (editorStore.drawMode == GeometryConstant.MULTI_LINE_STRING) {
    const coordinates: Position[][] = [];

    for (const feature of features) {
      const lineCoordinates = (feature.geometry as LineString).coordinates;
      lineCoordinates.pop();
      coordinates.push(lineCoordinates);
    }

    editorStore.setCreatedGeometry({ type: GeometryConstant.MULTI_LINE_STRING,
coordinates });
  }

  onComplete();
};
```

Функція `handleOnSubmit` виконує передачу ведених об'єктів до серверу для подальшої обробки. Спочатку йде перевірка наявності потрібних даних. Після чого

створюється об'єкт типу "Feature", який містить властивості, геометрію та ідентифікатор. Далі в залежності від обраного режиму об'єкт надсилається для додавання на мапу чи для подальшого аналізу на сервері. Коли сервер обробив запит, то тоді оновлюється джерело даних на головній мапі.

Лістинг 2.13 – Функція відправки виділеного об'єкту

```
const handleOnSubmit = handleSubmit(async data => {
  if (!editorStore.selectedFeatureType) {
    setErrorData({ ...errorData, type: t('TYPE_REQUIRED') });
    return;
  }

  if (!editorStore.createdGeometry) {
    setErrorData({ ...errorData, coordinates: t('COORDINATES_REQUIRED') });
    return;
  }

  setLoading(true);

  try {
    const createdFeature: Feature<GeometryProp, CreateFeatureProps> = {
      type: 'Feature',
      properties: { type: editorStore.selectedFeatureType.id, category:
selectedCategory?.id, links, ...data },
      geometry: toJS(editorStore.createdGeometry),
      id: new Date().getTime()
    };

    const collection =
      editorStore.editType === EditType.REGULAR ? await
mapStore.addFeature(createdFeature, files) : await
mapStore.analyzeFeature(createdFeature);
    if (collection) {
      (mainMap?.getSource(editorStore.selectedFeatureType.id) as
GeoJSONSource).setData(collection);
    }

    reset();
    resetData();
    onSubmit();
  } catch (e) {
    console.error(e);
    resetData();
  }
});
```

Сервер обробляє запит POST за маршрутом 'feature/analyze'. Декоратор `@UseGuards(JwtAuthGuard)` вказує на використання кастомного гварду `JwtAuthGuard` для автентифікації користувача за допомогою токена JWT. Далі отримується ідентифікатор користувача з об'єкта запиту, який був розширений під час аутентифікації. Після чого йде підготовка даних для обробки, де спочатку створюється об'єкт `processingData` з типом запиту та координатами геометрії, які отримані з об'єкта `featureDto`. Потім викликається метод `performAnalyzeRequest` сервісу мапи, який ініціює процес аналізу для переданих даних, оскільки процес може зайняти багато часу, то повертається об'єкт процесу обробки з ідентифікатором. Для цього використовується цикл `while`, який чекає, доки статус обробки не буде закінчено. Тому в кожній ітерації циклу викликається метод `performAnalyzeRequestStatus` для отримання поточного статусу обробки. Після завершення обробки отримується об'єкт `featureCollection` з результатами аналізу шляхом виклику методу `performAnalyzeRequestResult`. Потім метод `addProcessedMapFeatures` додає оброблені об'єкти до бази даних. Коли усі процеси були виконані, то знайдені об'єкти повертається у вигляді GeoJSON як результат обробки супутникового знімку.

Лістинг 2.14 – Функція аналізу виділеної площі

```
@UseGuards(JwtAuthGuard)
@Post('feature/analyze')
async analyzeMapFeature(
  @Body() featureDto: Feature<GeometryProp, CreateFeaturePropsDto>,
  @Request() req,
): Promise<FeatureCollection<GeometryProp, MapFeatureProps>> {
  const userId = (req.user as UserPayload).id;

  const processingData = this.mapService.buildProcessingData({
    type: this.getAnalyzeRequestType(),
    coordinates: featureDto.geometry.coordinates,
  });

  const processing = await this.mapService.performAnalyzeRequest({ data:
processingData });

  let status = processing.status;
  while (status === ProcessingStatus.IN_PROGRESS) {
    await timer(ANALAZING_TIMEOUT);
```

```

    const processingStatus = await this.mapService.performAnalyzeRequestStatus({
      processingId: processing.id,
      data: processingData,
    });

    status = processingStatus;
  }

  const featureCollection = await this.mapService.performAnalyzeRequestResult({
    processingId: processing.id,
    data: processingData,
  });

  await this.mapService.addProcessedMapFeatures(featureCollection, userId,
featureDto);

  return featureCollection;
}

```

2.7 Висновки за розділом 2

Важливим аспектом розробки є вибір відповідних інструментів та фреймворків. У проєкті було вирішено використовувати фреймворки NestJS, MongoDB та React, які забезпечують потрібний функціонал для реалізації серверної та клієнтської частини проєкту, а також відповідають сучасним вимогам до розробки програмного забезпечення.

Також було проаналізовано різні архітектури комп'ютерних мереж, та визначено що під розроблену систему підходить саме мікросервісна архітектура. Після чого було розглянуто бібліотеки для аналізування зображень, з яких було обрано сервіс Mapflow AI.

Після чого було розроблено основні компоненти системи, такі як ядро картографічного додатку, сервіс керування файлами, сервіс авторизації, а також систему для аналізу супутникових знімків.

3 ДОСЛІДЖЕННЯ СИСТЕМИ

У роботі картографа є багато інструментів, які допомагають прискорити процес. Зазвичай такі інструменти вбудовані в ГІС, наприклад малювання геометричних примітивів таких як крапка, полігон, класифікація фрагменту мапи відповідно до його категорії, але такий процес займає певний час щоб обробити лише єдиний супутниковий знімок та інші геопросторові дані. Однак сучасні технології в сфері картографії можуть прискорити і поліпшити цей процес. При роботі зі супутниковими знімками, картографи використовують спеціальні програми для обробки та інтерпретації цих зображень, які можуть автоматично розпізнавати та класифікувати об'єкти на знімках, що значно прискорює процес створення мап, тому спочатку варто роздивитися які саме знімки потрібно аналізувати щоб отримати довільний результат.

Для аналізу можна використовувати знімки з відкритих проєктів, таких як Landsat або Sentinel, але вони мають занадто низьку просторову роздільну здатність, що приведе к помилкам при обробці. Тому варто зробити фокус на високу оптичну роздільну здатність, що є найбільш затребуваним комерційним сегментом та найбільш застосовуваним у завданнях картографії. Можна відмітити що Mapbox Satellite використовує глобальну мозаїку "Maxar Vivid", яка представляє задовільну якість зображень.

3.1 Актуальність впровадженого підходу

Після того як було розглянуто основні переваги використання штучного інтелекту (ШІ) при картографуванні супутникових знімків, також важливо розглянути і актуальність цієї технології в контексті сучасних потреб у геоінформаційних системах.

За останні кілька років відбулися значні зміни в сфері картографування та аналізу геопросторових даних. Зростання обсягів даних, їх різноманітність та необхідність швидкого оновлення інформації створюють проблеми, які можна ефективно вирішити за допомогою ШІ. Актуальність використання штучного інтелекту у картографії полягає в здатності цієї технології адаптуватися до зростаючих потреб у точності та ефективності аналізу геопросторових даних. Сучасні алгоритми машинного навчання можуть самостійно вивчати та адаптуватися до нових умов, що робить їх ефективними в умовах мінливого середовища.

Однією з ключових актуальних проблем є необхідність оперативного та точного моніторингу природних катастроф, змін клімату та інших надзвичайних ситуацій. Тому з використанням ШІ можливо створити системи, які автоматично реагують на зміни у геопросторовому середовищі та надають оперативну інформацію для прийняття рішень.

Актуальність ШІ також виявляється у здатності ефективно обробляти великі обсяги даних у реальному часі. Це важливо для різних сфер, включаючи транспорт, міське планування, сільське господарство та екологію, де потрібно миттєво реагувати на зміни та вирішувати завдання в реальному часі.

Наприклад, можна створити модель яка може використовуватися для моніторингу будівництва на супутникових знімках, як варіант до сценарію автоматичного картографування міської забудови. Такий варіант можна застосувати до щорічної супутникової зйомки вибраних міст, де можна поррахувати порівняльну динаміку будівництва, або порівняти прогрес за роками. Крім того ШІ можна використовувати якщо в ході будівництва потрібна інформація з більшою чистотою чи у певні терміни, крім щорічної статистики.

Отже, актуальність використання ШІ у картографії супутникових знімків зумовлена необхідністю прискорення та оптимізації процесів аналізу геопросторових даних у відповідь на сучасні проблеми та потреби суспільства.

3.2 Призначення й умови застосування програми

Розроблений додаток має можливість створення об'єктів як вручну (з використанням спеціальних інструментів), так і за допомогою ШІ для розпізнавання фрагментів. На рисунку 2.1 представлена основна базова сторінка програми. Вона включає в себе блок пошуку, в якому розміщена кнопка для відкриття меню. У верхньому куті розташований блок авторизації, а при наявності авторизації користувача він замінюється на його зображення, яке також відкриває доступ до налаштувань облікового запису.

У лівому нижньому куті знаходиться блок вибору варіанту відображення мапи та інструменти для створення нового об'єкту. Праворуч розташована панель інструментів керування мапою, за допомогою якої можна взаємодіяти з самою мапою. Тут доступні інструменти для створення будівлі, прокладання шляху, розмітки площі, а також розпізнавання фрагментів мапи.

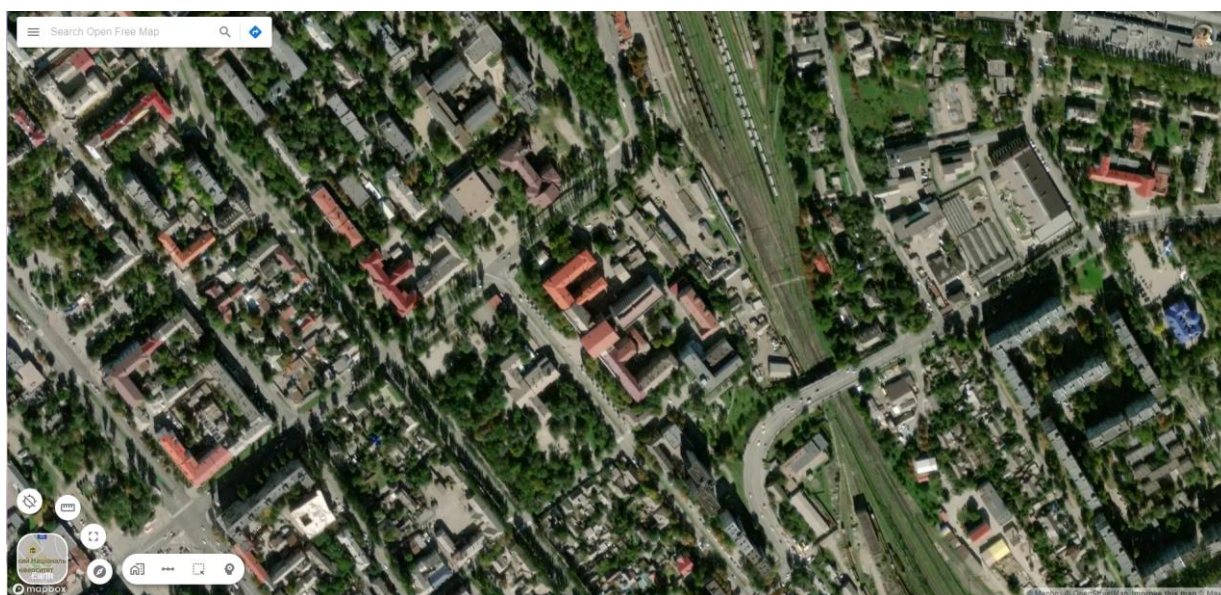


Рисунок 3.1 – Головна сторінка

Наприклад, можна створити картографічне відображення університетської будівлі. Для взаємодії з мапою слід скористатися панеллю інструментів, розташованою у правому нижньому куті, яку зображено на рисунку 3.2.

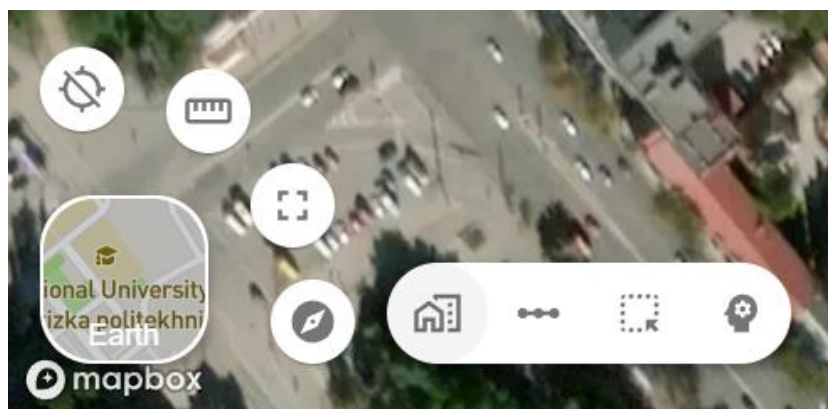


Рисунок 3.2 – Інструменти взаємодії з мапою

Після цього буде запропоновано обрати тип будівлі, із варіантів можуть бути Building – загальний випадок, Factory – промисловість, Private house – приватні будинки тощо. Тип будівлі впливає на те як створений полігон буде відображатися на мапі. В нашому випадку можна обрати звичайний Building. Приклад такого меню зображено на рис 3.3.

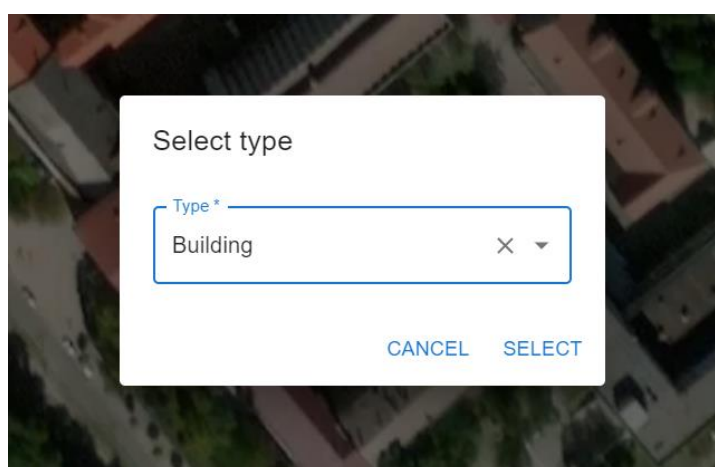


Рисунок 3.3 – Вибір типу будівлі

Після цього необхідно вручну, за допомогою курсора, виділити будівлю університету. Крім того, є можливість редагувати кожну з вершин полігону для отримання більш досконалої та якісної форми.

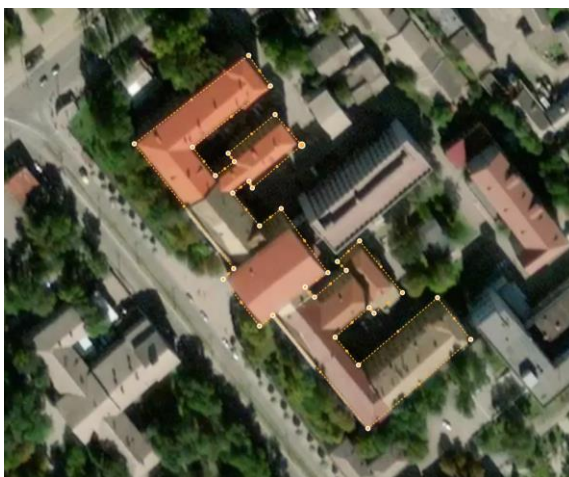


Рисунок 3.4 – Створення форми університету

Коли об'єкт буде повністю готовий, наступним етапом буде надання інформації про те, що саме було створено. На зображенні видно, що потрібно вказати такі дані як: назва, опис об'єкту тощо. Тип будівлі автоматично визначається відповідно до обраного формату, крім того є можливість вибору категорії об'єкту для зручного фільтрування під час пошуку. Також можна вказати інші контактні дані.

Add a place

Place details
Provide some information about this place. If this place is added to Maps, it will appear publicly.

Name *
Національний університет «Запорізька політехніка»

Description
НУЗП є провідним державним закладом вищої освіти на Південному Сході України.

Type *
Building

Category
University

Coordinates *
47.81778N, 35.16428E

Contacts

Address
вул. Жуківського, 64

Phone
38(061)7642506

Wikipedia
https://en.wikipedia.org/wiki/Zaporizhzhia_Polytechnic

Link
<https://zpu.edu.ua/>

Place photos
Add helpful photos like storefronts, notices, or signs.

Рисунок 3.5 – Додавання інформації про об'єкт

Створений об'єкт буде опрацьований на сервері та доданий до бази даних. Після цього він автоматично з'явиться на мапі для всіх користувачів. Приклад створеного об'єкту з доданою інформацією зображено на рисунку 3.6.

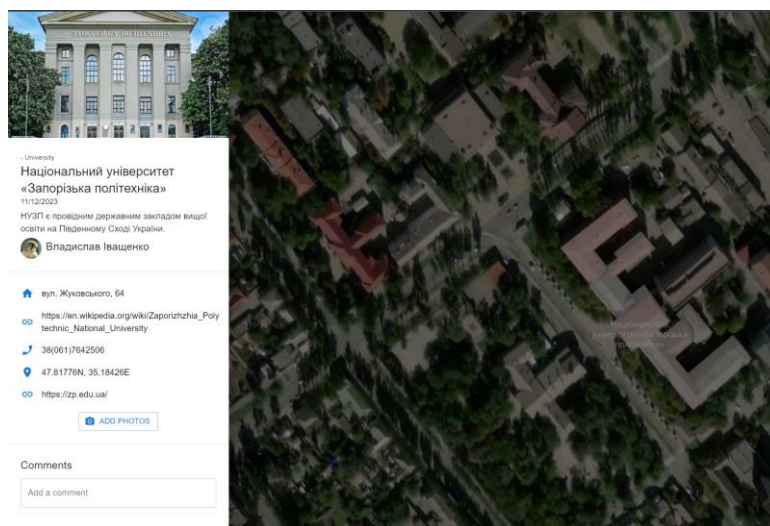


Рисунок 3.6 – Створений об'єкт будівлі університету

Як бачимо, створення навіть одного такого об'єкту вимагає певного часу, який може займати від декількох хвилин до години, залежно від складності об'єкта на супутниковому знімку.

З метою спрощення цього процесу можна використовувати штучний інтелект для аналізу фрагментів на супутниковому знімку. Для цього слід обрати спеціальний інструмент з панелі взаємодії з мапою (див. рисунок 3.2) та виділити необхідну область для подальшого аналізу. На рисунку 3.7 виділено область усієї території університету з усіма корпусами та іншими технічними спорудами.

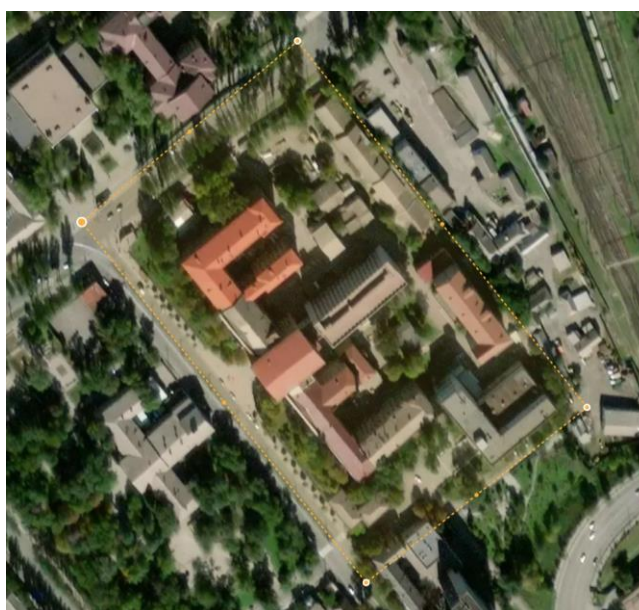


Рисунок 3.7 – Виділення області для аналізу

Після відправлення обраної області необхідно зачекати протягом певного часу, загалом десь кілька хвилин, і в результаті буде отримано об'єкти, які представлені на рисунку 3.8. Як бачимо, усі будівлі було правильно знайдено та відображено на мапі. Далі залишається лише додати додаткову інформацію для кожної будівлі (або перед початком розпізнавання ввести загальну інформацію).

Цей підхід дозволив закартографувати певну частину мапи (див рисунок 3.8) всього лише за декілька хвилин, порівняно з тим, якби було закартографовано лише одну будівлю.



Рисунок 3.8 – Створенні об'єкти для всієї площі університету

Більше того, можливо також провести розпізнавання значно більшої площі мапи. Наприклад, було обрано певну частину міста (див рис 3.9) для розпізнавання, і в результаті отримано об'єкти, які представлені на рисунку 3.10. Якщо ручне картографування зайняло би більше години, то розпізнавання фрагментів за допомогою ШІ зайняло лише декілька хвилин без врахування додавання додаткової інформації.



Рисунок 3.9 – Виділення певної площі



Рисунок 3.10 – Створені об'єкти на мапі

3.3 Аналіз результатів розпізнавання фрагментів мапи

Як бачимо, використання штучного інтелекту, відіграє ключову роль у прискоренні та поліпшенні процесу картографування супутникових знімків. Ця технологія надає можливості для автоматизації багатьох етапів обробки та аналізу великих обсягів геопросторових даних, що значно спрощує і прискорює роботу картографувальників та геоінформаційних спеціалістів. Тому можна відмити що

головною перевагою використання ШІ у картографуванні супутникових знімків є його здатність автоматично розпізнавати об'єкти та типи об'єктів на знімках.

Можна навести таблицю, як таке прискорення картографування буде поліпшувати час на відміну від класичного ручного картографування. Потрібно зазначити, що часові затрати можуть суттєво варіюватися в залежності від конкретного завдання, обсягу даних та характеристик забезпечення яке використовує ШІ. Тому можна навести загальний порівняльний приклад часових витрат між класичним картографуванням та картографуванням за допомогою штучного інтелекту.

В таблицю 3.1 було додано декілька пунктів, які варто розглядати при порівнянні різних методів картографування:

- обробка знімків, цей етап включає нанесення полігонів на супутниковий знімок, з подальшою корекцією об'єктів відносно один одного;

- класифікація об'єктів на знімках, на цьому етапі відбувається надання кожному об'єкту певного класу, що може включати в себе визначення типу рослинності, водоймищ, шляхів, будівель тощо;

- оновлення та аналіз мап, цей етап включає оновлення через певний проміжок часу, додавання нових об'єктів, видалення застарілих даних та корекцію інформації.

Таблиця 3.1 – Витрачений час на різні методи картографування

Етап роботи	Класичне картографування	Картографування з використанням ШІ
Обробка знімків	Від декількох годин до декількох днів	Автоматично виконується після отримання знімку
Класифікація об'єктів на знімках	Залежно від складності, декілька годин до декількох днів.	Може бути виконано за декілька хвилин або навіть секунд
Оновлення та аналіз мап	Залежить від періодичності оновлення, може займати певний час	Оновлення може бути автоматизованим

Цей порівняльний аналіз показує, що використання ШІ може значно зменшити часові затрати на кожному етапі процесу картографування, забезпечуючи більш ефективний та оперативний аналіз супутникових даних.

Під час етапу обробки знімків класичне картографування традиційно виконується вручну або за допомогою спеціалізованих програм, що значно уповільнює процес. Якщо брати алгоритми ШІ, вони можуть бути використані для автоматичної обробки знімків, тому використання ШІ може значно поліпшити швидкість та точність на кожному з цих етапів.

Також варто розглянути кожен з етапів, на першому етапі класифікації при класичному картографуванні потрібно буде визначати клас будівлі вручну, а при використанні ШІ це може бути зроблено автоматично. На останньому етапі варто відмітити, що традиційно картографічні дані оновлюються через певні періоди, тому потрібно робити оновлення вручну або за допомогою спеціалізованих інструментів. Цей процес може включати в себе ручне додавання нових та видалення застарілих об'єктів, а також корекцію інформації. Натомість з використанням ШІ можна автоматизувати цей процес. Система на основі ШІ може автоматично аналізувати нові супутникові знімки та виявляти зміни, що дозволить швидше оновлювати картографічні дані та реагувати на зміни у реальному часі.

Як бачимо, використання ШІ може значно поліпшати швидкість картографування, але потрібно також врахувати можливу похибку, бо сучасні алгоритми хоча і можуть обробити супутникові знімки, але різні чинники можуть впливати на результат цього процесу. Наприклад, можна виділити такі як: неякісні вихідні знімки, неточність алгоритму тощо. Тому варто подивитися на виникнення можливої похибки на кожному з етапів. Наприклад, якщо брати до уваги рисунок 3.10 можна побачити що одна будівля була оброблена неякісно і її потрібно картографувати вручну. Приклад такої цієї будівлі зображено на рисунку 3.12.

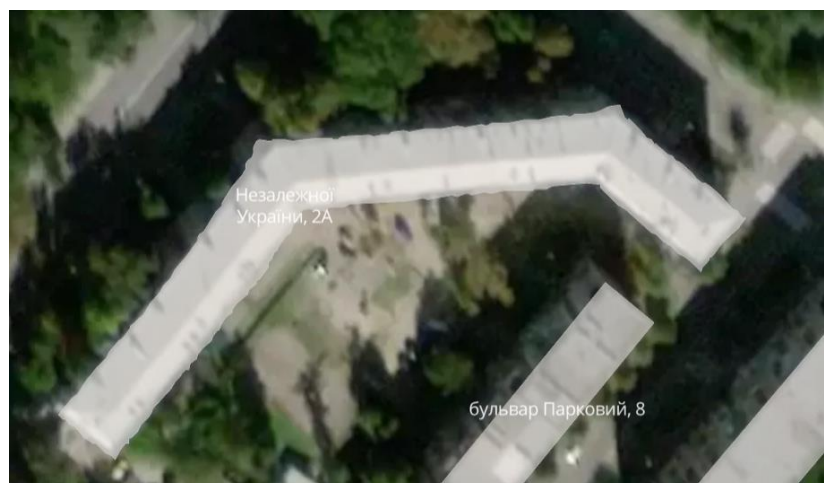


Рисунок 3.12 – Невірно створений об’єкт

Тому точність картографічного аналізу на основі ШІ порівняно з класичним картографуванням може значно відрізнятись в залежності від конкретного завдання, якістю вхідних даних та налаштувань самого алгоритму ШІ. Нижче представлено загальну таблицю 3.2, де показано можлива різниця в точності між двома методами.

Таблиця 3.2 – Точність різних підходів

Етап роботи	Класичне картографування	Картографування з використанням ШІ
Обробка знімків	Залежить від використаних інструментів, загалом висока точність	Точність може залежати від якості алгоритмів ШІ, може варіюватися
Класифікація об’єктів на знімках	Зазвичай висока точність, але залежить від навичок фахівця	Точність може бути високою, але залежить від тренуваної моделі
Оновлення та аналіз мап	Зазвичай висока точність при актуальних даних	Точність залежить від алгоритмів ШІ

Важливо враховувати, що точність використання ШІ може значно покращитися з розвитком та вдосконаленням алгоритмів, а також з використанням більш різноманітних навчальних даних. Тому варто зазначити що при використанні ШІ потрібно перевіряти результат, та за потреби, редагувати результат.

Проаналізувавши такі чинники як швидкість та точність картографування (див таблицю 3.3), можна навести таблицю як таке використання штучного інтелекту зможе заощадити витрати. Тобто, можна врахувати такий аспект, що знадобиться менше картографів для картографування знімків. Тому використання ШІ у картографії може призвести до економії грошей через автоматизацію та оптимізацію робочих процесів. Це може включати в себе необхідність меншої кількості картографів або розробників для виконання тих самих завдань.

Таблиця 3.3 – Кількість фахівців при різних підходах

Етап роботи	Класичне картографування	Картографування з використанням ШІ
Обробка знімків	Вимагає значного кількості фахівців	Може значно зменшити кількість фахівців
Класифікація об'єктів на знімках	Вимагає певну кількість фахівців	Може зменшити кількість фахівців, але все ще потребує певну кількість для перевірки результату
Оновлення та аналіз мап	Вимагає значного кількості фахівців	Може значно зменшити кількість фахівців

Можна відмітити, що цінність використання ШІ включає в себе не лише зменшення витрат на оплату праці, але і підвищення продуктивності завдяки швидшим та більш точному аналізу супутникових знімків.

Оскільки ці критерії різних методів значно впливають на картографування, потрібно роздивитися використання штучного інтелекту на практиці. Для цього було обрано певний сектор (див рисунок 3.11), який потрібно спочатку закартографувати вручну, а потім цю ж саму область за допомогою використання ШІ. Варто позначити, що картографування повинно відбуватися з повним додаванням потрібної інформації про створений об'єкт.

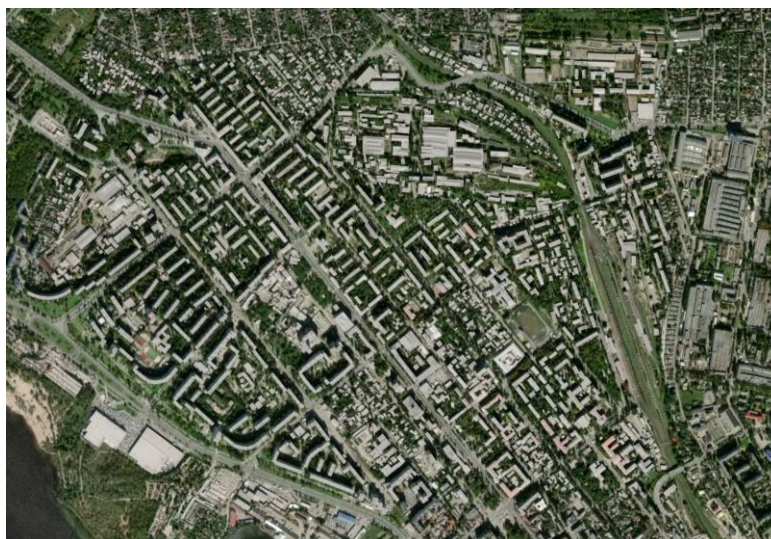


Рисунок 3.11 – Область для картографування

Тому для виконання даного завдання було обрано популярний сервіс OpenStreetMap (OSM), який дозволяє картографувати супутникові знімки. Він є відомим глобальним проектом, який надає відкритий доступ до геоданих, зібраних за допомогою ручного картографування. Проте, OSM покладається переважно на ручному внесенні даних, що може бути часозатратним і вимагати великої кількості людських ресурсів.

Натомість, власна розробка інтегрує можливості штучного інтелекту (ШІ) для автоматизації процесу картографування, тому можна продемонструвати переваги ШІ у картографуванні. Результати завдання наведені на рисунку 3.12.

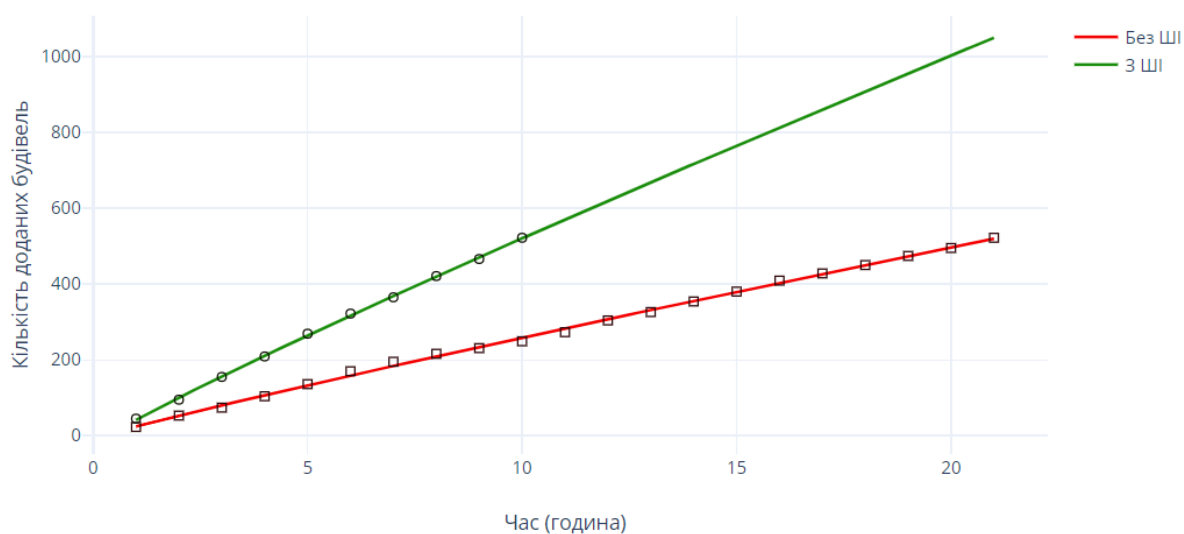


Рисунок 3.12 – Графік доданих будівель за проміжок часу

Як можна побачити з графіку, процес картографування з використанням штучного інтелекту (представлений зеленою лінією) виявився значно швидшим, через те що всі будівлі (522 одиниці) були додані приблизно за 10 годин. У той же час, процес картографування без використання штучного інтелекту показав гірші результати, адже таку ж кількість будівель вдалося додати лише після 21 години. Початкова швидкість картографування в обох випадках була схожою, але з плином часу штучний інтелект дозволив значно прискорити процес, автоматизувавши більшу частину роботи, яка раніше потребувала ручного втручання. Враховуючи, що кількість доданих будівель з кожною годиною збільшувалась, можна зробити припущення, що за допомогою картографування з використанням штучного інтелекту через 21 годину можна було б додати приблизно 1000 будівель, що є майже вдвічі більше, ніж при звичайному ручному картографуванні.

Це порівняння показує, що використання ШІ в картографуванні може принести величезну користь, зокрема в плані ефективності та швидкості. Тому можна визначити наскільки такий варіант прискорить картографування. Для цього було використано формулу 3.1, за допомогою якої можна вирахувати наскільки було поліпшено час картографування.

$$\text{Прискорення} = \left(\frac{\text{Час} - \text{Час}_{\text{ШІ}}}{\text{Час}} \right) \cdot 100\% \quad (3.1)$$

де Час – витрачений час на картографування будівель без використання ШІ;

Час_{ШІ} – витрачений час з використанням ШІ.

Тобто, потрібно знайти абсолютно економію часу штучного інтелекту, що є різницю між часом без ШІ і часом з ШІ, та поділити на витрачений час без використання ШІ. Тобто, формула обчислює частку часу, збереженого за допомогою штучного інтелекту, і виражає її у відсотках. Цей відсоток показує, наскільки швидшим є процес картографування, коли використовується ШІ.

Після картографування будинків з рисунку 3.11, було пораховано що для додавання більше ніж 500 будівель ручним методом потрібно приблизно 21 година, натомість за допомогою штучного інтелекту лише трішки більше за 10 годин.

$$\text{Прискорення} = \left(\frac{20.7 - 10.2}{20.7} \right) \cdot 100\% = 50.7\% \approx 51\% \quad (3.2)$$

З формули 3.2 видно, що відсоткове збільшення швидкості картографування за допомогою ШІ становить приблизно 51%. Таке значне збільшення підкреслює високу ефективність, яку можна досягти за допомогою картографування з використанням ШІ. Тому система, яка використовує ШІ для картографування, має великий потенціал, на відміну від поточної системи OpenStreetMap, де використовується лише ручний спосіб додавання будівель.

3.4 Висновки за розділом 3

В цьому розділі було застосовано розроблений додаток для розпізнавання фрагментів мап супутникових знімків. Її функціональність і ефективність використання підтверджена аналізом результатів, де було надано таблиці порівнянь класичного підходу картографування, а також підходу з використанням з ШІ.

Також було розглянуто, актуальність впровадженого підходу для прискорення картографування супутникових знімків. Цей підхід не лише спрощує процес аналізу геопросторової інформації, але й значно збільшує швидкість обробки великих обсягів даних. Такий підхід має великий потенціал для подальшого розвитку та оптимізації методів картографії супутникових знімків.

ВИСНОВКИ

Проведено аналіз архітектури систем, які використовують нейронні мережі для розпізнавання та класифікації фрагментів мапи. Також було виявлено, що сучасні технології нейронних мереж глибокого навчання демонструють високу ефективність у вирішенні складних завдань аналізу зображень.

У розробці проєкту було проведено вибір інструментів та фреймворків для забезпечення необхідного функціоналу серверної та клієнтської частини проєкту. Аналіз архітектури комп'ютерних систем визначив мікросервісну архітектуру як відповідну для системи. Після чого було розроблено комп'ютерну систему, яка дозволяє розпізнавати фрагменти мапи зі супутникових знімків.

Після розробки було проведено аналіз нового підходу та виявлено високу функціональність і ефективність в порівнянні з класичним підходом до картографування, який використовується в інших системах, таких як OpenStreetMap. Зазначено, що використання штучного інтелекту не лише прискорює процес аналізу геопросторової інформації та збільшує швидкість обробки великих обсягів даних, а також може значно заощадити витрати, оскільки для картографування супутникових знімків знадобиться менша кількість фахівців.

Запропонований підхід має значний потенціал для подальшого розвитку та актуальне застосування, що може прискорити різноманітні процеси, проте така система також має певні недоліки. По-перше, недостатня точність розпізнавання може призводити до невірного розпізнавання фрагментів мапи, що може потребувати ручного внесення змін. Також, погана якість вхідних даних, таких як низька роздільна здатність супутникових знімків, може суттєво вплинути на ефективність та правильність роботи системи. Таким чином, проєкт, що використовує нейронні мережі для розпізнавання фрагментів мапи на супутникових знімках, демонструє великий потенціал та можливості сучасних технологій нейронних мереж глибокого навчання для вирішення складних задач аналізу зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Як розібратися з Computer Vision URL: <https://dev.ua/news/yak-roz%D1%96bratisya-z-computer-vision> (дата звернення: 01.10.2023).

2 Комп'ютерний зір URL: https://uk.wikipedia.org/Комп%27ютерний_зір (дата звернення: 01.10.2023).

3 Капралов Є. Г., Кошкарьов А. В., Тікунов В. С. Основи геоінформатики. Видавничий центр «Академія», 2004. 352 с.

4 Addressing GIS Technology to tackle Biodiversity loss URL: <https://www.linkedin.com/pulse/addressing-gis-technology-tackle-biodiversity-loss/> (дата звернення: 01.10.2023).

5 Пінґде Фу, Цзюлінь Сунь. Веб-ГІС. Принципи та застосування. Редлендз (Каліфорнія) : Esri Press, 2010. 356 с.

6 Мацуга О., Архангельська Ю., Єрещенко Н. Навчальний посібник до вивчення курсу «Інформаційні технології розпізнавання образів». РВВ ДНУ, 2016. 356 с.

7 Viola P., Jones M. Rapid object detection using a boosted cascade of simple features. The Institute of Electrical and Electronics Engineers, 2001. 518 p.

8 Li Z., Shen H., Cheng Q. Deep learning cloud detection for high resolution remote sensing images. ISPRS Journal of Photogrammetry and Remote Sensing, 2019. 212 p.

9 Introduction to Neural Networks URL: <https://www.methodpark.de/blog/introduction-to-neural-networks/> (дата звернення: 08.10.2023).

10 The Mathematics of Neural Networks URL: <https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05> (дата звернення: 08.10.2023).

11 Convolutional neural network (CNN) URL: <https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network> (дата звернення: 08.10.2023).

12 Convolutional Neural Networks: The Biologically-Inspired Model
URL: https://www.codementor.io/james_aka_yale/convolutional-neural-networks-the-biologically-inspired-model-iq6s48zms (дата звернення: 08.10.2023).

13 Understanding of Convolutional Neural Network (CNN) – Deep Learning
URL: <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148> (дата звернення: 08.10.2023).

14 What is UNET? URL: <https://medium.com/analytics-vidhya/what-is-unet-157314c87634> (дата звернення: 12.10.2023).

15 Ronneberger O., Fischer P., Brox T. U-net Convolutional networks for biomedical image segmentation. Springer, 2015. 241 p.

16 About LinkNet URL: <https://wiki.cloudfactory.com/docs/mp-wiki/model-architectures/linknet> (дата звернення: 12.10.2023).

17 About DeepLabv3 URL: <https://paperswithcode.com/method/deeplabv3> (дата звернення: 13.10.2023).

18 Review: Dilated Convolution (Semantic Segmentation) URL: <https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5> (дата звернення: 13.10.2023).

19 Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations (ICLR), 2015. 60 p. (дата звернення: 14.10.2023).

20 How to make a building footprint detector URL: <https://medium.com/picterra/how-to-make-a-building-footprint-detector-768045a15bcc> (дата звернення: 15.10.2023).

21 2nd SpaceNet Competition Winners Code Release URL: <https://medium.com/the-downlinq/2nd-spacenet-competition-winners-code-release-c7473eea7c11> (дата звернення: 15.10.2023).

22 Towards accelerating disaster response with automated analysis of overhead imagery URL: <https://towardsdatascience.com/towards-accelerating-disaster-response-with-automated-analysis-of-overhead-imagery-18c65731eaaf> (дата звернення: 16.10.2023).

23 Microsoft Building Footprints URL: <https://planetarycomputer.microsoft.com/dataset/ms-buildings> (дата звернення: 17.10.2023).

24 Top 10 Tech Stacks That Reign Software Development in 2023 URL: <https://www.fingent.com/blog/top-7-tech-stacks-that-reign-software-development/> (дата звернення: 20.10.2023).

25 What Is Nest.JS? Why Should You Use It? URL: <https://www.turing.com/blog/what-is-nest-js-why-use-it/> (дата звернення: 20.10.2023).

26 DB-Engines Ranking URL: <https://db-engines.com/en/ranking> (дата звернення: 22.10.2023).

27 What is MongoDB – Working and Features URL: <https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/> (дата звернення: 25.10.2023).

28 Reasons Why You Should Use React for Web Development URL: <https://www.techmagic.co/blog/why-we-use-react-js-in-the-development/> (дата звернення: 26.10.2023).

29 Material UI URL: <https://mui.com/> (дата звернення: 26.10.2023).

30 What are the Pros and Cons of React URL: <https://www.knowledgehut.com/blog/web-development/pros-and-cons-of-react#what-is-react-js?%C2%A0> (дата звернення: 26.10.2023).

31 Everything About Software Architecture URL: <https://medium.com/swlh/everything-aboutsoftware-architecture-dfd2b9351ef4> (дата звернення: 01.11.2023).

32 What is monolithic architecture? URL: <https://euristiq.com/monolith-vs-microservices-architecture/> (дата звернення: 01.11.2023).

33 About Microservices Architecture URL: <https://www.baeldung.com/cs/microservices-vs-monolithic-architectures> (дата звернення: 01.11.2023).

34 What is MongoDB Atlas? URL: <https://www.mongodb.com/docs/atlas/> (дата звернення: 04.11.2023).

35 Create web maps with GL JS URL: <https://www.mapbox.com/mapbox-gljs> (дата звернення: 08.11.2023).

36 'What is Dropbox?': How to use the cloud-based file-storage service for collaboration URL: <https://www.businessinsider.com/guides/tech/what-is-dropbox> (дата звернення: 08.11.2023).

37 Google OAuth2 API Explained URL: <https://medium.com/@pumudu88/google-oauth2-api-explained-dbb84ff97079> (дата звернення: 09.11.2023).

38 OpenCV – Overview URL: <https://www.geeksforgeeks.org/opencv-overview/> (дата звернення: 10.11.2023).

39 SimpleCV: Making Vision Computing Easy and Effective URL: <https://www.opensourceforu.com/2016/05/simplecv-making-vision-computing-easy-and-effective/> (дата звернення: 10.11.2023).

40 DLib: Library for Machine Learning URL: <https://www.kdnuggets.com/2014/06/dlib-library-machine-learning.html> (дата звернення: 10.11.2023).

41 Detect and analyze real-world objects with Mapflow AI URL: <https://mapflow.ai/> (дата звернення: 11.11.2023).

42 A Complete Guide to Sources and Layers in React and Mapbox GL JS URL: <https://www.lostcreekdesigns.co/writing/a-complete-guide-to-sources-and-layers-in-react-and-mapbox-gl-js/> (дата звернення: 11.11.2023).

43 About GeoJson Data URL: <https://docs.oracle.com/en/database/other-databases/nosql-database/21.1/sqlreferencefor nosql/geojson-data.html#0> (дата звернення: 11.11.2023).